Red button and yellow button: usable security for lost security tokens (Position paper)

Ian Goldberg[1⋆], Graeme Jenkinson[2], David Llewellyn-Jones[2], Frank Stajano[2]

[1] University of Waterloo, Canada
[2] University of Cambridge, United Kingdom

Abstract. Currently, losing a security token places the user in a dilemma: reporting the loss as soon as it is discovered involves a significant burden which is usually overkill in the common case that the token is later found behind a sofa. Not reporting the loss, on the other hand, puts the security of the protected account at risk and potentially leaves the user liable.
We propose a simple architectural solution with wide applicability that allows the user to reap the security benefit of reporting the loss early, but without paying the corresponding usability penalty if the event was later discovered to be a false alarm.

1 Introduction

Imagine this scenario. I go abroad on a trip. I get back and, on attempting to do some online banking, I can no longer find that all-important physical token that lets me log in. I know I brought it with me abroad, and I'm pretty sure I was always very careful with it, but now I can't find it. What should I do? I could, of course, report the loss to the bank, but then I'll have to go through the security rigmarole with the call centre operator (after I get through the automated voice menus and more than my Recommended Monthly Allowance of muzak), answer long-forgotten "secret questions" to prove that I am me, jump through various hoops to have them send a new device in the post to my registered home address, possibly pay an additional insurance premium for having lost a security token... all the while thinking that, thanks to Murphy's law, shortly after the request for a new token has been accepted I will find the old one in a hidden pocket of my suitcase (not that I didn't already check, of course).

So instead I don't report the loss, hoping that sooner or later I'll find the lost token at home, which on the face of it is by far the most likely outcome. But I'm feeling slightly uneasy, imagining that maybe some foreign crooks found it and are hacking into my current account as we speak.

This example shows how the security procedures currently in use to deal with lost security tokens give me a rather powerful disincentive to report the

⋆ On sabbatical at the University of Cambridge Computer Laboratory while working on this topic.

Authors' preprint of 2016-06-13.
Final version to appear in *Proceedings of Security Protocols Workshop 2016*, Springer LNCS.

loss immediately. If I still have any hope that I might find the token later, I might save myself a lot of hassle by pretending I never lost it and meanwhile continuing to look for it. Note that we take the term "security token" in its most general sense, to encompass not only calculator-like login devices but also credit cards and so forth.

The research question we address is:

> can we devise a better procedure that might reconcile the two conflicting goals of the token user?

... namely not wanting to be hassled unnecessarily in the relatively common case that the lost token will be found again (= "I'd rather not report the loss, at least yet"), and not wanting to end up in trouble in the rare but not impossible occurrence that the token was actually stolen and misused (= "I'd rather report the loss immediately").

To be more precise, whether the token user ends up in trouble following the loss of the token and its misuse by criminals depends on the exact terms and conditions that regulate the agreement between the token user and the token issuer. In the banking case it is common for the liability of the user to be limited so long as the loss was promptly reported through official channels (e.g., declaration to the police), but this may vary by jurisdiction. In any case, whatever the arrangements, if value is taken out of the system by crooks who access the account without authorization, the legitimate actors in the system are going to incur a loss, however distributed, and it is reasonable to assume that, to avoid moral hazard ("I don't pay for it so I don't give a damn"), the bank will pass on at least some of the liability to the user if the user did not report the loss. So we assume that the user has *some* incentive not to leave the loss unreported.

From another viewpoint, one might observe that some banking tokens are protected by a PIN and allow only three attempts before locking up, and therefore that the user should not care about the possibility of misuse of a lost or stolen token unless the adversary is assumed to be able to overcome such protections. Not all tokens offer such protection, though; and, even for those that do, the probability of guessing a human-chosen PIN in three tries is significantly higher than a naïve calculation might suggest, because human-chosen PINs are far from uniformly distributed [1]. Therefore, by the "moral hazard" argument above, in our general discussion it still makes sense to assume that the user should have some incentive to report the loss of a token.

On a related note, we observe that there is a wide spectrum of possible strengths of the hardware tamper resistance mechanisms offered by security tokens, from non-existent through smartcard-grade to sealed and protected cells enclosing processor, memory and battery, and beyond. And, orthogonally, there is a wide spectrum of possible consequences for the loss of the token, from negligible to extremely serious. It is reasonable to expect that the appropriate countermeasure will depend on the threat scenario and that there will be a correlation between the two variables. It seems unlikely that a single solution will suit every possible case: for any given hardware implementation of a security token, one

will always be able to construct horrendous threats for which the countermeasures will be insufficient and trivial threats for which the countermeasures will be overkill and too expensive. Any solution will only be optimal for a subset of the possible cases. Any general solution will be a trade off. This is basic risk management.

