Evaluating the Android Security Key Scheme: An Early Usability, Deployability, Security Evaluation with Comparative Analysis

Robbie MacGregor
Dalhousie University

Abstract

The Android Security Key scheme promises to provide users of Android handsets with strong, public key-based multi-factor authentication similar to that achieved via USB security keys. In this paper I present an evaluation of Android Security Keys using the usability, deployability, security (UDS) framework, as well as a consideration of security and privacy issues. A comparative analysis against other 2-factor schemes is also provided, with a focus on USB security keys.

I argue that Android Security Keys differ from USB security keys in terms of both usability and deployability when a basic set of UDS benefits are considered. The convenience of using a mobile handset already in a user's possession as a second factor is privileged over the efficiency of login tasks. A cursory assessment of Android Security Keys gives the impression they offer a similar set of security benefits to USB-based implementations, but I identify potential improvements during a closer analysis of the security model and associated threats.

1 Introduction

Legacy, textual passwords are here to stay. Passwords are familiar, cheap, and convenient. They are easy to use, readily deployable, and their longstanding incumbency has placed them in an almost unassailable position [1–3]. There have been many attempts to replace textual passwords with more secure alternative authentication methods, but no scheme has managed to supplant them, and it seems unlikely that an alternative will provide all of the benefits that textual passwords do while simultaneously offering users meaningful security gains [1].

The problems associated with passwords have been well documented and studied [4, 5], with many issues apparent from the very beginning [6]. Where passwords are concerned, researchers and service providers are trying to address problems related to usability and security that have persisted for decades. With replacement unlikely, tools and security practices aimed at supporting the users of legacy, textual passwords are common. Users are most frequently presented with advice to enable multi-factor authentication schemes when these are available [7], and to use password managers to support the creation and use of strong, unique passwords for all online accounts [8, 9].

Some of the most successful attempts to offer usable, secure multi-factor authentication have involved USB security keys [10–12]. These keys (implementing the FIDO Alliance's U2F [13] or FIDO2 [14] protocols) provide a cryptographic second factor to enhance the security of users authenticating to online accounts. USB security keys have been demonstrated to effectively improve user security when compared to 2-Step SMS and other OTP-based schemes [10], but require specialized equipment and are not always convenient. In an apparent attempt to address these issues, and encourage the adoption of strong, public key-based multi-factor authentication, Google launched the Android Security Key scheme in April 2019 [15, 16].

Android Security Keys show some promise, but have yet to be subjected to systematic, independent evaluation. In this paper, I offer an early evaluation of the Android Security Key scheme using the usability, deployability, security (UDS) framework of Bonneau et al. [1]. I additionally consider how Android Security Keys respond to some of the issues with USB security keys highlighted by researchers like Reynolds et al. [11] (who examined usability through a pair of case studies) and Jacomme and Kremer [12] (who completed a formal verification of security benefits).

In the sections that follow I unpack both foundational and recent research informing this work, with a focus on the re-

Who Are You?! Adventures in Authentication (WAY) 2019.
August 11, 2019, Santa Clara, CA, USA.

search and usability, deployability, security framework of Bonneau et al. [1]; I discuss the motivations for this paper; describe both the methodology and the limitations of the assessment completed; summarize my evaluation of the Android Security Key scheme; and discuss early results, contributions and plans for further research.

2 Background

The users of online accounts are largely authenticated by means of textual passwords. The continued appeal of passwords is not just a function of their familiarity and ubiquity, but also reflects the fact that they are convenient for users, easy to deploy, and come with little overhead [1–3]. Bonneau et al. [1] conducted an extensive qualitative review of web authentication schemes, considering 35 proposals to either replace or augment legacy, textual passwords. The researchers found no alternative authentication method that could provide all of the benefits associated with passwords while improving upon their security or usability. Bonneau et al. provide some of the best arguments for the persistence of passwords in the fields of authentication and usable security.

Textual passwords persist, and so do a variety of problems associated with their usability and the level of security they provide the users of online accounts. Some of the problems with passwords are longstanding and were effectively catalogued decades ago. Parallels exist between the issues discussed by those studying passwords in the present, and the challenges Morris and Thompson enumerated in their article examining passwords and user authentication on early Unix systems [6]. The more recent proliferation of online accounts, and the increased burden this places on the users of textual passwords, has resulted in problems related to password reuse and other practices that can leave users vulnerable to account compromise [5].

