

JBCA: Designing an adaptative continuous authentication architecture

Javier Junquera-Sánchez¹, Carlos Cilleruelo-Rodríguez¹, Luis de-Marcos¹, and
José Javier Martínez-Herráiz¹

Dpto. Ciencias de la Computación. Universidad de Alcalá

Abstract. Several continuous authentication methods provide very accurate ways of detecting the impersonation of a user. However, even though they should be access control chain's links, they are commonly documented as algorithms unrelated to any system. In this paper, we propose a continuous authentication architecture to integrate several kinds of continuous authentication engines in a complete framework. The architecture covers different use cases where continuous authentication methods, placed in the user's device or in the cloud, collaborate to determine if the user's identity has changed. We use ensemble learning principles to combine the values generated by each CA model, and compute the trust level adaptively, addressing common errors caused by changes in the environments of continuous authentication systems.

Keywords: continuous authentication, ensemble learning, it architectures, authorization

1 Introduction

Nowadays mobile phones have become a critical asset for information security. There is a lot of personal information inside each mobile device and it can be the key to access to other systems (e.g. they are commonly used as a second factor for authentication). Furthermore, mobile devices can be carried everywhere, in contrast with other company devices that can improve their security through physical security policies, like video surveillance. And they are very lightweight, so a robber who steals one and transport it easily.

One way to offer better security to mobile devices is Continuous Authentication (CA). CA is the set of techniques and procedures destined to verify the users' identity after their authentication, and without their active participation. Most of these techniques are based on capturing behavioral metrics, or biometrics which not require an active implication. It is useful for situations where a user forgets unlocked his device somewhere (e.g. during a work break) and an adversary uses it without his consent.

CA can use more than one approach to create better detection results, especially taking into account that this CA process could happen in several different situations where some models could work better than others (e.g. photography could be a very precise approach, but in some poor visibility conditions, could

be better to use a keystroke dynamics approach). As a result of this, there exist some initiatives to orchestrate or process multiple inputs to provide CA. However, most of them do not cover the data retrieving phase or lack the same problem: all the metrics have to be stored in the same place.

Besides collecting and storing all this behavioral user information arises privacy concerns. On one hand, because the information needed to run the system is sensible (e.g. biometrics). And on the other hand, because the model itself could be dangerous, as it could allow detecting the user's identity without his consent.

During our process researching CA systems and models, we design an architecture for retrieving and combining different kinds of CA metrics. To combine different decision models we implemented a voting system and the logic for delivering the results to an alert system (e.g. like a SIEM). Our architecture covers these three phases:

- Guidelines proposal for collecting, storing, and generating CA trust values from behavioral metrics
- Retrieving and combining partial trusts
- Delivering final trust alerts

The present work aims to provide a complete architecture for a CA system. The goals are to cover the complete data flow, from metrics to trust values, allowing the combination of results from several CA decision models, but also being useful the simplest possible model (i.e. a model with just one CA provider). This paper contributes to knowledge by presenting the first holistic EDR architecture for CA.

2 Background

2.1 Continuous authentication

Continuous authentication is the phase of an access control system which ensures the identity of the user remains the same that was in authentication process. Nowadays, there are several CA approaches with high detection accuracy, but none of them addressed the full architecture of a CA system. Almost every approach can be adapted to different systems (e.g. mouse pointer dynamics could have particularities if the user uses a touchpad, instead of a mouse [19]) or processed with different algorithms.

Traditionally, keystroke dynamics is one of the most primitive approaches [24], as it could be implemented mostly everywhere (almost all computers have had a keyboard). Many different features from keyboard writing sessions have been studied including writing speed, time between keystrokes or time pressing each key.[22]

In mobile devices, the most significant evolutive step is touchscreen dynamics[13]. The touchscreen is the hegemonic peripheric device of smartphones like the keyboard is of the computers. The way users scroll a text provides lots

of data[25]. Also, what is already known about keystroke dynamics' can be combined with the touchscreen to detect how the user writes in a virtual keyboard[14].

Finally, the HMOG research (based on Hand Movement, Orientation, and Grasp) condenses both models and enriches them with other phone-specific sensors (e.g. accelerometer) that can lead to knowing the specific circumstances in which the user is using it: lying down, running, sitting, etc. A lot of sensors can provide data to infer them, like light sensors[5], position sensors[16] or microphone[6].