2 The core idea

Our core idea is to decouple the reporting of the loss from the reissuing of the token. We wish to encourage the token user to report the loss as soon as possible, and therefore we wish users to incur no penalty for doing so[3]. Reporting the loss should be a "no big deal" activity that gracefully tolerates false alarms.

We model our solution as two metaphorical buttons, a "pre-alert" yellow button and a "true alert" red button. The semantics of the yellow button are *"I don't know where the token is, but maybe I'll find it again soon. Please disable it temporarily."* Pressing the yellow button is a lightweight and no-hassle operation (involving no interaction with call centres) that has the effect of notifying the back end that the token has been lost: the back end should reject any attempts at authenticating with that token until otherwise notified. Pressing the yellow button should not, however, revoke the token or initiate any heavy-duty administrative actions such as issuing a new one.

The semantics of the red button are *"I really think I lost it now, and I have given up any hope of finding it again. But I need to use it, so please send me a new one instead."* Pressing the red button is a request to revoke the lost token permanently and to issue a replacement for it. It will involve cost in several dimensions, potentially including administrative actions requiring unpleasant interactions with voice menu systems and call centre operators, penalties, handling costs and delays, but will only be invoked as a last resort.

Pressing the yellow button (freezing the accounts) is a cheap action that can be undone, whereas pressing the red button (revoking the accounts) is an action that is both expensive and irrevocable. The advantage of this yellow/red strategy is that it lowers the cost to the user of taking protective security action and therefore it increases the likelihood that such action will be taken promptly, keeping the user more secure.

At this stage we are building a mental model for the user rather than a particular implementation: we said "metaphorical buttons" because we do not necessarily envisage them as physical buttons on a physical device. If they were, and they do not have to be, we would have to specify on which device they would appear (clearly not on the token whose loss we are considering reporting). It may well be, for example, that the yellow button is implemented simply by sending a particular SMS or email to a designated recipient. The red button, on the other hand, will involve similar procedures to those currently in use for revoking a

[3] To those objecting that reporting the loss freezes the account and prevents the user from logging in, we point out that, having lost the token, the user is unable to log in whether or not they report the loss.

lost token, because there must always be a hierarchy of more and more time-consuming but more and more powerful emergency procedures one can invoke even if every supporting piece of authentication evidence (including security tokens, passports, etc.) has been lost or has otherwise become unaccessible.

There is of course the question of how the red and yellow buttons are *themselves* secured, and how the yellow button is "unpressed"[4]. There are a number of plausible instantiations, but we present one next. The red and yellow buttons may actually just be codes generated at the time the token was delivered to the user (either initially or after the last press of the red button). These codes can be likened to GPG revocation certificates [2]: they should be stored in a way that the true owner is sure to maintain access to them, but it is not horrible if others also gain access. This is because if someone else accesses the red or yellow button, they can lock or revoke the token (just as they could publicize the revocation certificate to revoke a GPG key), but that is arguably the correct outcome given that the token is no longer in the hands of its legitimate owner. Storing the red and yellow button codes on one's phone is reasonable (as long as there is also a more permanent copy somewhere, say taped to the electricity meter at home).

When the yellow button is pushed, whoever pushes it is given an unlock code, which should be high enough entropy to be unguessable, so it is basically a capability to unlock *that particular yellow-button pressing*. The only ways out of the yellow state are with the unlock code issued when the state was entered, or by pushing the red button.

Some users may indeed be tempted to "keep the yellow button pushed" whenever they are not actively using the token (or perhaps whenever they get on an airplane), and this is not totally unreasonable. The downside of this conservative strategy is that it amounts to manual locking and unlocking of the token at every use, which negatively affects usability; moreover, the user is betting that he will not lose his copy of the unlock code—the risk to the bet is having to push the red button, as the only ways out of the yellow button state are with the unlock code issued when entering the state, or the red button.

This is only a sketch, rather than a full design addressing all the relevant issues of authentication to the freezing proxy or revocation proxy[5] after having

[4] The yellow button is logically a switch with two states, "alert on" and "alert off"; so the "yellow alert" state stays on until explicitly revoked. The red button, instead, is logically more akin to a "trigger" button that can be used to fire off an alert but not to say when the alert is over (it will be over when the replacement token is shipped to the user). So if the yellow button is implemented by sending an SMS, then another SMS must be sent to unpress the button. A timeout would also work but would be less secure and would remove control from the user and we therefore advise against it.

[5] Defined as the in-cloud servers that the yellow and red buttons respectively talk to, and that consequently issue "account freeze" or "account revocation" commands to the servers on which the user accounts are hosted. This level of indirection is necessary when one token unlocks accounts on distinct servers. We shall explore this idea further in the next section.

lost the authentication token, of the trust relationship between the proxy and the user, of the possibility that attackers might go after the proxy instead of the token and so forth. We do however consider it an improvement over the status quo and a solution with wide applicability, and we therefore consider the idea worthy of wider discussion even at this preliminary stage.

