

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Telekomunikacji

Praca dyplomowa magisterska

na kierunku Telekomunikacja
w specjalności Teleinformatyka i Cyberbezpieczeństwo

Typeauth - badanie skuteczności biometrii pisania na klawiaturze w
kontekście języka wzorca tekstowego

Jakub Brzozowski

Numer albumu 283461

promotor

dr hab. inż. Mariusz Rawski

WARSZAWA 2022

Typeauth - badanie skuteczności biometrii pisania na klawiaturze w kontekście języka wzorca tekstowego

Streszczenie. Bezpieczeństwo użytkowników w cyberprzestrzeni oraz bezpieczeństwo ich danych odgrywa coraz większą rolę w obecnym świecie. W obliczu rosnącego zagrożenia ze strony cyberprzestępców, a szczególnie ataków socjotechnicznych, niezwykle ważny jest rozwój nowych technologii w zakresie uwierzytelnienia i identyfikacji użytkowników. Kluczową rolę w tym zakresie ma biometria, która oprócz sprzętowych kluczy **U2F**, jest skuteczną metodą zapobiegania nieautoryzowanemu dostępowi do naszych danych. Można jednak zauważyć, że wraz ze wzrostem bezpieczeństwa poszczególnych mechanizmów uwierzytelnienia, maleje ich użyteczność. Biorąc jako przykład sprzętowe klucze **U2F**, posiadanie dodatkowego urządzenia kryptograficznego, którego jedynym celem jest potwierdzenie naszej tożsamości, może nie być atrakcyjne dla każdego użytkownika. Fakt ten zdaje się potwierdzać to, że ceny takich rozwiązań nie są jeszcze na poziomie ogólnodostępnym dla każdej osoby.

Biometria zdaje się rozwiązywać ten problem, wprowadzając uwierzytelnienie na podstawie tego czym użytkownik "jest", a nie na podstawie tego co posiada. Może to być dowolna cecha charakterystyczna, zaczynając od naszego kodu DNA, na odcisku palca kończąc. Niestety nie każda cecha biometryczna jest odpowiednia do implementacji rozwiązań, z których korzystamy na co dzień. Przykładowo zastosowanie czytnika tęczówki oka w aplikacji bankowej może być problematyczne i kosztowne zarówno dla dostawcy aplikacji, jak i dla użytkownika. Dobrym kompromisem pomiędzy bezpieczeństwem, a użytecznością, jest biometria bazująca na cechach pisma na klawiaturze. Każdy użytkownik posiada unikalny dla siebie zestaw cech, który składa się na jego charakter pisma na klawiaturze komputera. Cechami tymi mogą być czasy wciśnięcia klawisza lub ilość użycia klawisza "Backspace". Bazując na tych cechach, możliwe jest uwierzytelnienie użytkownika do serwisów, które przetwarzają mniej poufne dane, takich jak witryny studenckie lub fora internetowe.

W niniejszej pracy prezentuję aplikację **Typeauth** - rozwiązanie, które posłużyło do zbadania, jaki wpływ ma język pisania, na skuteczność biometrii bazującej na cechach pisania na klawiaturze. Bazując na klasyfikatorze **SVM**, stworzono autorski system, który uwierzytelnia użytkowników na podstawie przepisywanego wzorca tekstowego. W rozwiązaniu wykorzystano aktualne i bezpieczne technologie, takie jak Flask czy Python. Następnie przeprowadzono badania z udziałem 40 uczestników, w których zbadano skuteczność systemu w zależności od języka pisania i długości wzorca tekstowego. Dla najlepszego przypadku udało się uzyskać skuteczność na poziomie około 80%. Wnioski z wykonanej pracy pozwolą na projektowanie lepszych rozwiązań biometrycznych, opisując wpływ jaki ma język wzorca tekstowego na skuteczność rozwiązań biometrycznych bazujących na cechach pisania.

Słowa kluczowe: Biometria, Uwierzytelnienie, SVM, Identyfikacja, Typeauth

Typeauth - enhancing the efficiency of keystroke biometrics in context of the typing language

Abstract. The security of users in the internet and confidentiality of their data play an increasingly important role in today's world. In view of the growing threat from cybercriminals, and especially social engineering attacks, the development of new technologies for user authentication and identification is crucial. Biometrics plays a key role in this area, which apart from U2F hardware keys, is an effective method of preventing unauthorized access to user's data. However a tendency can be observed that, as the security of authentication method increases, it's usefulness decreases. Having an additional cryptographic device for the sole purpose of identification, may not be appealing to every user, especially since the prices of such solutions are not yet affordable to every person.

Biometrics seems to solve this problem by providing authentication based on what the user "is", rather than what user "have". Biometrics can be based on any characteristic, starting with our DNA code, ending with our fingerprint. Unfortunately, not every biometric feature is suitable for the implementation with the services that we use on a daily basis. For example, the use of an eye iris reader in a banking application can be problematic and costly for both the application provider and the user. A good compromise between security and usability is keystroke biometrics. Each user has a unique set of features that make up his style of typing on a keyboard. These features are for example mean key press time or count of the "Backspace" key usage. These features can be used to authenticate user to websites that process less confidential data, such as student services or internet forums.

In this paper, I present **Typeauth** - an application that was used to study the influence of typing language on the effectiveness of a keystroke biometric authentication. The application uses the SVM classifier to create a proprietary system that identifies users based on the timing data gathered from the typing session. The solution uses state-of-art and secure technologies such as Flask or Python. A study involving 40 participants was carried out, which examined the effectiveness of the system depending on the writing language and the length of the text pattern that was rewritten by the users. In the best case scenario, an efficiency of around 80% was achieved. Conclusions from the study will allow design of improved biometric solutions, by describing the impact of the typing language on the effectiveness of biometric solutions based on writing characteristics.

Keywords: Biometrics, Authentication, SVM, Identification, Typeauth



Warszawa, 10/02/2022

.....
miejscowość i data

..... Jakub Brzozowski
.....
imię i nazwisko studenta
..... 283461
.....
numer albumu
Teleinformatyka i Cyberbezpieczeństwo
.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płytkie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

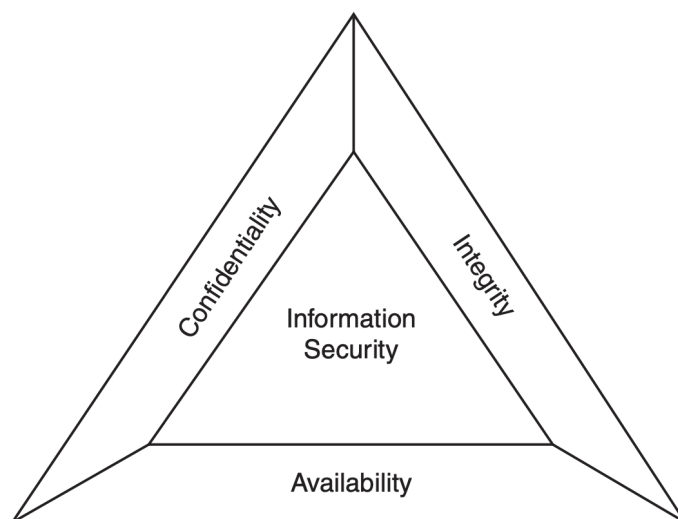
Spis treści

1. Wstęp	11
2. Cel pracy	14
3. Stan sztuki	15
3.1. Biometria	15
3.2. Parametry czasowe	15
3.3. Klasyfikator	16
3.4. Wzorzec tekstowy	18
3.5. Ocena systemów biometrycznych	18
3.6. Aktualnie rozwiązania	19
4. Koncepcja	21
4.1. Ekstrakcja danych	21
4.1.1. Typy zdarzeń	21
4.1.2. Typy cech biometrycznych	21
4.1.3. Cechy czasowe	22
4.1.4. Proces zbierania danych	22
4.2. Klasyfikacja	23
4.2.1. Klasyfikator bazujący na metryce Manhattan	23
4.2.2. Detektor SVM	23
4.3. Propozycja rozwiązania	24
5. Opis rozwiązania	28
5.1. Aplikacja internetowa	28
5.2. Stos technologiczny	28
5.2.1. Flask	29
5.2.2. Javascript	29
5.2.3. Baza danych	30
5.3. Implementacja aplikacji	31
5.4. Klasyfikator	33
5.4.1. Klasyfikator bazujący na metryce Manhattan	33
5.4.2. Klasyfikator SVM	33
5.4.3. Serwer WWW	34
5.4.4. Instalacja	36
6. Badania	37
6.1. Próba badawcza	37
6.2. Opis badania	37
6.3. Wybór klasyfikatora	38
6.4. Badania skuteczności	40
6.5. Subiektywna ocena rozwiązania	43

6.6. Sugestie użytkowników	45
7. Podsumowanie	46
7.1. Podsumowanie wyników	46
7.2. Praca na przyszłość	47
Bibliografia	49
Wykaz symboli i skrótów	53
Spis rysunków	54
Spis tabel	55
Spis załączników	55

1. Wstęp

W dzisiejszych czasach najcenniejszym zasobem są dane, a wraz ze wzrostem ich wartości, rośnie zainteresowanie nimi ze strony cyberprzestępców. W samym roku 2021, skuteczność socjotechnicznych ataków phishingowych wzrosła o prawie 150% w porównaniu do roku 2020 [1]. Aby zamodelować zagrożenie, na jakie wystawione mogą być dane użytkownika, utworzono model **CIA** (*Confidentiality, Integrity, Availability*). Według modelu CIA, bezpieczeństwo danych może zostać naruszone poprzez jeden z trzech wektorów: poufność, integralność lub dostępność (Rys. 1.1). Idąc od końca, możemy naruszyć dostępność danych przeprowadzając atak **DoS** (*Denial of Service*) lub **DDoS** (*Distributed Denial of Service*) i tym samym pozbawić innych użytkowników możliwości dostępu do pewnych zasobów. Kolejno, możemy naruszyć integralność danych, czyli dokonać w nich zmiany, którą odczują inni użytkownicy systemu. Takie naruszenie może zostać wykonane w wyniku wykorzystania niepoprawnej autoryzacji w serwisie internetowym. Ostatnim elementem modelu CIA, jest poufność. Poufność danych zazwyczaj zabezpieczana jest wykorzystując odpowiednią autoryzację, czyli ustalenie do jakich zasobów dany użytkownik ma dostęp. Jednak, aby chronić wszystkie dane w konkretnym systemie, konieczne jest wykorzystanie uwierzytelnienia.



Rysunek 1.1. Model CIA [2].

Standardowo uwierzytelnienie jest procesem, w którym, na podstawie jednego lub kilku z trzech czynników, zostaje potwierdzona tożsamość użytkownika. Czynnikami tymi zazwyczaj są to co użytkownik wie (na przykład hasło), to co użytkownik posiada (na przykład fizyczny token) lub rzadziej to czym użytkownik “jest”, czyli dowolna z jego unikalnych cech biometrycznych [3]. Uwierzytelnienie oparte na cechach biometrycznych posiada zarówno wiele zalet jak i wad, jednak unikalność gwarantowana przez cechy takie jak rysy twarzy lub układ linii papilarnych na naszych palcach znacząco przyczyniła się

do wzrostu popularności tego sposobu uwierzytelnienia na całym świecie [4]. Według [5], już w 2018 roku ponad 62% przedsiębiorstw wykorzystywało w swoich zastosowaniach biznesowych biometrię, a prawie 24% planowało wprowadzić biometrię do roku 2020. Uwierzytelnienie biometryczne niesie ze sobą takie zalety, jak niezawodność czy wygoda [6], przez co możliwe jest jego wykorzystanie w stosunku do osób niepełnosprawnych lub starszych [7]. Dzięki pozbyciu się konieczności posiadania dodatkowego urządzenia służącego za drugi składnik uwierzytelnienia, użytkownik końcowy zyskuje wygodę i pewność, że dostęp do jego danych jest odpowiednio zabezpieczony.

Sporym problemem jest jednak wybór odpowiedniego typu uwierzytelnienia biometrycznego. Poniżej przedstawiono kilka cech na podstawie których system może identyfikować i uwierzytelniać użytkowników [8]:

- układ linii papilarnych na palcach,
- kształt twarzy,
- budowa tęczówki oka,
- układ naczyń krwionośnych na dłoniach,
- barwa, tembr i inne cechy głosu,
- materiał DNA,
- cechy fizycznego podpisu.

Jednym z głównych parametrów, którymi możemy opisać rozwiązanie bazujące na biometrii, jest użyteczność oraz bezpieczeństwo [9]. Parametry te są zależne od siebie nawzajem. Dla systemów o wysokim stopniu bezpieczeństwa, takich jak skanery siatkówki oka, konieczne jest użycie wyspecjalizowanych i kosztownych detektorów. Są one gwarancją bezpieczeństwa systemu, jednak znacząco zmniejszają użyteczność oraz zwiększają koszt rozwiązania. Na drugim krańcu spektrum biometrii znajdują się systemy użyteczne, które są jednak mniej bezpieczne. Do takich rozwiązań można zaliczyć skanery twarzy czy linii papilarnych.

Jednymi z najbardziej użytecznych systemów biometrycznych są te bazujące na charakterze pisania na klawiaturze [10]. Takie systemy są wygodne, ponieważ użytkownik nie musi posiadać żadnego innego urządzenia lub detektora poza zwykłym komputerem wyposażonym w klawiaturę. Rozważmy poniższy ciąg zdarzeń, który jest przykładem typowego procesu uwierzytelnienia, gdzie drugim składnikiem jest właśnie biometria oparta na cechach charakteru pisma użytkownika:

1. użytkownik loguje się podając login i hasło,
2. aplikacja prosi użytkownika o przepisanie krótkiego tekstu, aby zebrać jego cechy pisma,
3. aplikacja analizuje cechy pisma użytkownika,
4. jeśli system potwierdził tożsamość, użytkownik jest uwierzytelniony.

W powyższym przykładzie użytkownik korzysta ciągle z tego samego urządzenia i nie musi wykonywać żadnych dodatkowych czynności poza tymi, które i tak wykonuje standardowo

przy logowaniu – korzysta z klawiatury i komputera. Jest to również metoda uwierzytelnienia, która sprzyja użytkownikom o mniejszym stopniu zaawansowania technicznego. Przepisanie frazy tekstu wyświetlanego na ekranie jest czynnością znaną każdemu użytkownikowi komputera i nawet osoba, która dopiero zaczyna swoją przygodę z technologią, będzie potrafiła jej sprostać.

Jednak użyteczność takiego rozwiązania niesie za sobą dużo mniejsze bezpieczeństwo. Przez pozbycie się dokładnych sensorów, które pozwalają na wykrywanie i klasyfikację bardzo szczegółowych cech biometrycznych, zwiększa się prawdopodobieństwo, że system nieprawidłowo zidentyfikuje użytkownika. Oprócz tego, duży wpływ kondycji użytkownika [8] na skuteczność jest jednym z głównych czynników, które powstrzymują implementacje biometrii pisaną na klawiaturze w obecnych systemach. Warto również dodać, że uwierzytelnienie bazujące na charakterze pisma na klawiaturze nie powinno być traktowane jako bezpieczny odpowiednik innych mechanizmów biometrycznych. Jest to spowodowane między innymi faktem, że uzyskanie naszych charakterystyk pisma przez osoby trzecie jest łatwiejsze, niż w przypadku innych cech biometrycznych. Złośliwy użytkownik może stworzyć złośliwą wtyczkę do przeglądarki lub wstrzyknąć kod na stronę internetową m.in. w wyniku podatności **XSS** (*Cross-Site Scripting*) i w ten sposób rejestrować nasz charakter pisma. Dlatego konieczne jest zrozumienie, że biometria bazująca na cechach pisaną jest rekomendowana jedynie dla aplikacji przetwarzających mniej poufne dane (np. serwisy studenckie lub fora internetowe).

