CryptoPhoto Internals

Overview of the cryptographic mechanisms and protection, data flow, and conceptual architecture of the CryptoPhoto Authentication solution.

17 February 2017

Chris Drake (CryptoPhoto)

Praveen Gauravaram (TCS)

Abstract

We present a full-lifecycle examination of CryptoPhoto internals, including implementation deployment, enrolment, usage, and maintenance. We analyse the security aspects of the two main security functions (authentication and transaction-verification), and also the supporting systems and infrastructure needed to securely deploy, use, and maintain them. We examine a broad range of threats prevented by CryptoPhoto and compare it's efficacy to assorted legacy authentication methods. In comparison with existing tech, we find more than 100 security and useability improvements – including:-

- Transaction verification in addition to 2FA mutual authentication from human to server and server to human
- Human involvement in authentication and verification instead of mere device
- Strong separation of duties architecture between the device used to login and device used to authenticate login and verify transactions through out-of-band and second channel provides valuable security and privacy benefits.
- Novel use cases are accommodated including telephone and in-person as well as online.
- Excellent coverage of periphery issues, including enrolment, identity binding, self-service, loss-handling, defence against pre-existing compromise, social attacks, travel, and more.
- Wide breadth of coverage against a comprehensive array of modern attack scenarios.
- Provides working high-security even to unmotivated and unsophisticated users; security is **not** dependant on user intelligence or care, and no training is necessary.
- **More important** than any of the foregoing: CryptoPhoto is fast, easy, and a pleasure to use.

CryptoPhoto is a fast, easy, and complete, LoA3[i]-class authentication 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. enrolment, use, maintenance, etc, plus all side-channels and exceptions.).

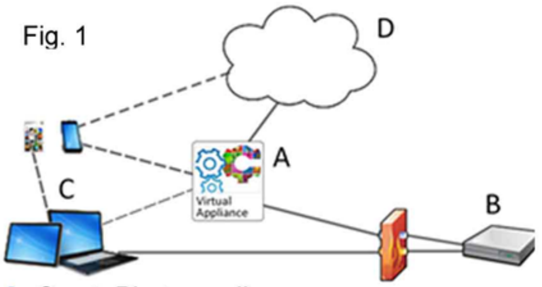
The functionality of the protocol along with crypto/security controls and their purpose is detailed below at various stages of CryptoPhoto deployment and usage.

1. Conceptual Overview

Customer **C** using their PC/Tablet/phone and CryptoPhoto token(s) accesses provider service **B** (through provider firewall), with CryptoPhoto protection enforced via appliance **A** with help from cloud services **D**.

CryptoPhoto is:
i) one or more virtual appliances (**A**), which communicate with…
ii) CryptoPhoto App in customers' mobile devices and/or physical printed tokens (**C**), plus
iii) our server-side API or SDK which is used by the Provider system (**B**) needing protection plus cloud services to originate PUSH & SMS (these are convenience, not security, services).

Fig. 1



A. CryptoPhoto appliance
B. Banking servers (behind firewall)
C. Customers with phones, tablets, PC's etc
D. Supporting services (SMS, Push-messaging, app stores, outgoing email)

Authentication

Customer **C** loads website **B** and enters username and optional password. Server **B** makes API call to appliance **A** to determine if customer is enrolled with CryptoPhoto. If not, **B** logs customer in (* see note). If yes, appliance **A** returns challenge widget containing a photo to server **B** which it displays on customer **C**'s browser, and appliance **A** additionally triggers a PUSH through cloud **D** to auto-open customer **C**'s token for them. Customer **C** uses token to solve the challenge by taping the photo on the App on their phone, which matches the one displayed by the widget. The photo-tap triggers App on the customer **C**'s mobile device to securely communicate a signed and encrypted EOTP (event based one-time password) response to appliance **A**, which (if correct) signals customer **C**'s browser to auto-proceed (this step is for convenience, not for security). Finally, server **B** checks again the customer's EOTP is correct, and logs them in. Appliance **A** is isolated from (has no knowledge of) customer identity.

For the case where customer **C** does not use a mobile phone, they submit the EOTP manually into their browser from their physical token to complete login.

Significant technical effort exists throughout, preventing a wide variety of different attack scenarios from compromising user authentication, bypassing it, hijacking sessions, impersonation, substitution and injection.

* Note: when CryptoPhoto is added to an existing system, most providers will allow users who have not yet enrolled with CryptoPhoto to log in using their existing authentication (usually a username+password plus possible a legacy SMS/OTP). The CryptoPhoto recommended enrolment procedure for these cases differs slightly from the above flow: un-enrolled users are not logged in, they are directed through the enrolment system and then logged out, at which point they then log back in using CryptoPhoto protection. Optional support for protecting enrolments over already-compromised channels also exists.

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 server **B** or via **C**'s browser, depending on implementation), prepares it for display to customer, and triggers a PUSH through cloud **D** to customer **C**'s mobile device, which securely retrieves the transaction to be displayed from appliance **A**. (communications between appliance and customer's device and browser are always over TLS and additionally secured[ii] using pre-shared RSA keys for both customer device and appliance, which are optionally biometrically encrypted in device storage). Note that Appliance **A** can display and sign anything – providers will need to decide between malware-resistance and "privacy" (Appliance **A** does not store transactions). Customer **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 directly to appliance **A**, which in turn communicates it to server **B**. (depending on implementation, communication to **A** and **B** might be via customers **C**'s browser). Server **B** checks for signature match and customer approval, and processes the verified transaction. In the event of a decline by customer **C**, or mismatch at server **B**, appliance **A** additionally triggers a cloud **D** PUSH to request decline reasoning from customer **C**, which is passed to server **B** (to detect attacks and potential customer compromise in real-time). This is illustrated in Figure 10 in Section 5. Note that Provider **B** initiates all transaction processing: we recommend that failed and declined transactions are sent to customer using our cTV system to ascertain what's going on (accident, or attack?) – this is optional, but if in use, it's **B** that tells **A** to do this. (**A** doesn't know about the meaning – it's just another CryptoPhoto Transaction Verification (cTV) event it processes) The protection as for authentication exists also for cTV, and additional significant technical effort has been applied to make integration of this technology into legacy systems quick, easy, and safe; a typical cTV upgrade to an existing website **B** transaction-acquiring web <form> consists of two lines of code in the form (to trigger the cTV), and two more in the processing server (to check the signature); no queuing etc needed. Multiparty[iii] approvals (or declines) are also supported (in and out of band).

Transaction Verification (out of band)

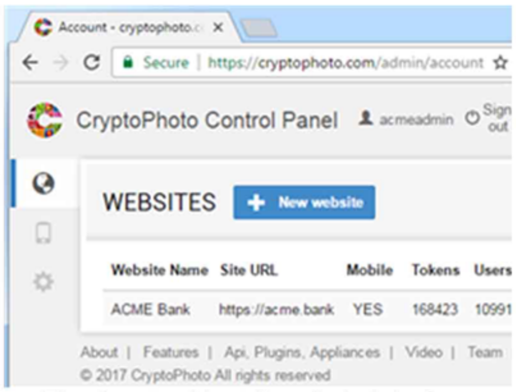
Server **B** triggers a PUSH (via **A**) through cloud **D** to the CryptoPhoto App on customer **C**'s mobile device, which signals an audio alert. Customer **C** unlocks their phone (if not already) which retrieves the transaction from server **B** (as per the above inline method) and displays it to customer **C**. Customer **C** follows on-screen instructions, and taps or selects an appropriate option. Customer **C**'s response and any associated data is digitally signed and the signature is communicated to appliance **A** which in turn communicates it to server **B** (** see note). An example of using this solution for a bank's human-to-human telephone customer verification is illustrated in Figures 6 and 7 in section 4 (page 5). In this example, customer **C**'s response will be communicated to the banks customer service representatives' screen.
Note:- CryptoPhoto App only talks to the appliance (using extra encryption as mentioned above – to protect against MitM certificate-substitution attacks). Appliance **A verifies the signature; the API call communicates result to server **B**.

Integration

1. Provision CryptoPhoto authentication appliance "**A**" (one or more real or virtual machines; internet-facing)
2. Identify server "**B**" which is to be protected by CryptoPhoto (or optionally its WAF if no change to **B** is allowed).
3. Provision one new menu option into the server "**B**" system, e.g. "Security Settings" which navigates to one "blank" page (page content will be the user enrolment and token-management subsystem and is supplied from appliance "**A**")
4. Integrate authentication into existing login flow (typically, this appears as an extra page after user has entered username+password, but before they reach their "logged in" state).
5. Integrate transaction-verification into one or more possible customer operations (this is accomplished by minor adjustments to the source page, and insertion of a signature-verification operation inside the server "**B**" system that ordinarily processes the operation. No need for queuing etc exists).
6. Test with "**C**".

2. Provider Deployment

Providers (e.g. a Bank) have 2 deployment options; "Cloud" (which uses the CryptoPhoto shared authentication infrastructure) or "Appliance" (providers deploy real or virtual on-premise security machines for their exclusive use). We will assume the Provider opts for this second case, which is most suited to a Bank.


Account - cryptophoto.c
Secure | https://cryptophoto.com/admin/account
CryptoPhoto Control Panel
acmeadmin
Sign out
WEBSITES
New website
About | Features | Api, Plugins, Appliances | Video | Team
© 2017 CryptoPhoto All rights reserved

Website Name	Site URL	Mobile	Tokens	Users
ACME Bank	https://acme.bank	YES	168423	10991

Fig. 2 – provider site admin interface

A CryptoPhoto security appliance is a professionally hardened SELinux CentOS machine which is stripped and customized to run the CryptoPhoto authentication services. To ensure a trusted and clean initial O/S install, the CryptoPhoto deployment system commences with a "bare metal" erasure of the machine it runs from (this includes commercial clouds, like Amazon, Azure, etc) to ensure no unwanted or insecure initial conditions[iv] exist that might cause future compromise. A clean automated O/S install is performed through the Anaconda[v] kick-start subsystem which includes the fresh generation of assorted Linux initial keys. Prior to key generation, CryptoPhoto kick-start customizations introduce two hardware True-Random TRNG[vi] drivers to ensure all new keys are generated from secure non-deterministic entropy. Optional LUKS full-disk-encryption with remote-unlock exists to protect virtual appliances against assorted host-intrusion threats, and all appliances against physical threats.

CryptoPhoto appliances enforce HSTS[vii] and HPKP[viii] and exclusively communicate using TLS with Apps, browsers, and it's API. To additionally protect against risks introduced through certificate-substitution intermediaries[ix], appliances generate their own RSA asymmetric keypairs, and the CryptoPhoto update servers also use a long-term RSA asymmetric keypair as well. The appliance public+private keypair is auto-generated at appliance deployment time and is a 2109-bit[x] RSA keypair. The public component of the CryptoPhoto updatekeypair is embedded in the CryptoPhoto App and/or mobile SDK[xi] (hereafter just "App/SDK"). CryptoPhoto appliances distribute CryptoPhoto soft-tokens[xii] inside JSON[xiii] data packets to customer mobile devices during enrolment provisioning[xiv]; the public component of the 2109-bit RSA key forms part of this JSON. The App/SDK additionally generates a per-token 2048-bit RSA keypair and uploads the public component to the appliance during enrolment. The App/SDK uses these asymmetric keys to encrypt communications between the appliance and mobile App (See note***), such communications include the EOTP[xv] key of an associated token photo when a user taps on the matching photo, the digital signature of transaction-verification messages, "bump" pairing[xvi] notices, and appliance endpoint responses of token discovery[xvii] used in initial user enrolments. Encrypting these things, even though they already travel over TLS, is done to protect them against possibly malicious certificate-substitution situations[ix]. Asymmetric cryptography is used to protect the communications even though a TLS intermediary may gain access to the public key of the token during enrolment (knowing a public key is insufficient to decrypt communications made with it).

Assorted very-old mobile devices, which are too slow to properly process this additional in-device RSA cryptography, are exempted from using it[xviii].

Appliances talk to customer devices, to Provider web pages, and to Providers through API calls and/or an SDK. API keys for the appliance (random text strings) are also generated during deployment. Provider logos and branding are uploaded to the CryptoPhoto website through an admin panel, and are subsequently used by Appliances for branding.

In some situations, an on-prem provider appliance, and/or a customer's mobile device, may need to communicate with a CryptoPhoto update server (for example; to resolve a TokenID meaning, retrieve updates, accept new tokens, verify update integrity, apply logos/customization/branding, initiate 3rd party messaging, and so forth). For this purpose, a long-term CryptoPhoto 2048bit RSA keypair exists, and new Appliances and customer apps come pre-installed with the public component of this key. This also facilitates secure communication in the presence of TLS certificate-substitution.

For details on Appliance administration, initialisation process and integration of CryptoPhoto with a provider's IT infrastructure, refer to Appendix 1 (page 10).

***Note: This involves standard way of generating a secret key and using PKC to communicate it and later using it for encryption/decryption. Standard browser-supported javascript crypto implementation is used for this.

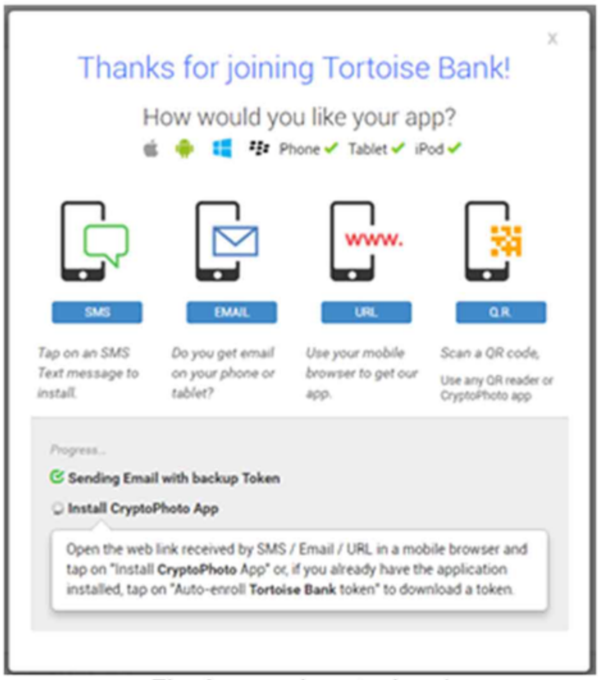
3. Initial User Enrolment.

There are multiple different ways Users can be enrolled to use CryptoPhoto with a Provider. Ultimately, enrolment concludes with a User possessing one or more CryptoPhoto Tokens which are bound[xix] to their account, and usually also having the CryptoPhoto App (or Provider app with CryptoPhoto SDK included) installed on their mobile device (phone/tablet/watch/etc). Ideally, an enrolled user will have two tokens; one inside an app in their phone/tablet, and the other a physical printed "recovery" token, which they keep in a safe place in case they lose their phone[xx].

- For Providers who opt to include CryptoPhoto facilities into their Providers own smartphone Apps (e.g. mobile internet banking) using the CryptoPhoto mobile SDK, Users will automatically obtain the CryptoPhoto app software (SDK really) when they install or update their Provider apps. In this scenario, there is no additional requirement for the user to obtain a separate app.
- For Providers using CryptoPhoto separately, the Appliance* (*see appendix 1) includes a guided mechanism to help a User obtain the requisite App (and token to go with it usually). See fig. 3 →
- The CryptoPhoto appliance also guides users to rapidly install Provider Apps (e.g. a bank-branded BaNCS Digital app with the CryptoPhoto SDK) in the event this is necessary[xxi].

Soft-token enrolment commences with a User who is logged-in and Authenticated to a Provider (to reach this point, they will have used their existing authentication method, possibly also with legacy 2FA if present, and they may have been required to use their CryptoPhoto hard-token if issued).

Upon login, the provider can either direct the user to obtain a soft-token immediately (recommended), or can provide links that the user can click to reach the soft-token enrolment page.


x
Thanks for joining Tortoise Bank!
How would you like your app?
Phone ✓ Tablet ✓ iPod ✓
www.
SMS
EMAIL
URL
Q.R.

Tap on an SMS Text message to install.

Do you get email on your phone or tablet?

Use your mobile browser to get our app.

Scan a QR code, Use any QR reader or CryptoPhoto app

Progress...

✓ Sending Email with backup Token

Install CryptoPhoto App

Open the web link received by SMS / Email / URL in a mobile browser and tap on "Install **CryptoPhoto** App" or, if you already have the application installed, tap on "Auto-enroll **Tortoise Bank** token" to download a token.

