An Evaluation of the Power Consumption of Coauthentication as a Continuous User Authentication Method in Mobile Systems

Brandon Corn
Columbus State University
Columbus, GA, USA
corn_brandon@columbusstate.edu

Alfredo J. Perez
Columbus State University
Columbus, GA, USA
perez_alfredo@columbusstate.edu

Ashley Ruiz
Universidad Ana G. Mendez
Gurabo, PR, USA
aruiz139@email.suagm.edu

Cagri Cetin
University of South Florida
Tampa, FL, USA
cagricetin@mail.usf.edu

Jay Ligatti
University of South Florida
Tampa, FL, USA
ligatti@usf.edu

ABSTRACT

Methods for continuous user authentication have become important with the proliferation of mobile devices in m-Health and human-centered systems. These methods must guarantee user identity with high assurance, authenticate without explicit intervention, and be power-aware. We present an evaluation of the power consumption of collaborative authentication (coauthentication) as a continuous authentication method. Coauthentication is a single-factor method in which multiple registered devices work together to authenticate a user, minimizing obtrusiveness while providing high user authentication assurance. To evaluate coauthentication's power consumption, we conducted experiments using two Bluetooth-enabled mobile devices and a stand-alone server in a local area network and running coauthentication continuously for eight hours. We found that the protocol uses approximately between 1.19% and 4.0% of the total power used by the devices. These results give evidence of the feasibility of using coauthentication as a continuous authentication method in mobile devices from the power consumption perspective.

CCS CONCEPTS

• **Security and privacy → Authentication**; • **Human-centered computing → Ubiquitous and mobile computing**.

KEYWORDS

Authentication, Continuous authentication, Mobile systems, Power consumption

ACM Reference Format:
Brandon Corn, Alfredo J. Perez, Ashley Ruiz, Cagri Cetin, and Jay Ligatti. 2020. An Evaluation of the Power Consumption of Coauthentication as a Continuous User Authentication Method in Mobile Systems. In *2020 ACM Southeast Conference (ACMSE 2020), April 2–4, 2020, Tampa, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3374135.3385304

ACMSE 2020, April 2–4, 2020, Tampa, FL, USA

ACM ISBN 978-1-4503-7105-6/20/03…$15.00
https://doi.org/10.1145/3374135.3385304

1 INTRODUCTION

A Continuous Authentication (CA) method is a security mechanism that monitors user's actions at a high frequency (in time) and determines if a user is an authorized one [5]. CA methods are advantageous over one-time mechanisms because the latter may allow non-authenticated users to access protected resources if the initial user does not properly log out of a session [9], or if the system that makes use of the one-time authentication method does not log out (or re-authenticates) the initial user during a session after a long time. CAs are important for systems requiring assurance that long tasks are conducted by an authorized user, which may be the case of m-Health applications, for example, on the automated medication delivery (dosage) on a patient outside a hospital [11].

Since CAs authenticate users continuously, these methods must be reliable, user-friendly and power-efficient (especially in battery-powered devices) [10]. In this work, we present an evaluation from the power consumption perspective of collaborative authentication (coauthentication) [8] when used as a CA method in mobile phones. Coauthentication makes use of multiple devices which collaborate to authenticate a user. The rest of this paper is organized as follows. Section 2 presents the related work. In section 3 we describe the coauthentication method. Section 4 presents the methodology and evaluation of the power consumption of coauthentication. In section 5 we provide conclusions and future work.

2 RELATED WORK

Since the first works on CA methods, they have been based on something that the user is, which is, the utilization of biometrics (or patterns) obtained or inferred from the user when using a system. According to Gonzalez-Manzano et al. [5], the first contributions to CAs date back to mid 90's with keystroke analysis for PCs used to continuously monitor a user [13]. Since then, multiple biometric CAs techniques have been researched based on face recognition, gait analysis, touch dynamics, keystroke dynamics, user behavior and sound for both PCs and mobile environments [7, 14].

Since biometric-based methods use machine learning techniques to learn patterns to authenticate a user, they lead to issues such as the collection of data for model training, the accuracy of these methods to authenticate a user, and the possibility of active attacks through adversarial machine learning [6], making them not usable

for certain classes of applications, or they may generate security gaps.