2. Cel pracy

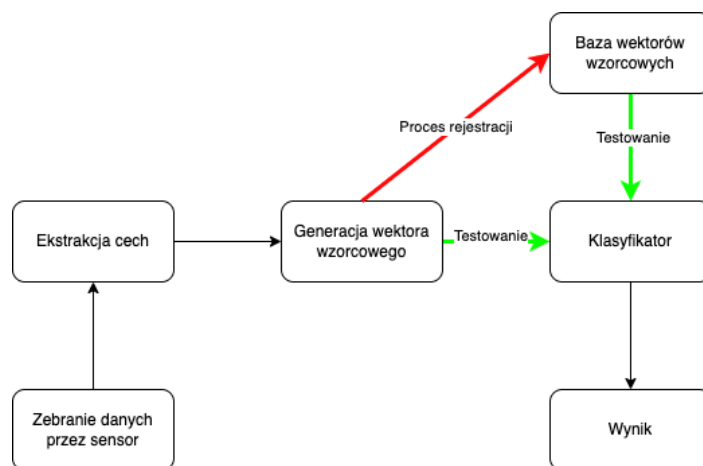
W pracy poruszano głównie zagadnienia związane z biometrią pisaną na klawiaturze. Głównym celem pracy było zbadanie, jak zmiana języka wzorca, który jest wykorzystywany w procesie zbierania danych czasowych, wpływa na finalną skuteczność rozwiązania. W trakcie badań zajmowano się jedynie kwestią języka polskiego oraz angielskiego. Dążono do tego, aby sprawdzić, w jaki sposób zmieni się skuteczność uwierzytelnienia biometrycznego, podczas gdy użytkownik polskojęzyczny będzie raz korzystał z wzorca anglojęzycznego, a raz z polskojęzycznego. Wyniki badań mogą posłużyć do lepszego projektowania rozwiązań biometrycznych bazujących na cechach pobieranych z charakteru pisma użytkownika.

Pobocznym celem pracy było zaprojektowanie i implementacja aplikacji internetowej, która wykorzystuje biometrię bazującą na cechach pisma użytkownika. Aplikacja powinna być prosta we wdrożeniu do dowolnego innego serwisu internetowego i umożliwiać bezpieczne uwierzytelnianie użytkowników bazując na ich charakterze pisma. Końcowe rozwiązanie mogłoby posłużyć jako bramka uwierzytelnienia dwuskładnikowego dla serwisów, takich jak dzienniczki internetowe lub fora tematyczne.

3. Stan sztuki

3.1. Biometria

Biometria pisania na klawiaturze, lub inaczej biometria rytmu pisania na klawiaturze, to jedna z wielu metod biometrycznych. Aby dobrze zrozumieć aktualne rozwiązania bazujące na tym mechanizmie, konieczne jest zapoznanie się z ogólnym schematem działania takiego rozwiązania. Poniżej przedstawiono schemat blokowy działania typowego mechanizmu biometrycznego [11].



Rysunek 3.1. Diagram przepływu przedstawiający sposób działania mechanizmu biometrycznego.

Schemat możemy podzielić na dwa główne procesy. Pierwszym z nich jest proces **rejestracji**, w którym system zbiera dane o użytkowniku i zapisuje w bazie pozyskane o nim informacje w wektorze wzorcowym. Drugim procesem jest proces **testowania**. Podczas tego procesu tworzony jest wektor testowy, który następnie jest porównywany w klasyfikatorze z wektorem wzorcowym przechowywanym w bazie. Następnie system zwraca wynik, którym może być pozytywna lub negatywna identyfikacja użytkownika.

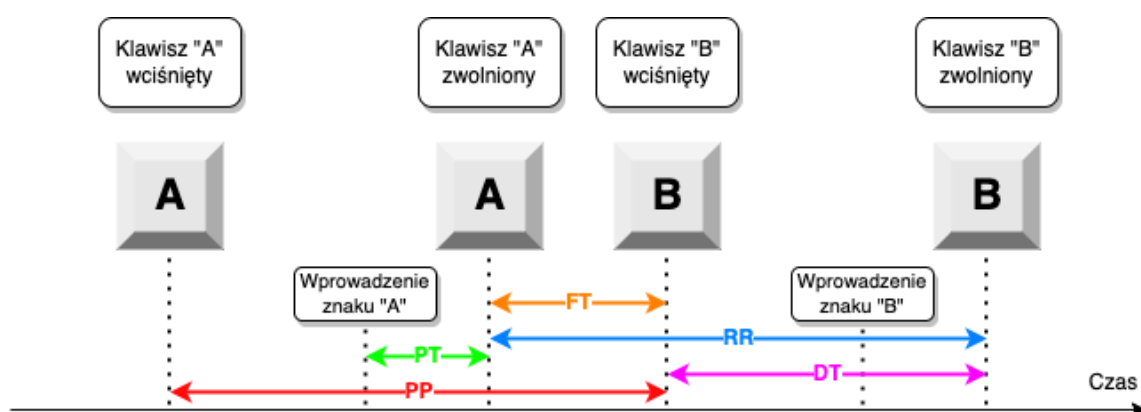
3.2. Parametry czasowe

Systemy biometrii rytmu pisania na klawiaturze bazują głównie na danych czasowych klawiszy wciskanych przez użytkownika [12]. Informacje te mogą być zbierane na wiele sposobów, wykorzystując przy tym różne typy sensorów. Jeśli system jest aplikacją komputerową lub internetową, to wykorzystywane do tego są zdarzenia systemowe. Na ich podstawie zapisywany jest znacznik czasowy, wskazujący dokładny moment wciśnięcia lub zwolnienia przycisku przez użytkownika. Oprócz tego zawierana jest informacja na temat konkretnego przycisku, którego dotyczyło to zdarzenie. Znane są również sprzętowe implementacje, które budowane są między innymi na platformach **FPGA** [13]. Tego typu systemy cechują się lepszą dokładnością, ponieważ wprowadzają mniej opóźnień do systemu, co pozwala na dokładniejszą identyfikację użytkownika. Takie podejście jest jednak

mniej użyteczne, ponieważ wymaga od użytkownika posiadania fizycznego urządzenia, co samo w sobie stoi w sprzeczności z ideą biometrii.

Na podstawie znaczników czasowych pobranych od użytkownika z wykorzystaniem odpowiednich sensorów, w systemie tworzony jest wektor cech charakteru pisma. W literaturze opisywanych jest wiele różnych parametrów, które można uzyskać ze znaczników czasowych. Popularnymi parametrami wykorzystywanymi w wielu implementacjach [14] są:

- czas pomiędzy zwolnieniem i wciśnięciem następujących po sobie klawiszy (*Flight time (FT)*),
- czas pomiędzy wciśnięciem i zwolnieniem tego samego klawisza (*Dwell time (DT)*),
- czas wciśnięcia klawisza (*Press time (PT)*),
- czas pomiędzy wciśnięciami następujących po sobie klawiszy (*Press to press time (PP)*),
- czas pomiędzy zwolnieniami następujących po sobie klawiszy (*Release to release time (RR)*).



Rysunek 3.2. Schemat przedstawiający parametry czasowe wykorzystywane w biometrii dynamiki pisania na klawiaturze.

Powyższe parametry są wyliczane na podstawie różnicy odpowiednich znaczników czasowych. Przykładowo, jeśli mamy dwa znaczniki czasowe - wciśnięcie klawisza "A" (1) oraz wciśnięcie klawisza "B" (2), to czas pomiędzy wciśnięciami (parametr PP) uzyskamy odejmując (1) od (2). W ten sposób obliczane są wszystkie pożądane parametry. Następnie zależnie od implementacji kalkulowane są ich wartości średnie, mediany i dewiacje. Wszystkie te wartości tworzą unikalny dla użytkownika wektor, który jest reprezentacją jego charakteru pisma. Wektor jest zapisywany do bazy, w której jest wykorzystywany do identyfikacji użytkownika w procesie testowania.

3.3. Klasyfikator

Proces testowania polega na ponownym zebraniu danych czasowych, a następnie obliczeniu wektora testowego. Jest on następnie przekazywany do klasyfikatora, który na

podstawie wektora pobranego z bazy oraz wektora testowego, decyduje o identyfikacji użytkownika. W [3] możemy zaobserwować, że najpopularniejszymi metodami klasyfikacji używanymi w badaniach nad biometrią dynamiki pisania są metody statystyczne oraz rozwiązania bazujące na uczeniu maszynowym. Z metod statystycznych możemy wyodrębnić cztery główne podkategorie:

- probabilistyczne (*Probabilistic*),
- grupowanie (*Clustering*),
- bazujące na odległości (*Distance-based*),
- generyczne (*Generic*).

Algorytmy **probabilistyczne** przyporządkowują rozkład prawdopodobieństwa, z jakim każde kolejne wciśnięcie klawisza należy do określonego użytkownika w bazie danych. Prawdopodobieństwo może być modelowane różnymi technikami, np. krzywą Gaussowską lub prawdopodobieństwem ważonym. Algorytmy bazujące na **grupowaniu** polegają na grupowaniu wektorów o podobnych charakterystykach. Przykładowymi metodami grupowania może być technika k-średnich lub c-średnich. Dla algorytmów **bazujących na odległości**, obliczany jest dystans (euklidesowy lub inny), pomiędzy wektorem testowym, a tym przechowywanym w bazie. Odległość ta jest przeliczana na ilość punktów. Jeśli ich ilość znajduje się poniżej lub powyżej (zależnie od algorytmu) tzw. wartości odcięcia (*threshold*), użytkownik zostanie zidentyfikowany jako ten, za którego się podaje. Algorytmy **generyczne** (statystyczne), bazują na obliczaniu odchyłeń, dewiacji i median wektora testowego od tego znajdującego się w bazie. Przykładami algorytmów statystycznych są algorytm K najbliższych sąsiadów oraz test t-studenta.

Najpopularniejszymi podejściami wykorzystującymi uczenia maszynowe są:

- sieci neuronowe (*Neural networks*),
- drzewa decyzyjne (*Decision tree*),
- logika rozmyta (*Fuzzy logic*),
- algorytmy ewolucyjne (*Evolutional computing*).

Sieci neuronowe są mechanizmem, który stara się naśladować procesy zachodzące w ludzkich neuronach, aby estymować przyszłe wartości oraz klasyfikować wektory testowe. Algorytm **drzewa decyzyjnego** polega na rozdzieleniu danych wejściowych w taki sposób, aby w każdym węźle utworzonego w ten sposób drzewa, zysk informacyjny był największy. Przykładowymi technikami wykorzystującymi drzewo decyzyjne są algorytm lasu losowego (*random forest*) lub algorytm J48. Systemy bazujące na **logice rozmytej** (*fuzzy logic*), starają się podzielić zbiór danych na podzbiory, przypisując elementom wartości pośrednie, na podstawie których określana jest ich przynależność do zbiorów. **Algorytmy ewolucyjne** są wzorowane na procesach, które zachodzą w naturze. Przykładami takich algorytmów są algorytm mrówkowy lub algorytm genetyczny. Jednak niezależnie od typu algorytmu, kluczowym elementem projektowania i implementacji takich systemów jest proces zbierania danych oraz trenowania algorytmu. Konieczne jest zebranie dokład-

nych danych, które pozwolą na poprawne wytrenowanie systemu, tak aby w przyszłości poprawnie identyfikował użytkowników [15].

3.4. Wzorzec tekstowy

Biometria pisania opiera się na cechach sposobu wprowadzania tekstu na urządzeniu przez użytkownika. Nie bez znaczenia jest tutaj typ wzorca tekstowego. Wśród większości prac badawczych można zaobserwować wyszczególnienie dwóch typów wzorca tekstowego [16]:

- tekst dowolny (*Free text*),
- tekst stały (*Fixed text*).

Tekst dowolny daje dużo większą swobodę dla użytkownika i projektanta systemu, ponieważ nie ma konieczności zapamiętywania ani zapisywania wcześniej ustalonej frazy. Przykładem dowolnego wzorca tekstowego jest losowo generowane zdanie, które użytkownik musi przepisać w celu zalogowania się do systemu. Wadami tekstu dowolnego jest brak powtarzalności wektora testowego, ponieważ cechy pisma mogą się nieznacznie zmieniać w zależności od tego co użytkownik aktualnie przepisuje.

Tekst stały eliminuje tę przypadłość, ponieważ użytkownik za każdym razem wpisuje tę samą frazę. Przykładem wzorca tekstu stałego może być nazwa (login) użytkownika [17]. Oczywiście można wykorzystać całkowicie dodatkową frazę, którą użytkownik sam wprowadza podczas procesu rejestracji. Wtedy jednak konieczne jest dodatkowe przechowywanie takiej frazy po stronie aplikacji, a użytkownik musi zapamiętać lub zapisać w menadżerze haseł dodatkową, poza loginem i hasłem, zmienną. Takie podejście ma jeszcze jedną ważną zaletę. Tekst stały w formie osobnej frazy pozwala użytkownikowi na sprawdzenie, czy pod witrynę internetową nie podszywa się inna złośliwa strona [18]. Działa to w sposób podobny do obrazków użytkownika wyświetlanych w niektórych serwisach bankowych. Jeśli przy logowaniu aplikacja wyświetla przypisany do użytkownika obrazek (lub w naszym przypadku tekst), to możemy przyjąć z dużą pewnością, że strona nie jest złośliwa, ponieważ zdobycie tych informacji przez atakującego jest skrajnie trudne do osiągnięcia.

3.5. Ocena systemów biometrycznych

W literaturze przyjęło się, że skuteczność rozwiązań wykorzystujących biometrię ocenia się zazwyczaj trzema parametrami:

- FRR (*False Rejection Rate*),
- FAR (*False Acceptance Rate*),
- EER (*Equal Error Rate*).

Wszystkie powyższe parametry wyrażane są w procentach. Parametr **FRR** określa odsetek prób testowych, podczas których system błędnie odrzucił prawidłowego użytkownika. Parametr **FAR** opisuje odsetek prób testowych, podczas których system nieprawidłowo

potwierdził tożsamość użytkownika. Parametr **EER** wyznacza wartość, w której parametry FRR i FAR są sobie równe i wyznacza optymalny punkt pracy systemu biometrycznego.

3.6. Aktualnie rozwiązania

Prace nad biometrią dynamiki pisania na klawiaturze mają swoje początki już pod koniec dziewiętnastego wieku, kiedy popularna była komunikacja z użyciem telegrafu. Telegrafisci mogli wtedy rozpoznawać siebie nawzajem, bazując na rytmie wysyłania kodu Morse'a [19]. W czasie drugiej wojny światowej wykorzystywano tę metodę przy odróżnianiu sojusznika od wroga podczas komunikacji z wykorzystaniem telegrafu [20]. Wzrost zainteresowania dynamiką pisania w kontekście biometrii ma swój początek w latach 90 ubiegłego wieku. Wtedy nastąpił wzmożony wysiłek, którego celem było stworzenie coraz to bardziej skutecznych rozwiązań stosujących ten mechanizm biometryczny.

Obecnie znane są komercyjne i otwarte źródłowe rozwiązania, które oferują identyfikację bazującą na biometrii dynamiki pisania. W poniższej tabeli zamieszczono niektóre z tych projektów wraz z typem ich detektorów oraz szacunkowymi poziomami skuteczności.

Tabela 3.1. Zestawienie przykładowych rozwiązań w zakresie biometrii cech pisania na klawiaturze.

Rozwiązanie	Klasyfikator	Typ wzorca	Skuteczność
Nethone [21]	Uczenie maszynowe	Tekst stały	95,30%
Typingdna [22]	Sztuczna inteligencja	Tekst dowolny lub stały	98,00%
CNN model [23]	SVM	Tekst dowolny	92,31%
TypeNet [24]	Rekurencyjna sieć neuro- nowa (RNN)	Tekst dowolny	95,00%
S.Krishnamoorthy [25]	Wieloklasowy SVM	Tekst stały	97,40%

Bazując na powyższym zestawieniu możemy stwierdzić, że obecnie możliwe jest stworzenie rozwiązania bazującego na biometrii pisania na klawiaturze o bardzo wysokiej (ponad 90%) skuteczności. Oczywiście nie wszystkie rozwiązania cechują się tak wysokim poziomem skuteczności. Przykładowo, w [26] uzyskano skuteczność rozwiązania bazującego na dowolnym wzorcu tekstowym na poziomie 85%. Niższa skuteczność jest prawdopodobnie spowodowana zbyt małą próbą badawczą oraz zbyt małą liczbą cech wydobywanych z parametrów czasowych użytkownika.