Fig. 3 – enrolment wizard

The Enrolment system is retrieved from the CryptoPhoto Appliance via API (Illustrated in the above Figure) by the Provider web site and presented to the user with instructions. If the user is on a mobile device, this is detected, and the Enrolment system either directs the users' device to their appropriate store to obtain the app software, or (if they already have the app software) their app is opened to obtain a token which is bound to the Provider automatically.

If the user is not on their mobile device (or does not want to use it for the app), they are presented with a selection of other methods to obtain the app on the device they want. All of these methods, as well as the "app store redirect" method mentioned in the previous paragraph work as follows:

The users' mobile device is directed to an intermediary page which sets a temporary web browser cookie on the users' mobile device. It then re-directs to the store for the user to install the free app. When the free App is installed, its initial invocation opens the web browser to retrieve the temporary cookie, which it uses to determine which Provider and what token and user is needed, and finally an appropriate token for the user is downloaded and bound. In other words – all a user needs to do is install the app: the CryptoPhoto appliance's Enrolment system takes care of making everything work automatically for the user, without requiring them to follow additional instructions or understand store mechanisms, and without risking them installing a wrong app. In UX timing comparisons, this enrolment is approximately 10x (1000%) faster than existing app-based 2FA enrolment systems[xxii].

Cryptography controls: When a token is downloaded and installed on a mobile device, a 2048-bit RSA key pair is generated on the device. The public key is communicated over TLS directly to the Provider's authentication appliance as specified in the token. The mobile device uses the RSA private key to sign 64-bit EOTP codes corresponding to challenge images whenever a user clicks on the matching candidate Image sent by a provider during a login (mutual authentication). The device then encrypts this signature with the public key of the Appliance and sends the ciphertext over TLS to the Appliance. This additional step protects against certificate substitution attacks.

Provider Discovery: All CryptoPhoto soft-token enrolment mechanisms (including Mobile-App store redirection, SMS, URL, Email, QR Code, ultrasonic, Bump, manual-typing, and Barcode) are designed to deliver a single TokenID (12 digit random number) to the App/SDK; this TokenID is then looked-up in the CryptoPhoto cloud server to resolve the random number to the appropriate appliance endpoint of the Provider, from where a token with suitable branding etc is subsequently obtained.

TokenIDs, where embedded in HTTPS URLs, use base36 (typically all uppercase for QR encoding efficiency) encoding, with the base portion of the URL directing incoming requests to a CryptoPhoto cloud CGI service programmed to activate the app, or redirect users to an appropriate store to obtain the appropriate app if they don't already have it. In other words – no matter who encounters the enrolment system, or how, it always works quickly and easily and safely for the user, even if they do unexpected things (e.g. scan our QR with the wrong app). CryptoPhoto Apps/SDK extracts TokenIDs from URLs or QR codes directly; they do not resolve any URLs encountered.

Fig. 4 – CryptoPhoto TLS test results

CryptoPhoto TLS is certified A+ grade with perfect forward secrecy and all trust markers and security enforcements working and activated.

4. Mutual-Authentication.

CryptoPhoto mutual-auth works online, in-person, and via telephone:

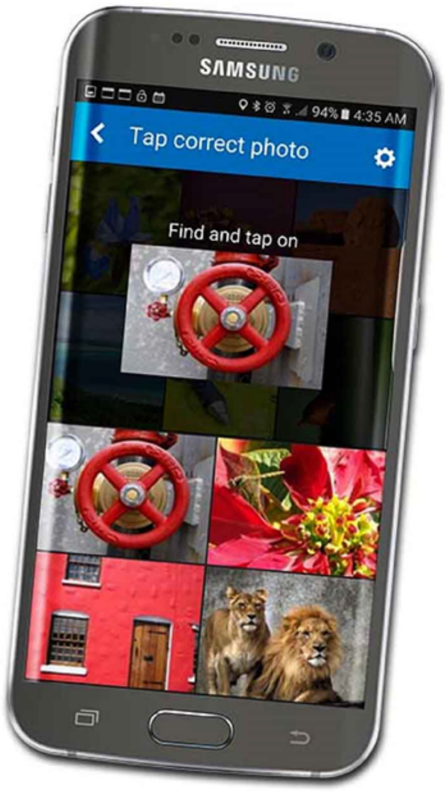


Fig. 5 – Online

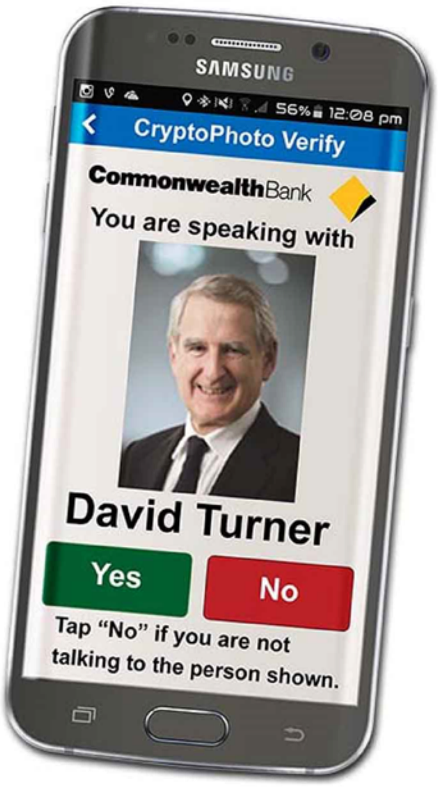


Fig. 6 – In person

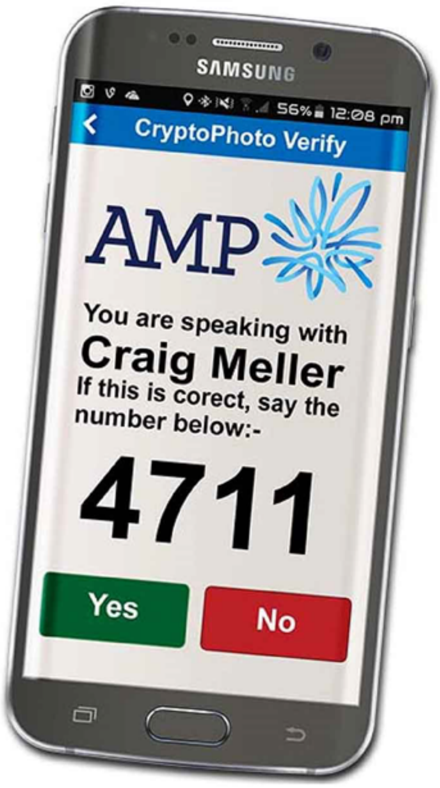


Fig. 7 – Call centre

Mutual-authentication (both parties proving to one another their legitimacy; e.g. a website and the user) is an important tool for combatting security attacks including phishing, social engineering (against both users and staff), and man-in-the-middle attacks. CryptoPhoto uses an entirely new form of mutual authentication; one which involves the human's brain as part of the authentication mechanism itself (legacy mechanisms authenticate devices, not a person, making them almost universally ineffective[xxiii]). The CryptoPhoto authentication user experience makes use of our naturally evolved ability to quickly recognize a matching pair of photos; users are not required to understand how or why they are involved in a mutual-authentication cryptographic protocol, they merely need to take one or two seconds necessary to tap the matching photo. This mechanism protects even unsophisticated and unmotivated users without requiring training or help, and blocks even sophisticated attacks from working against them.

For in-person mutual authentication, they confirm that the they are speaking with the correct person (the other party does the same), and for over-the-phone authentication, they read out a random code to satisfy the other party that they have their phone in their possession at the time.

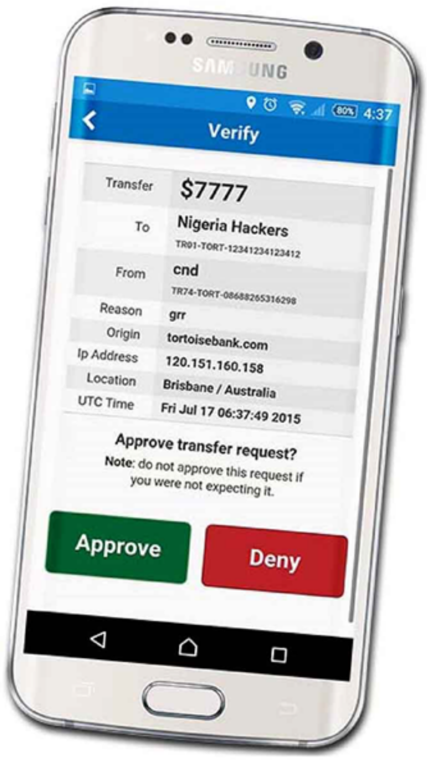
Refer to Appendix 2 for the technical mechanisms through which CryptoPhoto attains mutual authentication Online (Appendix 2A), In-Person (Appendix 2B) and via Telephone (Appendix 2C) communications.

Refer to Appendix 5 for the CryptoPhoto mechanisms preventing image-theft, MitM, and screen-scraping attacks.

Mutual-Authentication is not a substitute for Transaction-Verification: we recommend that important actions (e.g. transferring money, changing address, closing accounts, etc) requested/submitted by customers also be verified using Transaction-Verification features, which additionally allow customers to check *over a secure second channel* that their request has been submitted accurately, offers non-repudiation to the provider, and neutralizes malware and MitM attacks.

5. Transaction Verification

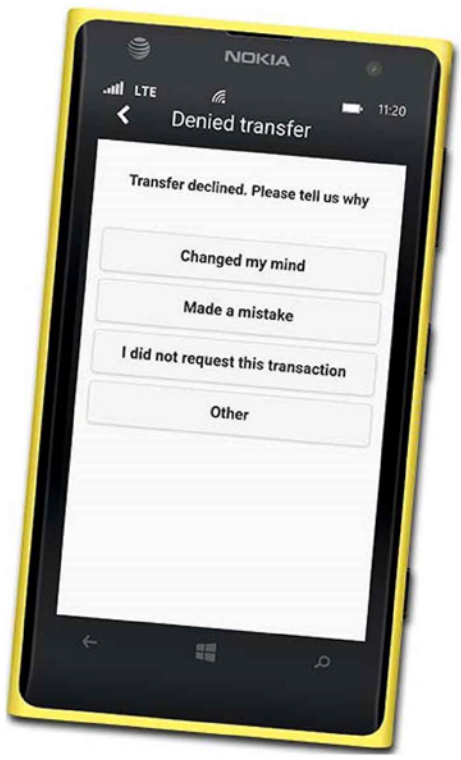
The point of transaction verification is to confirm users' intent using an independent device. "Transaction" has the broadest possible meaning – it might literally be a financial transaction, but it can be anything else as well, like changing New Payments Platform (NPP) aliasing, erasing a virtual machine, cancelling a remote house alarm, etc: any time when a secure non-repudiable confirmation from a user needs to be obtained, CryptoPhoto Transaction Verification (cTV) steps in.



Transfer	$7777
To	Nigeria Hackers TR01-TORT-12341234123412
From	cnd TR74-TORT-08688265316298
Reason	grr
Origin	tortoisebank.com
Ip Address	120.151.160.158
Location	Brisbane / Australia
UTC Time	Fri Jul 17 06:37:49 2015

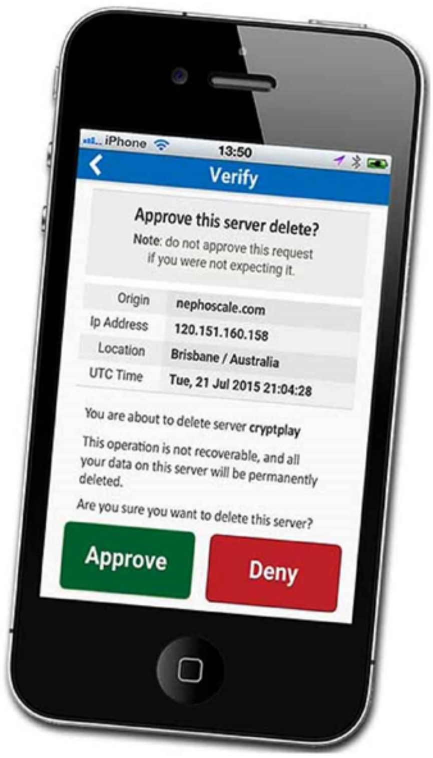
Approve transfer request?
Note: do not approve this request if you were not expecting it.

Fig. 8 – Financial Transactions



Transfer declined. Please tell us why

Fig. 9 – Real-Time attack detection with low false positives



Approve this server delete?
Note: do not approve this request if you were not expecting it.

Origin	nephoscale.com
Ip Address	120.151.160.158
Location	Brisbane / Australia
UTC Time	Tue, 21 Jul 2015 21:04:28

You are about to delete server cryptplay
This operation is not recoverable, and all your data on this server will be permanently deleted.
Are you sure you want to delete this server?

Fig. 10 – Any other secure notice and response need

cTV principally neutralizes malware and MitM attacks; any activity that might interfere or originate transactions will be unable to carry out its attack without the user's approval via their cTV token. The cTV mechanism presents a full-screen, rich, out-of-band[xxiv], second-channel[xxv], self-opening interface to the user for confirmations; the provider designs this screen with their biggest malware risk in mind – so for internet banking transactions, the screen is designed to clearly display the transaction amount and intended recipient. The user is instructed on this display to carefully check these before confirming the transaction.

There are two invocation mechanisms: browser, and server.

1. cTV browser invocation: this method exists to make cTV integration with existing/legacy systems as easy as possible. One small change is made to the web <form> where transactions are submitted, and one small change is

made to the server responsible for processing these transactions. The form change instructs the CryptoPhoto appliance to trigger the appropriate cTV out-of-band (OOB) display for the user, and to accept back from that user their digitally signed confirmation produced to cover all the elements of the form. The server change is to check that the customer approved the transaction and their digital signature matches the form variables before processing. In other words, just one or two lines of the form, and one or two lines of the server code need adjustment to make cTV work. It is also possible to do all cTV processing on a WAF[xxvi] (e.g. ISAM4W) making it possible to add strong transaction-verification without changing any legacy code on the server.

The point of method 1 is to avoid having the server need to queue transactions while waiting for confirmations, which is why it's just a couple of easy lines of code to implement.

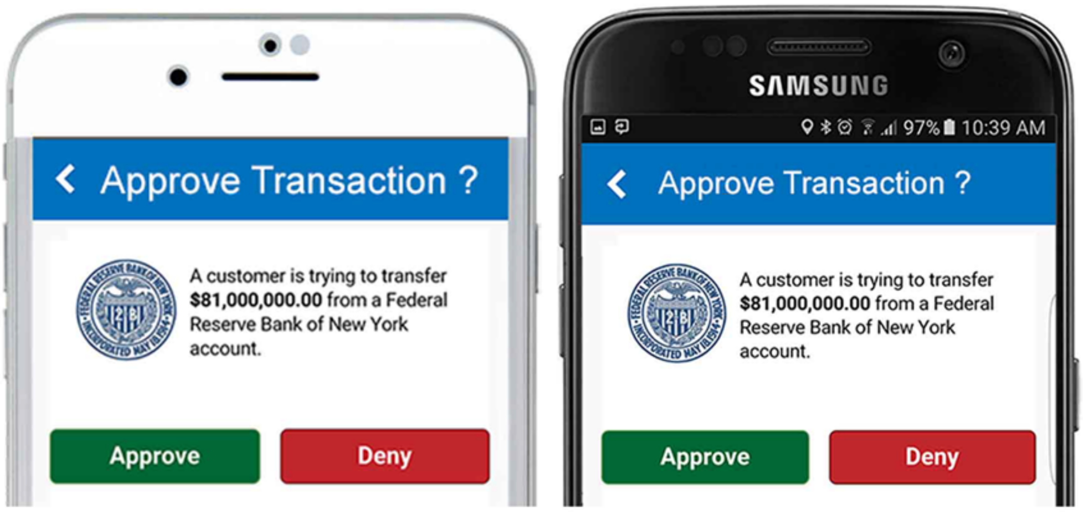
2. cTV server invocation: with this method, the server sends incoming transactions to the users' device out-of-band, and awaits their confirmation in response. The server communicates with the appliance directly for this.

When a customer approves a transaction, the digital signature built from the private key of mobile App over the entirety (or selected elements as per Providers implementation preferences) of the transactions' form data, together with the customers' approval response is communicated to the Appliance, which in turn communicated this to either (1) the Users web form (whereupon it is inserted into a hidden form field and subsequently on-sent to the Providers server), or (2) the Providers server directly. Either way, when the server receives the signature, it makes an API call to the Appliance to check validity, checks that the fields are unmodified (e.g. not malware-adjusted), and that the user approved the transaction, before the server processes the transaction.

If no user action is received, the server either (1) never gets any request, or (2) never gets approval, and so either way, the transaction is not processed. Timeouts are enforced as per existing provider policies.