2.2 Data analysis

Each CA system needs to evaluate the data received. There are multiple methods for processing the information retrieved from each source[20].

Several of them are almost trivial when the features have been correctly extracted. The approaches which better illustrate the key point of a powerful CA system are those based on image recognition[27]. The process for extracting the color of the user's eyes can be difficult and entails several complex algorithms; but when done, determining if it is the user's eye color is as trivial as comparing two values[12].

However, other features easier to collect, are more difficult to evaluate. The most used algorithm for processing time-based metrics is SVM[3], as these kinds of metrics often fit well in regression models. For approaches related to tracking the user's behavior (i.e. based on the change of states of the system), the most effective algorithms are those suited for classification problems[17] (like k-nearest-neighbors method).

Also, the camera can be used as a rather effective input for a CA system. Approaches such as detecting the color of the user's clothes can very accurate during a working session. For instance, if the t-shirt color changes, another person could be using the mobile phone. CA should also consider the frequency of check or how often the system should try to identify the user. Moreover, another important parameter that must be defined, is how much time should session be hold (i.e. during a workday, a week, etc).

According to [28], the main dimensions that define the validity of a CA model are universality, distinctiveness, permanence and collectability. These dimensions define, for example, whether a model is valid for a long time, just for a working session; or if it fits for any person (e.g. a hair colour based model requires all the population to have hair).

Hence, the most important action to create a useful CA system is to find which features provide a most versatile and reliable way for evaluating the user's identity.

2.3 Multi-feature approaches

Not all this data can be processed inside mobile devices, or in devices with limited capacity (for example, machine learning could be an demanding operation), and in most cases, it has to be processed outside of the users' device.

As HMOG combines multiple features for determining if the user's identity remains to be the genuine, other researches have tried to get better performance processing several features at the same time [4].

Some of this research starts from realizing that, even though the fusion of multiple features could deliver better results, not all of them have the same weight. The framework proposed in [18] implements a decision fusion system to combine several phone metrics with touchscreen gesture dynamics, in order to evaluate each user's gesture in a specific context.

There are also approaches that, instead of combining multiple features' results, process the same feature with different AI models. Combining them to obtain a more accurate evaluation is known as ensemble learning [9][23]. Our CA proposal uses ensemble learning as a method of combining several decision models.

2.4 Access control

The authentication of the different components of the architecture should accomplish security, but also usability and scalability. It is necessary to understand the nature of the different architecture's actors.

OAuth2 [15] is an authorization protocol. This protocol allows the user to delegate its identity into a client, and let him interact with the resources server as it was the user itself.

There are multiple actors with different interests over the system. These actors correspond to the three components of an OAuth2 system: resource server (our server), clients (other service providers), and users (the final users).

The most common use case of OAuth2 is a process where a social network asks us if we grant access to some resources (like knowing our identity) to a third-party service, in order to login to this third party with the social network credentials. The background of this process is the `authorization_code` mode of OAuth2.

OAuth2 standard has several authorization modes [1], some of them deprecated. In order to understand our CA system architecture is it necessary to understand the following:

- Authorization code: The most common OAuth2 system. The user is redirected to the third party login page (in this case, to our's login page, from a third party application) and grants access to his data.
- Device code: Similar to authorization code, but implemented for devices without the capability of loading a web page.
- Client credentials: Used to authenticate clients instead of users. It is useful when clients should make use of the resource server for anything else than act as the user.

The OAuth2 standard has been largely studied, and the fruit of these studies has suffered several updates. As a result, the most secure configuration of OAuth2 has derived in a kind of new standard with some constraints concerning the original version [21].

While authorization code mode would allow both third party and self-programmed systems (respectively) to interact with a server, client credentials mode would also allow identifying clients (servers or systems where the user has no participation) in order to let them interact with our API when necessary.

OpenID Connect is a protocol constructed over OAuth2 to standardize also the process for retrieving user's data: username, mail, etc.

Also, the selected way for the identity is JWT as an access token, and a string refresh token (to allow the revocation without checking always during the JWT token validity time).

3 Architecture overview

The architecture consists of three types of elements:

- Authorization server: Responsible of the authentication in the architecture.
- Resources server (JBCA API): Manages the trust obtained from different features of the users' identity (generated by the CA clients) in order to create a global trust value.
- Clients: Act as the interface between the users and the architecture. There are several types of client, documented in Section 4.