Multi-factor authentication schemes are one way to help address some of the problems with textual passwords, enhancing security. After enrolling in a service provider's multi-factor program, users must present a secondary secret, in addition to a password, in order to verify their identity and be authenticated to a service [17]. Frequently, this secondary secret represents proof that the user is in possession of a unique piece of hardware (a token or mobile handset), though a variety of other options incorporating biometrics and even ambient environmental factors have been proposed and tested.

Multi-factor authentication has been demonstrated to improve the security of users, but many barriers to adoption still exist [7, 18]. In the case of common multi-factor schemes usability, generalized by users in discussions centred around cost and convenience, is seriously at issue [18]. Google's recently released Android Security Key scheme [15, 16] represents an attempt to entice users to adopt a multi-factor authentication method by presenting it as highly convenient (the second factor is, in effect, a mobile handset likely to be already in use), while at the same time promoting a strong, public key-based form of multi-factor. While users maybe familiar with the idea of using a handset to help enhance the security of online accounts via one-time passwords [19] or interaction with mobile push dialogues [20], the Android Security Key scheme leverages a cryptographic second factor and communicates with service providers via the WebAuthn API [21].

Android Security Keys seem promising, but the scheme is only recently out of beta, and much of the published information concerning it directly reflects either the enthusiasm of its own developers or a marketing effort on the part of Google. Android Security Keys need to be quickly and independently evaluated by researchers focused on authentication, usable privacy and security. Fortunately, there are many tools available to facilitate this kind of assessment. The qualitative framework of Bonneau et al. [1] provides an excellent starting point. Bonneau et al. did not merely assess authentication methods, but also developed a straightforward evaluation framework, modelled loosely upon Nielsen's heuristic evaluation [22] and other inspection techniques from the field of user interfaces. By considering a set of 25 ideal benefits across the domains of usability, deployability, and security researchers can succinctly describe an authentication scheme, locating it in relation to others in a way that facilitates comparative analysis. The usability, deployability, security (UDS) framework offers flexibility, extensiblity, and has been largely accepted as a useful tool in the field of authentication.

This paper is the first to evaluate Android Security Keys, but a UDS evaluation of USB security keys (a related, precursor to Android Security Keys) was completed by Lang et al. [10]. These researchers were Google insiders, assessing the company's adoption of USB security keys, and a set of tools and protocols with which they had had some association. Lang et al. may have been too optimistic in their assessment, but established a baseline and jumping off point for making future evaluations and comparisons. More recently, USB security keys were assessed for usability through a pair of case studies by Reynolds et al. [11]. These researchers examined enrolment under controlled conditions, and authentication activities separately in the field. An explicit focus on usability, and a user-centred exploration of the authentication process under a design that enhanced external validity, generated results that can inform more in-depth discussions of both USB security keys and related authentication methods. Jacomme and Kremer [12] formally verified the security benefits associated with USB security keys, modelling threats to both security and privacy, and providing a detailed account of what the scheme achieved and what could be improved upon in future implementations. The results of Jacomme and Kremer can help direct future work, and move discussions beyond the kinds of comparative analysis the UDS framework permits.

Evaluating a new authentication method is in many ways about asking how it improves upon past offerings. My evaluation of Android Security Keys largely focuses on how they

differ from USB security keys, and as such is informed by recent research efforts focusing on this scheme.

3 Motivation

The Android Security Key scheme was announced publicly in April 2019 and is only recently out of beta [15, 16]. An early evaluation of Android Security Keys is needed to determine what improvements they may offer over similar authentication methods. The UDS framework provides the right toolkit to complete a timely evaluation and comparative analysis that will aid researchers in the fields of authentication and usable security to locate this new scheme in relation to others. These early results can additionally provide some direction for future investigations by shining a light on issues bearing closer scrutiny. This evaluation is necessary, and will be of interest to researchers working in authentication, passwords, usable security.

4 Methodology

In support of a comparative analysis, I apply the UDS framework of Bonneau et al. [1] to evaluate the Android Security Key scheme. Considering the basic set of 25 usability, deployability, and security benefits an ideal authentication method would offer, I rate the performance of Android Security Keys based upon a knowledge of underlying protocols and a hands-on inspection of the authentication scheme. The results of my evaluation are compared against the baseline findings of Bonneau et al.[1] where the incumbent, textual passwords, and Google's familiar 2-Step multi-factor method are concerned. I take the results of the UDS evaluation carried out Lang et al. [10] to be definitive regarding USB security keys implementing the FIDO Alliance's U2F [13] or FIDO2 [14] protocols (though some of their claims need to be addressed in the discussion section 6). Data from prior work is used to make comparisons, and determine what gains, if any, Android Security Keys provide over past offerings.