Prezentowane rozwiązania korzystają zarówno z klasyfikatorów bazujących na SVM, jak i tych opartych na uczeniu maszynowym. W przypadku rozwiązań [21] oraz [22], nie udało się jednoznacznie określić typu klasyfikatora ze względu na to, że są to rozwiązania komercyjne. Wśród rozwiązań otwartoźródłowych zdecydowanie popularniejszym rozwiązaniem jest detektor SVM. Aktualne prace zdają się nie faworyzować określonego typu wzorca tekstowego. Tekst stały był wybierany równie często jak dowolny, a najczęściej była nim nazwa (login) identyfikowanego użytkownika podawana przy logowaniu. Wszystkie zaprezentowane implementacje korzystały głównie ze znaczników czasowych

do ekstrakcji cech umożliwiających identyfikację. W [21], oprócz znaczników czasowych, wykorzystywane były również informacje kontekstowe powiązane z sesją użytkownika. Istnieją jednak rozwiązania, w których od użytkownika wcale nie są pobierane parametry czasowe. W [27] wykorzystano jedynie pomiar siły nacisku na ekran urządzenia, co dało efekt w postaci skuteczności na poziomie 99%.

4. Koncepcja

Sekcja ta jest poświęcona podejściu, jakie obrano przy projektowaniu poszczególnych elementów systemu biometrycznego w aplikacji Typeauth. Aby stworzyć działający system biometryczny, konieczne jest posiadanie elementu zbierającego dane biometryczne (sensora) i elementu zajmującego się klasyfikacją, ekstrakcją cech oraz testowaniem. Konieczne jest także stworzenie odpowiedniej bazy danych, w której dane będą przechowywane. Każdy z elementów systemu został zaprojektowany z myślą o jak największej skuteczności przy zachowaniu prostoty implementacji.

4.1. Ekstrakcja danych

Głównym problemem rozwiązania był projekt mechanizmu, który umożliwiłby efektywną ekstrakcję odpowiednich cech biometrycznych, które potem miałyby zostać wykorzystane do klasyfikacji oraz identyfikacji użytkownika. Obecnie istnieje już wiele prac badawczych poświęconych badaniu skuteczności poszczególnych typów cech, możliwych do ekstrakcji w procesie pisania na klawiaturze. W niniejszej pracy zdecydowano się na wybór cech, które gwarantują największą skuteczność przy jednoczesnym jak najmniejszym skomplikowaniu implementacji.

4.1.1. Typy zdarzeń

Programistyczne zdarzenia, które są generowane przez użytkownika podczas procesu pisania na klawiaturze, możemy zgrubnie zebrać do poniższych trzech typów [28]:

- przycisk w górę (*Key up*) - zdarzenie, kiedy przycisk wraca do domyślnej pozycji (podniesionej),
- przycisk w dół (*Key down*) - zdarzenie, kiedy następuje kompresja przycisku przez użytkownika,
- wprowadzenie znaku (*Key press*) - zdarzenie, kiedy do aplikacji wprowadzony zostaje znak, a przycisk znajduje się w najniższej pozycji.

4.1.2. Typy cech biometrycznych

Z powyższych zdarzeń możliwe jest wydobywanie cech użytkownika, które możemy podzielić na dwie podstawowe grupy. Pierwsza grupa to **cechy globalne** (*global features*). Dotyczą one określonych zachowań użytkownika w trakcie pisania na klawiaturze. Do cech globalnych możemy zaliczyć takie cechy, jak wciśnięcia klawisza *Backspace* lub pisanie dużych liter z wykorzystaniem klawisza *Caps Lock* lub klawisza *Shift*. W aplikacji Typeauth, od użytkownika zbierane są następujące cechy globalne:

- liczba wciśnień klawisza Shift (parametr *shift_count*),
- liczba wciśnień klawisza Backspace (parametr *backspace_count*),
- czy wciśnięty został klawisz Caps Lock (parametr *is_capslock*).

4.1.3. Cechy czasowe

Drugą grupą są **cechy czasowe** (*temporal features*) [29]. Wynikają one bezpośrednio ze stylu pisanía użytkownika na klawiaturze, czyli z wciśnień pojedynczych klawiszy lub tak zwanych dwuznaków, czyli dwóch kolejnych klawiszy na klawiaturze. Wykonując odpowiednie pomiary czasowe w czasie, gdy użytkownik wprowadza tekst na klawiaturze, możliwa jest rejestracja następujących znaczników czasowych (*timestamp*):

- znacznik czasowy wciśnięcia klawisza (zdarzenie *Key down*),
- znacznik czasowy odpuszczenia klawisza (zdarzenie *Key up*).

Powyższe znaczniki mogą być następnie przesłane do serwera aplikacyjnego. Na ich podstawie w aplikacji Typeauth obliczone zostaną następujące parametry (cechy) czasowe:

- czas pomiędzy wciśnięciami następujących po sobie klawiszy (*Press to press time (PP)*),
- czas pomiędzy zwolnieniami następujących po sobie klawiszy (*Release to release time (RR)*).

4.1.4. Proces zbierania danych

Proces zbierania danych biometrycznych przy rejestracji określono mianem **onboardingu**. Wynikiem tego procesu jest stworzenie wektora unikalnych cech biometrycznych, które określają styl pisma każdego użytkownika. Podczas tego procesu, użytkownik ma za zadanie przepisać czterokrotnie cztery losowe frazy. Po zakończeniu onboardingu dane te przekazywane są do serwera aplikacyjnego, gdzie wyliczane są następujące parametry:

- średni czas pomiędzy wciśnięciami następujących po sobie klawiszy (parametr *hold_mean*),
- średni czas pomiędzy wciśnięciami zwolnieniami po sobie klawiszy (parametr *idle_mean*),
- mediana czasu pomiędzy wciśnięciami następujących po sobie klawiszy (parametr *hold_median*),
- mediana czasu pomiędzy zwolnieniami następujących po sobie klawiszy (parametr *idle_median*).

Takie podejście do ekstrakcji cech biometrycznych zostało wykorzystane w [30], gdzie, w połączeniu z detektorem SVM, osiągnięto zadowalające rezultaty dla implementacji w postaci aplikacji internetowej.

4.2. Klasyfikacja

W celu uwierzytelnienia, czyli potwierdzenia tożsamości logującego się użytkownika, konieczna była implementacja odpowiedniego mechanizmu klasyfikacji. Zdecydowano się nie wybierać jednego typu klasyfikatora, a zamiast tego przeprowadzić wstępne badania dla dwóch klasyfikatorów - po jednym z każdej kategorii (uczenie maszynowe oraz metody statystyczne). W [31] przeprowadzono szeroko zakrojone badania, które porównują skuteczność wielu typów klasyfikatorów pod kątem biometrii. W pracy zbadano 14 klasyfikatorów. Zarówno detektor **SVM** oraz detektor wykorzystujący metrykę **Manhattan** uzyskał bardzo dobre wyniki skuteczności identyfikacji użytkownika. Zdecydowano się na implementację tych klasyfikatorów w pierwszym etapie projektowania aplikacji Typeauth. Reprezentują one dwie odrębne grupy metod detekcji, co znacząco poszerza spektrum badań. Oprócz tego, na korzyść tych klasyfikatorów świadczy ich potwierdzona wysoka skuteczność w wielu innych pracach naukowych [32], [30].

4.2.1. Klasyfikator bazujący na metryce Manhattan

Klasyfikator bazujący na metryce Manhattan (klasyfikator Manhattan) przypisuje określoną liczbę punktów wektorowi testowemu, bazując na jego odległości (według metryki Manhattan) od wektora przechowywanego w bazie. Jeśli ilość punktów jest większa od pewnej wartości odcięcia (*threshold*), to system uznaje, że użytkownik nie jest tym za którego się podaje. Poniżej przedstawiono wzór na odległość Manhattan pomiędzy dwoma punktami x oraz y :

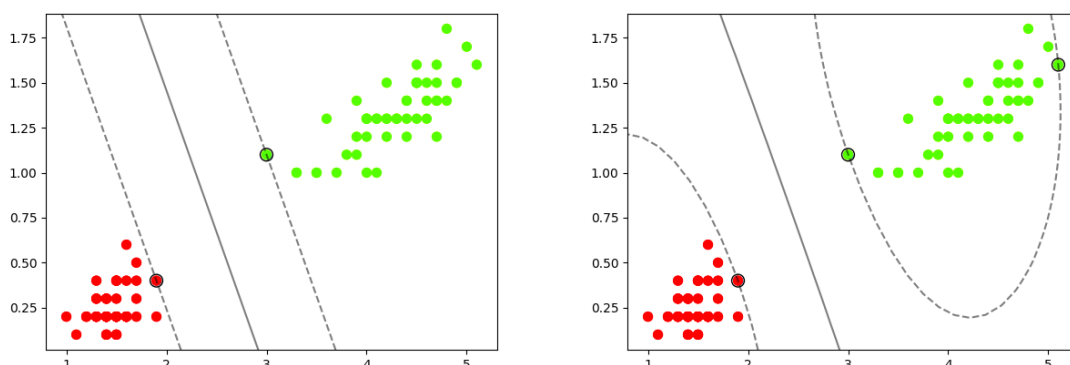
$$D(x, y) = \sum_i^{d_x} |x_i - y_i|$$

W powyższym równaniu d_x jest wymiarem wektora cech danego użytkownika. W projektowanej implementacji klasyfikatora Manhattan, obliczana będzie odległość pomiędzy wektorem testowym oraz wektorem przechowywanym w bazie. Oba te wektory będą składały się z czterech zmiennych zmiennoprzecinkowych (sekcja 4.1.4) oraz trzech zmiennych zmiennoprzecinkowych zawierających informacje o klawiszach charakterystycznych (sekcja 4.1.2). Na podstawie obliczonej odległości Manhattan pomiędzy tymi wektorami, wektor testowy będzie odpowiednio punktowany, a następnie weryfikowany względem wartości odcięcia.

4.2.2. Detektor SVM

Maszyna wektorów nośnych SVM (*Support Vector Machine*) jest koncepcją detektora opartego na klasyfikacji statystycznej. Klasyfikator ma za zadanie wyznaczenie hiperpłaszczyzny, która najlepiej rozdziela dwie klasy x_i, y_i , tak aby odległości pomiędzy poszczególnymi punktami były jak największe (Rysunek 4.1). Dzięki wykorzystaniu tzw. *kernel trick* [33], możliwa jest klasyfikacja wektora cech o nieskończonej długości, przez co SVM

dobrze sprawdza się w sytuacjach, w których liczba przestrzeni (cech) jest dużo większa od liczby próbek [34].



Rysunek 4.1. Wizualizacja klasyfikacji dwóch zbiorów punktów z wykorzystaniem klasyfikatora SVM bazującego na liniowej funkcji jądra (lewy) oraz z wykorzystaniem klasyfikatora SVM bazującego radialnej funkcji bazowej (**RBF**) (prawy).

W projektowanej aplikacji Typeauth, algorytm SVM najpierw będzie starał się sklasyfikować wektory znajdujące się już w bazie danych. Następnie odpowiednia funkcja będzie pobierała wektor testowy o takiej samej strukturze, jak dla klasyfikatora Manhattan. Kolejnym etapem będzie klasyfikacja wektora testowego do jednego z użytkowników znajdujących się już w bazie. Jako wynik funkcja będzie zwracała identyfikator użytkownika, do którego, według algorytmu, należy wektor testowy.

4.3. Propozycja rozwiązania

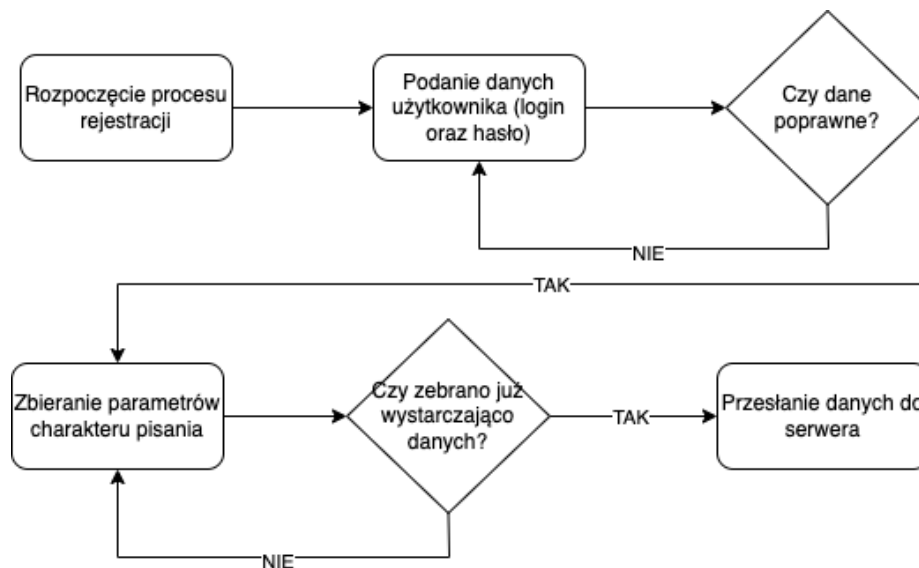
Biorąc pod uwagę wcześniejsze obserwacje, zaprojektowano system mający na celu realizację dwóch głównych zadań:

- klasyfikację użytkowników na podstawie ich cech biometrycznych wynikających ze stylu pisanie na klawiaturze,
- prawidłowe uwierzytelnienie użytkownika wykorzystując dwa składniki - jego hasło oraz biometrię bazującą na cechach wpisywanego przy logowaniu tekstu.

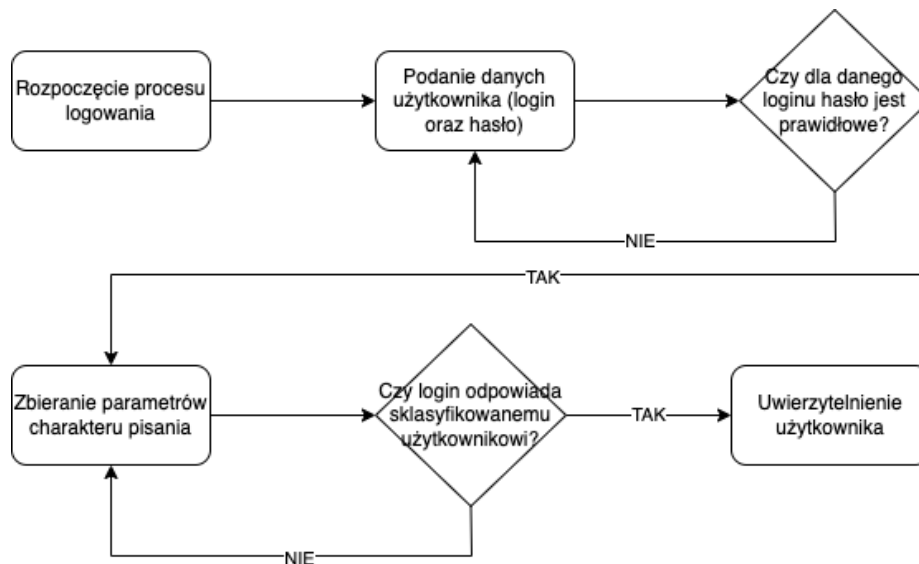
Poniżej zaprezentowano schematy procesów rejestracji (onboardingu) oraz logowania użytkownika (Rysunek 4.2 i Rysunek 4.3).

Po otrzymaniu danych od użytkownika, serwer obliczałby średnie wartości oraz mediany przesłanych parametrów, a następnie zapisywałby je w bazie danych. Taki wektor cech charakteryzujący każdego użytkownika byłby powiązany z jego parametrami logowania - nazwą użytkownika (adresem email) oraz hasłem.