The use cases proposed in Section 8 aims to provide a comprehensive example of the role that these components play in the architecture (also shown in Figure 2).

4 Clients

As the clients would be not just the source of information about the users, but also the consumers of the architecture's products, it is necessary to start with them for explaining the architecture. There would be three types of client.

4.1 CA Clients

This category defines the clients which can generate trust values. Independently how they work, they would have their own CA processes to calculate the trustworthiness of each user identity, and send the results to the resources server. These types of clients should have a backend for protecting their own authentication secrets (i.e. the keys for authenticating themselves as clients against the authorization server), but they can implement their internal logic however they need it.

Nonetheless, it is necessary to mention that could be special kind of CA client which not always would have access to a backend: the offline clients.

Offline clients It is not a type of client, but a subtype of CA client. Generate models in the cloud, or storing this data out of the user's assets, could be a problem with certain types of data. This problem is especially serious when is specially protected data (like medical, biometrics, religion...) or there are cross-border connections (e.g. between an EU user to a USA server). So, sometimes, it is desirable that the user's data is not stored in a backend (e.g. very sensible data). For this purpose, we have designed what we have seen fit to call *offline client*: a client with CA capabilities, but without a backend.

Even though this could seem unnecessary (as they produce exactly the same type of data than common CA Clients), from the point of view of authorization, we have to take care of its particularities.

4.2 Trust clients

The purpose of the architecture is to generate the most precise possible value for indicating trustworthiness in a user's identity. The consumers of this value are trust clients. They would ask anytime which is the result of the evaluation process, for example, to close a session of a medical application if the user might have been impersonated.

4.3 Partner clients

This is the name given for those clients with special grants over the resources server. This type of client won't act as a specific user (i.e. won't be limited to interact with the information of the user which has granted them) but can retrieve the values generated for any user.

The aim of creating this role is not to bypass the users' grant, but allow the integration of the architecture in systems that already have their own purpose and need to provide CA capabilities to their tools.

5 Authorization server

The authorization server will be in charge of authenticating the clients and generate the tokens to authorize them to use the server resources. It will also allow the whole system to maintain the user's identity unique.

5.1 Authorization token

The authorization token would be codified using JWT, as it would provide enough information to the resource server for validating it without adding connection overload. This JWT token would be generated using the claims defined both in the JWT's RFC [7][26] and with the information that the OAuth2 introspection endpoint should provide [8].

The logic for revoking access to a client would work during the refresh process. If a user wants to revoke the rights granted to a client, the next time the client

would try to refresh his access token (using OAuth2's refresh token), would be rejected, or his grants would be modified. Even though this approach could open a time slot where the user's revocation won't be effective, this time can be limited making the token's time of life shorter.

Finally, the server should be configured to be able to provide also information about the users' identity if required.

5.2 User identity

To provide information about the user, an OpenID Connect layer will be implemented. The grants related to this functionality would be only given when they are necessary for the CA purpose (i.e. never like a social, or an identity validator feature). For example, the user's address could be necessary for a correlation with GPS data (if the authenticator makes this check), but to know the user's email address won't be necessary in this case.

The unique identifier all over the network would be placed in the subject claim (i.e. `sub`) of the JWT token, and should not apport any information about the user (would be, for example, an UUID value).

5.3 Client roles

Those clients mentioned in section 4 will have their particularities for authenticating.

- “Online” CA clients would retrieve OAuth2 tokens using the “`auth.token`” or “`device.code`” mode.
- “Offline” CA clients which have no possibility of authenticating themselves as clients, they should use the “`device.code`” mode.
- Partner clients should authenticate themselves using the `client_credentials` mode, as they will not directly interact with users.
- Trust clients will not need a particular authentication mode, and it will depend on the circumstances where they work (i.e. the capabilities for storing secretly a key or the type device where they work)

Each type of client should be allowed by the architecture administrators to use their authentication mode. For example, a regular CA client would be rejected if tries to get an OAuth2 token using the `client_credentials` mode.

5.4 Scope

As part of the OAuth2 protocol specification, there should be a list of defined grants, which will be set the `scope` field. The scope possibilities would be:

- `ADD_MODEL_TRUST`: Add trust of a particular model
- `READ_GLOBAL_TRUST`: Retrieve the trust generated by the combination of all partial trusts.