In addition to the security and usability issues, a third factor usually ignored when evaluating CA methods is power consumption. Whereas power is assumed to be negligible in PCs, in mobile environments (especially for battery-powered devices such as mobile phones and wearables), power is critical as it impacts the utilization of the device [12]. In the past, it has been shown that the utilization of a mobile phone's camera (in the activation of the camera sensor only) can deplete a cell phone battery's in as little as 3.5 hours when continuously used [2]. Finally, the continuous utilization of Machine Learning (ML) algorithms in mobile devices as traditionally used has a negative impact in battery life when floating-point data representation and operations are used (usually to extract features needed for ML methods to work) [1]. Thus, biometric-based CAs may generate significant power overhead in mobile devices.

3 COAUTHENTICATION

Collaborative authentication (coauthentication) [8] is a single-factor user authentication technique in which multiple registered devices work together to authenticate a user. Coauthentication does not need passwords or any type of biometrics to authenticate a user, yet it provides benefits against phishing prevention, replay, man-in-the middle, device misplacement and denial-of-service attacks. Coauthentication can be implemented through different protocols. However, independently of its implementation, the method requires the following type of devices:

- *Authenticator:* This device is typically a server that decides to authenticate a user.
- *Requestor:* This device starts the authentication process.
- *Collaborators:* These devices are registered with the authenticator, and they are needed to execute the coauthentication protocol. At least one collaborator device is needed.

The full coathentication protocol for a requestor and one collaborator device is shown in Figure 1. Initially, requestor and collaborator devices register with the authenticator to establish authentication secrets which are shared symmetric cryptographic keys. For the authenticator A and requestor R, they share a secret key K_{AR}, and the authenticator A and collaborator C share a secret key K_{AC}. In this scenario, the protocol works as follows:

(1) Requestor R initiates the coauthentication method by sending the authenticator A its ID and an encrypted authentication-request message containing a challenge nonce N_1 (which serves to authenticate A to R).
(2) Authenticator A receives and decrypts the request message, finds that the requestor R is registered to a user having collaborating device C, creates a challenge nonce N_2 (which serves to authenticate R to A), generates two new keys ($\widehat{K_{AR}}$ and $\widehat{\widehat{K_{AR}}}$) to share with R (to rotate keys, to ensure forward secrecy and prevent key-duplication attacks), and double encrypts these data in a collaboration-request message to C, the first (inner) encryption using K_{AR} and the second (outer) encryption using K_{AC}. By double encrypting nonce N_2, the authenticator ensures participation of both user devices' secret keys (K_{AR} and K_{AC}) in the coauthentication.
(3) Collaborator C receives and decrypts the previous message, verifies the identity of the requestor, and forwards the decrypted message (which is still ciphertext encrypted with K_{AR}) to requestor R through a private channel.
(4) Requestor R receives and decrypts this message using K_{AR}, verifies the identity of the collaborator, and obtains N_2, $\widehat{K_{AR}}$, and $\widehat{\widehat{K_{AR}}}$. The requestor then generates and sends the authenticator a collaboration-response message containing N_2 encrypted with its first updated key, $\widehat{K_{AR}}$. The requestor saves the second updated key, $\widehat{\widehat{K_{AR}}}$, for a future coauthentication request.
(5) Authenticator A receives the collaboration-response message, decrypts, and verifies the collaborator's identity and that the received nonce matches the N_2 it sent earlier. Because A has now verified participation of both keys K_{AR} and K_{AC}, it sends an authentication-complete message, for example containing a session key K_{SK}, to the requestor R.
(6) Requestor R sends an acknowledgment to the authenticator.

In figure 1, message 3 is sent from the collaborator to the requestor using a private channel (i.e, a Bluetooth connection). Every other message is sent via a public (shared) channel (i.e., WiFi, 5G, SMS). When used as a CA method, the requestor performs the above steps continuously as needed without user intervention (a zero-interaction system [3]). Past research has demonstrated coauthentication as a secure and usable user authentication method [8, 15].

4 METHODOLOGY

4.1 Experimental Testbed

The testbed that we used to measure the power consumption of coauthentication is composed of the following devices:

- A WiFi access point to setup a Wireless Local Area Network (WLAN) for the devices.
- Two Huawei MateSE smartphones which served as requestor and collaborator running Android 8.0.
- A laptop that served as the authenticator. This laptop ran Windows 10.