5 Evaluation

The Android Security Key scheme implements a form of public key-based multi-factor authentication. A user's secret key is stored on their mobile handset, and used to sign messages verifying that user to an online service provider at login. The scheme combines legacy, textual passwords with a cryptographic second factor similar to that employed by modern USB security keys. Enrolment and login tasks closely reflect what users of Googles 2-Step and push-based multi-factor schemes are familiar with [19, 20]. A one-time enrolment task is completed via the browser, and a simple dialogue accompanied by a set of "Yes" and "No" buttons is presented to the user at login. Recently released, this method of authentication is currently only available to the users of Google services, using the company's Chrome browser on a client machine, and in possession of a mobile handset running version 7 or higher of the Android operating system. An UDS evaluation of the Android Security Key scheme follows.

5.1 Usability

Android Security Keys are not *Memorywise-effortless*, nor are they *Scalable-for-Users*. The scheme augments textual passwords, but still relies upon them. Strong, unique textual passwords are assumed in this evaluation, as their absence would result in Android Security Keys being denied other UDS benefits. Like other mobile phone-base schemes considered by Bonneau et al. [1] it is granted a *Quasi-Nothing-to-Carry* benefit. It is assumed that a majority of users will possess and carry a mobile handset. So, while a secondary device is required, there is no new, special-purpose hardware involved in the login process. Android Security Keys are not *Physically-Effortless*, as users still input textual passwords. The scheme is *Easy-to-Learn*, involving a single step with clear on-screen instructions, and is *Quasi-Efficient-to-Use*, as cookies can limit the frequency with which users have to input a device PIN or otherwise interact with their phones. The scheme is granted the *Infrequent-errors* benefit owing to its design, and the direction from Bonneau et al. to assume best-case implementations. Android Security Keys do not achieve *Easy-Recovery-from-Loss*. The mobile handset of a user stores a unique secret key locally. Fallback options exist, but a new device and re-enrolment are required after a loss. The developers of Android Security Keys have suggested that ideally, users adopting the scheme will already posses a USB security key to guarantee continuous, secure access [15], though this requirement might jeopardize other UDS benefits.

5.2 Deployability

Android Security Keys are *Quasi-Accessible*, as are most phone-based schemes. Bonneau et al. [1] assumed that users with low or no vision had tools available to them to support basic interaction with both a client device and phone screens. This is now the norm in UDS evaluations, and is considered sufficient to partially satisfy the criteria for the *Accessible* benefit. Enrolment and login tasks provide UX that very closely mirrors mature authentication methods, like Google's 2-Step and push-based multi-factor schemes, so I have assigned a similar benefit. I grant Android Security Keys *Negligible-Cost-per-User*, as there is no new, special-purpose hardware to be purchased and data transmission is likely to happen over a wireless network without any data fee. The scheme is not presently *Server-Compatible* or *Browser-Compatible*,

[1] An extended technical report from the University of Cambridge presents the researchers' findings in detail for all schemes evaluated [23].

	Usability								Deployability						Security										
	Memorywise-Effortless	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically-Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Mature*	*Non-Proprietary*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
Web Passwords			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Google 2-Step			○		●	○	○	○	○			●	●		○	○	●	●		●	●	●	●	●	●
USB Security Keys	○	○			●	●	●		●	○		○	●	●	●	●	●	●	●	●	●	●	●	●	●
Android Security Keys			○		●	○	●		○	○					●	●	●	●	●	●	●	●	○	●	○

● = offers the benefit; ○ = almost offers the benefit; blank = does not offer the benefit; red = disputed.

Table 1: Comparative Usability, Deployability, Security Evaluations

though there is good reason to think that it will be in future as there are provisions for it present in the WebAuthn API documentation [21], and client-side communication relies upon an extension to the established Client to Authenticator Protocol (CTAP) [24,25] used by USB keys implementing FIDO2 [14]. Android Security Keys are not *Mature*, having only recently been released. The scheme is not *Non-Proprietary*, though it is built on and around open protocols and standards.