3 Red/Yellow for Pico

Most banking tokens are dedicated, in the sense that each bank issues its own; if a person has accounts with three banks, they generally need to use three different tokens. This makes things easy for the bank, who has full control over the design and operation of both endpoints, but complicated for the user, who might accrue a keyring of tokens as bulky as that of a prison warden. From the user's viewpoint it would be much more interesting to use a universal token capable of granting access to several independently run accounts.

We originally conceived the red/yellow idea in the context of the Pico [3] project[6], which is indeed a universal authentication token: rather than being tied to one particular issuer or back-end service, it contains potentially thousands of independent login credentials for the same user, for accounts on many unrelated back ends. We shall now therefore discuss how to apply the red/yellow button idea to Pico, not so much because that is where the idea occurred to us originally but because doing so highlights the complications associated with the additional level of indirection required to deal with multiple independent back-end verifiers.

Within the Pico system, pressing either button requires contacting all the back-end servers on which the user has accounts. This in itself is a penalty to pay, at least in terms of privacy loss. While Pico allows the user to maintain a separate persona for every service (or even several personae with the same service), pressing the yellow (or, a fortiori, red) button would allow a global network observer to link the many simultaneous revocation requests back to the same Pico, thus deanonymizing the user. Temporal linking might give the user away even if mixes, remailers or other network anonymizers were used to obfuscate the source of the requests. Adding random delays of sufficient magnitude between the revocation requests, in an attempt to frustrate such relinking, would leave a window of vulnerability between discovery of loss and protection of account that somehow goes against the motivating principle of introducing the red/yellow button architecture.

An alternative strategy, geared towards privacy, exploits the Pico's "network share server" [3, Section 4.1] that must send a periodic keep-alive signal without which the Pico locks up. In this strategy the user who has lost the Pico sends the revocation requests (or at least the yellow ones) just to that server, telling it to stop providing its keep-alive signal, but without notifying the actual back ends. This has the security disadvantage of not locking the accounts immediately but only within the time interval (say half a day or one day) within which the

[6] `https://mypico.org`

Pico expects to hear the keep-alive signal again from its network share server. It also has the additional security disadvantage that, if the adversary managed to extract any credentials from the lost Pico (either through hardware attacks or by capturing it before it locked itself), then the block will be totally ineffective because the back ends won't know about it. This strategy offers some privacy at some cost in security and it is debatable whether this is the appropriate trade-off.

A variant of this strategy, geared towards both privacy and immediacy of revocation but with an efficiency cost for the user, is to require the network share not just once or twice a day but at every authentication request. This implements a different trade-off, with greater security (accounts are locked immediately) at the cost of greater energy consumption for the Pico, greater latency for every login and potentially lower availability as the user won't be able to log in whenever the network share server is unreachable, even if the server to which the user intended to log in is reachable.

Yet another strategy, also geared towards privacy and immediacy of revocation but with the inefficiency cost shifted towards the servers, might involve some kind of "anonymized" public revocation, by which we mean an architecture with the following properties. There is an append-only publicly writable bulletin board in the cloud (suitably protected against denial of service) where yellow and red button presses end up. Each button press is encrypted (with suitable randomization) with the public key of the service it intends to freeze or revoke, to ensure that a global network observer may learn that Alice froze or revoked something, but not *what*[7]. Each service is notified by the bulletin board whenever a write event occurs and must attempt to decrypt the new message in case it was addressed to that service (this inefficiency being the cost of privacy protection for the user). It is interesting to discuss the incentives of the parties involved, in search of a reward system that would motivate the services to incur this extra cost in order to protect the privacy of their users.

4 Conclusions

We have sketched a simple and very general idea for allowing users of security tokens to report loss of token without incurring the heavy penalty usually associated with doing so. This should in turn improve security.

The idea is still at an early stage and we have not implemented it yet. We are keen to discuss further architectural and scalability considerations with our peers at the workshop.

Acknowledgements

The authors with a Cambridge affiliation are grateful to the European Research Council for funding this research through grant StG 307224 (Pico). Goldberg

[7] So when Alice loses her Pico and presses the yellow button, thus writing hundreds of revocations to the bulletin board, the NSA learns that Alice lost her Pico, but is none the wiser about what services she has accounts with.

thanks NSERC for grant RGPIN-341529. We also thank the workshop attendees for comments.

References

1. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? The security of customer-chosen banking PINs. In: FC '12: Proceedings of the 16$^{\text{th}}$ International Conference on Financial Cryptography. (March 2012)
2. The Free Software Foundation: The GNU Privacy Handbook. `https://www.gnupg.org/gph/en/manual/c14.html#REVOCATION` (1999)
3. Stajano, F.: Pico: no more passwords! In: Proceedings of the 19th international conference on Security Protocols. SP'11, Berlin, Heidelberg, Springer-Verlag (2011) 49–81