Przy procesie logowania, aplikacja klasyfikowałaby użytkownika oraz podejmowałaby decyzje o tym, czy może on zostać uwierzytelniony. Dla klasyfikatora Manhattan proces polegałby na obliczeniu tzw. **metryki Manhattan** pomiędzy odpowiadającymi sobie



Rysunek 4.2. Diagram przepływu procesu rejestracji użytkownika.

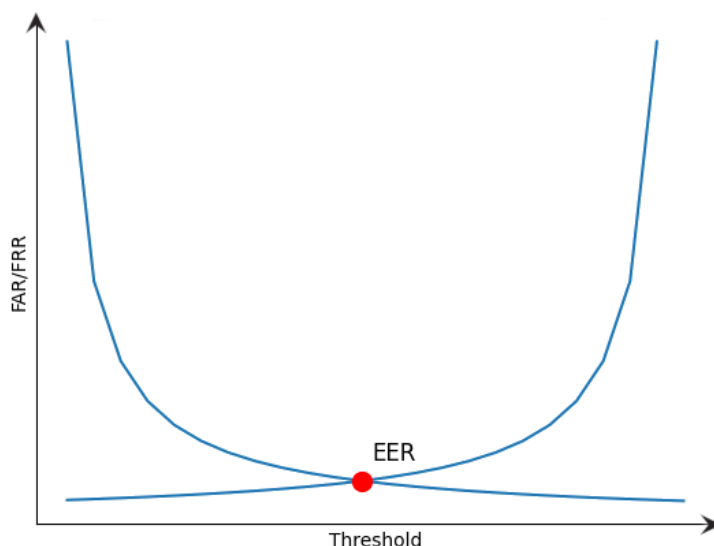


Rysunek 4.3. Diagram przepływu procesu logowania użytkownika.

wartościami znajdującymi się w dwóch wektorach cech. Pierwszy z wektorów byłby wektorem użytkownika, za którego podaje się logującą się osoba, natomiast drugi wektor jest wyliczany na podstawie danych pobranych z procesu logowania. Aby użytkownik został uwierzytelniony, wynik takich obliczeń musiałby być odpowiednio mniejszy, niż wcześniej z góry ustalony próg (*threshold*).

Dla klasyfikatora Manhattan oraz dla każdego innego klasyfikatora bazującego na odległości istnieje problem ustawienia wartości progowej (*threshold*). Dla tego typu detektorów użytkownik jest uwierzytelniony, jeśli odległość pomiędzy dwoma wektorami cech jest mniejsza (lub większa zależnie od implementacji) od wartości progowej. Występuje zatem problem związany z tym, jaką wartość ma stanowić próg decyzyjny,

tak aby wartości parametrów FRR i FAR były optymalne. Zazwyczaj, gdy wartość progowa jest coraz mniejsza, parametr FAR maleje prawie do zera, natomiast parametr FRR rośnie, ponieważ klasyfikator akceptuje tylko bardzo wiarygodne wektory. Stąd ciężko uwierzytelnić się nawet prawidłowemu użytkownikowi, którego wektor testowy czasem odbiega od wzorcowego. W przypadku zwiększania wartości progu odcięcia, następuje analogiczna sytuacja - FAR rośnie, natomiast FRR maleje. Prawidłowemu użytkownikowi łatwiej się uwierzytelnić, natomiast osoba podszywająca się pod użytkownika również jest klasyfikowana jako użytkownik, ponieważ detektor częściej akceptuje odstępstwa. Mamy tutaj konflikt pomiędzy dwoma właściwościami projektowanego systemu - użytecznością oraz bezpieczeństwem. Im system jest bardziej bezpieczny (mniejsza wartość progu decyzyjnego), tym większa szansa, że przy próbie uwierzytelnienia odrzucony zostanie prawdziwy użytkownik. Optymalna wartość progu decyzyjnego występuje w momencie, gdy krzywe FRR i FAR przecinają się. Jest to punkt EER (*Equal Error Rate*) [35].



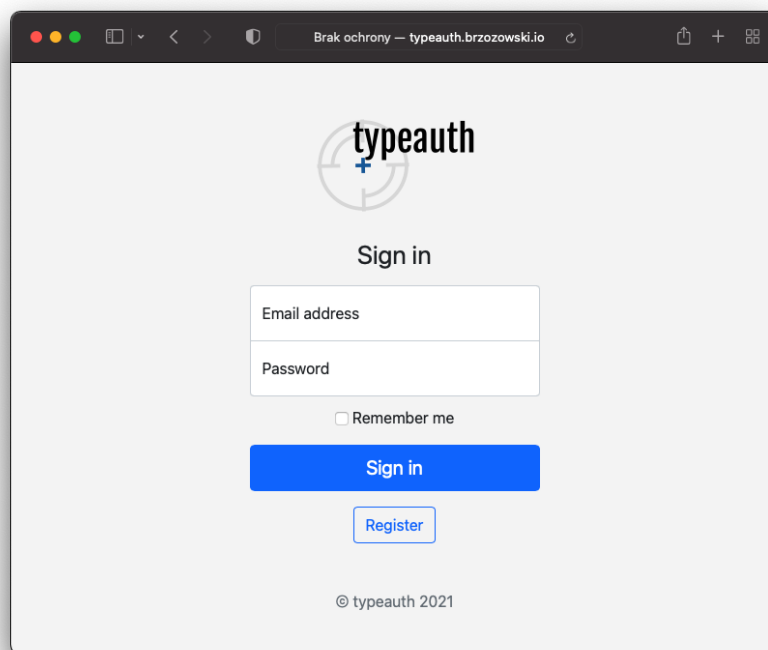
Rysunek 4.4. Wykres parametrów FRR oraz FAR od wartości progowej z zaznaczeniem optymalnego punktu EER.

Istnieje wiele algorytmów i metod obliczania wartości progowej, przy której osiągnięty jest punkt EER. Przykładowo, w [36] zastosowano mechanizm, w którym projektowany system nie posiada stałej wartości progowej, a zamiast tego jest ona wyliczana dla każdego wektora cech znajdującego się już w bazie, a następnie wybierana jest wartość progowa, dla której osiągnięta jest wartość EER lub dla wartości, w której uzyskano najmniejszą wartość FAR. W proponowanym w niniejszej pracy rozwiązaniu, wartość progowa detektora Manhattan została obliczona wykonując serię badań dla różnych wartości progowych detektora, a następnie wybrana, by została wartość, dla której wartości średnie parametrów FAR i FRR są najbardziej zbliżone do siebie.

Odpowiednikiem progu decyzyjnego dla detektora SVM jest **zmienna pomocnicza C** (*Regularization parameter*). Odpowiada ona za siłę kary ponoszonej w przypadku błędnej klasyfikacji. Im mniejsza wartość parametru C, tym większa może być powierzchnia hiperpłaszczyzny rozdzielającej dwa zestawy danych. Zwiększa to prawdopodobieństwo tego, że próbka zostanie błędnie sklasyfikowana, co wiązałoby się z tym, że hiperpłaszczyzna będzie lepiej oddzielała poszczególne punkty danych.

5. Opis rozwiązania

W ramach pracy zaprojektowano i zaimplementowano od podstaw rozwiązanie umożliwiające zbieranie danych biometrycznych i klasyfikację użytkowników na ich podstawie. Aplikacja została roboczo nazwana **Typeauth** (*type* - pisać, *authentication* - uwierzytelnienie). Tak jak wskazuje nazwa projektu, ma on za zadanie uwierzytelniać użytkowników na podstawie ich charakterystyki pisania, a dokładniej wpisywania znaków na klawiaturze.



Rysunek 5.1. Ekran logowania aplikacji Typeauth.

5.1. Aplikacja internetowa

W fazie projektowania zdecydowano, że rozwiązanie zostanie zaimplementowane w formie aplikacji internetowej. Wybrano takie podejście z kilku powodów. Przede wszystkim charakter aplikacji internetowej znacząco ułatwia proces badań, ponieważ dostęp do systemu łatwo udostępnić uczestnikom badania. Należy jednak mieć na uwadze zagrożenia, jakie wiążą się z aplikacjami internetowymi. Najbardziej sporna jest kwestia bezpieczeństwa przy korzystaniu z aplikacji tego typu. Nasze dane wysyłane są wtedy do zewnętrznego serwera, który może stać się celem ataków cyberprzestępców.

5.2. Stos technologiczny

W niniejszym rozdziale przedstawiono stos technologiczny wykorzystany do stworzenia aplikacji. W poszczególnych sekcjach opisano technologie, takie jak wykorzy-

stany język programowania, framework i system bazy danych. Zamieszczono również fragmenty kodu aplikacji, aby przybliżyć sposób, w jaki zaimplementowano określone funkcjonalności.

5.2.1. Flask

Flask jest lekkim frameworkiem dla języka Python, który umożliwia tworzenie aplikacji internetowych, bazując na tak zwanych **szablonach** [37]. Szablony są plikami HTML, w których, za pomocą odpowiedniej składni, możemy przekazywać dynamiczne zmienne z aplikacji i wyświetlać je użytkownikowi końcowemu w przeglądarce. Parsowaniem szablonów i tworzeniem dynamicznych dokumentów HTML w frameworku Flask zajmuje się silnik Jinja2. Dodatkowo, dzięki temu, że framework ten jest bardzo popularny wśród społeczności programistów, funkcjonalności aplikacji mogą zostać rozszerzone z wykorzystaniem tworzonych przez nich wtyczek, co znacznie zwiększa możliwości tworzonych projektów. Poniżej przedstawiono przykład tworzenia pętli `for` z wykorzystaniem szablonu.

Listing 1. Przykład pętli `for` jako szablon Jinja2.

```
1  {% for item in items %}
2      <li><a href="{{ _item.href }}">{{ item.caption }}</a></li>
3  {% endfor %}
```

Oprócz obsługi szablonów, silnik Jinja2 zapewnia bezpieczeństwo aplikacji, pozwalając na uruchamianie szablonów w izolowanym środowisku (*sandbox*) [37]. Pomimo wydajności języka Python, która jest mniejsza od innych języków kompilowanych, zdecydowano się na Pythona ze względu na niewielkie wymagania środowiska uruchomieniowego, szybkość tworzenia projektów oraz wiele publicznie dostępnych bibliotek z dobrą dokumentacją.

5.2.2. Javascript

Język Javascript jest aktualnie najpopularniejszym językiem programowania na platformie Github [38], która gromadzi największy na świecie zbiór repozytoriów kodu źródłowego. Javascript jest językiem skryptowym, który wykorzystywany jest głównie przez przeglądarki internetowe do dynamicznej zmiany zawartości wyświetlanej strony internetowej, w zależności od wywoływanych przez użytkownika zdarzeń. W aplikacji Typeauth, język ten wykorzystano do obsługi procesu zbierania danych biometrycznych. Reagując na odpowiednie zdarzenia w postaci wciśnięcia (*keydown*) i zwolnienia (*keypress*) klawisza klawiatury, uruchomiony w przeglądarce kod Javascript może mierzyć parametry dynamiki pisania na klawiaturze. Przykładowy fragment kodu służący do rejestrowania takich zdarzeń przedstawiono na Listingu 2.

Listing 2. Fragment kodu Javascript rejestrującego zdarzenie wciśnięcia (*keypress*) poszczególnych klawiszy przez użytkownika.

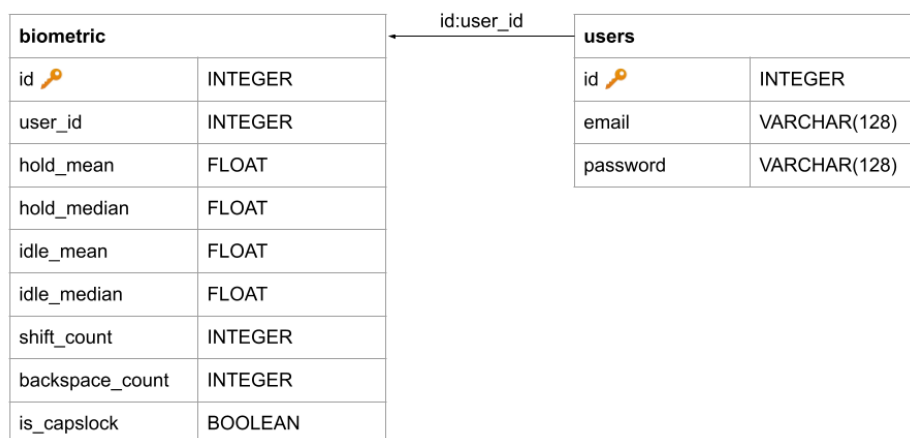
```

1 $( '#floatingInput' ). keypress (function ( evt ) {
2     tmr . push ( {
3         "tmr" : evt . timeStamp ,
4         "key_code" : evt . keyCode ,
5         "evt" : "keypress"
6     } );
7 }
```

Przeniesienie całej logiki zbierania znaczników czasowych na stronę klienta (przeglądarki) upraszcza końcowe rozwiązanie, jak i zmniejsza ilość danych transmitowanych do backendu. Takie podejście jest często stosowane w oprogramowaniu wykorzystywanym do badania charakterystyki pisania na klawiaturze [39]. Jednak język Javascript jest również wykorzystywany przez cyberprzestępców w tak zwanych **keyloggerach**, czyli złośliwych programach przechwytyjących dane wpisywane na klawiaturze przez ofiarę. Danymi takimi mogą być numery kart kredytowych lub hasła do kont bankowych.

5.2.3. Baza danych

Po otrzymaniu i wstępnym przetworzeniu danych zebranych od użytkownika, zostają one wysłane do serwera aplikacyjnego. Do przechowywania tych danych po stronie serwera wykorzystano relacyjny system bazodanowy **SQLAlchemy** [40]. SQLAlchemy dostarcza proste i funkcjonalne biblioteki umożliwiające w bezpieczny sposób zarządzanie bazą danych. W procesie tworzenia, jako typ bazy danych wybrano format **SQLite** [41], który, ze względu na niskie wymagania sprzętowe oraz niewielki rozmiar, stanowił adekwatny wybór do projektowanego rozwiązania. Poniżej zaprezentowano schemat zaprojektowanej na potrzeby badań bazy danych.

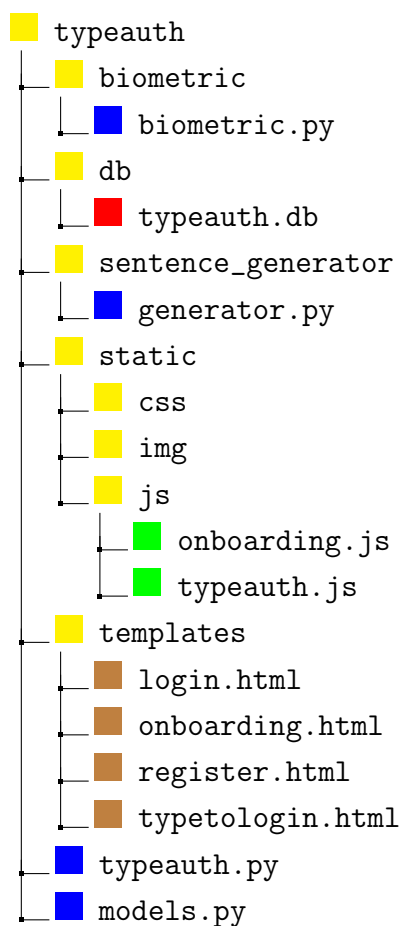


Rysunek 5.2. Schemat bazy danych SQLite wykorzystanej w ramach badań.

Tabela `user` przechowuje standardowe dane identyfikujące użytkownika, takie jak login (adres email) oraz hasło. Identyfikator każdego użytkownika jest powiązany relacją z jego danymi biometrycznymi zapisanymi w tabeli `biometric`. Danymi tymi są mediany i wartości średnie czasów pomiędzy wciśnięciami (PP) oraz odpuszczeniami (RR) kolejnych klawiszy. Oprócz tego, w tabeli zapisywane są średnie ilości wciśnień klawiszy specjalnych – Shift i Backspace oraz zmienna logiczna, która informuje o tym, czy podczas rejestracji użyto przycisku Caps Lock.