5.3 Security

The public key cryptography employed by the Android Security Key scheme results in its being *Resilient-to-Physical-Observation*, *Resilient-to-Targeted-Impersonation*, *Resilient-to-Throttled-Guessing*, *Resilient-to-Unthrottled-Guessing*, and *Resilient-to-Internal-Observation* by design. Were the scheme to become more widely available, unique key pairs, generated on a per account basis, would grant it *Resilient-to-Leaks-from-Other-Verifiers*. The similarities between Android Security Keys and other public key-based schemes, communicating with online services using the same API, result in the scheme being granted *Resilient-to-Phishing*. Data related to both an authorization request's origin and the communication channel are signed and verified. The scheme is *Resilient-to-Theft*. Per Bonneau et al., both a device PIN and a strong textual password are assumed to protect accounts from compromise in the case of phone-based schemes [1]. The scheme is *Quasi-No-Trusted-Third-Party*. Provisions exist for a certificate authority structure to support device attestation, with attestation certificates being a crucial part of device registration and user enrolment [21]. The most common form of batch attestation provided under the USB security key scheme, and likely reflected in the implementation of Android Security Keys, results logically in only K-unlinkability (with Alaca and van Oorschot [26] claiming that an attestation certificate must be shared by 100,000 or more security keys to limit linkability). For this reason, the scheme is considered *Quasi-Unlinkable*. The scheme is clearly *Requiring-Explicit-Consent*, with users clicking through a dialogue and additionally receiving confirmation of their action via a display (directly addressing an important concern raised by Reynolds et al. [11]).

6 Discussion

The Android Security Key scheme is still in its infancy. If it were to be expanded to formally support additional services (and there is good reason to think that it might be), the scheme would be largely on par with the more established multi-factor option of USB security keys. This is the claim of the developers of Android Security Keys [15], and I do not dispute it.

I have chosen to assign a different set of benefits to Android Security Keys than Lang et al. [10] did USB security keys, but UDS evaluations are somewhat subjective. Were I evaluating

USB security keys, I would likely be more critical of their performance. Prior work has suggested that these keys might offer additional usability benefits, permitting users confident in the security of a cryptographic second factor to safely reuse passwords or reduce password complexity [10]. I contend that the benefit of *Resilient-to-Theft* could not be granted to USB security keys in the absence of strong, unique textual passwords. Similarly, more recent research focused on the security of USB-based multi-factor implementations has highlighted issues related to their purported *unlinkablity* [12, 26]. I do not believe that any scheme relying on the WebAuthn API [21], supporting the methods of attestation described in its documentation, can be considered wholly *No-Trusted-Third-Party* or *unlinkable*. Taking these factors into consideration, the major difference between Android Security Keys and the USB-based scheme becomes that, while Android Security Keys provide the convenience of carrying no special-purpose authenticator, USB security keys simplify user interaction, improving the efficiency of login tasks. Users get public key-based multi-factor authentication, using a device already in their possession, but have to contend with lock screens and navigate a phone's UI as opposed to simply making contact with a capacitive sensor.

The security benefits delivered by each of these schemes appear similar when a basic set of UDS benefits are considered. However, a less generous model, like the security focused one adopted by Jacomme and Kremer [12] reveals some differences. Android Security Keys are well positioned to address some of the calls for improvement and further investigation stemming from the work of these researchers. In particular, the Android Security Key scheme no longer physically connects an authenticator to a client machine (mounting it as an input device). This is a positive step. Attacking the scheme in effect requires the compromise of two distinct devices, rather than merely the compromise of a client and control of its I/O. Google's Cloud-assisted Bluetooth Low Energy (caBLE) extension [25] has still yet to be ratified, and there is little publicly available documentation, but it appears to provide for secure, "pairingless" Bluetooth communication between clients and Android Security Keys [15]. This is could represent a meaningful improvement over past schemes. Current versions of CTAP provide for security keys capable of communicating with a client machine via Bluetooth, but require pairing, even while the protocol's documentation acknowledges that Bluetooth represents a poor indicator of proximity and can introduce its own set of security concerns (notably a default level of system-wide access introducing challenges parallel to those associate with plug-in USB security keys) [24]. Removing the need to pair devices, and enhancing proximity checking and other forms of verification via secure network communication are important steps toward improving security outcomes, though caBLE will require study when documentation becomes available, and formal verification of its purported benefits by security researchers is recommended.

Without further study and formal verification, the possible improvements represented by Android Security Keys can't be confirmed, but the possibility of progress related to security goals, while at the same time addressing convenience, makes this scheme a form of multi-factor worth looking at more closely in future.

Acknowledgements

I want to thank this paper's reviewers and readers for their time, and for their thoughtful feedback. Additionally, I thank my supervisor at Dalhousie University, Dr. Srinivas Sampalli; faculty members, Dr. Kirstie Hawkey, and Dr. Raghav Sampangi; and my colleague, Mir Masood Ali.