We used Battery Historian [4] which is a tool developed by Google that generates data about power-related events on an Android mobile phone. The tool works on the output of log reports collected via the *dumpsys* command-line utility that is part of the Android SDK tools. Battery Historian gives statistics about the following aspects of the device: history of battery-related events; device's global statistics; approximate power use per UID and system component; per-app mobile ms per packet; system UID aggregated statistics; and app UID aggregated statistics (among others). Battery Historian works by first using the *dumpsys* tool to obtain a log, then uploading the log to the web-based app tool to analyse the behavior of the app. In our experiments, we collected the logs for the collaborator and the requestor devices.

The smartphones ran an Android application (Figure 2) that implemented the coauthentication protocol, in particular the requestor and collaborator components of the protocol as described in section 3 (Figure 1). The laptop executed the authenticator components of the protocol. We instrumented the Android app to execute the full coauthentication protocol continuously for eight hours

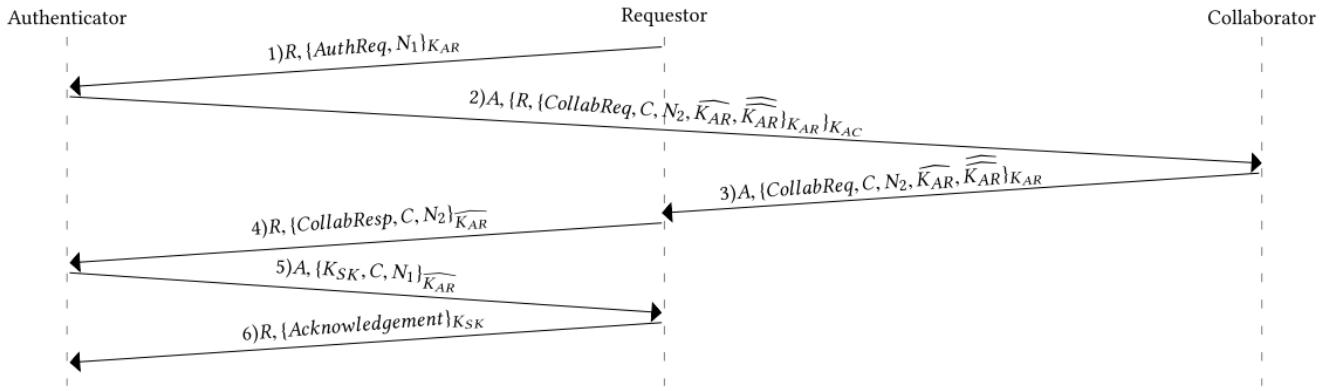

Figure 1: The Full Coauthentication Protocol [8]

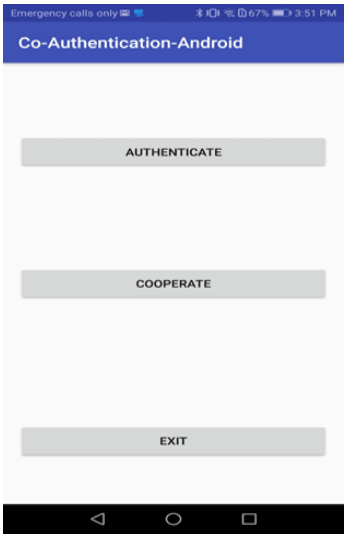

Figure 2: An Android Screenshot of App used for Experiments

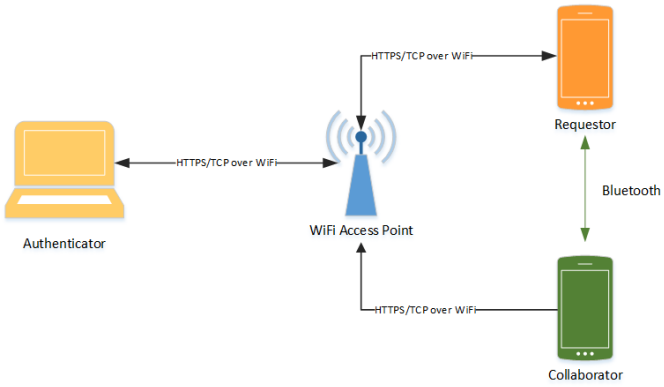

Figure 3: Experimental Testbed

(to assume a normal working day in which a device may not be charged), issuing 9,675 requests approximately (1 request every three seconds approximately). All the software was implemented in Java. The only human intervention during the execution of the experiments was the initial pressing of the *Authenticate* button for the requestor device, and the *Cooperate* button for the collaborator device.