5.3. Implementacja aplikacji

Aby zaimplementować tak złożony projekt jak aplikacja internetowa, zdecydowano się na podzielenie całego projektu na mniejsze pliki i podkatalogi. Utworzono osobne katalogi dla każdej biblioteki, bazy danych oraz szablonów HTML (Rysunek 5.3). W



Rysunek 5.3. Listing wszystkich katalogów i plików aplikacji Typeauth.

głównym katalogu znajduje się plik `typeauth.py`, który odpowiada za inicjalizowanie aplikacji, jej konfiguracji oraz tworzenie wszystkich obiektów i zmiennych potrzebnych do jej funkcjonowania. Aplikacja posiada możliwość wyboru klasyfikatora oraz trybu generowania wzorca tekstowego. Wybór klasyfikatora jest podejmowany na podstawie

zmiennej tekstowej `method`, która może przyjmować wartość odpowiednio “manhattan” lub “svm”. Zmienna tekstowa `word_type` decyduje o trybie, w jakim pracuje generator wzorca tekstowego. Może przyjmować ona wartości “pl_long”, “pl_short”, “eng_long” lub “eng_short”. Parametr ten mógł być również przekazywany przez użytkownika w parametrze żądania HTTP GET, co umożliwiłaby zmianę trybu generacji wzorca z poziomu przeglądarki. Oprócz tego, plik `typeauth.py` odpowiada za odpowiedni routing ruchu sieciowego do następujących ścieżek aplikacji:

- / - główny ekran aplikacji służący do wyboru ścieżki logowania lub rejestracji,
- /onboarding - widok aplikacji odpowiadający za rejestrację użytkownika,
- /typetologin - widok aplikacji służący za wyświetlanie wzorca tekstowego i zbieranie znaczników czasowych pisania na klawiaturze.

Plik `models.py` przechowuje schemat tworzenia bazy danych SQLite. Szczegółowy schemat bazy przedstawiono na Rysunku 5.2. Sama baza danych przechowywana jest w katalogu `db`. W katalogu `templates` umieszczono szablony HTML zawierające instrukcje do generacji dynamicznych stron poprzez silnik Jinja2. Folder `static` przechowuje statyczne pliki i skrypty wykorzystywane przez przeglądarkę. Zawarte w katalogu `static/js` pliki JavaScript odpowiadają za logikę zbierania i przetwarzania znaczników czasowych w momencie kiedy użytkownik pisze na klawiaturze. Katalog `static_generator` zawiera plik `generator.py`, który odpowiada za generację wzorców tekstowych. W pliku w tablicach `nouns_eng_long`, `nouns_pl_long`, `nouns_eng_short`, `nouns_pl_short` przechowywane są zmienne tekstowe, które służą za bazę do losowej generacji wzorców tekstowych. W katalogu `biometric` umieszczony jest plik `biometric.py`, w którym zawarte są dwa klasyfikatory – SVM oraz Manhattan. Klasyfikatory jako argumenty przyjmują dwa wektory cech użytkownika. Wektor cech użytkownika jest tablicą zawierającą następujące zmienne zmiennoprzecinkowe:

- `hold_mean` - średni czas pomiędzy wciśnięciami następujących po sobie klawiszy,
- `idle_mean` - średni czas pomiędzy wciśnięciami zwolnieniami po sobie klawiszy,
- `hold_median` - mediana czasu pomiędzy wciśnięciami następujących po sobie klawiszy,
- `idle_median` - mediana czasu pomiędzy zwolnieniami następujących po sobie klawiszy,
- `shift_count` - ilość użycia przycisku Shift,
- `backspace_count` - ilość użycia przycisku Backspace.

Zdecydowano się na zrezygnowanie ze zmiennej boolowskiej przechowującej informacje o tym, czy użyto klawisza CapsLock, ze względu na niską wartość informacyjną tej zmiennej. Wynika to z niewielkiej ilości użytkowników, którzy korzystali z tego klawisza w procesie logowania i rejestracji.

5.4. Klasyfikator

5.4.1. Klasyfikator bazujący na metryce Manhattan

Klasyfikator Manhattan zaimplementowano z wykorzystaniem biblioteki Scipy [42]. Jest powszechnie wykorzystywana biblioteka, która posiada wiele metod realizujących różnorodne obliczenia matematyczne. Wykorzystany moduł `scipy.spatial.distance` zawiera funkcję `cityblock()`, obliczającą metrykę Manhattan pomiędzy dwoma wektorami. Funkcję tą zastosowano w metodzie `check_user_manhattan()` (Listing 4). Do funkcji tej przekazywany jest wektor testowy oraz wektor przechowywany w bazie. Następnie, zwracana jest punktacja, która odpowiada odległości Manhattan pomiędzy tymi dwoma wektorami.

Listing 3. Implementacji klasyfikatora Manhattan w aplikacji Typeauth.

```
1 def check_user_manhattan(test_vector, user_vector):
2
3     score = cityblock(test_vector, user_vector)
4
5     return score
```

5.4.2. Klasyfikator SVM

Do implementacji klasyfikatora SVM w realizowanym projekcie wykorzystano gotową bibliotekę scikit-learn w wersji 1.0.1 [34] (Listing 5). Biblioteka ta zawiera wiele implementacji maszyn wektorów nośnych, które charakteryzują się szerokimi możliwościami konfiguracji procesu klasyfikacji. Oprócz zmiany funkcji jądra na wersje liniowe, gaussowskie i wielomianowe, możliwe jest także dostosowanie wartości kary za błędną klasyfikację oraz kształt funkcji decyzyjnej. Funkcja `check_user_svm()` przyjmuje jako argumenty wektory cech użytkowników z bazy danych, tablicę zawierającą ich identyfikatory numeryczne oraz wektor testowy.

Listing 4. Przykłady implementacji detektora SVM dla różnych typów funkcji jądra.

```
1 def check_user_svm(users, samples, vector):
2
3     clf = svm.SVC()
4     clf.fit(samples, users)
5     result = clf.predict([vector])
6
7     return result
```

Kluczowym elementem jest tutaj metoda `predict()`, która jako argument przyjmuje wektor testowy użytkownika i z pomocą klasyfikatora SVM próbuje zidentyfikować od którego użytkownika pochodzi wektor testowy. Jako rezultat zwracany jest identyfikator

użytkownika. Jeśli jest on taki sam jak identyfikator użytkownika, dla którego podano wcześniej hasło, aplikacja uwierzytelnia użytkownika.

5.4.3. Serwer WWW

Aplikację uruchomiono na wirtualnym serwerze (**VPS**) z systemem operacyjnym Ubuntu 20.04 zlokalizowanym w środowisku chmurowym. W celu zwiększenia bezpieczeństwa projektu zdecydowano się na użycie rozwiązania typu reverse proxy [43]. Jako proxy wykorzystano oprogramowanie Apache w wersji 2.4.41. Zdecydowano się również na wykorzystanie dodatkowej warstwy uwierzytelnienia HTTP, aby zapobiec indeksowaniu aplikacji przez wyszukiwarki internetowe. Zapobiega to również nieautoryzowanemu dostępowi do panelu logowania aplikacji przez osoby trzecie, gdyż poświadczenia dostępu były przekazywane jedynie osobom, które uczestniczyły w badaniach. Poniżej przedstawiono produkcyjną konfigurację reverse proxy Apache.

Aby ułatwić dostęp przez przeglądarkę internetową, wykupiono domenę `typeauth.com`. W ten sposób po otwarciu adresu `http://www.typeauth.com` w przeglądarce internetowej, uzyskamy dostęp do stworzonej aplikacji. Przy korzystaniu z aplikacji internetowej nie ma konieczności instalacji żadnego dodatkowego oprogramowania, sterowników ani spełniania minimalnych wymagań systemowych. Każdy użytkownik, który posiada dostęp do nowoczesnej przeglądarki internetowej, może z powodzeniem korzystać z aplikacji Typeauth. Na korzyść wyboru aplikacji internetowej przemawia również łatwość przeprowadzania aktualizacji i dodawania nowych funkcji w oprogramowaniu. W przypadku konieczności aktualizacji kodu aplikacji, wystarczy jedynie pobrać nową wersję repozytorium na serwer produkcyjny, a następnie zrestartować aplikację. Jest to znacznie wygodniejsze, niż w przypadku rozwiązań typu "gruby klient", w których użytkownik musi przechodzić proces instalacji za każdym razem, gdy nowa wersja aplikacji jest udostępniona.

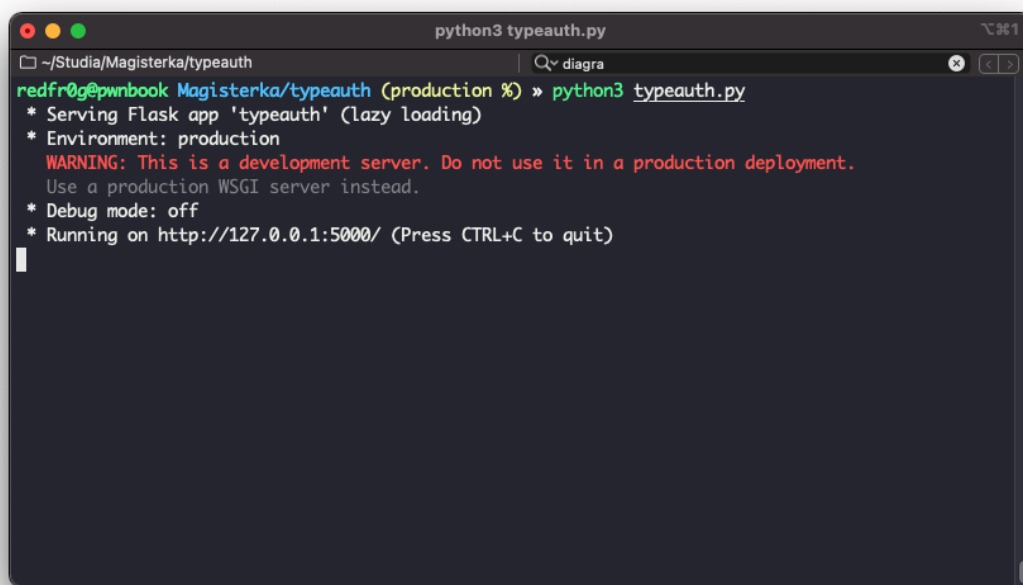
Aby zapewnić maksymalne bezpieczeństwo projektowanej aplikacji, zdecydowano się na skorzystanie z platformy chmurowej OVH, na której uruchomiono wirtualny serwer z systemem Ubuntu Linux. Administracyjny dostęp do serwera zrealizowano przy pomocy **SSH**, gdzie logowanie możliwe było jedynie przy pomocy klucza prywatnego. Na poziomie systemu operacyjnego utworzony został osobny użytkownik o niskich uprawnieniach, tak aby zminimalizować ryzyko eskalacji uprawnień w efekcie potencjalnego nieautoryzowanego dostępu.

Listing 5. Plik konfiguracyjny `typeauth.conf` zawierający ustawienia reverse proxy.

```
1 <VirtualHost *:80>
2     ProxyPreserveHost On
3
4     ProxyPass / http://localhost:5000/
5     ProxyPassReverse / http://localhost:5000/
6
7     Timeout 5400
8     ProxyTimeout 5400
9
10    ServerName www.typeauth.com
11    ServerAlias *.www.typeauth.com
12
13    <Proxy *>
14        Order deny, allow
15        Allow from all
16        AuthType Basic
17        AuthName "Password_Required"
18        AuthUserFile /etc/apache2/.htpasswd
19        Require valid-user
20    </Proxy>
21 </virtualhost>
```