In the event of a user not wishing to proceed with a transaction, CryptoPhoto can optionally enquire why not. In our example Figure.9, we give the user 3 "decoy" answers, plus the answer "I did not request this transaction" – the purpose of the decoys is to give the user a way to make mistakes that do not bother the banks' Network Operations Centre (NOC), but the special answer can be connected to the NOC to provide a real-time alert that the bank is experiencing a phishing outbreak, or that a user's account has been compromised or a user has a malware infection, all with low false-positive rates.

Multi-Party approvals are also supported.



A customer is trying to transfer **$81,000,000.00** from a Federal Reserve Bank of New York account.

Fig. 11 – multiparty approval with 2 operators

Users who require one or more approvals to be given from a pool of authorised people can set up appropriate rules for their workflow. For example – high value transactions might require 3 different users from a group of 5 to concur, before the transaction is processed, while any one member from the group of 5 might have permission to deny the entire transaction (either before concurrence, or within whatever time-frame a Provider might choose)

Transactions are not necessarily financial. Adding new users to accounts, changing account aliasing, and binding new tokens are examples of non-financial transactions, which are also supported by CryptoPhoto.

Non-repudiation is supported via CryptoPhoto digital signatures, to help protect the Provider against fraudulent customers (for example: a customer sending $10,000 to his friend in Nigeria, with plans to later seek a refund from the bank claiming malware). Depending on Provider policies, the CryptoPhoto app can be enabled to support the collection of additional transaction metadata (device geolocation/GPS, networking, front-facing-camera images, etc), which can be helpful to successfully detect and prevent these kinds of frauds.

6. Self-Service maintenance.

Similar to initial Enrolment, a Provider server makes an API call to the CryptoPhoto appliance to retrieve the maintenance system to be presented to the users' device.

Options typically available through this interface include:

Activation button: allows (if permitted by Provider policy) a user to "disable" CryptoPhoto on their account; their tokens remain bound, but are not used until this setting is re-enabled. Typically, provider policies require that disabling protection is permitted only after Transaction-Verification (to prevent malware maliciously disabling a customer's protection without their consent)

Wizard: steps a user through the enrolment process again (e.g. so they can set up a new phone).

Add Token: lets users create and download new hard-tokens.

Download / Email: transfers a hard-token to a user.

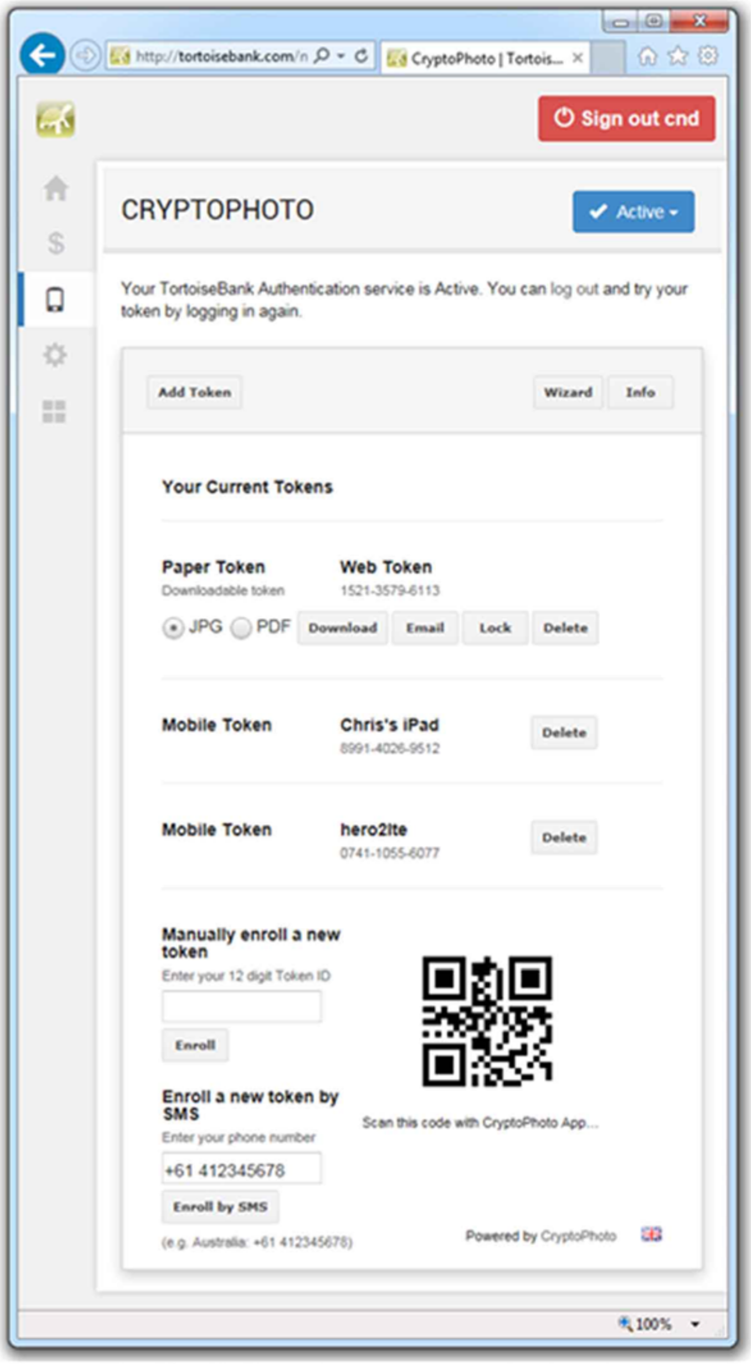
Lock: prevent future transfer/download of a token to a user.

Delete: permanently remove a token's binding from their account (also removes it from their mobile device where possible)

Manual enrolment: allows users to bind a new hard-token to their account.

QR code and SMS: quick way to let users acquire and bind new soft-tokens.

Info: Audit trail of users' token-usage history
CryptoPhoto setup allows Providers to choose a variety of operating options for their CryptoPhoto protection, including which of the above mechanisms exit (or not), and which ones require the use of Transaction-Verification using existing tokens before users can get new ones, or erase old ones, etc (to protect users against malware).



Your TortoiseBank Authentication service is Active. You can log out and try your token by logging in again.

Your Current Tokens

Paper Token Downloadable token	Web Token 1521-3579-6113	
Mobile Token	Chris's iPad 8991-4026-9512	Delete
Mobile Token	hero2lte 0741-1055-6077	Delete

Manually enroll a new token
Enter your 12 digit Token ID

Enroll a new token by SMS
Enter your phone number
+61 412345678
(e.g. Australia: +61 412345678)

Fig. 12 – Self-service management subsystem

7. Functionality of CryptoPhoto Images and the respective security analysis.

One-Time-Passwords (OTP) Definition: They are a one-way alphanumeric code based mechanism for a human to prove their authenticity to a website during login. The computer accepts the secret OTP code from the human, compares this to its own version, and when there's a match, the computer knows the human is authentic. Computers are very good at matching alphanumeric codes. OTP codes change after each use – they never come up twice in a row (so that if a code is stolen, it will not be able to be used next time). There are a finite number of OTP codes, usually random, so they do eventually repeat. (e.g. 1 time in every 10,000 uses on average for a 4-digit numeric OTP)

CryptoPhoto performs two-way (mutual) authentication. When a server needs to prove its authenticity to a human, the computer uses a random photo instead of a random alphanumeric code, because humans are very good at matching photographs. When a human does a login with CryptoPhoto, the computer screen shows them a random image, the human finds this image among the photos on their CryptoPhoto token, and when there is a match with the photo on mobile App or static hard Token that human has , the human knows the server from which the image has originated on the computer screen is authentic. The CryptoPhoto images change after each use – they never come up twice in a row (so that if an image is stolen, it will not be able to be used at next login). There are a finite number of images, shown at random, so they do eventually repeat.

CryptoPhoto connects these two concepts together in a "protocol dance" so that it's fast and easy for the human, and doesn't need the human to understand why they're matching the photo, or what the match means (it means, of course, that the human knows the website is authentic, except the human doesn't need to know that they know that for this to work).

There are two kinds of CryptoPhoto tokens: static, and dynamic. On static tokens, the images are fixed and never change. On dynamic tokens, images that have been used are replaced by new unused images after use. At some point, there will come a time when a user has used every single possible image in the Token. When this time is reached, CryptoPhoto will start repeating photos they have already used before. The algorithm which does this is programmed to ensure that any repeated image is never shown to a user immediately after that same image has been used by the user (i.e. no image will ever show up twice in a row).

CryptoPhoto includes defences against image-alphanumeric code theft (including agent-based TLS-session-key computed local display, robotic and non-human browser image request detection/deception, and photo tracking, signing, and manipulation).However, should somehow a user's login photo nevertheless be "stolen", that photo will not be useful to the attacker because it will not be used again next time until at least 95% of the other photos have been shown since. In other words, no photo shows up again until all the other photos have been shown.

Note that it is NOT the photo which logs users in – the photo is the selector into an EOTP array which is then digitally signed to facilitate the login. Theft of photos, no matter how many, will not facilitate an unauthorized login.

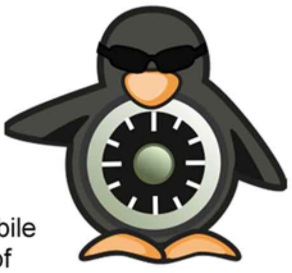
It is up to the provider to decide what policy to enforce with respect to exhausted static tokens (note that a mobile phone token, when used in offline-mode (rare), behaves the same as a static token): in the case of a physically-printed hard-token with (say) 36 photos and 36 codes, the provider can choose to (a) issue the customer a new token with new photos and new codes, or (b) allow the customer to keep using this same token. Option (b) carries the low risk that if the user is compromised 36 times in a row, and the attackers manage to steal their credentials (username and password) and all 36 photos and all 36 codes, then from the 37th login onwards, they would have sufficient information to authenticate. But this equivalent to stealing the hard CryptoPhoto token. CryptoPhoto protects against image and code theft as well, as explained above which is why this is a low risk; an attacker making multiple attempts to log in in order to "poll" for the an image they may have stolen in the past will trip the suspicious activity counter, while an attacker somehow able to observe user login photos while waiting for an opportunity to use a stolen EOTP code will be "kicked" almost immediately when the real user completes their login..

Note that photo-alphanumeric code compromise does not apply to soft-tokens, because EOTP codes corresponding to photo-alphanumeric code are additionally digitally signed and timestamped by the private key on users' mobile phone App, and encrypted with the appliance public key, thus not subject to theft or replay.

8. Summary of Appendices.

The rest of this paper is organised as follows: In Appendix 1, we cover what CryptoPhoto appliances are, how to install them, and how to integrate CryptoPhoto protection into an existing application. Appendix 2 explains how to use CryptoPhoto offline; over the telephone and in-person, as well as online. Appendix 3 explains the layout reasoning for Figure 17 (page 21). Appendix 4 details the CryptoPhoto tokens and their data structures. Appendix 5 covers the additional supporting protections present in CryptoPhoto used to secure the operating environment. Appendix 6 details some common attack scenarios, how CryptoPhoto prevents them, and how competing products fair for the same attacks. It ends with a more thorough enumeration of attack scenarios and user-experience considerations with a table comparing CryptoPhoto against the range of competing technologies on the market.

Appendix 1 - Appliances, their administration, initialisation and integration.



CryptoPhoto Appliances are one or more professionally-hardened Security-Enhanced Linux real or virtual servers which enable the CryptoPhoto functionality. They live internet-facing within the provider infrastructure, or optionally, providers can make use of shared Public appliances operated by CryptoPhoto.

Appliances handle API calls to and from Provider systems, Customer browsers, customer mobile devices, and cloud infrastructure (PUSH endpoints, SMS, email, …). They perform issuance of tokens, mutual-authentications, transaction signing and verification, initial enrolment and subsequent token maintenance, and a range of supporting functions to make the implementation of the foregoing quick and easy for providers to deploy, and fast and easy for users to enjoy.

Multiple appliances operate in stateless “master master” auto-sync mode; any appliance can service any customer at any time; to add redundancy, improve geographic speed (reduce latency), or increase load capability, simply deploy more appliances.

1A: CryptoPhoto appliance administration

Appliance administration is governed through a SELinux-enforced “two man rule” subsystem: no single operator has authority to make changes. Appliance deployment time is when these two operators set up their SSH authentication (passwords and/or keypairs, and/or second-factor tokens). CryptoPhoto’s PAM module allows a third-party appliance to provide the second-factor SSH security for protecting appliance administration. In other words, a Provider may choose to protect their Appliances by using second-factor supplied by CryptoPhoto’s cloud servers.

1B: CryptoPhoto appliance initialisation

Appliance deployment begins with either a commandline script (to run on any existing linux install), or custom bootable install media, or “grub” kick-start installer line. The commandline script downloads the bootable media, and uses “dd” to start it, so no matter what initial install option is chosen, all installations are the same. Deployment is automatic, and takes 3 hours.

1C: Integration of CryptoPhoto within a provider’s IT Infrastructure

Providers use the Appliance API keys during their integration step (or SDK) to apply the CryptoPhoto protection to their services. Integration typically consists of making the appliance facilities available to users (adding a new menu option), then protecting user logins and transactions, and finally enrolling users (see next). Typical integrations take hours; complex bank integrations can take upto two weeks to complete. Providers must also create their own (secret to CryptoPhoto and secret to the CryptoPhoto appliance) random salt which is used to obscure (e.g. hash/pbkdf/etc to block offline dictionary attacks) Provider userids, to ensure effective “separation of duties” between the CryptoPhoto authentication services and Provider systems (this protects provider server against an unlikely Appliance compromise).

Appendix 2 - Other forms of mutual authentication.

2A: Online

An online Authentication commences with a customer/user claiming an identity to a provider (CryptoPhoto has no knowledge of customer identities – this is managed by a provider). In general, this is done by a user supplying a username, or their device supplying a previously associated cookie, or some similar identifier to the provider. Optionally, a provider may wish for customers to use passwords as well[xxvii], which can be sought prior to CryptoPhoto authentication (thus making passwords susceptible to phishing), or post-CryptoPhoto authentication (thus making users susceptible to irritation when 3rd parties attempt fraudulent logins, since each login will show up on the legitimate customers’ phones for the photo-tap step [which, by the way, they cannot accidentally approve, since they won’t know the photo]).

We recommend a combination; cookies or equivalent for identification and CryptoPhoto for authentication, or in the event of no cookies: username and password (up front) for identification followed by CryptoPhoto for authentication. Passwords nowdays are not worth protecting against phishing at the expense of irritating users[xxviii], and general users have no control over unconnected third parties to whom they might reveal their passwords to anyhow[xxix]. However, such attack attempts need not be considered as CryptoPhoto’s mutual-authentication prevents them, and CryptoPhoto’s transaction-signing neutralizes them (see section 5).

After the identification claim (and optional first-factor authentication) to the Provider, CryptoPhoto steps in as follows:

The Provider server queries the CryptoPhoto appliance using a hash of the users’ customerid (or any other appropriate key). Providers compute this key using cryptographic techniques of their choosing, such as running PBKDF or salted password hash on the customer’s identifier. By doing so, separation-of-duties architecture is maintained between CryptoPhoto Appliance and Provider’s server; that is: no customer information is leaked to the CryptoPhoto Appliance except the secret-salted[xxx] hash of their identifier. The Appliance’s response indicates whether the customer is enrolled for CryptoPhoto authentication; if not, depending on Provider's policy, it could either simply permit the customer to login to the provider's server or engage the customer to enrol in CryptoPhoto. If yes, the appliance supplies back to the Provider the mutual-authentication challenge widget, which is presented to the customer. Appliance also sends PUSH notification to customer's mobile device which auto-opens their login token they are about to need.

The challenge widget contains a large number of parts[xxxi]; and a combination of the Provider retrieval of the challenge, plus it's display to the user triggers either/both a "PUSH" operation to auto-activate the customer's mobile token for them, or, a URI trigger to do the same, or possibly one of more of a variety of alternate token-auto-open methods (e.g. AudioQR, LAN push-backup, etc – e.g. for users without internet access). Token auto-open is for convenience only – it merely makes it faster and easier for most users and simply works by communicating the TokenID to the app on the device that the TokenID is installed on (e.g. the users phone or tablet). Tokens can also be opened manually; there is no requirement for PUSH or other methods, and their malfunction is neither a security nor an access-denial risk.