- OPENID and derivated: To obtain information about the user

Table 1 shows the relation between the types of clients and their scope. These are the main groups, but can also be combined (for example, an EDR could want to write its own generated trust values and read the global trust result).

Table 1. Clients' configurations summary

client type	OAuth2 modes	Scope
Online CA	<code>auth_token, device_code</code>	<code>ADD_MODEL_TRUST</code>
Offline CA	<code>device_code</code>	<code>ADD_MODEL_TRUST</code>
Partner	<code>client_credentials</code>	Specific for the partner
Trust clients	Any	<code>READ_GLOBAL_TRUST</code>

6 AI Models

Even though the architecture should be agnostic about how the clients calculate the trust values, at least it should have enough information to know how to combine them. The models should also comply with some quality requisites to be accepted in the system.

6.1 Quality

In contrast with the client, which should know how much data could be needed for training the models, or to evaluate the user's identity, the resources on server should only know about two quality parameters:

- Model's precision: It will be crucial for calculating the weight that each measure will have in the final trust value.
- Model's expiration: The time the value is valid to be part of the final trust value.

6.2 Requirements

The requirements that every registered model should meet are related to the nature of CA: rejecting intruders without interrupting the genuine user's work. So the models recall should be near 1.0 (i.e. practically total).

Also, as the strength of a CA system is to get a reliable value combining multiple input methods, so any data that contributes is valuable. Thus, the precision of the model should be the highest possible, but it would be enough with a precision greater than 0.5 (i.e. better than a random decision).

6.3 Privacy concerns

The fact that all the process take place in the user's scope does not exempt from taking care of the information's security. On the one hand, securing this data is a crucial part of protecting the user's device; on the other hand, the responsibility remains with the data processor, regardless of where it performs its tasks.

As it could be necessary a significant amount of data about the user's behavior, it is necessary to take into account the possible problems it could generate. Every new model must follow a minimal risk policy, always choosing the alternatives that could make less impact on the user's privacy.

In order to fulfill with personal data protection principles [2], the next questions should be made in the design phase. Each time a CA decision model is created:

1. Can old data (i.e. the data used to generate the model) be removed? Or would it be necessary for creating future models?
2. Is it possible to recreate the original data from the model?

Those models where old data can be removed, to avoid its exposure, will be better from the privacy point of view. But if once created, the model makes it possible to generate again the data (or similar data), there should be implemented differential privacy countermeasures [11].

The data stored to generate the models also implies a risk (its leakage is the risk itself in both cases). There are also two questions to define before implementing a solution:

1. Can the data be aggregated?
2. Can it even be encrypted?

If the model consist on determining, for example, the progression followed by its individual metrics, sometimes will not be necessary the original dataset, but one whose elements follow the same pattern. In some other cases, it would be desirable at least to save it encrypted, and decrypt only when necessary.

There are also approaches based on secure multi-party computation or homomorphic encryption [10] that can be implemented in different models, to allow compute with them without decrypting.

6.4 Ethical concerns

CA is, by definition, a process for validating the user's identity without his participation (i.e. without having to do any action). Even though this process aims to protect the user, this process could lead a malicious actor to identify the user not just without his participation, but without his knowledge or his consent.

Although the models capable of identifying the user would be more effective against intruders, those just focused on detecting when the user has changed

(i.e. who is using the device is not the genuine user) would be more compliant with the user's privacy.

Moreover, those models which are valid during a short period of time (e.g. during a work session) would also be better, as others could use metrics from which the user cannot be detached.

7 Evaluation

To calculate the final trust we propose a two-step approach. The first task would be a simple combination based on a weighted average. The second one would consist of adding feedback to re-evaluate the models' precision with based on their agreement with the global trust.

Once at least one client has submitted a partial trust for a user, the evaluation process can be triggered.

7.1 Ensemble learning

As knowing the precision of a model is a requirement for the architecture, it will determine the weight of each value in the final computation. We work with three variables: T_i (partial trust for model i), p_i (precision of model i) and r_i (reliability of model i , initialized to 1). The algorithm presented in Algorithm 1 shows how all trusts get merged.