5.4.4. Instalacja

Aby zainstalować i uruchomić aplikację, konieczne jest pobranie wszystkich katalogów i plików aplikacji. Następnie należy pobrać wszystkie potrzebne zależności wykorzystując narzędzie `pip` lub dowolny inny menadżer pakietów Python. Aplikacja Typeauth uruchamiamy wykonując polecenie `python typeauth.py` w głównym katalogu (Rysunek 5.4).



```
python3 typeauth.py
~/Studia/Magisterka/typeauth | Qv diagra
redfr0g@pwnbook Magisterka/typeauth (production %) » python3 typeauth.py
* Serving Flask app 'typeauth' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Rysunek 5.4. Przykład uruchomienia aplikacji Typeauth na lokalnym środowisku w systemie MacOS.

6. Badania

Poniższa sekcja podsumowuje przeprowadzone badania weryfikujące skuteczność rozwiązania oraz wyniki testów, które pozwoliły zdecydować o projekcie finalnej implementacji rozwiązania.

6.1. Próba badawcza

W ramach pracy poddano badaniom około 40 osób w wieku 18-60 lat. Badane osoby posiadały zróżnicowany poziom umiejętności korzystania z komputera oraz różniły się stylami oraz szybkością pisania na klawiaturze. Wszystkie wymogi dotyczące każdej badanej osoby przedstawiono poniżej:

- przedział wiekowy 18-60 lat,
- umiejętność obsługi komputera i klawiatury komputerowej na poziomie co najmniej podstawowym,
- płynna znajomość języka polskiego,
- znajomość języka angielskiego na poziomie co najmniej podstawowym,
- umiejętność pisania na klawiaturze komputerowej w języku polskim i angielskim.

6.2. Opis badania

Każda pojedyncza sesja badawcza składała się na wykonanie przez użytkownika poniższych kroków:

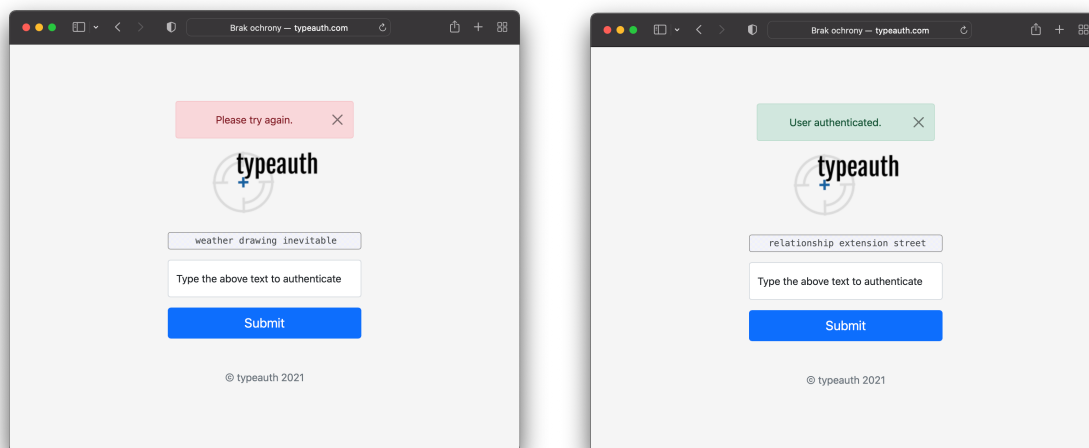
1. rejestracja w aplikacji Typeauth (przejście przez proces *onboarding*),
2. wykonanie 5 prób logowania,
3. wypełnienie anonimowej ankiety.

W ankiecie należało zamieścić odpowiedzi na poniższe pytania:

- Podczas badania, ile razy aplikacja poprawnie rozpoznała twój styl pisma (na 5 prób logowania)?
- Czy proces rejestracji był uciążliwy z perspektywy użytkownika? (TAK/NIE)
- Jak oceniasz wykorzystanie biometrii dynamiki pisania w procesie logowania? (Ocena subiektywna)

Ankieta zawierała także odpowiednie pole, w którym można było podzielić się dodatkowymi spostrzeżeniami lub uwagami. Odpowiedzi na to pytanie były brane pod uwagę podczas etapu opracowywania wniosków i określania pracy na przyszłość.

Podczas procesu logowania, użytkownik mógł się spotkać z dwoma typami komunikatów wysyłanych przez aplikację, co przedstawiono poniżej:



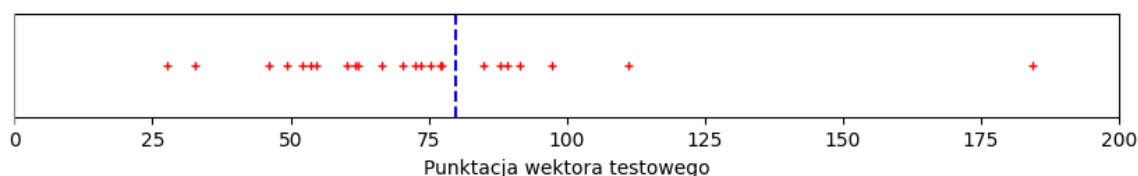
Rysunek 6.1. Komunikat otrzymywany w przypadku niepoprawnego (lewy) i poprawnego (prawy) logowania.

Po zebraniu wyników ankiet z każdej fazy badań, dla otrzymanych danych na temat udanych prób logowania obliczano parametry FAR i FRR oraz całkowitą skuteczność rozwiązania daną wzorem:

$$\text{Skuteczność(\%)} = 100 - (FAR + FRR)/2$$

6.3. Wybór klasyfikatora

W rozdziale Koncepcja wspomniane zostało, że projekt Typeauth został zaprojektowany z myślą o użyciu dwóch typów klasyfikatorów pozwalających na identyfikację użytkownika. Klasyfikator oparty na metryce Manhattan oraz klasyfikator SVM. Dla każdego klasyfikatora przeprowadzono kilkanaście testów, czego wynikiem było uzyskanie parametru FRR dla każdego z dwóch klasyfikatorów. Badania przeprowadzono dla detektora SVM z wartościami zmiennej pomocniczej C wynoszącymi **1.0** i **0.1**. Dla detektora Manhattan wartość progu decyzyjnego została ustawiona na **50**. Wizualizacja detektora Manhattan została przedstawiona na rysunku 6.3.



Rysunek 6.2. Wizualizacja prób logowania tego samego użytkownika przy wykorzystaniu detektora Manhattan. Niebieska przerywana linia przedstawia próg odcięcia ustawiony na wartość 80.

Taką wartość wybrano na podstawie wcześniejszych obserwacji i pomiarów parametrów FAR i FRR (dla wartości progu decyzyjnego 50 uzyskano najbardziej zbliżone wartości

FAR i FRR, przez co można przyjąć, że próg ten jest zbliżony do wartości EER). W tabeli 6.1 zamieszczono zbiorcze wartości FRR uzyskane na tym etapie badań.

Tabela 6.1. Wartości parametru FRR dla detektora Manhattan i SVM.

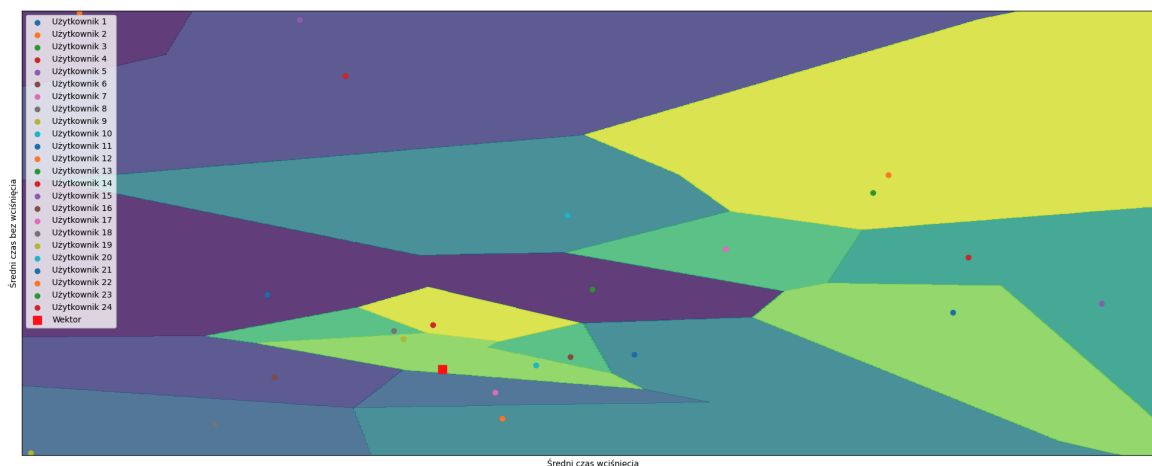
Typ detektora	FRR[%]
SVM (C=1.0)	60
SVM (C=0.1)	32
Manhattan	40

Można zaobserwować, że wartość FRR zmniejsza się wraz ze spadkiem wartości zmiennej pomocniczej C . Na podstawie wyników zdecydowano się finalnie wybrać detektor SVM z wartością zmiennej $C=0.1$ do końcowego rozwiązania. Klasyfikator został zaimplementowany w następujący sposób. Najpierw pobierane są wektory cech wszystkich użytkowników znajdujących się aktualnie w bazie. Dane te przekazywane są do klasyfikatora, który następnie stara się podzielić na klasy wszystkich użytkowników. Kolejnym krokiem jest wywołanie metody `predict()`. Metoda jako argument przyjmuje wektor testowy i przypisuje go do wcześniej sklasyfikowanych użytkowników. Jeśli użytkownik, który stara się uwierzytelnić, znajduje się w hiperpłaszczyźnie przypisanej do swojego użytkownika, to system potwierdza jego tożsamość i wysyła komunikat o poprawnym uwierzytelnieniu. Poniżej zamieszczono wizualizacje tego procesu w wymiarze średniego czasu wciśnięcia i zwolnienia klawisza. Współrzędne punktów na wykresie odpowiadają średnim parametrom czasowym użytkownika przechowywanymi w bazie danych. Czerwonym kwadratem oznaczono współrzędne wektora testowego użytkownika który aktualnie próbował się zalogować. Obszary, na które podzielono wykres, przedstawiają granice obszarów sklasyfikowanych przez detektor SVM. Jeśli współrzędne użytkownika znajdują się w obszarze, to zostanie on zidentyfikowany jako użytkownik, do którego jest przypisany ten obszar.

W późniejszych etapach badań, gdy w bazie znajdowało się kilkadziesiąt użytkowników, zauważono, że znacząco rośnie wartość parametru FRR. Takie zachowanie najprawdopodobniej brało się z faktu, że w system błędnie klasyfikuje użytkownika przypisując go do użytkownika o podobnym wektorze cech. Problem ten rozwiązano zmniejszając liczbę sklasyfikowanych użytkowników podczas próby logowania. Zamiast klasyfikować całą bazę użytkowników, aplikacja wybiera wektor próbującego się zalogować użytkownika oraz 6 innych losowych wektorów czasowych. Niweluje to błędy w klasyfikacji, gdy w bazie znajdują się użytkownicy o podobnym stylu pisania na klawiaturze lub, gdy jeden użytkownik jest zarejestrowany wielokrotnie. Dodatkowo przy mniejszej liczbie wektorów do klasyfikacji, algorytm SVM jest wydajniejszy.

Takie podejście tworzy jednak powierzchnię do potencjalnego ataku większościowego, w momencie, gdy atakujący kontroluje większość rekordów w bazie danych. Jeśli atakujący zarejestruje znaczącą większość użytkowników w bazie danych, z wektorem czasowym

odbiegającym od wektora użytkownika, to przy klasyfikacji istnieje duże prawdopodobieństwo prawidłowego uwierzytelnienia. Ryzyko wystąpienia takiej podatności może być zmniejszone poprzez stosowanie systemu w aplikacjach, gdzie istnieje ograniczona możliwość logowania lub ilość użytkowników jest na tyle duża, że przeprowadzenie takiego ataku jest niepraktyczne z powodu dużej liczby wymaganych rejestracji.

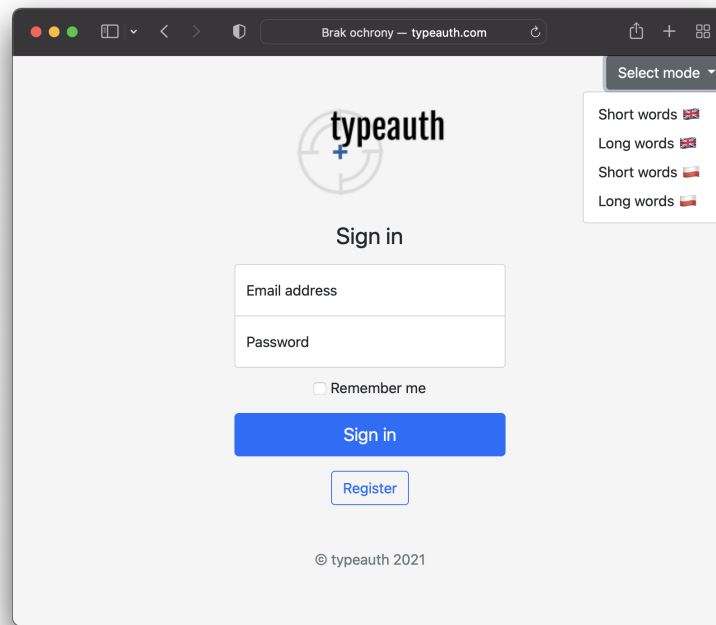


Rysunek 6.3. Wizualizacja klasyfikacji użytkowników z wykorzystaniem detektora SVM z liniową funkcją jądra i zmienną pomocniczą $C=0.1$.

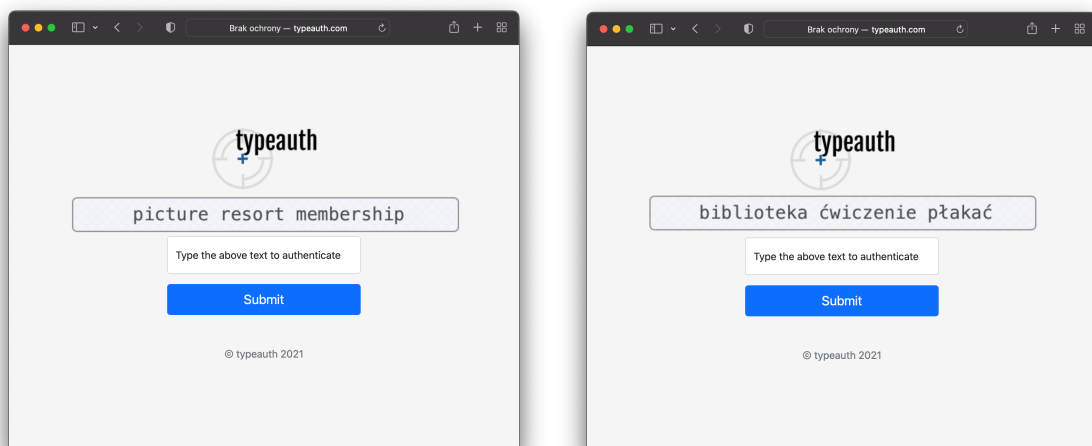
6.4. Badania skuteczności

Celem pracy było zbadanie tego, jaki wpływ ma język wzorca tekstowego na skuteczność uwierzytelnienia biometrycznego. W tym celu zaimplementowano możliwość zmiany języka w jakim generowane są frazy podczas procesu logowania lub rejestracji. Oprócz zmiany języka generowanych fraz, możliwa jest również zmiana długości generowanych fraz, przez co użytkownik musi przepisać mniej znaków, aby się uwierzytelnić.

Na poniższych grafikach można zauważyć jak zmiana trybu generowanego tekstu wpływała na wyświetlany przez aplikację tekst.

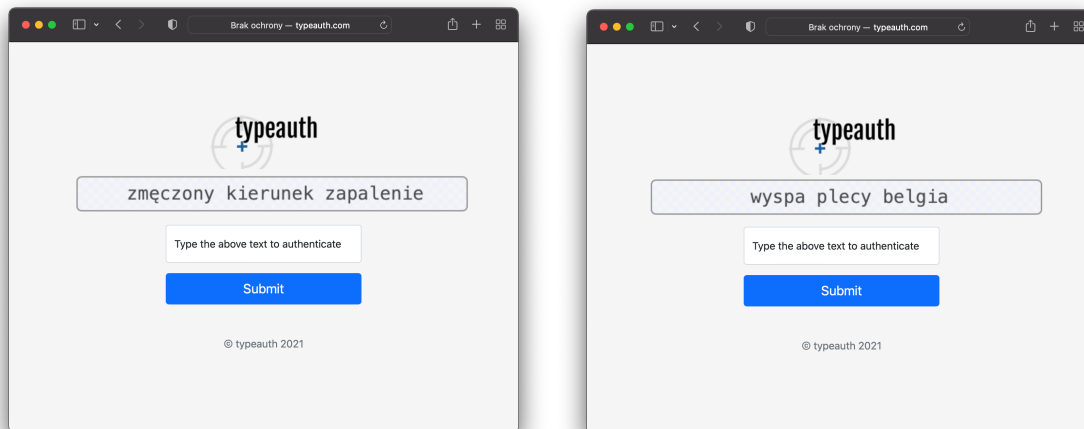


Rysunek 6.4. Funkcjonalność zmiany języka i długości generowanych fraz.



Rysunek 6.5. Przykład generacji fraz w języku angielskim i polskim.

Do badania wybrano 1339 polskich słów [44] oraz 1524 słów angielskich [45]. Następnie listy słów podzielono na listy zawierające słowa długie (dłuższe niż 6 liter dla języka polskiego i dłuższe niż 5 liter dla języka angielskiego) oraz słowa krótkie. Sumarycznie uzyskano 2863 słów, które są wyświetlane jako grupy 3 wyrazów przy logowaniu i rejestracji w celu tworzenia wzorca do zbierania danych biometrycznych. Poniżej przedstawiono przykład generacji wzorców tekstowych zawierających długie i krótkie słowa w języku polskim.



Rysunek 6.6. Przykład generacji wzorca tekstowego zawierającego długie i krótkie słowa.

Badania przeprowadzono w czterech etapach. Każdy etap dotyczył innej długości słów oraz innego języka według poniższego schematu:

- badanie 1 - wzorec tekstowy zawierający krótkie słowa w języku angielskim,
- badanie 2 - wzorec tekstowy zawierający długie słowa w języku angielskim,
- badanie 3 - wzorec tekstowy zawierający krótkie słowa w języku polskim,
- badanie 4 - wzorec tekstowy zawierający długie słowa w języku polskim.

Podczas przeprowadzania badań zauważono, że dla niektórych użytkowników przy wklejeniu losowego ciągu znaków i dopisaniu litery, następuje uwierzytelnienie. Takie zachowanie systemu było jednak zauważalne tylko dla pojedynczego użytkownika i najprawdopodobniej wynika to z cech dynamiki pisania dla tego użytkownika. Możliwe, że użytkownik ten posiada cechy pisania, które są zbliżone do zdarzenia wklejenia tekstu i dopisania dodatkowego znaku.

W tabeli 6.2 zamieszczono wyniki parametrów FRR i FAR dla wszystkich czterech badań dla detektora SVM ($C=0.1$) oraz systemie klasyfikacji 7 użytkowników przy logowaniu. Wyniki te są efektem kilkutygodniowych badań przeprowadzonych z użytkownikami na żywo lub zdalnie. Warto zaznaczyć, że dla każdego badania użytkownicy otrzymywali szczegółową instrukcję opisującą korzystanie z systemu, tak, aby generowane wyniki pochodziły jedynie od użytkowników poprawnie logujących się w aplikacji.

Tabela 6.2. Wartości parametrów FRR i FAR dla badań 1-4 z wykorzystaniem klasyfikatora SVM z metodą klasyfikacji 7 użytkowników.

Numer badania	FRR[%]	FAR[%]
1	35,56	30,00
2	40,00	25,00
3	36,36	15,00
4	27,50	15,00

Można zauważyć, że dla badania 3 oraz 4 parametry FRR i FAR są dużo niższe, niż dla ich odpowiedników w języku angielskim. Długość poszczególnych słów w generowanej frazie nie wpływa znacząco na FAR, FRR lub skuteczność. Dla języka polskiego widocznie lepiej sprawdzają się słowa dłuższe (różnica około 10% FRR). Taka tendencja wydaje się być prawidłowa, co można zauważyć również w [46], gdzie jako wzór do przepisania wykorzystano tylko jeden wyraz (stały wzorec tekstowy). W [46] fraza o długości kilkunastu znaków osiągnęła skuteczność większą o około 20%, niż fraza o długości 9 znaków. Dla języka angielskiego, słowa krótkie i długie osiągają zbliżone wyniki FRR i FAR (różnica 5% na korzyść wzorca zbudowanego ze słów krótkich). Następnie ze wszystkich badań obliczono średnie wartości FRR i FAR oraz skuteczność dla każdego języka.

Tabela 6.3. Wartości parametrów FRR i FAR dla wszystkich czterech badań.

Język	Średnie FRR[%]	Średnie FAR[%]	Skuteczność [%]
<i>ENG</i>	37,78	27,50	67,36
<i>PL</i>	31,93	15,00	76,54

Można zauważyć, że dla języka polskiego skuteczność jest wyższa niż dla języka angielskiego. Różnica ta wynosi około 10% na korzyść języka polskiego. Podobna tendencja dotyczy średnich wartości parametrów FRR i FAR. Wartość FRR dla języka polskiego jest mniejsza o około 6%, natomiast FAR jest mniejszy o 12%. W [47] porównywano skuteczność biometrii pomiędzy językami angielskim i włoskim. W przypadku wprowadzenia dwóch kolejnych próbek w języku włoskim i angielskim, wartość parametru FAR była większa dla języka angielskiego o kilka punktów procentowych. Podobny trend możemy również zauważyć w [48], gdzie porównywano skuteczność biometrii w kontekście języków arabskiego i angielskiego. Tam dla detektora SVM, parametr FAR dla języka angielskiego i arabskiego wyniósł odpowiednio 24,5% i 16,9%. Parametr FRR wynosił odpowiednio 61,3% i 42,3%. W niniejszej pracy różnice w FRR i FAR są mniejsze niż w [48]. Wynika to najprawdopodobniej z faktu, że składnia języka arabskiego i angielskiego różni się dużo bardziej, niż języka polskiego i angielskiego. Tutaj różnice są dużo mniejsze biorąc pod uwagę fakt, że język polski i angielski korzystają z tego samego alfabetu.