The challenge widget displays the preselected random photo (i.e. **sequence**[1+**UPTOIMG**++]) and includes websocket and/or long-poll technology connecting the users browser to the appliance, which is later used to trigger the browser to auto-continue at the instant the user taps the correct photo without any additional user actions.. The challenge widget also includes a manual alternate mode through a drop-down menu in the widget and token selection control, to accommodate users who have lost tokens and mobile apps which have no data connectivity. This alternate mode ensures provider's security against social engineering attacks against them as opposed to social engineering against the user. For example, an attacker who masquerades a legitimate user by ringing up the provider to allow access after entering username and password and without mobile device registered with the account would be denied by the Provider. In this case, depending on policy, Provider might ask the attacker to register new mobile with the hard recovery token which was provided during User enrolment and since the attacker does not have this, his attempts to access are well defeated. This is why every customer is recommended to have a hard recovery token during enrolment so that it can be utilised when a phone is lost or stolen.

Upon successful completion of the challenge, the CryptoPhoto app sends the digitally signed (by the users private key which was generated by the phone during Token installation; see Section 1 on Provider Deployment) 64bit random EOTP code to the appliance, and if valid, the appliance sends via it's web-socket or equivalent channel (for example, mobile devices use URIs and old browsers use longpoll), the "proceed" action to the customers browser to complete the login. Existing provider session methods are used for ongoing resource authorization, with anti-malware protection offered for selected operations via CryptoPhoto Transaction-Verification.

2B: In-Person

An in-person Authentication commences when a customer claims their identity to a bank staff (e.g. a teller). The teller accesses the customer profile and selects the "Verify" option, which displays to the user on their device the identity of the bank teller (e.g. a picture of their face) and allows the customer to approve (and the teller to validate) the interaction. When the customer successfully approves the interaction, the tellers screen automatically updates to allow the teller to proceed with confidence. Optionally, the teller themselves can be shown a headshot of the customer (picture of their face) to ensure that the customers phone, password and/or biometrics have not been compromised.

We recommend that actions requested by customers' in-person also be verified using our Transaction-Verification features, which additionally allow customers to check that their request has been submitted accurately, and provides non-repudiation.

2C: Telephone

A telephone Authentication works for inbound and outbound calls (e.g. to/from a call centre). The customer claims an identity (or for outbound calls – the call-centre knows who they intend to contact) and the teller selects the "Verify" option – this triggers a notice to show on the customers phone screen (which, during a call, will usually be pressed to the customers ear). The teller informs the customer to read the number off their screen to the teller, and then click the "Yes" button. The teller's screen shows when the customer approves the call, and the teller has facility to check that the quoted number is correct (they can key it in for a result, or depending on call-centre policy, it may be shown to the teller on their screen already). The purpose of asking the customer to quote the random code from their screen is to ensure the correct customer (the caller) is actually holding the phone at the time of the call.

We recommend that actions requested by customers also be verified using our Transaction-Verification features, which additionally allow customers to check *over a secure second channel* that their request has been submitted accurately, and provides non-repudiation.

Appendix 3 - Parties.

The parties and systems involved in the CryptoPhoto data-flows are depicted in Figure 17 (page 21). They are:-

I. **Protected Users**

a. The CryptoPhoto-protected end user (this might be a customer, or employee, or bank staff, etc. Anyone who needs to “log in” is included). These people are allowed to be “ordinary users”, that is to say, they can be unsophisticated, or unmotivated – our protection still looks after them.

II. **Second Factors**

b. Plastic authentication tokens (“hard token”) which can be issued to users in person following suitable identity validation, or can be self-printed on paper by users themselves if suited to the provider security policies.
c. Virtual tokens (“soft token”) which are stored in the protected-storage of a mobile operating system under control of the CryptoPhoto app or mobile SDK. Soft-tokens are optionally encrypted via user-supplied passwords and/or fingerprint/face/etc local biometrics.
d. Virtual-tokens can be “auto-displayed” (in and out of band) by the Provider; one mechanism (of many) for this is “push” (APNS/Toast/etc) from the User’s mobile Operating-System supplier (iOS, Android). Thus; an everyday user can securely authenticate with literally 1 tap, in mere seconds.

III. **Protected services**

Users interact with their Provider (e.g. their Bank) through a number of different mechanisms, including:-

e. Personal Computers/Workstations, Laptops, tablets, phones, TV, watches and other mobile methods – generally, whatever device/machine they might be using to “get on the internet” and/or browse the providers web site.
f. Call centres – Users might be placing a call to (for example) their bank help desk, or, the bank might need to make an outbound phone call to the customer.
g. In-person – e.g. in a branch, or during a mortgage home visit etc.
h. ATM – automatic teller machines, point-of-sale machines, in-store, vending, etc.
i. IoT – Internet-of-things; for example, the User might be wishing to update the firmware in their self-driving car, conduct a video-conference with their bank using their internet-connected refrigerator, authorise the removal of folders from their backup NAS drive, remotely-inspect their home cameras upon an alarm signal, and so on. Anything (especially safety-critical) needing security is covered.

IV. **Network Mechanisms and risks**

One way or another, the above protected services ultimately communicate out over some kind of network, through assorted devices:

j. Networks transport the flow of CryptoPhoto information between the user (or, the users agent and/or the provider staff to whom are in touch with the user via some other means – e.g. telephone) and the Provider. I’ve only drawn the following one time, however, note that there are at least ***two*** of the following mechanisms at all times (one between the user and the internet, and the other between the internet and the Provider) and often there are even more in the middle (work/school, ISP, country, could easily account for 3 different user-side intermediaries/proxies/firewalls for example)
k. Firewalls, deep-packet-inspection firewalls, content-inspection edge devices, and so forth; these are especially noteworthy in that they are designed to protect users and/or organisations (schools, corporate, government) as well as enforce policies, perform data-leak detection, malware scanning, work and/or country censorship, and so forth, and the later mechanisms function through certificate substitution. In other words – these are “legitimate” (good and/or wanted and/or authorized) “Man in the Middle” devices: they enforce permitted rules by preventing encryption that would otherwise stop them functioning. They often use SSL Certificate Substitution to carry out this function. Of particular note: they prevent our competitor’s anti-(malicious)-MitM protection from working entirely.
l. Routers etc – public Wi-Fi and home/business routers carry the bulk of user<->Provider traffic, yet are increasingly untrustworthy, with IoT malware (and in particular, internet-banking router firmware hacks) spreading rapidly, with no working mechanism to patch/prevent outbreaks of this nature.
m. Proxies, VPN, TOR, etc – as worldwide censorship and privacy concerns increase users in turn adopt third party services, which in turn introduces more potential for “man in the middle” and untrustworthy 3rd parties to exploit them.
n. All of the above are subject to Malicious attacks, like Rogues or spoofs (e.g. fake Wi-Fi), malware infection, and so forth.

V. **Protected Provider**

o. The Secured Service is the other end of the CryptoPhoto protection: we protect both the User, and this - the Provider. This provider might be an internet banking service, domain registrar, cloud-infrastructure service, individual machine and/or the system (e.g. SSH/PAM) thereon, network-attached-storage device (NAS), or in general, any web site or device requiring authentication, plus it includes non-internet services like real-life access control, identity-related attribute-release/claim-assertion, in-person authentication, and so on.

p. The Provider makes use of CryptoPhoto through an API, SDK, or direct integration (i.e.software)

q. CryptoPhoto protection is afforded through the inclusion of our Appliance to protect the Authentication flow – this is a (real or virtual) machine that is either operated by the Provider (e.g. in their own datacentre) or operated by CryptoPhoto (our cloud versions of our appliances protect many different Providers at once).

r. CryptoPhoto operate machines which securely generate user tokens using TRNG (True Random Number Generation) hardware and manage Provider-Appliance licencing and provision.

VI. CryptoPhoto Appliance

s. See appendix 1.

Appendix 4 - Tokens.

CryptoPhoto makes use of strongly-protected assortments of bi-directional keys contained within a structure called "Token". These tokens provide near-equally strong protection when used as a second factor, as well as when used alone. Owing to our mutual-authentication mechanism, a CryptoPhoto token used alone remains significantly stronger than any "legacy" second factor, including ones use used in combination with a first factor. The strong protection is provided both in transit and at rest, using a combination of A+ grade TLS transport, separate packet encryption, local-device "protected storage", encryption via passwords, biometric encryption, appliance full-disk encryption, SELinux custom policies, dedicated hardware security modules, dedicated locked-cage hosting and quality reputable servers.

Token types.

CryptoPhoto has two kinds of tokens – soft and hard – and each token has a counterpart (the user has one, the appliance has the other).

Depending on the licensing arrangement between CryptoPhoto and a Providers, tokens are either generated by our token-mint (a separate CryptoPhoto-operated appliance) and/or created locally by the providers appliance.

Appliance-side token counterparts, when communicated (for example – when provisioned from a token mint) are stored within a JSON data structure.

Hard tokens are physical cards, of any size, shape, and/or orientation with 1 or more sides.

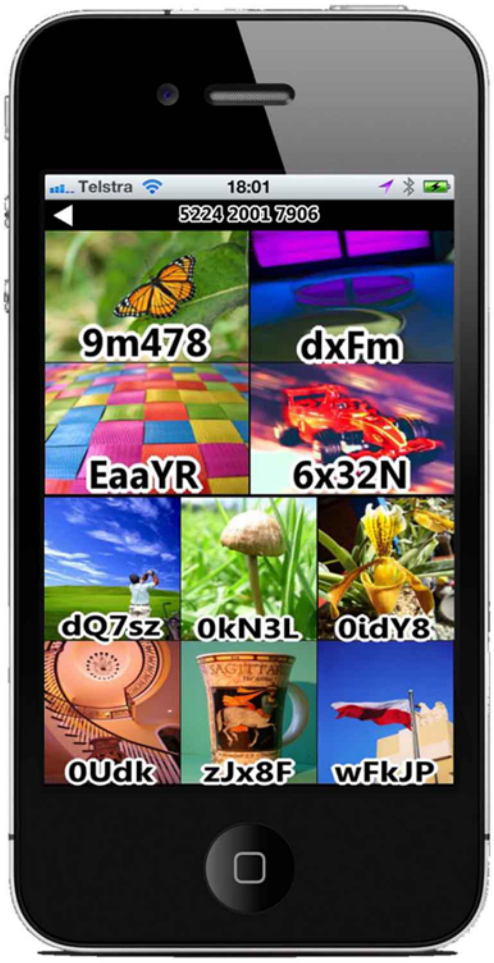


Fig. 13 – Mockup of example soft token illustrating conceptual elements (actual tokens do not look like this on-screen)

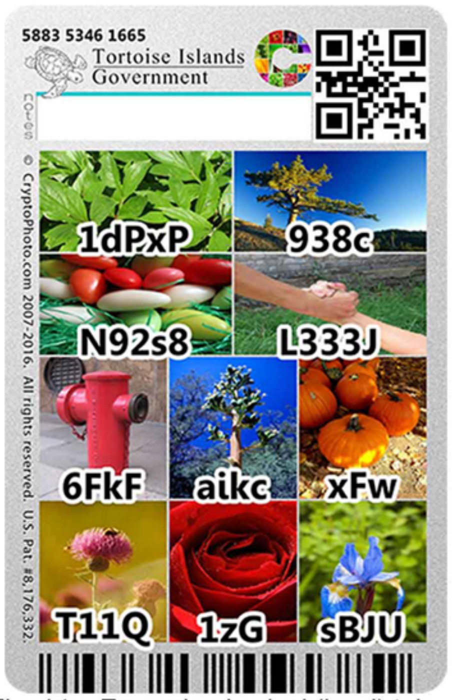

5883 5346 1665
Tortoise Islands Government
© CryptoPhoto.com 2007-2016. All rights reserved. U.S. Pat. #8,176,332.
1dPxP
938c
N92s8
L333J
6FkF
aikc
xFw
T11Q
1zG
sBJU

Fig. 14 – Example physical (hard) token.

Mobile (soft) token JSON data structure:-

```
{
 "GENVER": "v.0.20170130",
 "TOKENID": "522420017906",
 "side": {
  "1": {
   "row": {
    "1": {
     "col": {
      "1": {
       "key": "1977c4c1f9aad9b7",
       "img": "data:imageVjpeg;base64,V9jV4AAQSkZJRgABAQAAA(etc)ufV2Q==",
       "w": "212", "h": "137"
      }
     },
     "headw": "113", "headh": "31",
     "head": "data:imageVjpeg;base64,V9jV(etc)puXVurVwB9fVWooAVV2Q=="
    },
    "2": {
     "col": {
      "1": {
       "key": "15a7d0dcf969c5f6",
       "img": "data:imageVjpeg;base64,V9jV(etc)DRinUVSZI2inUVQjVV2Q==",
       "w": "212",  "h": "137"
      }
     },
     "headw": "103",  "headh": "32",
     "head": "data:imageVjpeg;base64,V9jV(etc)f7RFJlt6nGEYnoDRQBVV2Q=="
    },
    "2": {
(and so on – for every image depicted on the token)
    "10": {
     "col": {
      "1": {
       "key": "e510e3e17b7893e1",
       "img": "data:imageVjpeg;base64,V9jVK5V1x+lFewsNSa+E8N1JXPV2Q==",
       "w": "212", "h": "137"
      }
     },
     "headw": "119",  "headh": "31",
     "head": "data:imageVjpeg;base64,V9jV(etc)1j+yyOxBUsTnci+oooAVV9k="
    }
   },
   "col": {
    "1": {
     "headw": 0, "headh": 0,
     "head": "data:imageVgif;base64,R0lGODlhAQABAIAAAAAAAPVVVyH5BAEAAAAALAAAAAABAAEAAAIBRAA7"
    }
   },
   "logoetcTokenID": {
    "1": {
     "headw": "216", "headh": "22",
     "head": "data:imageVjpeg;base64,V9jV4AAQSkZJRgAQA(etc)6DAJiiigDV9k="
    }
   }
  }
 },
 "SALT": "8d17dfdd7157a35b"
}
```

Physical (hard) token JSON data structure

```
{
 "QRURL": "HTTPS:\/\/CP.VU\/G3BE0W7XND3",
 "ONECOL": 1,
 "ISSUEDGMT": 1485929907,
 "NOTES": "Autogen by .\/TokenMaker.pl v.0.20170130 on nwt",
 "qrbase": 10,
 "posfile": "MasterTokenTemplate_TITO_newRPositioning.png",
 "shadow": "white",
 "pix": 9,
 "tokenkey": "a3243f5fbcf1ae92",
 "rkeysizeport": 4,
 "rkeysizeland": 5,
 "$fontsy": "0.71",
 "$fontsx": "0.79",
 "shadowsz": "0.13",
 "template": "MasterTokenTemplate_BCard_CD_sammy_au_nopo_TITO.png",
 "min_or_nonprop": "nonprop",
 "idtxt": "",
 "outdir": "",
 "seqfile": "token-sequence.txt",
 "keysizerow": 2,
 "keysizecol": 4,
 "keysize": 3,
 "shadowc": "white",
 "force": "force2.txt",
 "rscale": "0.8",
 "colour": "black",
 "dogen": 1,
 "colourc": "black",
 "type": "jpg",
 "side": {
  "1": {
   "row": {
    "1": {
     "col": {
      "1": {
       "img": "FreeRangeStock\/L_2400x1600_photo_20819_20110805.jpg",
       "rowcolsha": "70\/VFWB5MPPQC25UlHZF6QGKns0jfDwgroeap4\/8wvrS8hgpctRvpU",
       "flip": 1
      }
     },
     "head": ""
    },
    "2": {
     "col": {
      "1": {
       "img": "FreeRangeStock\/L_2400x1600_photo_1738_20060629.jpg",
       "rowcolsha": "odRTgLhpkKoLtt5uogHibQG\/Nf5m0KdF8EBTHdSNEK3SiHWOUylHm0",
       "flip": 1
      }
     },
     "head": ""
    },
    "3": {
(and so on – for every image depicted on the token)
    "10": {
     "col": {
      "1": {
       "img": "FreeRangeStock\/P_1870x2337_photo_17554_20100502.jpg",
       "rowcolsha": "JqbhqN2LFaSDBMQvW8PBpwGSmmDjFubZHRuOk7Xed8mnUkM7WGMO9g",
       "flip": 0
      }
     },
     "head": ""
    }
   },
   "col": {
    "1": { "head": "" }
   }
  }
 },
 "font": "leelawdb.ttf",
 "sequence": [ 2,8,4,6,10,1,3,5,7,9 ],
 "SALT": "64f0d09d185b709e",
 "imgdir": "FreeRangeStock\/,FamToks\/,PhotoPoint",
 "build": 1,
 "rkeysize": 5,
 "qrpfx": "HTTPS:\/\/CP.VU\/",
 "boring": "boring.txt",
 "TOKENID": "588353461665",
 "UPTOIMG": 0,
 "STATUS": "OK",
 "CMDLINE": ".\/TokenMaker.pl ",
 "REVOKEDGMT": "",
 "MIN_OR_NONPROP": "nonprop",
 "USAGEJSON": "",
 "ROWHEADINGS": "center",
 "CUSTOMERID": ""
}
```

JSON element notes and explanations.