The requestor and collaborator devices were connected via Bluetooth to exchange messages. The authenticator, requestor and collaborator were connected to the same WLAN via WiFi, with messages between them exchanged using TCP and HTTPS connections (Figure 3). This scenario recreates when coauthentication may be used for m-Health applications in a closed environment (i.e., a home, a retirement facility, hospital). Based on the WiFi protocol specification, when used in infrastructure mode (i.e., with base station as intermediary), all data is forwarded to the base station, then broadcast to its receiver. Every time the implementation was executed new TCP, HTTPS and Bluetooth connections were open between the devices. However, the collaborator and requestor were left paired.

We used the standard *javax.crypto* library with symmetric cryptographic operations implemented with 256-bit Cipher Block Chaining (CBC) mode Advanced Encryption Standard (256-bit CBC-mode AES). All nonces, session keys and updated K_{AR} keys were dynamically generated using Java's SecureRandom class, with sizes 64-bit (nonces), 256-bit (session keys) and 256-bit (K_{AR}) respectively. The initial keys were assumed to be shared before the execution of the protocol and they were hard-coded.

Before conducting the experiments both devices were charged to 100% battery capacity so that if the applications were to deplete the power, we could observe how long the application ran before the devices' battery was without charge. Before the apps were run, we reset the battery stats and cleared the *logs* of both devices so that when analyzing the log reports, all the statistics pertained to the experiment's 8 hours data. Only essential background services were in execution while the experiments were performed. During the eight hours that the experiments were conducted, the app at both mobile phones was in the foreground (i.e., app always in the screen).

Table 1: Summary of Measurements Collected via Battery Historian for the Coauthentication Android App

Measure	Requestor	Collaborator
Est. power use (pct)	3.53%	1.66%
Est. power use due to CPU (pct)	0.00%	0.00%
Total WiFi data transferred	4.42 MB	3.45MB
WiFi data received	1.95MB	3.06MB
WiFi data transmitted	2.48MB	401KB
Bluetooth power use	0.03%	0.06%

4.2 Results

We report the results of the execution of one of the experiments in Table 1. The app was executed in both phones for eight consecutive hours with minimum power consumption for both the requestor and collaborator devices. At the end of the experiments, the device's batteries were discharged at 45% (requestor) and 43% (collaborator) approximately, but this was due to how the coauthentication app was implemented: the app was always in the foreground (screen was always on). In Battery Historian, the power consumed due to screen usage is reported differently than the stats presented in Table 1, thus the table presents the actual power usage by the protocol itself. We ran this experiment eight times with minimum power consumed by the collaborator at 1.19% and maximum power consumed by the requestor at 4%. The power consumed was consistent with the results reported Table 1.

Most of the power consumed in the requestor was spent on transmitting WiFi data. This can be seen from the protocol's design as the requestor sends three messages to the authenticator using the WiFi network interface while the collaborator does not send any app messages (only TCP/IP control messages are sent which explains the low quantity of WiFi data transmitted). The collaborator device spent power transmitting Bluetooth data. Being developed for Personal Area Networks (PANs), Bluetooth has been optimized for low-power consumption. It worth also mentioning that due to the utilization of symmetric cryptographic keys in the protocol, the power consumption due to CPU usage is negligible.

If coauthentication is used in a m-Health system, the collaborator (or collaborators) would be sensor or devices located in the body of a person or in close proximity (to minimize power in sending data via a PAN), and the requestor may be a mobile phone or a medical device with reliable power that collects data from the sensors. In this scenario, coauthentication can be a great protocol for CA due to its simplicity, usability and low-power consumption.