6.5. Subiektywna ocena rozwiązania

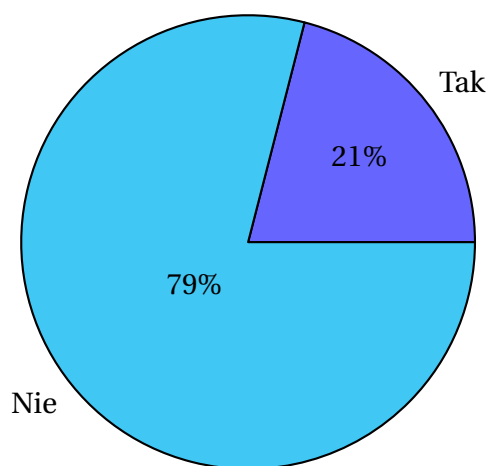
Oprócz zmierzenia skuteczności rozwiązania, zebrano również dane mające ocenić stosunek badanych do zastosowania klawiatury w kontekście biometrii. W tym celu badani musieli odpowiedzieć na dwa pytania:

1. Czy proces rejestracji był uciążliwy z perspektywy użytkownika?
2. Jak oceniasz wykorzystanie biometrii dynamiki pisania w procesie logowania?

Na pierwsze pytanie możliwa była jedynie odpowiedź “Tak” lub “Nie”. Drugie pytanie było pytaniem zamkniętym, i możliwe było zaznaczenie jednej z poniższych odpowiedzi:

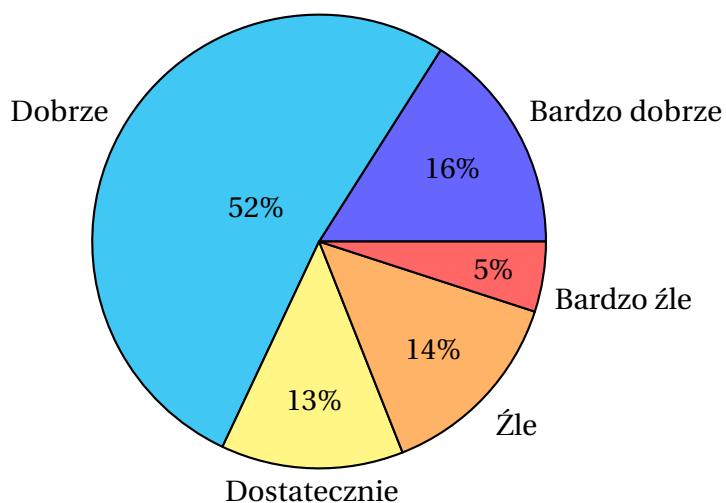
- Bardzo źle
- Źle
- Dostatecznie
- Dobrze
- Bardzo dobrze

Poniżej zamieszczono procentowe rozłożenie odpowiedzi na pytanie o uciążliwość procesu rejestracji. Pytanie to miało na celu oszacowanie tego, jak użytkownik odbiera fakt, że przy zakładaniu konta należy wielokrotnie przepisać wyświetlany tekst, co może być dla niego uciążliwe.



Rysunek 6.7. Rozłożenie procentowe odpowiedzi na pytanie o uciążliwość procesu logowania.

Odpowiedzi na pytanie o subiektywną ocenę wykorzystania biometrii w projektowanym rozwiązaniu przedstawiono na wykresie poniżej.



Rysunek 6.8. Rozłożenie procentowe odpowiedzi na pytanie o subiektywną ocenę projektowanego systemu.

Na podstawie uzyskanych danych można wywnioskować, że użytkownicy końcowi dobrze oceniają wykorzystanie biometrii pisania na klawiaturze w procesie logowania. Jedynie 19% użytkowników źle lub bardzo źle oceniło proces logowania biometrycznego w aplikacji Typeauth. Prawie 80% ankietowanych odpowiedziało że proces rejestracji (onboardingu) nie był uciążliwy.

6.6. Sugestie użytkowników

Dodatkowo uzyskano kilka wartościowych sugestii pochodzących z ostatniego pytania otwartego. Jeden z użytkowników sugerował podatność rozwiązania na atak typu *replay*. Taki scenariusz byłby możliwy poprzez zastosowanie urządzenia lub skryptu rejestrującego znaki wprowadzane przez użytkownika tzw. *keylogger*. Jeśli atakujący dostatecznie długo będzie rejestrował styl pisma ofiary, to z powodzeniem mógłby odtworzyć jego znaczki czasowe pisania i tym samym zalogować się do aplikacji. Pozostałe sugestie dotyczyły głównie dużej zależności rozwiązania od czynników środowiskowych takich jak:

- urządzenie na którym użytkownik się loguje,
- aktualny stan użytkownika (stres, zdenerwowanie itp.),
- konieczność wykorzystania fizycznej klawiatury.

Powyższe czynniki przyczyniają się do ogólnego zmniejszenia skuteczności rozwiązania oraz zmniejszają jego dostępność dla osób z ograniczoną sprawnością co wynika z faktu konieczności korzystania z klawiatury.

7. Podsumowanie

Pobocznym celem pracy dyplomowej było zaprojektowanie i implementacja systemu biometrycznego umożliwiającego identyfikację użytkowników na podstawie cech ich pisma na klawiaturze. Za cel główny postawiono sobie zbadanie tego, jaki wpływ na skuteczność takiego systemu ma język wzorca tekstowego, który użytkownicy musieli przepisywać podczas logowania. Wzorzec tekstowy generowany był w języku polskim lub angielskim, a wszyscy badani użytkownicy płynnie posługiwali się językiem angielskim, natomiast język polski był ich językiem ojczystym. Efektem prac było stworzenie aplikacji Typeauth, która umożliwiała rejestrację, logowanie oraz uwierzytelnienie użytkowników bazując na cechach biometrycznych ich charakteru pisma. Aplikacja zbierała następujące parametry w celu identyfikacji użytkownika:

- znacznik czasowy wciśnięcia klawisza klawiatury,
- znacznik czasowy zwolnienia klawisza klawiatury,
- zmienną logiczną czy użyto klawisza Caps Lock,
- zmienną całkowitoliczbową zawierającą ilość wciśnięcia przycisku Shift,
- zmienną całkowitoliczbową zawierającą ilość wciśnięcia przycisku Backspace.

Na podstawie tych parametrów tworzony był wektor cech opisujący każdego użytkownika. Wektor był przechowywany w bazie danych SQLite, z której był pobierany podczas procesu logowania w celu klasyfikacji użytkownika. Aplikację internetową Tyepauth zbudowano z następujących elementów:

- frontend aplikacji bazujący na Bootstrap i własnych modułach Javascript umożliwiających zbieranie znaczników czasowych,
- backend zaprogramowany w języku Python z wykorzystaniem frameworku Flask oraz SQL Alchemy,
- baza danych SQLite przechowująca dane logowania użytkowników i ich dane biometryczne,
- infrastruktura serwerowa bazująca na systemie Ubuntu i oprogramowaniu Apache umożliwiającą dostęp do aplikacji z internetu.

Aplikację udostępniono w domenie internetowej typeauth.com, a dostęp odpowiednio zabezpieczono tak, aby osoby nieautoryzowane nie mogły otworzyć strony z aplikacją. Następnie przeprowadzono badania z udziałem 40 osób, których ojczystym językiem był język polski. Każda z badanych osób płynnie posługiwała się językiem polskim oraz angielskim.

7.1. Podsumowanie wyników

Pierwszym etapem badań było zweryfikowanie tego, który z klasyfikatorów – Manhattan czy SVM, wykazuje lepszą skuteczność. Na podstawie wyników z tego etapu zdecydowano, że do finalnych badań wykorzystany zostanie klasyfikator SVM. Następnie

przeprowadzono cztery badania, na podstawie których wyliczono parametry FRR i FAR oraz skuteczność końcowego rozwiązania. Analizując otrzymane wyniki stwierdzono, że w rozwiązaniach biometrycznych bazujących na stylu pisania na klawiaturze lepszym wyborem języka wzorca tekstowego, jeśli z systemu korzystają użytkownicy polskojęzyczni, jest język polski. Na podstawie średnich parametrów FAR i FRR można zauważyć, że język polski uzyskuje FRR niższe o około 8%, natomiast FAR niższe o około 10%. Dla języka polskiego skuteczność rozwiązania była lepsza o około 10%. Wyniki, które otrzymano, jednoznacznie wskazują, że w przypadku użytkowników polskojęzycznych, skuteczniejszy będzie wybór języka polskiego do generacji wzorca tekstowego w procesie rejestracji i logowania. Jeśli chodzi o wpływ długości fraz, w przypadku języka polskiego lepszym wyborem będzie stosowanie fraz składających się z wyrazów dłuższych niż 6 znaków.

7.2. Praca na przyszłość

Aplikacja Typeauth, którą zaprojektowano na potrzeby pracy, znajduje się w bardzo wczesnym etapie rozwoju. Pomimo faktu, że dowiedziona została skuteczność projektowanego rozwiązania, konieczna jest dalsza praca, aby aplikacja była w pełni funkcjonalna i skuteczna na zadowalającym poziomie. Pomimo tego, skuteczność aplikacji Typeauth nie jest na poziomie rozwiązań które prezentowano w Tabeli 3.1. Może to wynikać z faktu, że z parametrów czasowych wydobywano zbyt mało cech stylu pisma użytkownika przez co system błędnie klasyfikował niektórych użytkowników. Powodem może być również to, że nie wszystkie badania prowadzono stacjonarnie, przez co niektórzy z użytkowników mieli trudności aby zrozumieć działanie systemu. Przykładowo, jeśli użytkownik popełniał dużo błędów na etapie rejestracji, a potem na etapie logowania jego ilość błędów zmniejszała się, to użytkownik ten mógł doświadczyć odmowy uwierzytelnienia ze względu na brak użycia klawisza Backspace przy logowaniu.

Aby polepszyć skuteczność rozwiązania, konieczne jest przeprowadzenie dalszych badań oraz wykorzystanie większej ilości cech pisma zbieranych od użytkownika. Użycie dodatkowych cech, takich jak opóźnienia czasowe wciśnięcia i zwolnienia przycisku (*Press Latency*, *Release Latency*) [24], mogłoby zwiększyć skuteczność detektora SVM. Ciekawą propozycją byłoby również obliczanie prędkości pisania na klawiaturze lub mierzenie tego, jak bardzo poszczególne wciśnięcia klawiszy nakładają się na siebie w dziedzinie czasu (*overlapping*), jak zaproponowano w [49]. Wykorzystując dodatkowe cechy wydobyte ze znaczników czasowych, zadowalającym osiągnięciem byłoby osiągnąć skuteczność powyżej 90%, co wielokrotnie uzyskiwano już w podobnych implementacjach [50], [51].

Oprócz zwiększenia skuteczności rozwiązania, ciekawym kierunkiem badań wydaje się być ułatwienie korzystania z systemu biometrycznego przez użytkownika końcowego. W prezentowanej pracy użytkownik, aby się zalogować lub zarejestrować, musi wielokrotnie przepisać tekst wyświetlany na ekranie. Przykładowo, zamiast przepisywać losowo

generowane frazy (*free text*), aplikacja mogłaby zacząć zbieranie znaczników czasowych już na etapie podawania hasła oraz loginu [52].

Bibliografia

- [1] jamf, "Phishing Trends Report 2021", 2021. adr.: <https://resources.jamf.com/documents/white-papers/phishing-trends-report-2021.pdf>.
- [2] M. Solomon i M. Chapple, *Information Security Illuminated*. Jones i Barlett Publishers, 2004, ISBN: 9780763726775.
- [3] S. Y. Pin Shen Teh Andrew Beng Jin Teoh, "A Survey of Keystroke Dynamics Biometrics", *The Scientific World Journal*, 2013.
- [4] D. Bhattacharyya, R. Ranjan, A. A. Farkhod i M. Choi, "Biometric Authentication: A Review", *International Journal of u-and e-Service*, t. 2, 3 2009.
- [5] "Data Snapshot: Biometrics in the workplace commonplace, but are they secure? - Spiceworks", adr.: <https://community.spiceworks.com/security/articles/2952-data-snapshot-biometrics-in-the-workplace-commonplace-but-are-they-secure>.
- [6] A. C. Weaver, "Biometric authentication", *Computer*, t. 39, s. 96–97, 2 lut. 2006, ISSN: 00189162. DOI: 10.1109/MC.2006.47.
- [7] M. A. Sasse i K. Krol, "Usable biometrics for an ageing population 1", adr.: <http://digital-library.theiet.org/content/books/pc/pbsp010e>.
- [8] A. Babich, "Biometric Authentication. Types of biometric identifiers",
- [9] B. Jerman-Blažič i T. Klobučar, *Erratum to: Advanced Communications and Multimedia Security*. 2017, E1–E1. DOI: 10.1007/978-0-387-35612-9_23.
- [10] R. Giot, B. Hemery, C. Rosenberger i C. R. Low, "Cost and Usable Multimodal Biometric System Based on Keystroke Dynamics and 2D Face Recognition", t. 4, 2010. DOI: 10.1109/ICPR.2010.282. adr.: <https://hal.archives-ouvertes.fr/hal-00503103>.
- [11] H. Vallabhu i R. V. Satyanarayana, "Biometric Authentication as a Service on Cloud: Novel Solution", *International Journal of Soft Computing and Engineering (IJSCE)*, s. 163, 2 2012. adr.: http://www.sans.org/reading_room/whitepapers/authentication/biomet.
- [12] J. Hu, D. Gingrich i A. Sentosa, "A k-Nearest Neighbor Approach for User Authentication through Biometric Keystroke Dynamics",
- [13] "Hardware Implementation of Dynamics Keystroke Applied for Cloud Computing",
- [14] S. I. of Technology, I. E. D. Society, I. of Electrical i E. Engineers, *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC 2017) : 10-11, February 2017*, ISBN: 9781509032433.
- [15] K. Revett, F. Gorunescu, M. Gorunescu, M. Ene, S. T. D. Magalhães i H. M. D. Santos, "A machine learning approach to keystroke dynamics based user authentication", *Int. J. Electronic Security and Digital Forensics*, t. 1, 1 2007.
- [16] A. Alsultan i K. Warwick, "User-Friendly Free-Text Keystroke Dynamics Authentication for Practical Applications Wireless power sundries: Ideas on circuits, theory and configurations View project Alarm Handling & Fault Analysis View project

- User-Friendly Free-text Keystroke Dynamics Authentication for Practical Applications”, 2013. DOI: 10.1109/SMC.2013.793. adr.: <https://www.researchgate.net/publication/257840711>.
- [17] J. A. Robinson, V. M. Liang, J. A. Chambers i C. L. MacKenzie, “Computer user verification using login string keystroke dynamics”, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, t. 28, s. 236–241, 2 1998, ISSN: 10834427. DOI: 10.1109/3468.661150.