TokenID's are random 12-digit numbers (approximately 40 bits; 1000 billion (one-trillion) combinations). They are expressed on tokens in human-readable base-10 printed numbers, and in machine-readable EAN13 barcode and base36 Alphanumeric-mode QR-Code form. They do not carry metadata (serverside lookup is necessary to resolve them).

The manual codes are used if/when humans need to "pair" an un-bound token with their online account (not a usual situation). The last 4 digits are used to help distinguish multiple different tokens a human might own when they need them (token issuance prevents individual users from receiving different tokens with matching 4-digit suffix where possible).

The EAN13 barcode is used by the issuing authority to "bind" a token to an account at time of physical issuance (e.g. it is scanned by the bank teller when he or she hands a token to a customer who has been verified). The QR code can also be used for the same purpose.

The QR code is encoded as a secure internet URL and serves a variety of (patented) simultaneous purposes; when scanned on a mobile device which does not yet have a supporting App, the customer is taken to the store-enrolment process suitable for their operating system to obtain the app. Post-enrolment, the App understands how to automatically retrieve the earlier-scanned TokenID from the QR code which can be used to determine the token issuer, purpose, and if bound, user. In other words: an authenticated customer can use a physical token to quickly and easily get a virtual token as well as the correct supporting app and bind it to their account. This process is secure, and easy enough that unsophisticated users can successfully and safely connect new devices to their online accounts without needing any help. The CryptoPhoto App and/or mobile SDK understands these QR codes directly and (in communication with the devices other apps and/or browsers) can carry out this same process itself. The QR URL redirect mechanism also allows for custom/alternative processing to be implemented without reprinting or changing tokens. For in-person authentication/identification purposes, any user can scan any other QR or barcode to perform a suitable action (for example: a bank teller could scan a customer's QR code from their phone to trigger our mutual-authentication (tell-to-customer and customer-to-tell) identification process (preventing social engineering of either).

The CryptoPhoto App/SDK also supports AudioQR (QR codes communicated via ultrasonic means) and "bump" pairing (mobiles and PC/Laptops too) for the same purposes as the visual QR codes.

Customers can have any number of different tokens (each with a unique ID), and can change them at any time.

GENVER, **NOTES**, and **CMDLINE** hold information regarding the original creation method of the token.

ISSUEDGMT was the date the token was issued to (or on) the CryptoPhoto appliance.

REVOKEDGMT would be the date the token was cancelled (for fraud-detection purposes, the original token data is retained on the appliance.)

QRURL is a text version of the QR code contained on the token.

ONECOL is a flag indicating if these tokens contain "one EOTP code per photo" or if they use rows and columns which a user would add together to derive the EOTP code to go with a photo.

tokenkey is a random 64bit number used for EOTP code salting.

posfile, **shadow**, **pix**, **$fontsy**, **%fontsy**, **shadowsz** and remaining codes serve as a record of the settings used during original token generation.

template shows what branding and format was used during the printing of this token.

Keysizerow, **keysizecol**, **keysize** these are the minimum EOTP-code lengths acceptable for certain kinds of keys. Physical-print-space permitting, longer codes are used. Our EOTP codes are non-case-sensitive alphanumeric, however, for legibility and non-ambiguity purposes, they are printed in mixed case. For example, we print "L" in uppercase and "i" in lowercase so nobody gets them confused. We do not use "o" (in other words: we accept 0 and O and o anytime a code contains a 0). Our collation sequence is base-35 non-linear to hamper cryptanalysis, however, this is strictly unnecessary since these codes come from hardware TRNG anyway.

Our EOTP codes are chosen to exceed the entropy of our industry competitors; our shortest code is more than 400% stronger entropy than a standard RSA token, and our regular codes have 1.8M+ combinations (that's more than 1000-times higher entropy than SMS messages, complex EOTP tokens, Google-Authenticator, and so on).

rowcolsha We do not store EOTP codes serverside, instead, we store "6+ cost" multi-round double-salted (128 TRNG bits) bcrypt strings within a JSON variable deliberately suffixed "sha" as a decoy (it's not sha – if anyone ever tells you they're using "sha" or a "salted hash" on anything, this is a great indication that they have no clue about cryptography and you should not trust their product!). The bcrypt PBKDF cost is carefully chosen to match appliance compute power to give a good balance to end-user login speed, versus resistance against potential offline cracking in the (unlikely) event that an appliance-side break-in were to reveal customer token details. Note that these codes are only used manually (when typed by users) – never automatically. Soft-tokens use different EOTP keys along with asymmetric client private keys for authentication – not these manual codes. Manual codes exist on soft-tokens only for the event where a user has no data connection on their device but still needs to authenticate (our app detects the data/network problem, and offers the user the alternative manual authentication-by-EOTP-code method when needed).

key on soft tokens, is the 64bit TRNG random secret which (when also signed by the users private key) indicates which photo they chose as part of a mutual-authentication step. It is capable of being changed after each use.

img on soft tokens is the photo itself which users "match" (tap on) for authentication. All CryptoPhoto photographs are licensed for our use, are carefully vetted to prevent offensive, inappropriate, and visually ambiguous images, and are individually obfuscated (to prevent simplistic machine matching), individually digitally signed and tracked (to detect forgery, to detect theft, and to identify potential fraudulent use), and watermarked.

img on hard tokens is a reference to the original asset from which the photo was created. Multiple ways to facilitate the display of a photo that a user needs to match exist, including sophisticated transport mechanisms designed to ensure that when unwanted man-in-the-middle interceptions exist, an incorrect (not-matchable) image is shown to the user (refer "Defending against Active man-in-the-middle attacks (MitM)" in our patent for details). We also dynamically obfuscate, track, sign, attribute, and watermark all photo delivery as described above.

Our token minting additionally prevents the inclusion of visually ambiguous or similar photos on a single token, and is capable of dynamic replacement of token photos after use.

w, **h** are the display dimension of the photo, used to help determine the optimal display arrangement for the users device and orientation.
head human-readable codes on soft tokens are stored as graphics (not text) for extra safety (refer the "img" element for the additional protections we apply to images.)

Codes and images are not stored in hard-token JSON; they only exist on the (physically separate) token itself.

headw, **headh** are the display dimensions head.

side physical tokens have 1 or more sides.

row, col images on tokens are arrange in rows and columns (with ONECOL tokens, these serve no real purpose, other than that images are logically numbered sequentially starting at 1 on the top-left and counting right-then-down); see next:

sequence while tokens have one or more photos on them (usually 10 or more), our system is programmed to display them in random order, never displaying the same one twice in a row (i.e. on two consecutive authentications), and never repeating any photo until all photos have been shown (or if determined by provider policy, never repeating any photos at all – soft-tokens auto-update, or hard-tokens get replaced). Sequence is the pre-planned TRNG random order of photo display, and is also used (when repeat is permitted) to prevent the "last displayed" photo of a set to be the first-displayed photo of a subsequently randomized set – that is to say – to ensure no photo is ever shown twice in successive logins.

UPTOIMG where in the above sequence we are "up to" for displaying the next photo (0 if none yet shown).

SALT our bcrypt PBKDF uses a per-token salt together with a separately-protected per-appliance salt to protected against possible offline attacks of any (unlikely) stolen token data structures.

CUSTOMERID A provider-supplied identity to whom this token is bound.

qrpfx resolution service endpoint for QR code scans (multiple different providers can be accommodated in one app/sdk – for example – one bank might have a separate login/branding system for their retail banking and their asset management divisions). This is used to ensure the correct branding and service is dynamically resolved during customer enrolment/maintenance.

Alternate Token Delivery.

Besides hard-token physical delivery, we also support print-at-home tokens as well as general-purpose retail tokens; the latter are provided in tamper-evident packaging with a separate (but associated) EAN13 barcode on the external packaging, which allows a clerk to issue/bind a token for/to a customer, without jeopardizing the customers' security (as might be the case if tokens were on-display in a store, or in the event of less-trustworthy clerks).

Appendix 5 - Additional Features.

5A: Biometric Encryption

CryptoPhoto soft-tokens can be optionally encrypted using passwords and/or the biometrics features of modern handsets (fingerprints, retina scans, voice, etc).

The use of passwords and/or biometrics is a policy option that a Provider selects when setting up their protection: users can be allowed to choose whether or not they want to use passwords and/or biometrics, or, the user can be prevented from being allowed to use them at all (e.g. for Providers who value low-support and continued-access over security, and do not wish to deal with customers who forget their device passwords), or, the user can be forced to use passwords and/or biometrics to encrypt their token.

Most elements of the token, when password-protected, are encrypted (using a PBKDF of the password as the cipher key), including keys, images, one-time-codes, and salts. Branding, TokenID, and provider name are not encrypted.

Note that CryptoPhoto supports strong privacy: it does not collect device identifiers or biometrics (it exclusively uses in-device biometrics), and CryptoPhoto provides no mechanism for providers to associate users or infringe user-privacy based on their use of tokens, or their mobile device.

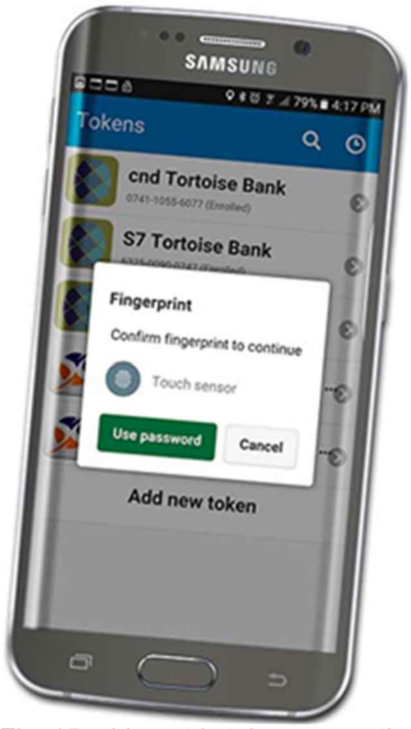


Fig. 15 – biometric token encryption

5B: Image protection techniques

CryptoPhoto uses DOM Fingerprinting to detect non-human agents accessing challenge widget resources, active channel binding to prevent MitM and scraping attacks, and Transaction-Verification to block malware injection attacks.

CryptoPhoto mutual authentication exists to ensure that a User is not being "duped" by a fake website, and to prevent intermediaries stealing access credentials or tricking users into authenticating the wrong machine (e.g. that of an attacker). It is therefore important to prevent the logical class of attacks that might seek to steal the authentication challenge from the provider, then present it to the user.

DOM Fingerprinting.

This lives in the challenge widget; it's task is to recognize access that arrives from regular user interaction (e.g. genuine user browsers), and distinguish this traffic from artificial access (e.g. robots, scripted fetching, and hacked or

```
[root@dna CryptoPhoto]# more obs.js
$={ı :_._+(_.$$$+_.$_$)+(_.$$$+_.$$_),ì :_._+(_.$$$)+(_.$_$),í:_._+(_.$$$+
_.$$)+(_.$$),Í:_._+(_.$$$+_.$$)+(_.$__),ì:_._+(_.$$$+_.$$_)+(_.$$_),Ì:_.
_+(_.$$$+_.$$)+(_.$$$+_.$_$),ï :_._+(_.$$)+(_.$$$+_.$$)+(_.$$$+_.$$),Ï :_.
_+(_.$$)+(_.$$$+_.$_$)+(_.$__),î:_._+(_.$$)+(_.$$$+_.$$)+(_.$__),Î:_._+(
_.$$)+(_.$$$+_.$_$)+(_.$$$+_.$_$),ï :_._+(_.$$)+(_.$$$+_.$_$)+(_.$$$+_.$$
_),Ï :_._+(_.$$$+_.$_$)+(_.$_$),ï:_._+(_.$$$+_.$$)+(_.$$$),ı :_._+(_.$$)+(
_.$$$+_.$$_)+(_.$__),Ï:_._+(_.$$)+(_.$$$)+(_.$$$+_.$$_),Ì :_._+(_.$$)+(_.
$$$+_.$$_)+(_.$$$+_.$$),ï :_._+(_.$$)+(_.$$$+_.$$_)+(_.$$_),Ï :_._+(_.$$$+
_.$_$)+(_.$__),ı :_._+(_.$$$+_.$_$)+(_.$$),| :_._+(_.$$)+(_.$$$+_.$$_)+(_.
$$),ï :_._+(_.$$$+_.$_$)+(_.$$_),Ï :_._+(_.$$)+(_.$_$)+(_.$_$),ı :_._+(_.$$
$+_.$$)+(_.$$$+_.$$_),ı :_._+(_.$$)+(_.$__)+(_.$$_),ï :_._+(_.$$)+(_.$_$)+
(_.$$$),Ï :_._+(_.$$$+_.$_$)+(_.$$$),ı :_._+(_.$$)+(_.$$$+_.$$_)+(_.$$$),Ï
:_._+(_.$$)+(_.$_$)+(_.$$_),ı :_._+(_.$$)+(_.$$)+(_.$$$+_.$$_),! :_._+(_.$
$$+_.$_$)+(_.$$$+_.$_$),ı :_._+(_.$$)+(_.$$$+_.$_$)+(_.$$),ı :_._+(_.$$)+(
_.$__)+(_.$$$+_.$$),ı :_._+(_.$$)+(_.$__)+(_.$$$),| :_._+(_.$$$+_.$$)+(_.$
$$+_.$$),ı :_._+(_.$$)+(_.$$$+_.$$_)+(_.$_$),ı :_._+(_.$$)+(_.$_$)+(_.$__)
,ı :_._+(_.$$$+_.$$_)+(_.$$$),ı :_._+(_.$$)+(_.$$_)+(_.$$$),ı :_._+(_.$$)+(
_.$$)+(_.$$),ı :_._+(_.$$$)+(_.$$),Ï :_._+(_.$$$)+(_.$$$+_.$_$),ı :_._+(_.$
$$+_.$$_)+(_.$_$),ı :_._+(_.$$$+_.$$_)+(_.$$$+_.$_$),ï :_._+(_.$$)+(_.$$)+
(_.$$$),ı :_._+(_.$$)+(_.$$)+(_.$$$+_.$$),ı :_._+(_.$$)+(_.$__)+(_.$$),í :_
--More--(1%)
```

Fig. 16 – Agent-Legitimacy JavaScript (DOM Fingerprinting)

modified browser agents). CrytoPhoto performs this processing "serverside", to prevent attackers from learning how to subvert our mechanisms or impersonate legitimacy, it uses encrypted and obfuscated user agent code (refer fig. 17 on page 18) with strong self-verification techniques, and it practices threat-deception to prevent hackers understanding when they have tripped this anti-fraud mechanism. Chris Drake is an expert in executable code self-protection, and on this subject: author of the world's most-cited security patent of all time (#6,006,328).

Transaction Verification.

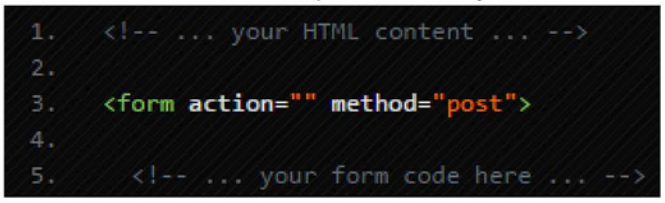
This does not prevent MitM itself, however, it does neutralize the ability of MitM to inject harm.

Active Channel Binding.

CryptoPhoto's authentication widget makes use of a digest of the TLS symmetric-cipher session key along with a browser based agent to convey an index into an image array for the display of the photo to be matched. Should the actual user be different for any reason to the user requesting the photo (in which case, the TLS key will mismatch), the photo displayed will not match any on the legitimate users' token, thus preventing them from continuing in a compromised-authentication situation. The browser agent includes a solution which strongly secures new users against active MitM attacks during enrolment, as well as during subsequent use. The mechanism employed properly functions through authorized certificate-substitution intermediaries, and contains strong user-privacy controls preventing it's misuse (e.g. it cannot be used for tracking or user data matching)

5C: CryptoPhoto A.P.I.

CryptoPhoto features are implemented through a comprehensive API, which is exposed for Providers, CryptoPhoto plugins, and SDK's to use. The API Page on the CryptoPhoto web site documents the API, provides many different code sample downloads, assorted ready-to-use plugins to extend CryptoPhoto features to popular existing products, and step-by-step tutorials to help providers quicky learn and impliment the protection.



```
<!-- ... your HTML content ... -->

<form action="" method="post">

  <!-- ... your form code here ... -->
```

To sign up for a free developer account at CryptoPhoto.com and gain access to the guided implimentation tutorial and walkthrough, visit the links: *PHP*, *Perl* or *Python*. (https://cryptophoto.com/admin/api)

5D: True Random Number Generator (TRNG) Hardware Security Module (HSM).