Algorithm 1 Trust calculation

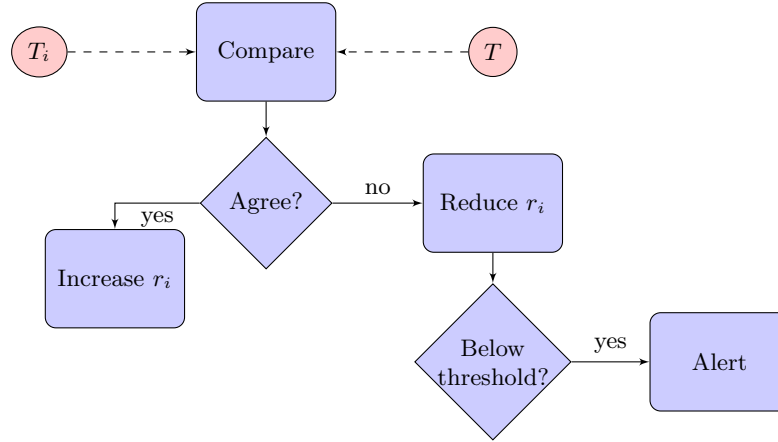
1:	$T \leftarrow 0$	{Initializing trust}
2:	$P \leftarrow \sum_0^i p_i$	{Sum of individual precisions}
1:	for $n \leftarrow 0$ to i do	{Weighted trust sum}
2:	begin	
3:	$T \leftarrow T + (T_i \times p_i \times r_i)$	{With this first $r_i = 1$ }
4:	end	
5:	$T \leftarrow T/P$	{Normalize final trust}

7.2 Adaptative model

Experiments done under very specific circumstances determined the initial values of precision for each model. The most accurate approach could fail under some conditions, and not all approaches will run on trusted environments (i.e. the local model's results are easier to be manipulated from an adversary than ones in the cloud). For this reason, the reliability of a model cannot be the same all time. For this reason, our model is designed to be adaptable using the reliability parameter.

On each evaluation process iteration, the reliability value of each model (r_i in 1 algorithm) will be updated following the decision diagram shown in Figure 1. The reason to determine if reliability is increased or decreased will not be quantitative (i.e. how near is its partial trust of the global trust), but qualitative (i.e. if agrees with the others about if the users are or not the genuine).

Fig. 1. Adaptative model



If after some iterations the rely value of a model goes below a threshold, this could have several meanings:

1. Its trust value is outdated
2. The circumstances make the model unprecise
3. Its trust value uploaded is being faked

In this case the model will be disabled until a new trust value is added or an administrator checks which has been the reason of the fall.

8 Use cases

The simplest way to illustrate the complete architecture is by observing its data flow through different use cases. In this section we document these three use cases that could be working independently:

- An “offline” model
- An EDR system
- A broker connected to the resource server sending metrics to a SIEM

At the end of the section, we will show how they can be combined in a complete CA system (i.e. like the one shown in Figure 2).

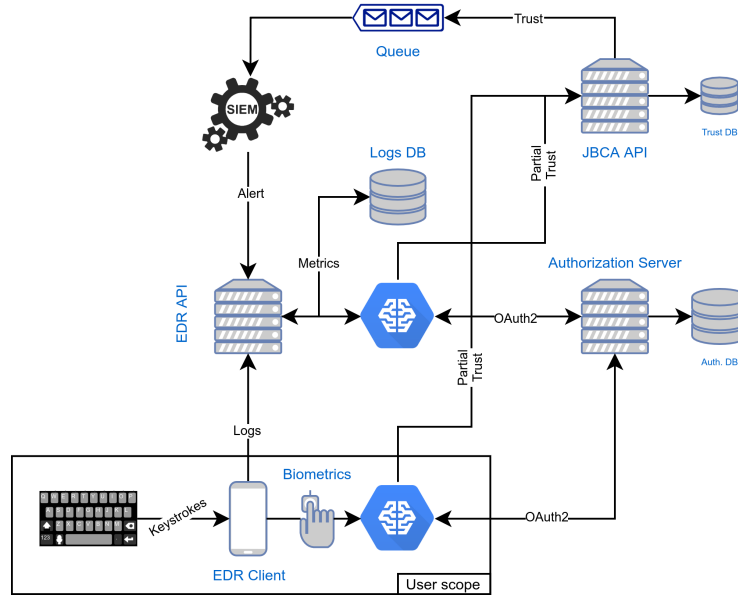


Fig. 2. Architecture overview

8.1 “Offline” use case

The “offline” models (i.e. those without a user-isolated backend) should authenticate using the OAuth2 mode **device code**. So, once the user is authenticated against the authorization server, the model would start working.