4.3 Limitations

In our experiments we evaluated the power consumption of coauthentication in a WLAN. When used with cellular networks, even though the amount of data transmitted may not significantly change, the utilization of the cellular interface may increase the power consumption of the protocol depending on the geographical location from where the requestor and collaborator devices implementing/using the protocol are transmitting to the authenticator via a cellular base station: the farther away a device is from a cellular base station (i.e, antenna), more transmission power the mobile devices use because they increase the power in their antenna to reach the base station.

5 CONCLUSION AND FUTURE WORK

We have experimentally evaluated the power consumption of coauthentication when used as a continuous authentication protocol. We found that the full coauthentication protocol consumes negligible amount of power when used continuously. Since coauthentication has been shown to be a reliable and usable protocol, we conclude that given its low-power consumption, coauthentication is a suitable protocol for CA. As future work we plan to evaluate the power consumption of variations in implementations of the protocol [8], as well as evaluating the protocol when used with cellular networks.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the US National Science Foundation and the US Department of Defense under grant award 1560214, and by the US National Science Foundation under grant award 1950416.

REFERENCES

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. Reyes-Ortiz. 2013. Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic. *J. UCS* 19, 9 (2013), 1295–1314.
[2] F. Ben Abdesslem, A. Phillips, and T. Henderson. 2009. Less is More: Energy-efficient Mobile Sensing with SenseLess. In *Proc. 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, Barcelona, Spain, 61–62.
[3] M. Corner and B. Noble. 2002. Zero-Interaction Authentication. In *Proc. 8th Annual International Conference on Mobile Computing and Networking (MobiCom '02)*. ACM, New York, NY, USA, 1–11.
[4] Android Developer. 2019. Profile Battery Usage with Batterystats and Battery Historian.
[5] L. Gonzalez-Manzano, J. De Fuentes, and A. Ribagorda. 2019. Leveraging User-related Internet of Things for Continuous Authentication: A Survey. *ACM Comput. Surv.* 52, 3, Article 53 (June 2019), 38 pages.
[6] L. Huang, A. Joseph, B. Nelson, B. Rubinstein, and J. Tygar. 2011. Adversarial Machine Learning. In *Proc. 4th ACM Workshop on Security and Artificial Intelligence (AISec '11)*. ACM, New York, NY, USA, 43–58.
[7] A. Klosterman and G. Ganger. 2000. *Secure Continuous Biometric-enhanced Authentication.* Technical Report. Carnegie-Mellon University Dept. of Computer Science.
[8] J. Ligatti, C. Cetin, S. Engram, J. Subils, and D. Goldgof. 2019. Coauthentication. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*. ACM, New York, NY, USA, 1906–1915.
[9] K. Niinuma, U. Park, and A. Jain. 2010. Soft Biometric Traits for Continuous User Authentication. *IEEE Trans. Info. For. Sec.* 5, 4 (Dec. 2010), 771–780.
[10] A. Perez, S. Zeadally, and N. Jabeur. 2017. Investigating Security for Ubiquitous Sensor Networks. *Procedia Computer Science* 109 (2017), 737.
[11] A. Perez, S. Zeadally, and N. Jabeur. 2018. Security and Privacy in Ubiquitous Sensor Networks. *Journal of Information Processing Systems* 14, 2 (2018).
[12] B. Priyantha, D. Lymberopoulos, and J. Liu. 2011. Littlerock: Enabling Energy-efficient Continuous Sensing on Mobile Phones. *IEEE Pervasive Computing* 10, 2 (2011), 12–15.
[13] S. Shepherd. 1995. Continuous Authentication by Analysis of Keyboard Typing Characteristics. In *European Convention on Security and Detection, 1995.* IET, Brighton, UK, 111–114.
[14] T. Sim, S. Zhang, R. Janakiraman, and S. Kumar. 2007. Continuous Verification Using Multimodal Biometrics. *IEEE Trans. Pattern Analysis and Machine Intelligence* 29, 4 (April 2007), 687–700.
[15] J. Subils, J. Perez, P. Liu, S. Engram, C. Cetin, D. Goldgof, N. Ebner, D. Oliveira, and J. Ligatti. 2019. A Dual-Task Interference Game-Based Experimental Framework for Comparing the Usability of Authentication Methods. In *2019 12th International Conference on Human System Interaction (HSI)*. IEEE, Richmond, VA, USA, 95–100.