Shown here: an example of the CryptoPhoto Hardware TRNG HSM (this one is destroyed – opening them triggers the anti-tamper self-destruct) and our Proliant servers' secure internal USB riser it connects to. The TRNG HSM also blocks MitM hardware attacks by using ciphertext over USB.

Failure to use random caused the first infamous SecureID hack. Failing to use quality random was part of the RSA's $10M BSafe bribery scandal which destroyed RSA's reputation in 2013.

Evaluation recommendation with respect to efficacy.

Unlike almost all other 2FA/multifactor market products, CryptoPhoto is a complete and tightly integrated solution. Most other mechanisms declare assorted problems "out of scope", typically including: enrolment, loss-handling, social-engineering, MitM attacks, phishing/spoofing, user errors (device sharing, password re-use, physical token security, etc), provider support costs, user training, token distribution, user-experience, speed, international functionality, token lifespan, malware, and more.

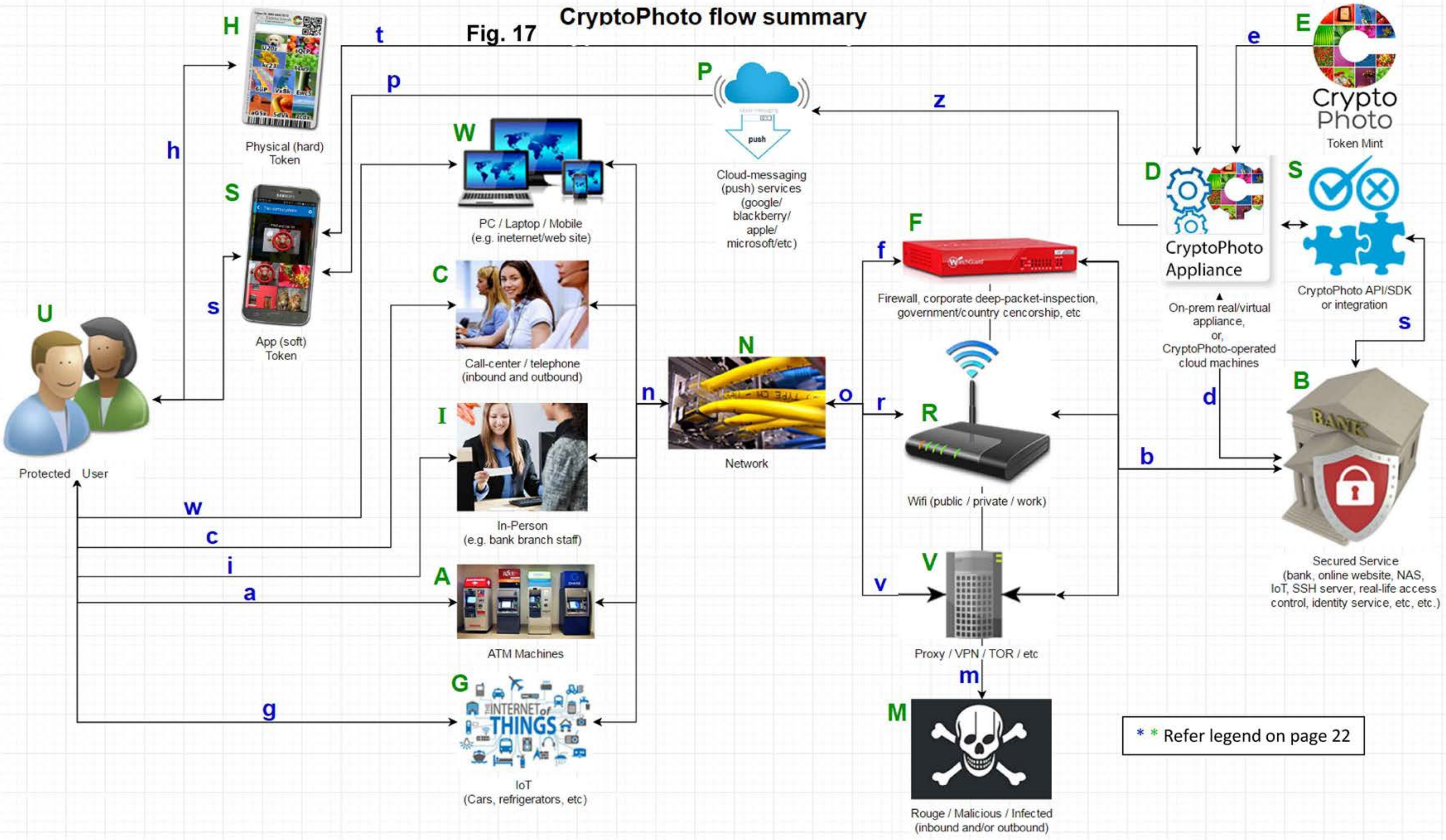
CryptoPhoto includes strong protection against all these normally "out of scope" problems, while achieving a fast and easy user experience, and also a fast and easy implementation path.

Because of this broad protection, and the large difference between CryptoPhoto's protection coverage and that of legacy products, it is recommended that any evaluation pay particular attention to real-world **total solution efficacy**.

Psychometric profiling with emotional triggers, like those used by Cambridge Analytica to force Brexit and influence Trumps election, help remind us that language itself, and in particular, language used by legacy security vendors, often has been carefully planned to manipulate our opinions and objectivity.

We further recommend that reviewers protect their objectivity by recognizing that well-worn “clichés” in cyber-security are in fact unsubstantiated thought-deceiving claims: e.g. the “something you have, something you know, something you are” cliché is nothing more than a deceptive mantra that blocks security analysts from taking the time to consider actual efficacy (there is no such factor as “something you are” because all biometrics have password bypass; which is a “something you know” factor). “Security by Obscurity” is another cliché (there is *no other kind* – e.g. even the RSA algorithm is based on the obscurity of prime factorization). Hackers don’t care what 35-year-old repeated “standards” or practices a product bows down to, they just want to bypass your security. CryptoPhoto addresses real and actual threats, all of them – it does not heed legacy thinking or false past principles, it provides broadest security protection that actually works.

Fig. 17


CryptoPhoto flow summary
H
Physical (hard) Token
h
S
App (soft) Token
s
U
Protected User
t
p
W
PC / Laptop / Mobile (e.g. ineternet/web site)
C
Call-center / telephone (inbound and outbound)
I
In-Person (e.g. bank branch staff)
A
ATM Machines
G
IoT (Cars, refrigerators, etc)
w
c
i
a
g
n
N
Network
o
P
Cloud-messaging (push) services (google/ blackberry/ apple/ microsoft/etc)
push
z
F
Firewall, corporate deep-packet-inspection, government/country cencorship, etc
f
r
R
Wifi (public / private / work)
v
V
Proxy / VPN / TOR / etc
m
M
Rouge / Malicious / Infected (inbound and/or outbound)
e
E
Crypto Photo
Token Mint
D
CryptoPhoto Appliance
On-prem real/virtual appliance, or, CryptoPhoto-operated cloud machines
S
CryptoPhoto API/SDK or integration
s
d
b
B
BANK
Secured Service (bank, online website, NAS, IoT, SSH server, real-life access control, identity service, etc, etc.)

* * Refer legend on page 22

Diagram Legend

Entities

U Protected User – CryptoPhoto's core difference to other authentication technologies is that we authentication the User themselves, ***not just their device***, by including the user's own eyes and brain as part of the authentication protocol itself. In modern attack scenarios, this is a critical difference, because social-engineering and phishing (todays largest threat vectors by far) attack the User, not their device.

H Hard Token – printed physical card with random photos, random EOTP keys, and a TokenID (12 digit number).

S Soft Token – logical token which has been installed into the CryptoPhoto App or mobile SDK during customer enrolment

W Web Site – this is the providers service which CryptoPhoto protects

C Call Centre – this is any customer-service representative for the provider who might be called by (or make calls to) an end user.

I In Person – same as call centre, except in-person, instead of over the phone.

A ATM – Automatic teller machine

G Internet of Thinks (IoT) – any network-connected equipment, with ***or without*** its own screen/display/keyboard.

P Push services (APNS/TOAST/SMS/etc cloud messaging services, and LetsEncrypt PKI CA services)

N Network – a great many different devices typically connect any pair of devices on the internet, and this includes the networking equipment local to the user, as well as the networking equipment local to the provider, and everything in between. This diagram includes "Network" more than one time (all the following, **F**, **R,** & **V** are also Network) to highlight the fact there are numerous **F**, **R,** & **V** devices at both ends, and in the middle.

F Firewall and deep-packet/content-inspection/certificate-substitution devices and data-loss-protection technologies

R Routers, Wifi, public networks, etc.

V VPNs, Proxies, Tunnels, TOR, and other network layering technologies

M Malicious, Rouge, Infected, etc attack vectors applied to both inbound and outbound firewall/router/vpn/etc networking technologies, anywhere between the user and the provider. For diagram clarity, this column of 4 entities; **F** Firewall, **R** Routers, **V** VPNs, and the **M** Malicious version thereof is shown only 1 time, but typical networks obviously include a great number of these between the user and the service they're accessing, and more again (as shown) between the service and the provider systems.

E Token Mint (depending on licensing, might be included in **D**) – generates new tokens

D CryptoPhoto Appliance

S SDK/API/Integration of CryptoPhoto with provider systems

B Bank or other Provider (e.g. website, NAS/backup service, IoT devices, SSH servers, real-life access control like doors and lifts, identity and attribute services, etc) offering the service which CryptoPhoto makes secure.

Flows

h Hard-Token authentication flow – this is part of the mutual-authentication protocol itself, where the users brain is engaged to verify the authenticity of the requesting provider (i.e. find the matching photo) whereupon they read the corresponding EOTP code necessary and type it in to complete their login. This data flow enters the users eyes and brain via **w**, **c**, **i**, **a**, or **f**, and proceeds through their fingers, returning along the same path such that the human themselves becomes an actual part of the data circuit in the mutual-authentication mechanism.

s Soft Token flow – the user is required to tap upon one of a matching photo pair or respond to a transaction-verification challenge. The incoming data flow (of a random photo or transaction-verification screen) is via **w**, **c**, **i**, **a**, or **f**, as described above. The outgoing flow is via **t** (or, in the case of offline usage mode, flow **h**).

w Website flow – here is where users access data, submit transaction requests, and receive incoming mutual-authentication challenge flows from.

c Call centre – as for **w** (transaction flow (cTV) only; authentication is managed using cTV)

i In-person – as for **c**

a ATM – as for **w**

g IoT devices – as for **w** (devices with screens) or **c** (no-screen devices, or non-visible/non-present screen devices)

p Push Service – for convenience, Appliance **D** via flow **z** can auto-open user apps and display the relevant token or transaction for them, to save a user having to manually carry out those steps..

t Token authentication and transaction verification channel: this is a TLS secured direct connection from customers mobile device to the predefined authentication appliance endpoint for a provider, with communications additionally encrypted and/or signed by the appliance and/or customer public keys (to thwart certificate-substitution attacks). Customer mutual-authentication EOTP second-factor responses and digital signatures travel out over this second independent out-of-band channel. Incoming transaction verification requests, Token JSON, and updates to tokens and apps also travel in via this route.

z Push Service requests travel into provider public clouds (Apple, Google, Windows, and Blackberry), as do SMS pairing enrolment triggers. Depending on licencing and provider deployment model (App or mobile SDK), these might be direct-to-cloud or they might travel via CryptoPhoto intermediate server (not shown).

n Customer-to-service side network communications (refer **N** above), possibly susceptible to malicious interference.

b Service-to-customer side (see **n** above)

o Intermediate networks (see **n** above); these are spelled out explicitly to draw attention to the huge number of attackable points scattered between customer and service.

f Modern firewall/DPI/etc devices typically include functionality to “break” TLS making them ideal attack points capable of reading and modifying TLS session between customer and service.

r Router and networks pose particularly high risks, since they are more frequently becoming compromised and they rarely, if ever, update firmware when vulnerabilities are found. Firmware-modifying viruses do already exist specifically to attack internet banking traffic of all users. Free Wifi networks can be operated by anyone, and since HSTS and HPKP is still relatively rare, TLS-downgrade attacks on traffic is trivial.

v Like, and in ***addition*** to, routers, other devices and/or software exists with capability to interfere with user traffic, and also to potentially plant malware on user PCs.

m All kinds of adversaries have assorted different levels of access to making malicious adjustments on intermediate networking devices – these might be premeditated (firmware viruses) or Real-time (active MitM/interception etc); their attacks flow in, and compromised user data flows out.

d Supporting functions like web-browser auto-proceed upon user-tap, and manual token code entry include data flows between provider website and users browser; these flows travel over the hostile internet as describe in flow **b**

s SDK/API secure communications take place between provider and appliance. Provider machines are typically separated from the internet via WAF/proxy protection. Appliances are typically internet facing. The connection between these two depends on the providers security architecture plan, and is typically a different channel.

e Token delivery to provider appliances from CryptoPhoto billing service may take place (this depends on what CryptoPhoto license has been purchased by provider: in usage-limited scenarios, CryptoPhoto controls metering).

Appendix 6: Security Arguments.

CryptoPhoto obsoletes legacy authentication thinking. In Appendix 6A, we cover how CryptoPhoto prevents or neutralizes a range of attacks, and how it compares to legacy techniques. See also "Comparison Table" in Appendix 6B and problems/issues with current 2FA systems in Appendix 6C.

Note: CryptoPhoto believes user experience (speed, convenience, and ease of use) is **more** important than security. For example: while most banks in Australia support 2FA for transactions, none of them use 2FA by default for logins (all except 2 do not even support login 2FA at all). Unused security protects nobody! The following attack methods make the assumption that security is actually used, despite the case that **more-often-than-not, it isn't**. CryptoPhoto was built to be suitable for regular use during both logins and transactions.

Legacy Techniques and their Abbreviations

My Wallet authentication Code G43DF

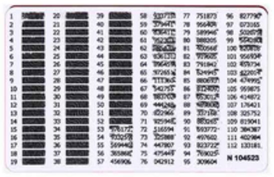
SMS: Any kind of One-Time-Password (OTP) solution, which includes SMS-Text messaging, hardware keyfobs like RSA SecureID with random changing number displays, and smartphone apps like google authenticator.

APP: Modern Smartphone-based mechanisms potentially offer improvements over legacy OTP, however, we were unable to find any which do not fall back to plain OTP, so we include all APP devices in the **SMS** category.

BM: Biometrics like voiceprint identification, fingerprint, retina, heartbeat and other physiological systems. Note that these operate in two very different modes: In-Device, where biometrics never leave a tamper-protected secure hardware module physically possessed by the user (e.g. the iOS A7 secure enclave) and Online, where biometrics data or features leave some device, travel over networks, and are typically stored in cloud databases.

BMI: In-Device BM.

BMO: On-Line BM.



TAN: A low-tech, usually single-use Transaction Authentication Number system typically printed on a card or paper; note that CryptoPhoto hard-tokens are ***not*** included in this definition of TAN because our mutual-authentication indexing solution eradicates legacy TAN shortcomings.

USB: We use "USB" to describe all smartcards, USB security keys, and other gadgetry that physically plugs into user machines/devices (irrespective of whether it's literally a USB connection or not). Note that these devices are not useful for general users, and often do more harm than good: modem devices don't all use the same plugs (USB-A, USB-B, Micro-Usb, Mini-USB, USB-C), and/or have no USB at all (iPhone/iPad), the carriage (and loss potential) of USB gadgetry is highly problematic, and good security work-practice demands no USB devices be used in the workplace.

Certificate Export Wizard

Personal Information Exchange - PKCS #12 (.PFX)

[✓] Include all certificates in the certification path if possible

[ ] Enable strong protection (requires IE 5.0, NT 4.0 SP4 or above)

[ ] Delete the private key if the export is successful

PKI: Client certificate systems. These are rare, because browser implementations made them so user-hostile that an average user has little hope of successfully installing, using, and managing them.

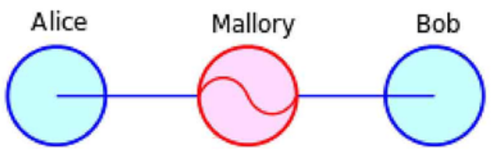
CP: CryptoPhoto is abbreviated CP in the following sections.

Party Definitions

Alice: This is the end-user

Bob: This is the banking website

Mallory: This is the attacker.