For example, let’s imagine a model based on detecting the color of the user’s clothes, and alert when it changes. The behavior would be:

1. When the user logs in, it takes a photograph, analyzes his clothes and saves locally which color they are.
2. Every minute a photograph is taken, colors are analyzed, and compared with the values saved in the first step, generating a similarity value between 0 and 1.
3. It upload the similarity result to the resource server.

8.2 EDR use case

Let’s suppose that the EDR model consists on analyzing the user’s typing dynamics in a cloud server (the backend). When the user logs in (using **authorization code** or **device code** method), the client sends to the EDR’s backend all the typing events (e.g. key press, key release, deletion rate, etc). The backend will:

1. Store all the events
2. When there are enough events, generates a user model with them

3. Every five minutes, it evaluates the new user events with the model generated in the previous step and it uploads the resulting value to the resource server

Parallely, the EDR is checking periodically alerts generated by a SIEM.

8.3 SIEM use case

The resources server would have a program that combines the trust values uploaded from both the EDR and the local system (as documented in section 7). At the same time, a partner's broker would be reading these global trust values, to feed a SIEM's message queue. The broker has to be connected to the resources server using `OAuth2 client_credentials` mode.

The broker asks to the resources server, at intervals, to calculate each user's global trust, and puts every result in a queue. The SIEM reads this queue and generates the pertinent alarms.

8.4 Joint use case

Initially a user would unlock his device and start writing. At this moment, the local model will capture his features, and the EDR's agent will start sending information to his server. Ten minutes later, there would be a model of the user in both systems, and the resources server will be providing a trust value supporting his identity.

Now, let's suppose the user puts on a jacket and continues writing. The clothes-based model will start to send values that question the users' identity, but the keyboard model will counteract the result. Later, the user forgets the device unlocked in public place and leaves. If an unauthorized user (from now on, the attacker) takes the device and tries to use it, there would be three scenarios:

1. If the attacker wears clothes similar to the user's, and he uses the keyboard, the system will detect the impersonation
2. If the attacker wears clothes different to the user's, and he uses the keyboard, the same will happen
3. If the attacker wears clothes different to the user's, and he does not use the keyboard, he may eventually be able to use the device for a period of time. But when the keyboard's model trust value expires, the only available trust values will be related with the clothes, and this will prevail and reject the attacker.

9 Conclusions and future work

The CA architecture presented in this paper is a highly integrable system that covers all the stages necessary for evaluating a user's identity. The proposed authentication system, and the central trust system, make possible the use of different types of clients running in several kinds of environments. It also allows

creating useful models without taking sensible data out of the user's device, granting the global result's integrity with an adaptative model against possible local tampering.

For future work we suggest analyzing ensemble learning methods to obtain the most accurate result. Determining the appropriate parameters for the adaptative strategy also requires further work. Finally, it is interesting to implement the architecture to continuously authenticate non-human users (e.g. validate the identity of sensors in a thermostat network).

Acknowledgements This project has received funding from the European Union's Horizon 2020 Research and Innovation Program under grant agreement No. 826284 (ProTego).

References

1. OAuth grant types. URL <https://oauth.net/2/grant-types/>
2. REGULATION (EU) 2016/ 679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL - of 27 april 2016 - on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/ 46/ EC (general data protection regulation)
3. Alshehri, A., Coenen, F., Bollegala, D.: Iterative keystroke continuous authentication: A time series based approach **32**(4), 231–243. DOI 10.1007/s13218-018-0526-z
4. Azzini, A., Marrara, S., Sassi, R., Scotti, F.: A fuzzy approach to multimodal biometric continuous authentication **7**(3), 243. DOI 10.1007/s10700-008-9034-1
5. Basar, O.E., Alptekin, G., Volaka, H.C., Isbilen, M., Incel, O.D.: Resource usage analysis of a mobile banking application using sensor-and-touchscreen-based continuous authentication **155**, 185–192. DOI 10.1016/j.procs.2019.08.028
6. Bonastre, J.F., Bimbot, F., Boe, L.J., Campbell, J.P., Reynolds, D.A., Magrin-Chagnolleau, I.: Person authentication by voice: A need for caution p. 4
7. Bradley, J., Sakimura, N., Jones, M.: JSON web token (JWT). URL <https://tools.ietf.org/html/rfc7519>
8. Campbell, B., Jones, M., Mortimore, C.: JSON web token (JWT) profile for OAuth 2.0 client authentication and authorization grants. URL <https://tools.ietf.org/html/rfc7523>
9. Casale, P.: Approximate ensemble methods for physical activity recognition applications **13**, 22–23
10. Chatterjee, A., Aung, K.M.M.: Fully Homomorphic Encryption in Real World Applications. Computer Architecture and Design Methodologies. Springer Singapore. DOI 10.1007/978-981-13-6393-1
11. Dwork, C.: Differential privacy: A survey of results. In: M. Agrawal, D. Du, Z. Duan, A. Li (eds.) Theory and Applications of Models of Computation, Lecture Notes in Computer Science, pp. 1–19. Springer. DOI 10.1007/978-3-540-79228-4_1. Event-place: Berlin, Heidelberg
12. Eberz, S., Rasmussen, K.B., Lenders, V., Martinovic, I.: Looks like eve: Exposing insider threats using eye movement biometrics **19**(1), 1:1–1:31. DOI 10.1145/2904018

13. Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D.: Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication **8**(1), 136–148. DOI 10.1109/TIFS.2012.2225048
14. Gascon, H., Uellenbeck, S., Wolf, C., Rieck, K.: Continuous authentication on mobile devices by analysis of typing motion behavior p. 13
15. Hardt, D.: The OAuth 2.0 authorization framework. URL <https://tools.ietf.org/html/rfc6749>
16. Katevas, K., Haddadi, H., Tokarchuk, L.: Poster: SensingKit: a multi-platform mobile sensing framework for large-scale experiments. In: Proceedings of the 20th annual international conference on Mobile computing and networking, MobiCom '14, pp. 375–378. Association for Computing Machinery. DOI 10.1145/2639108.2642910. Event-place: Maui, Hawaii, USA
17. Lin, F., Song, C., Zhuang, Y., Xu, W., Li, C., Ren, K.: Cardiac scan: A non-contact and continuous heart-based user authentication system. In: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17, pp. 315–328. Association for Computing Machinery. DOI 10.1145/3117811.3117839. Event-place: Snowbird, Utah, USA
18. Mahfouz, A., Mahmoud, T.M., Sharaf Eldin, A.: A behavioral biometric authentication framework on smartphones. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, pp. 923–925. Association for Computing Machinery. DOI 10.1145/3052973.3055160. Event-place: Abu Dhabi, United Arab Emirates
19. Mondal, S., Bours, P.: Continuous authentication using mouse dynamics. In: 2013 International Conference of the BIOSIG Special Interest Group (BIOSIG), pp. 1–12
20. Morales, A., Fierrez, J., Tolosana, R., Ortega-Garcia, J., Galbally, J., Gomez-Barrero, M., Anjos, A., Marcel, S.: Keystroke biometrics ongoing competition **4**, 7736–7746. DOI 10.1109/ACCESS.2016.2626718
21. Parecki, A.: OAuth 2.1. URL <https://www.ietf.org/proceedings/106/slides/slides-106-oauth-sessa-oauth-21-00.pdf>. IETF 106
22. Pisani, P.H., Lorena, A.C.: A systematic review on keystroke dynamics **19**(4), 573–587. DOI 10.1007/s13173-013-0117-7
23. Polikar, R.: Ensemble learning. In: C. Zhang, Y. Ma (eds.) Ensemble Machine Learning: Methods and Applications, pp. 1–34. Springer US. DOI 10.1007/978-1-4419-9326-7_1
24. Shepherd, S.J.: Continuous authentication by analysis of keyboard typing characteristics pp. 111–114. DOI 10.1049/cp:19950480
25. Siirtola, P., Komulainen, J., Kellokumpu, V.: Effect of context in swipe gesture-based continuous authentication on smartphones
26. Solapurkar, P.: Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario. In: 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 99–104. DOI 10.1109/IC3I.2016.7917942
27. Tsai, P., Khan, M.K., Pan, J., Liao, B.: Interactive artificial bee colony supported passive continuous authentication system **8**(2), 395–405. DOI 10.1109/JSYST.2012.2208153
28. Xu, H., Zhou, Y., Lyu, M.R.: Towards continuous and passive authentication via touch biometrics: an experimental study on smartphones. In: Proceedings of the Tenth USENIX Conference on Usable Privacy and Security, SOUPS '14, pp. 187–198. USENIX Association. Event-place: Menlo Park, CA