Final and particular thanks are owed to my wife, Tiina Johns, and my small son, Finlay, for giving me the time and opportunity to conduct research and share my findings far from home.

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 553–567.

[2] C. Herley, P. C. van Oorschot, and A. S. Patrick, "Passwords: If we're so smart, why are we still using them?" in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5628 LNCS, pp. 230–237, iSSN: 03029743.

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

[4] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, ""I Added '!' at the End to Make It Secure": Observing Password Creation in the Lab," 2015, pp. 123–140. [Online]. Available: https://www.usenix.org/conference/soups2015/proceedings/presentation/ur

[5] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's Go in for a Closer Look: Observing Passwords in Their Natural Habitat," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2017, pp. 295–310, series Title: CCS '17. [Online]. Available: http://doi.acm.org/10.1145/3133956.3133973

[6] R. Morris and K. Thompson, "Password security: A case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359172

[7] E. M. Redmiles, E. Liu, and M. L. Mazurek, "You want me to do what? a design study of two-factor authentication messages," in *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/soups2017/workshop-program/way2017/redmiles

[8] M. Fagan, Y. Albayram, M. M. H. Khan, and R. Buck, "An investigation into users' considerations towards using password managers," *Human-centric Computing and Information Sciences*, vol. 7, no. 1, 2017, publisher: Springer Berlin Heidelberg.

[9] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel, "Better managed than memorized? studying the impact of managers on password strength and reuse," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 203–220. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lyastani

[10] J. Lang, A. Czeskis, D. Balfanz, M. Schilder, and S. Srinivas, "Security Keys: Practical Cryptographic Second Factors for the Modern Web," in *Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, Feb. 2016, pp. 422–440. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-54970-4_25

[11] J. Reynolds, T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons, "A Tale of Two Studies: The Best and Worst of YubiKey Usability," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 872–888.

[12] C. Jacomme and S. Kremer, "An extensive formal analysis of multi-factor authentication protocols," *Proceedings - IEEE Computer Security Foundations Symposium*, vol. 2018-July, pp. 1–15, 2018.

[13] "Universal 2nd Factor (U2f) Overview," Tech. Rep., Apr. 2017. [Online]. Available: https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.pdf

[14] "FIDO2: Moving the World Beyond Passwords using WebAuthn & CTAP." [Online]. Available: https://fidoalliance.org/fido2/

[15] C. Brand and S. Karra, "The future of security keys: Using your phone in the fight against phishing." [Online]. Available: https://cloud.withgoogle.com/next/sf/sessions?session=SEC200

[16] A. Birgisson and C. Brand, "The ultimate account security is now in your pocket." [Online]. Available: https://www.blog.google/technology/safety-security/your-android-phone-is-a-security-key/

[17] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital Identity Guidelines," Tech. Rep., Jun. 2017. [Online]. Available: https://www.nist.gov/publications/digital-identity-guidelines

[18] M. Fagan and M. M. H. Khan, "Why do they do what they do?: A study of what motivates users to (not) follow computer security advice," in *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, 2016, pp. 59–75. [Online]. Available: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/fagan

[19] "Google 2-Step Verification." [Online]. Available: https://www.google.com/landing/2step/

[20] "Enforce uniform mfa to company-owned resources." [Online]. Available: https://cloud.google.com/identity/solutions/enforce-mfa#google_prompt

[21] D. Balfanz, A. Czeskis, J. Hodges, J. Jones, M. B. Jones, A. Kumar, A. Liao, R. Lindemann, and E. Lundberg, "Web Authentication: An API for accessing Public Key Credentials." [Online]. Available: https://www.w3.org/TR/webauthn/

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

[23] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-817, Mar. 2012. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf

[24] "Client to Authenticator Protocol (CTAP)," Tech. Rep., Jan. 2019. [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html

[25] E. Protalinski, "You can now use your android phone as a 2fa security key for google accounts." [Online]. Available: https://venturebeat.com/2019/04/10/you-can-now-use-your-android-phone-as-a-2fa-security-key-for-google-accounts/

[26] F. Alaca and P. C. van Oorschot, "Comparative Analysis and Framework Evaluating Web Single Sign-On Systems," *arXiv:1805.00094 [cs]*, Apr. 2018, arXiv: 1805.00094. [Online]. Available: http://arxiv.org/abs/1805.00094