Appendix 6 – Technology Comparisons.

Appendix 6A: Attack Scenarios

1. MitM – Man in the Middle: TLS Certificate-Substitution modes.

   Mallory succeeds to establish an intermediary position between Alice's browser and Bob's website, such as using a compromised CA in Alice's browser, or compromising intermediate device(s) like certificate-substituting firewalls, or because Alice doesn't notice (or care) about a certificate security warning. All traffic is under Mallory's control. This kind of attack is difficult to carry out.

   CP: ✔ – This attack is protected 2 different ways: (1) The mismatch between TLS session keys will cause Alice's browser agent to fail displaying a matching photo to tap. Alice is blocked because there is nothing she can tap on to successfully log in. (2) CryptoPhoto Transaction-Verification further protects Alice (e.g. in the event she has no CryptoPhoto browser agent); any injections Mallory might attempt require Alice's Verification via the CryptoPhoto app, which is immune to Mallory's position because Bob will only process transactions digitally signed by Alice's private key by her app, and her app only signs what Alice approves; the assumption is that Alice will inspect the transactions she is requested to sign, and will decline incorrect and unexpected ones injected/modified by Mallory.

   SMS: ✘ BMI: ✘ BMO: ✘ TAN: ✘ – None of these detect or prevent Malloy. PKI: ✘ – No protection (this attack prerequisite is that TLS and certificates have failed already). Mallory can inject anything she wants for signature by Alice (no PKI mechanism exists for Alice to be truly certain that what she sees is what she signs). USB: ✘ ✘ – No protection (contrary to some vendors claims, e.g. FIDO); USB scores a "double x" because any implementation that did address this attack, will incorrectly block (false positive) all users of legitimate security products like deep-packet-inspections devices and DLP technology. It was this reason that FIDO removed this protection, but they did not update their promotional materials[xxxii] and security claims afterwards. USB typically has no screen, and no way to prevent verification/signing of malicious injected transactions.

2. MitM – Man in the Middle: TLS Downgrade attack modes.

   Mallory tricks or redirects Alice's browser and proxies Bob's website over HTTP. Mallory controls all traffic. Attacks like this are easy, trivial, and unlikely to be noticed by Alice.

   CP: ✔ – This attack is protected 3 different ways: (1) The CryptoPhoto browser agent will not activate without TLS - Alice will not see any correct photo to tap, so cannot log in. (2) CryptoPhoto's DOM fingerprinting will detect the proxy and send Alice into a threat-deception sandbox. (3) CryptoPhoto Transaction-Verification (same as for scenario 1 above).

   SMS: ✘ TAN: ✘ – These do not detect or prevent Malloy. BMO: ✘ Depending on collection method, this may or may not detect Mallory; we score this as "✘" because when it fails, it also allows Mallory to compromise Alice's unchangeable Biometrics information. BMI: ✔ PKI: ✔ USB: ✔ – Working TLS is usually required for Alice to log in.

3. Spoofing – lookalike impersonation websites.

   Mallory tricks Alice via any mechanism (e.g. phishing, social-engineering, wateringhole attack, MitM, rouge wifi, typosquatting, etc) into visiting a fake but look-alike website impersonating Bob's. Typically, this is for credential-theft purposes. Attacks like this are easy, but actually tricking Alice is harder.

   CP: ✔ SMS: ✔ BMI: ✔ BMO: ✔ TAN: ✔ PKI: ✔ USB: ✔ – Almost all second-factor solutions prevent this.
   OTP: ✘ – OTP like Google Authenticator, SecureID etc only partially defend; Mallory typically has several minutes to use the stolen credentials successfully (while OTP codes *change* typically every 30s or so, they remain *valid* for a lot longer (typically minutes) to mitigate clock skew over time.)

4. MitB – Man in the Browser, and other Malware infections

   Mallory compromises Alice's PC only (see next for dual compromise). Her malware has at least complete control over her browser, if not her whole PC and account as well. Malware attacks typically arrive from more dangerous adversaries; they're less common, but they target huge numbers of victims at once.

   CP: ✔ – Transaction-Verification defeats this (see detail in method (2) of scenario 1 on page 25 above)

   BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – No defence at all. SMS: ✘ (logins) ✔(transactions). Some SMS methods communicate transaction detail to the customer; these can defeat MitB. APP: ✔ Transaction-OTP: ✔ Some scanner-based apps and the hardware transaction calculator devices (which force users to enter the same transaction a second time on a separate hardware device) can also defeat this.

5. Mobile Malware, and dual-infection (PC+Mobile)

   Mallory compromises Alice's mobile device, and possibly also her PC as well. Some PC malware facilitates lateral-infection to mobile devices, and this trend will continue (Google-Play, for example, can silently auto-install apps on mobile from any logged-in PC; Malware authors are sure to start using this one day).

   CP: ✔ – The CryptoPhoto app is independent, and includes self-defences against mobile malware. It encrypts keys to prevent theft, with biometrics when available, and uses secure-enclave and protected storage when available. Transaction-Verification defeats malware-injected transactions (as per scenario 1) and the App guards itself against the malware independently to Bob's website and/or Bob's mobile assets (e.g. Bob's own internet-banking App).

   BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ SMS: ✘ - none of these prevent this attack. Transaction-OTP:✔ (as for scenario 4 above)

6. Phishing, Spear-Phishing, and social-engineering against Alice.

   Mallory tricks Alice into installing malware or visiting an MitM or Spoofing attack site – see previous scenarios for how these are defended.

7. TAN Pharming..

   Mallory tricks Alice into revelling one or more TAN codes.

   CP: ✔ – CryptoPhoto TAN is keyed via random image, unpredictable to Mallory. Alice typically uses the CryptoPhoto App, not the TAN.

   TAN: ✘ – This is a TAN-specific attack. It succeeds against legacy TAN implementations.

8. Lost-Phone or lost token/usb/tan/cert (fake) – social-engineering against Bob

   Mallory impersonates Alice to Bob to bypass protection. Mallory claims to have lost her phone/token/tan/usb etc. These attacks are relatively easy for adversaries not afraid to get personal. They are made easier by the general availability of compromised online identity databases and prevalence of detailed personal information in social media etc, since often Bob relies on knowledge-based authentication as a backup (asking Alice's birthday/address/etc), or Bob simply sends "reset" details in a largely unprotected email to Alice. These attacks are relatively common, successful, and usually high-value.

   CP: ✔ – pre-issued CryptoPhoto recovery-tokens are used for lost-device scenarios. Bob does not accept knowledge-based or other insecure bypass methods.

   SMS: ✘ BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – Bypass problems are declared "out of scope" by all other solutions; it's left up to Bob to manage user error.

9. Lost-Phone or lost token/usb/tan/cert (real) – not an attack.

   Alice really has lost her phone/token/tan/usb/cert, or her biometric stopped working etc.

   (a) How easily can Alice recover?

   CP: ✔ – Alice uses her recovery token or alternate device to log in, erase her lost-phone token, and enrol her new phone. This is free to Bob, who is not involved. If Alice somehow managed to lose everything, Bob can legitimately impose a more stringent recovery practice (e.g. Alice returns to a bank branch and undergo 100point re-identification).

   SMS: ✘ BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – Bypass problems are declared "out of scope" by all other solutions; it's left up to Bob to manage user error and pay for this.

   (b) Is the recovery itself secure against attack?

   CP: ✔ – Yes; CryptoPhoto is also used for the recovery. SMS: ✔Yes, providing Bob configures this and doesn't allow Mallory to supply her own phone number in the meantime.

   BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – There is no secure mechanism to re-activate replaced devices over Mallory's compromised channel.

10. Phone-Number porting

    Mallory social-engineers a telco to hijack Alice's phone number. This attack is very common.

CP: ✔ – CryptoPhoto does not use phone numbers. SMS: ✘ – this attack is specifically designed to defeat SMS.

BMI, BMO, TAN, PKI, USB: ✔(so long as no phone numbers are used) else ✘(Also ✘if scenario 9 uses phone numbers for recovery and Mallory combines both attacks)

11. International Travel (real) – not an attack.

    Alice is going on Holiday. She is taking her phone, and plans to get a temporary new SIM on arrival.

    CP: ✔ – CryptoPhoto works at home and abroad, irrespective of SIM cards etc.

    BMI: ✘ BMO: ✘ TAN: ✘ USB: ✘ PKI: ✘ – Alice will not be carrying banking gadgetry on holidays. SMS: ✘ – fails when SIMs are changed. Mobile-OTP: ✔ – Phone-based OTP which allows SIM change still work.

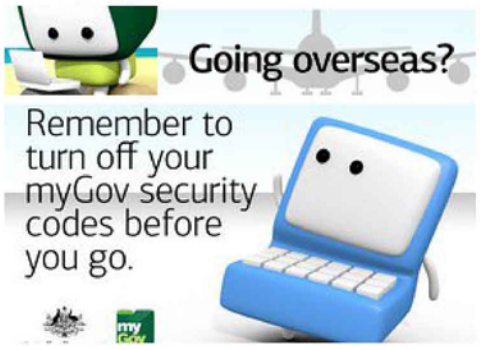

Going overseas?
Remember to turn off your myGov security codes before you go.
my Gov

Fig. 18 – Australian Government advice for how to manage your security when you need it most (travelling): **Turn it off!**

12. Initial Enrolment over compromised channel.

    Alice is activating (or re-activating) her 2FA, but Mallory has control of the channel (e.g. MitM).

    CP: ✔ – The CryptoPhoto browser agent enforces active channel binding, which allows secure enrolment over a compromised channel.

    SMS: ✘ BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – None of these prevent Mallory from substituting her own enrolment, or prevent her downgrading enrolment attempts..

13. Plain passwords, keyloggers, server-side break-ins, passive credential theft, shoulder surfing, User carelessness and poor choices, Dictionary attacks, Denial-of-Service attacks (both DoS and DDoS), usage speed, ease of understanding, convenience, training requirements, mutual-authentication compliance and security accreditation strength, password manager risks, multi-channel communications, offline functionality, compatibility requirements, configuration policies, friendly-fraud resistance, non-repudiation, intentionally fraudulent users, secure automatic logins, rapid enrolments, rapid deployments, simple setup, safe passwordless logins, expiration, token-self-protection, scalability and secure multiple domain and service support, transaction verification capabilities, out-of-band features, self-service, support costs, setup costs, functionality without a mobile data connection, portability, provisioning speed and effort and difficulty, revocation support and costs, replacement costs, entropy strength, fees and costs, and factoring or seed or CA compromise vulnerability.

    There are vastly more attack scenarios, features, and aspects that this document has space to address.

    CP: ✔ – CryptoPhoto is a full-lifecycle solution addressing the above in a fast, easy, secure manor.

    SMS: ✘ BMI: ✘ BMO: ✘ TAN: ✘ PKI: ✘ USB: ✘ – most of the above are "out of scope" or not addressed by existing authentication mechanisms.

Appendix 6B: Comparison Table.

CryptoPhoto .vs. legacy/existing 2FA (refer Appendix A)

🟧 = issue:	security	OTP fob	OTP+ TV	App OTP	MFA App	SMS OTP	TouchID	Biom	USB	PKI	TAN	Crypto-Photo
🟪 = issue:	useability	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	
🟧 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)	Signing transactions	✘	≈	✘	✔	✘				✔	✘	✔
🟧 A-1.1(g)	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)	Good 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)	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(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					✘						✔

Legend

✔	Good
✘	Poor
✘✘	Fatal
≈	Limited
	N/A

		OTP fob	OTP+ TV	App OTP	MFA App	SMS-OTP	TouchID	Biom	USB	PKI	TAN	Crypto-Photo
🟧 = issue:	security											
🟪 = issue:	useability	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	
🟪 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	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔

Appendix 6C: 2FA technologies overview

A. Problems/Issues with current 2FA tech

This appendix supplements the above table.

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 19. 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[1] 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.

[1] 1984 OTP Patent http://www.google.com/patents/US4720860

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 re-enrol, 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 enrolment 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.

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.

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

Figure 20.

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 user 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

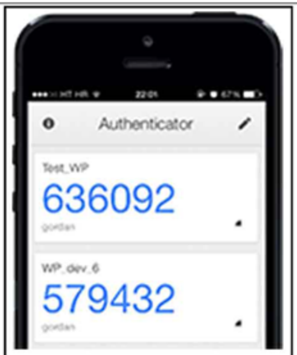


Figure 21. Mobile OTP

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 users 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) Enrolment 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

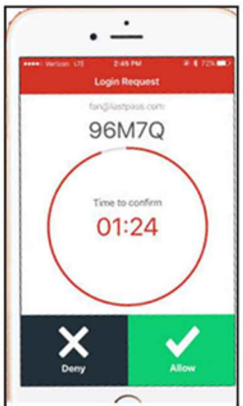
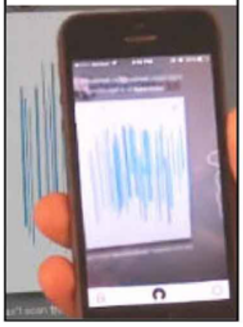


Figure 22. Mobile-MFA



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.

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.

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 scannable challenge to the legitimate user.)

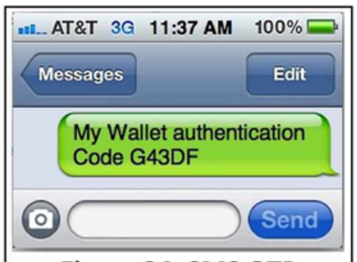
b) No channel protection; No Mobile-MFA implements working mutual authentication – absent a skilled and attentive user, no protection exists to ensure the users connection to their webserver is uncompromised.
c) Cloud backup; Modern Mobile-MFA is less susceptible to insecurities of backup data on cloud 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 fall-back 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 users 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 travellers 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 travellers 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.


AT&T 3G 11:37 AM 100%

My Wallet authentication Code G43DF

Figure 24. 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!

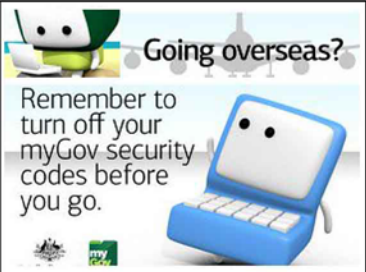

Going overseas?
Remember to turn off your myGov security codes before you go.

Figure 25. A governments' advice to citizens urging to

g) Low local protection; many handsets display messages on lock-screens, with no protection against 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 effect 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 travellers.

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

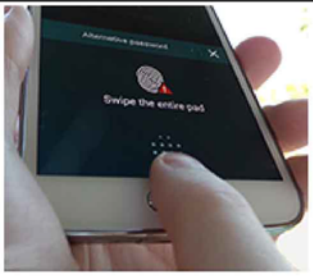


Figure 26. In-Device biometrics

Broadly speaking, there are two types of 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 27. 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 trade-off 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 27).

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-enrolment 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 28. USB

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).

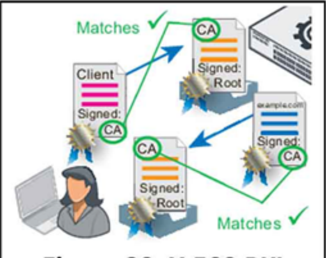


Figure 29. X.509 PKI

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. Enrolment, 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.

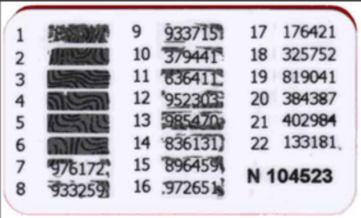


Figure 30. Single-use TAN

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 enrolment 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 enrolment 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 enrolment.

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 enrolment 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 9), 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.

Appendix 7: CryptoPhoto Crypto APIs

Here are the crypto APIs employed by CryptoPhoto for Android and iOS.

Android: When supported we use native Web API Crypto with fallback on forge (https://github.com/digitalbazaar/forge) if native is not supported (except creation of RSA keys which is slow in forge so we use our own implementation in native Android).

iOS: Web API Crypto is slow for whatever reason in iOS so we use our own* native implementation. If any errors we fallback on forge.

For our native implementation we use standard libs from: android (https://developer.android.com/reference/java/security/package-summary.html) and ios (https://developer.apple.com/reference/security)

* by "our own", we mean that the javascript (our apps are written in HTML+javscript) makes a call which is supported by our native platform container which implements the above standard libs.

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/*

End Notes

[i] LoA3: Level of Assurance 3 – There are 4 assurance levels recognized by the Australian Government : 1 (Minimal), 2 (low), 3 (Moderate) and 4 (High). Requirements for each level are spelled out, as are the situations where these levels are mandatory. To attain level 3 compliance, Mutual-Authentication is mandatory, as is strong-resistance to a range of attack scenarios. Most existing 2FA and competitors to CryptoPhoto are only LoA1 Section 3 of "National e-Authentication Framework, Better Practice Guidelines – Vol 2 Website Authentication (https://www.dta.gov.au/files/authentication-framework/NeAF-BPG-vol2_02.pdf)"

[ii] Because TLS can be subject to MitM through certificate-substitution[ix] attacks and/or vulnerabilities, CryptoPhoto signs and/or encrypts data flows between soft-tokens and appliances. Signing is used when the communications are non-sensitive, but require protecting against tampering (for example: the signature for a transaction is made using a token public key; it has no need for additional encryption), and encryption plus signing is used when communications confidentiality is needed (for example: when communicating EOTP codes, they are first signed by the users' public key, then encrypted to the appliances' public key before transport).

[iii] Multi-Party approvals (which are more versatile than simple dual-signatory procedures) allow a bank to record a set of individuals who have authority to approve certain transactions, and a set (not necessarily the same) who have authority to decline certain transactions, along with the rules for which transactions require multiparty approval, along with how many of each set of user are required to concur before the transaction is approved (or declined). The CryptoPhoto multi-party solution supports any arbitrary rules for this process, and manages out-of-band messaging and digital signing with non-repudiation. It was the lack of any effective multiparty approval process, and the lack of any out-of-band signing methods, and the lack of strong authentication security, that resulted in the insertion of $951M worth of fraudulent SWIFT transfers through the NY fed from the hacked Bangladesh central bank in 2016; $81M of which was stolen (and subsequently laundered through Philippine casinos) before a typographic error by the hackers raised the suspicion of Deustche Bank's international desk, who then blocked the rest.

[iv] Insecure initial conditions arise due to the unknown nature of the original cloud operating-system installation conditions, which have typically been performed by a cloud engineer or cloud software vendor, and not a security expert. Cloud-provisioned operating systems have typically been installed at some time in the past and "snapshotted", meaning that all tenants essentially run an identical installation, with assorted unknown dynamic patches applied to it at provision-time (e.g. root password change). Unfortunately, the Linux installation system generates many security-dependent configurations at ***install*** time, including SSH Daemon keypairs, web and mail certificates, Diffie-Hellman (DH) key-exchange parameters, SELinux policies and filesystem labelling, permissions, passwords, cipher-strength defaults, and more, and it does all this using default deterministic pseudorandom number sources from potentially unknown/compromised/backdoored non-patched/non-updated install media packages. Running a security service from a snapshotted operating system is simply too dangerous to consider.

Additionally, cloud hosts typically always erase the history of adjustments they've made post-install, and typically install custom software of their own into guest operating systems which provides their virtualization host infrastructure additional insight into, and control over, guests (usually for performance purposes). The opportunities for undesirable side effects of their custom adjustments and software, and the risk owing simply to the possibly large number of unknown individuals involved in the preparation of deployable images and the motives of those individuals is also too dangerous to ignore. Erasing everything and starting from scratch using expert professionally commissioned "to be maximally secure" setup from the start is the safest way to deploy security appliances.

[v] Anaconda is the automated and optionally headless (unattended) linux installation system otherwise known as "kick-start" (*.ks) accessible from media-boot and grub (the linux bootloader)

[vi] TRNG, or True-Random Number Generator, is a hardware extension to provide non-deterministic entropy, typically for the production of cryptographic keys. For redundancy, CryptoPhoto uses both the "haveged" and "ekeyd-egd-linux" daemons for TRNG.

[vii] HSTS (HTTP Strict Transport Security) is a mechanism to force web browsers never to use insecure HTTP port-80 communications; it is necessary to prevent man-in-the-middle downgrade attacks against TLS such as "SSL strip", active attack spoofing/proxies, malicious wifi, and so on.

[viii] HTTP Public Key Pinning (HPKP), is a security mechanism which allows HTTPS websites to resist impersonation by attackers using mis-issued or otherwise fraudulent certificates.

[ix] Most security-conscious organisations (e.g. schools, universities, corporations, etc), many home and office security firewalls (e.g. firebox) and even some entire countries (middle east, China) force all users to install an organisational Certificate Authority (CA) on their machines in order to access encrypted (e.g. TLS) sites. This is done so that content-inspection rules on an intermediary device (proxy, firewall, etc) can block incoming malware, prevent outgoing data exfiltration, and enforce other content policies. While normally good, sometimes these devices can be compromised, and sometimes fake CA's get installed so that hackers can also manipulate encrypted content. CryptoPhoto is only used for authentication purposes, which is not appropriate for intermediaries to access, which is why TLS communications are additionally encrypted with secondary appliance keys.

[x] The modulus length of 2109 is chosen to be deliberately and inconveniently larger than the far more common 2048bit RSA modulus length, to ensure no purpose-optimized-for-2048-bit-cracking techniques or hardware would be immediately usable against it, should they one day be found or invented.

[xi] CryptoPhoto has multiple Software Development Kits (SDK), but they fall broadly into 2 categories: "Server" which assists a provider to make use of CryptoPhoto security within their systems and services – for example, Perl, PHP, or Python etc programming libraries, and "Mobile" which is the User-side token-handling mechanism which becomes embedded into a providers mobile App. CryptoPhoto mobile SDK has two distinct usage models: "inside", where our SDK is integrated into a Provider Mobile app, and "outside", where a provider's mobile app functionality is integrated into the CryptoPhoto mobile app and branded to the Provider before deployment. This latter model allows CryptoPhoto app-based security enforcing features to additionally protect the providers mobile assets along with its own token-handling code, and provides numerous additional benefits, including hyper-rapid provisioning, automatic user pairing and enrollment, secure real-time updates, single codebase for a wide variety of mobile platforms including iOS, Android, WindowsPhone, and Blackberry, and support for an enormous range of old mobile devices (including as old as iPhone 3 etc).

[xii] CryptoPhoto's security mechanism is based on end users possessing a security "token", either soft (typically in a smartphone/tablet) or hard (physically printed/produced). Tokens are a collection of photos and codes and keys, either real (hard) or electronic (soft).

[xiii] JavaScript Object Notation (JSON) is the structure which stores CryptoPhoto soft-tokens. See example in Appendix 4 Tokens.

[xiv] Bootstrapping the security of customer initial enrolment is somewhat dependent on what kind of provider is using CryptoPhoto, and the tradeoffs they choose to enforce with respect to customer convenience, versus enrollment assurance strength and the risk profile of their users in relation to the possibility of malicious certificate substitution attacks during enrollment. A bank, for example, wishing to have complete confidence in correct identity and no malicious activity, could require users to initially obtain their tokens in person at a branch, where protection against TLS attacks can be enforced. Subsequent to initial enrollment, customers enroll additional devices (obtain additional CryptoPhoto soft-tokens) using their existing devices, which provides a secure out-of-band solution for preventing future TLS attacks (new tokens can only be obtained when the user authorizes it on their existing token)

[xv] CryptoPhoto EOTP (event-based one time password) keys are TRNG-generated 64bit random numbers pre-provisioned into Tokens. CryptoPhoto also supports arbitrarily-long TOTP (time-based one time passwords).

[xvi] CryptoPhoto new-users can "bump" their smartphone on the "enter" key of their computer to activate rapid new-token enrollment. All new tokens (including "bump" sourced ones) have the option to require authorization by one or more users existing token(s). In other words – to add a new phone to your account, you authorize this using your old phone.

[xvii] Initial enrollment from a mobile app/sdk perspective begins with the app/sdk encountering a TokenID; since these TokenIDs have no meaning, the app makes a discovery call to the CryptoPhoto cloud to find out which provider belongs to this TokenID and the endpoint for their Appliance. This mechanism allows one app/sdk to support multiple providers and appliances, and also allows single providers to operate multiple services and/or brands with multiple appliances, all invisible to the user (everything "just works" automatically, no matter how the provider has chosen to set up their architecture, including if they wish to change it in future).

[xviii] The CryptoPhoto mobile-App or mobile SDK architecture is a secure, purpose-built containerized platform similar to "phonegap" or "cordova"; the barest minimum external "outer container" exists in the native portion of the code which provides uniform cross-platform and cross-device access to device hardware functionality and security enforcement and signature checking of the "inner container", and the bulk of the processing and responsive UI/UX elements are general-purpose HTML and JavaScript living within an "inner container"; older devices (e.g. iPhone 3) lack the power

to perform large-key RSA operations in JavaScript, so these few devices are exempted from our additional-encryption over TLS [they still use the most-secure (as known today and available in their device crypto stack) best-practice TLS for all connections of course].

[xix] A "bound" token is one which has been issued to a User, and has been associated with their user account inside a CryptoPhoto Appliance. Tokens are ***issued*** "Unbound", and then securely connected to the account of a user by a range of different mechanisms (e.g. in-person, by a bank teller using an on-screen option, or "self-service" by a customer downloading a new token and using one of their existing tokens to authorize (bind) it). Note that "account" is a general-purpose term: CryptoPhoto Appliances have no actual knowledge of user identities or their account identifiers; these are purposefully withheld from Appliances to enforce separation-of-duties security architecture to protect against serverside break-ins (a one-way mechanism is implemented, such as SHA256 or a PBKDF (they make the choice and can use anything they like; internally, we also hash whatever ID they provide before use) by the provider so an Appliance can associate with accounts, without knowing anything about the actual accounts).

[xx] Users who lose their phone can use any alternate token they have enrolled, which may be another device like a tablet or spare phone, or may be a printed "hard token" (usually referred to as a "recovery" token, although it is just another token). In the case of a lost phone, a user at login time changes the token they wish to use with the on-screen dropdown box, logs in using their recovery token, then deletes the token associated with their lost device from their account (and typically enrolls for a new token on their replacement device at the same time)

[xxi] The decision about "one or two apps" is left to a Provider; CryptoPhoto supports either/both. A Provider can opt for their own app, and no secondary CryptoPhoto app (in which case CryptoPhoto becomes included in their App via SDK), or, a Provider can make use of the existing general-purpose CryptoPhoto App (in which case, the security protection against in-device malware is doubled, on account of the CryptoPhoto app itself, plus the Providers app, both independently implementing their own anti-malware features). The CryptoPhoto App opens automatically when needed, and "goes away again" automatically when finished. From the day-to-day usage perspective of a CryptoPhoto end User, the visual flow and operating mechanism of both methods is completely identical.

[xxii] A comparison with "Microsoft Authenticator" and "Google Authenticator" with "CryptoPhoto" has been done. A new CryptoPhoto user can be signed-up and enrolled in CryptoPhoto in under 2 minutes in less than a dozen pageviews, without requiring a skilled user and no risk of "wrong app" installation. A new "Microsoft Authenticator" user takes approximately 30 minutes to sign up and enrol, and this takes approximately 70 consecutive pageviews (varies based on users skill and pre-existing knowledge) and requires the user to be proficient with app-store searching and to understand how to find the correct app. "Google Authenticator" takes approximately 20 minutes, and approximately 60 consecutive pageviews, and also requires a skilled and attentive user.

[xxiii] For a human to properly authenticate a website (the first half of a mutual-authentication), they would have to manually observe the TLS padlock, then manually check the authenticity of the certificate, its chain and issuers, plus somehow perform HPKP and HSTS in their mind. Not only is this absurd, it's impossible on (for example) iPhone devices; iOS10 no longer offers the option to manually inspect certificates, plus its also impossible for everyone behind any certificate substitution protective device. Downgrade attacks work by preventing TLS, so a user under threat is not required to notice and inspect the TLS padlock, they are required to notice the ***absence*** of it - something which is near impossible for any ordinary user to accomplish! All this assumes also that the protected site somehow properly handles the second half of the mutual-authentication (securely identifying the user), which practically never happens (plaintext passwords in web forms are the contemporary norm still), and just because a site does authentication in both directions, this does not yet make it "mutual" – some way to connect both together, like a literal handshake, so that it is impossible for either to function in the absence of the other, is needed to make a protocol mutual. Existing authentication other than CryptoPhoto, including ones claiming to be "mutual", have not yet solved this problem.

[xxiv] CryptoPhoto cTV is out-of-band because it is a completely external verification mechanism that is not part of the direct transaction flow that a user makes use of when entering a transaction (e.g. they enter the transaction on their PC, and their phone is used to verify it, or, they enter their transacting into Safari on their iPhone, then they use the CryptoPhoto app to verify the Safari transaction). Note that "band" and "channel" are two different things

[xxv] For transactions originally entered into a PC, then verified by a customer using their mobile phone, the verification obviously makes use of a second channel (the PC to webserver channel is clearly not the same as to the appliance-to-phone channel, especially when the PC uses WiFi, and the Phone uses a different 4G network). For transactions entered via a user on their mobile device, then verified by the user on the same mobile device, the concept of

"channel" is not so clearly distinct, which is where the intended meaning of the word "channel" requires analysis. In security, "channel" does not usually *mean* channel at all – when this term is used, it's a concept placeholder for the idea of neutralizing attacks by making use of multiple independent protection paths such that an attack successful against one has at least another counterpart which is generally proposed or assumed to be out of reach of the attacker. Attackers in such an instance include MitM and in-device mobile malware. CryptoPhoto protects against MitM using signatures and predefined token endpoints to prevent and/or neutralize MitM activity. CryptoPhoto enforces a second independently secured path for the out-of-band verification of transactions on mobile devices through an App-level mobile-malware self-defense and anti-tampering mechanism. Mobile malware which might, for example, subvert Safari on an iPhone, will generally have no effect against the independent CryptoPhoto app, and the app itself contains anti-malware technologies as well.

[xxvi] WAF (Web Application Firewall) is a server-side proxy which sits between the internet, and the Providers systems, typically to enforce strong security rule and protect Provider software against things like injection attacks and form exploits. Many WAFs are able to add features and functionalities to Provider applications directly, without any change to Provider code, on account of the end-users connecting to the WAF and it's trusted and privileged "in the middle" position.

[xxvii] CryptoPhoto is suitable for use as a password-replacement mechanism: in other words, providers can opt to stop using passwords and use CryptoPhoto only if they choose. For providers who choose to keep using passwords, whether or not CryptoPhoto is then deployed as a pre-password step (to block phishing of password) or a post-password step (to block username dictionary pharming irritations and detect compromised accounts) is a deeply philosophical and lengthy discussion subject to a great deal of opinionated points of view. CryptoPhoto avoids this debate by functioning in whichever mode the Provider wishes to deploy.

[xxviii] CryptoPhoto exists to protect users against phishing precisely because it's not possible to protect passwords against getting phished, thus, in the presence of an effective anti-phishing protection like CryptoPhoto, it is a compelling argument to suggest that ***still*** attempting to protect the unprotectable (passwords) against phishing is unnecessary..

[xxix] Password re-use is a huge problem; modern users have hundreds of different online accounts, so many of them use the same password for different things: the problem of course is that it only takes one of these things to have a password breach which then reveals to hackers a new set of usernames and passwords which can be used to access the users other accounts. Exacerbating the problem are password managers, where it takes only 1 untrusted bit of software on a user's machine (or malware, or vulnerability or password-service hack) to reveal every account and every login the user owns to the attacker, regardless of what passwords are used.

[xxx] A salt is a random initialization vector (IV) mixed with a digest to hamper offline password attacks. A secret-salt is a salt that is not revealed to the entity making use of the hash (to reveal the salt to the entity would facilitate offline dictionary attack by the entity itself). CryptoPhoto separation-of-duties architecture exists to prevent any unlikely intrusion into a CryptoPhoto appliance from getting access to customer identifying information. Similarity, the CryptoPhoto appliance handles authentication, so any intrusion into a providers systems does not reveal authentication credentials to the attacker, since they're secured separately on the CryptoPhoto appliance.

[xxxi] Visually, a "challenge" looks like a random photo to a user, and contains a dropdown selection box to let them use an alternate token, plus an area where they can manually type in an EOTP code if they're using a non-smart printed hard-token, or if their mobile device has no data connectivity. Technically, the challenge is a randomized and perturbed image file which itself has been digitally signed and contains anti-theft and anti-tracking and anti-robot technology, delivered inside a container which includes anti-scraping and anti-robot blocking and detection mechanisms. CryptoPhoto implements "threat deception" measures for active MitM mitigation, to prevent hackers knowing their attack has been detected, rather than blocking them. We also offer two hybrid mechanisms to warn users either in-band (via the difficult-for-a-hacker-to-detect mechanism of superimposing the warning onto the photo) or out-of-band (via warning to the legitimate users mobile device through our Transaction Verification mechanism)

[xxxii] See, for example, http://zeropasswords.com/pdfs/WHATisWRONG_FIDO.pdf or the workinggroup archives.