 - [18] D. Iliev i Y. B. Sun, “Website forgery prevention”, 2010, ISBN: 9781424459438. DOI: 10.1109/ICISA.2010.5480293.
 - [19] T. Olzak, “Keystroke Dynamics: Low Impact Biometric Verification”, 2006.
 - [20] J. Montalvão, C. Augusto, S. Almeida i E. O. Freire, “Equalization of keystroke timing histograms for improved identification performance”,
 - [21] A. K. (Nethone), “User identification based on keystroke dynamics”, 2019. adr.: https://downloads.ctfassets.net/kisruz03w7zs/3Ve0Gs5oWFB3o2cpamGG0t/46e5f9da9a7b09a2a095c092dfd10d61/Cybersec_expo_useridentityfication_AK_Nethone.pdf.
 - [22] typingdna, “Student identity validation with keystroke dynamics”, 2020. adr.: <https://blog.typingdna.com/wp-content/uploads/2020/05/typingdna-elearning-whitepaper.pdf>.
 - [23] K. Lv, J. Liu, P. Tang i Q. Li, “Association for Information Systems AIS Electronic Library (AISeL) Keystroke Biometrics for Freely Typed Text Based on CNN model”, s. 6–26, adr.: <https://aisel.aisnet.org/pacis2018/291>.
 - [24] A. Acien, A. Morales, R. Vera-Rodriguez, J. Fierrez i J. V. Monaco, “TypeNet: Scaling up Keystroke Biometrics”,
 - [25] S. Krishnamoorthy, “Identification of User Behavioural Biometrics for Authentication using Keystroke Dynamics and Machine Learning”, adr.: <https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=8442&context=etd>.
 - [26] F. Monrose i A. Rubin, “Authentication via Keystroke Dynamics”, 1997.
 - [27] H. Saevanee i P. Bhattarakosol, “Authenticating user using keystroke dynamics and finger pressure”, 2009, ISBN: 9781424423095. DOI: 10.1109/CCNC.2009.4784783.
 - [28] T. Shimshon, R. Moskovitch, L. Rokach i Y. Elovici, “Clustering di-graphs for continuously verifying users according to their typing patterns”, 2010, s. 445–449, ISBN: 9781424486809. DOI: 10.1109/EEEI.2010.5662182.
 - [29] A. Morales, A. Acien, J. Fierrez, J. V. Monaco, R. Tolosana, R. Vera-Rodriguez i J. Ortega-Garcia, “Keystroke Biometrics in Response to Fake News Propagation in a Global Pandemic”,
 - [30] M. D. Carmen, S. Medrano, M. Carro i J. Caballero, “Enhancing Online Banking Authentication Using Keystroke Dynamics”, 2017.
 - [31] K. S. Killourhy i R. A. Maxion, “Comparing anomaly-detection algorithms for keystroke dynamics”, 2009, s. 125–134, ISBN: 9781424444212. DOI: 10.1109/DSN.2009.5270346.

-
- [32] Y. Zhong, Y. Deng i A. Jain, “Keystroke dynamics for user authentication”, czer. 2012, s. 117–123, ISBN: 978-1-4673-1611-8. DOI: 10.1109/CVPRW.2012.6239225.
 - [33] B. Schh, “The Kernel Trick for Distances”, 2000.
 - [34] *1.4. Support Vector Machines — scikit-learn 1.0.1 documentation*, <https://scikit-learn.org/stable/modules/svm.html>, Accessed: 2021-12-06.
 - [35] V. R. Wadekar i R. N. Patil, “Personal Identification And Verification Using Multimodal Biometrics”, *Journal of Engineering Research and Applications* www.ijera.com, t. 3, s. 1560–1567, adr.: www.ijera.com.
 - [36] J. Malik, D. Girdhar, R. Dahiya i G. Sainarayanan, “Reference Threshold Calculation for Biometric Authentication”, *Image, Graphics and Signal Processing*, t. 2, s. 46–53, 2014. DOI: 10.5815/ijigsp.2014.02.06. adr.: <http://www.mecs-press.org/>.
 - [37] F. A. Aslam, H. N. M. J. M. M. M. A. Gulamgaus i P. S. L. A. Professor, “Efficient Way Of Web Development Using Python And Flask”, *International Journal of Advanced Research in Computer Science*, t. 6, 2. adr.: www.ijarcs.info.
 - [38] *Github Language Stats*. adr.: https://madnight.github.io/githut/#/pull_requests/2021/3.
 - [39] E. Chukharev-Hudilainen, “Empowering Automated Writing Evaluation with Keystroke Logging”, *Observing Writing*, s. 125–142, sty. 2019. DOI: 10.1163/9789004392526_007. adr.: <https://brill.com/view/book/edcoll/9789004392526/BP000006.xml>.
 - [40] M. Bayer, “SQLAlchemy”, w *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*, A. Brown i G. Wilson, red., aosabook.org, 2012. adr.: <http://aosabook.org/en/sqlalchemy.html>.
 - [41] S. T. Bhosale, M. T. Patil i M. P. Patil, “SQLite International Journal of Computer Science and Mobile Computing SQLite: Light Database System”, *International Journal of Computer Science and Mobile Computing*, t. 4, s. 882–885, 2015. adr.: <https://www.researchgate.net/publication/279621848>.
 - [42] *scipy.spatial.distance.cityblock — SciPy v1.7.0 Manual*. adr.: <https://docs.scipy.org/doc/scipy-1.7.0/reference/reference/generated/scipy.spatial.distance.cityblock.html#scipy.spatial.distance.cityblock>.
 - [43] A. Stricek, “A Reverse Proxy Is A Proxy By Any Other Name”, 2002. adr.: www.mysite.com,.
 - [44] *Indeks:Polski - Najpopularniejsze słowa 1-2000 – Wikisłownik, wolny słownik wielojęzyczny*. adr.: https://pl.wiktionary.org/wiki/Indeks:Polski_-_Najpopularniejsze_s%C5%5C%82owa_1-2000.
 - [45] *stuff/english-nouns.txt at main · hugsy/stuff*. adr.: <https://github.com/hugsy/stuff/blob/main/random-word/english-nouns.txt>.
 - [46] M. Rybnik, P. Panasiuk i K. Saeed, “User authentication with keystroke dynamics using fixed text”, 2009, s. 70–75, ISBN: 9780769536927. DOI: 10.1109/ICBAKE.2009.42.

- [47] D. Gunetti, C. Picardi i G. Ruffo, “Keystroke Analysis of Different Languages: A Case Study”,
- [48] A. Alsultan, K. Warwick i H. Wei, “Free-text keystroke dynamics authentication for Arabic language”, ISSN: 2047-4938. DOI: 10.1049/iet-bmt.2015.0101. adr.: www.ietdl.org.
- [49] M. Rybnik, M. Tabedzki i K. Saeed, “A keystroke dynamics based system for user identification”, 2008, s. 225–230, ISBN: 9780769531847. DOI: 10.1109/CISIM.2008.8.
- [50] H. Ali, M. Salami, W. Martono, M. Jimoh i E. Salami, “Keystroke Pressure-Based Typing Biometrics Authentication System Using Support Vector Machines Optimal trajectory using Genetic Algorithms View project Keystroke Pressure-Based Typing Biometrics Authentication System Using Support Vector Machines”, *LNCS*, t. 4706, s. 85–93, 2007. DOI: 10.1007/978-3-540-74477-1_8. adr.: <https://www.researchgate.net/publication/225151596>.
- [51] G. R. Online, Y. Li, B. Zhang, Y. Cao, S. Zhao, Y. Gao i J. Liu, “Study on the BeiHang Keystroke Dynamics Database Author Copyright Statement Study on the BeiHang Keystroke Dynamics Database”, 2011. DOI: 10.1109/IJCB.2011.6117485. adr.: <http://hdl.handle.net/10072/43574>.
- [52] I. Traore, I. Woungang, M. S. Obaidat, Y. Nakkabi i I. Lai, “Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments”, 2012, s. 138–145. DOI: 10.1109/ICDH.2012.59.

Wykaz symboli i skrótów

- C** – ang. *Regularization parameter*, zmienna pomocnicza C
- CIA** – ang. *confidentiality, Integrity, Availability*
- CNN** – ang. *Convolutional Neural Network*
- DDoS** – ang. *Distributed Denial of Service*
- DoS** – ang. *Denial of Service*
- DT** – Czas pomiędzy wciśnięciem i zwolnieniem tego samego klawisza
- EER** – ang. *Equal Error Rate*
- FAR** – ang. *False Acceptance Rate*, ilość akceptacji nieprawidłowego użytkownika
- FRR** – ang. *False Reject Rate*, ilość odrzuceń prawidłowego użytkownika
- FT** – Czas pomiędzy zwolnieniem i wciśnięciem następujących po sobie klawiszy
- HTML** – ang. *HyperText Markup Language*, hipertekstowy język znaczników
- IP** – Adres internetowy
- PP** – Czas pomiędzy wciśnięciami następujących po sobie klawiszy
- PT** – Czas wciśnięcia klawisza
- RBF** – ang. *Radial Basis Function*, radialna funkcja bazowa
- RNN** – Rekurencyjne sieci neuronowe
- RR** – Czas pomiędzy zwolnieniami następujących po sobie klawiszy
- SQL** – ang. *Structured Query Language*, język strukturalnych zapytań bazodanowych
- SVM** – Maszyna wektorów nośnych
- U2F** – ang. *Universal Second Factor*
- VPS** – ang. *Virtual Private Server*, prywatny serwer wirtualny
- WWW** – ang. *World Wide Web*
- XSS** – ang. *Cross-Site Scripting*

Spis rysunków

1.1	Model CIA [2].	11
3.1	Diagram przepływu przedstawiający sposób działania mechanizmu biometrycznego.	15
3.2	Schemat przedstawiający parametry czasowe wykorzystywane w biometrii dynamiki pisanie na klawiaturze.	16
4.1	Wizualizacja klasyfikacji dwóch zbiorów punktów z wykorzystaniem klasyfikatora SVM bazującego na liniowej funkcji jądra (lewy) oraz z wykorzystaniem klasyfikatora SVM bazującego radialnej funkcji bazowej (RBF) (prawy).	24
4.2	Diagram przepływu procesu rejestracji użytkownika.	25
4.3	Diagram przepływu procesu logowania użytkownika.	25
4.4	Wykres parametrów FRR oraz FAR od wartości progowej z zaznaczeniem optymalnego punktu EER.	26
5.1	Ekran logowania aplikacji Typeauth.	28
5.2	Schemat bazy danych SQLite wykorzystanej w ramach badań.	30
5.3	Listing wszystkich katalogów i plików aplikacji Typeauth.	31
5.4	Przykład uruchomienia aplikacji Typeauth na lokalnym środowisku w systemie MacOS.	36
6.1	Komunikat otrzymywany w przypadku niepoprawnego (lewy) i poprawnego (prawy) logowania.	38
6.2	Wizualizacja prób logowania tego samego użytkownika przy wykorzystaniu detektora Manhattan. Niebieska przerywana linia przedstawia próg odcięcia ustawiony na wartość 80.	38
6.3	Wizualizacja klasyfikacji użytkowników z wykorzystaniem detektora SVM z liniową funkcją jądra i zmienną pomocniczą $C=0.1$	40
6.4	Funkcjonalność zmiany języka i długości generowanych fraz.	41
6.5	Przykład generacji fraz w języku angielskim i polskim.	41
6.6	Przykład generacji wzorca tekstowego zawierającego długie i krótkie słowa.	42
6.7	Rozłożenie procentowe odpowiedzi na pytanie o uciążliwość procesu logowania.	44
6.8	Rozłożenie procentowe odpowiedzi na pytanie o subiektywną ocenę projektowanego systemu.	44

Spis tabel

3.1 Zestawienie przykładowych rozwiązań w zakresie biometrii cech pisanego na klawiaturze.	19
6.1 Wartości parametru FRR dla detektora Manhattan i SVM.	39
6.2 Wartości parametrów FRR i FAR dla badań 1-4 z wykorzystaniem klasyfikatora SVM z metodą klasyfikacji 7 użytkowników.	42
6.3 Wartości parametrów FRR i FAR dla wszystkich czterech badań.	43

Spis załączników

1. Wektory cech badanych użytkowników	56
---	----

Załącznik 1. Wektory cech badanych użytkowników

Tabela Załącznik 1.1. Tabela zawierające wektory cech użytkowników zarejestrowanych podczas badania rozwiązania Typeauth.

Id	Hold mean	Hold median	Idle mean	Idle median	Shift count	Bckspc count	Cpslck count
1	97,77	94,0	120,76	97,25	0	0	0
2	89,90	89,75	422,53	307,5	0	0	0
3	107,96	106,62	126,60	86,12	0	2	0
4	105,47	102,25	196,41	143,62	0	0	0
5	110,00	98,12	279,23	191,25	0	0,25	0
6	104,42	90,62	167,41	121,62	0	0,25	0
7	82,98	75,25	347,64	179,0	0	0	0
8	103,24	89,12	93,74	89,25	1	0,25	0
9	71,45	69,75	78,04	81,75	0	0	0
10	50,55	43,25	63,94	72,25	0	0	0
11	111,48	104,37	180,71	133,87	1	0	0
12	119,04	91,25	112,27	84,75	1	0	0
13	104,07	126,62	81,01	72,25	0	0	0
14	178,22	91,75	138,83	82,25	1	0,75	0
15	156,93	139,25	160,12	124,5	1	0,75	0
16	172,07	106,25	137,42	101,5	1	0,25	0
17	111,76	98,5	111,30	100,5	1	0,25	0
18	129,40	82,37	164,29	81,87	1	2,25	0
19	91,79	83,37	124,15	97,25	1	0,5	0
20	92,81	83,87	120,10	85,62	1	0,75	0
21	107,88	106,25	107,07	90,12	1	0,5	0
22	155,18	100,5	132,93	100,37	1	2	0
23	147,86	96,0	200,73	115,0	1	1,75	0
24	146,12	119,87	191,87	125,5	1,5	3,25	0
25	77,35	76,62	141,95	137,0	0	0,25	0
26	56,06	57,5	280,34	201,62	0	0	0
27	114,25	111,0	144,45	121,75	0	0,75	0
28	86,26	84,62	249,33	148,75	0	0,5	0
29	81,09	80,5	276,85	163,5	0	0	0
30	78,19	82,0	101,27	77,5	0	1	0
31	93,09	100,87	79,49	79,5	0	0	0

32	109,97	121,12	106,37	99,75	0	0,75	0
33	91,71	74,75	175,91	96,0	0	0,25	0
34	114,94	96,5	153,94	115,37	0	0,75	0
35	63,08	60,75	374,98	340,62	0	0	0
36	107,82	91,0	259,20	108,62	0	1	0
37	111,89	84,0	138,48	99,25	0	1	0
38	125,78	120,5	414,44	333,5	0	0,25	0
39	83,31	80,75	93,91	78,75	0	0,75	0
40	80,13	78,25	460,00	408,25	0	0	0
41	133,18	110,5	260,11	146,25	0	0,25	0
42	81,00	70,25	109,93	72,25	0	1,75	0
43	106,71	73,0	273,56	167,87	0	0,5	0
44	170,97	100,75	422,42	309,37	0	0,5	0
45	105,16	105,0	110,92	83,0	0	0,75	0
46	75,76	79,0	69,43	62,87	0	0,5	0
47	89,58	89,75	116,45	99,37	0	0,25	0
48	76,53	70,0	191,04	170,37	0	0,25	0
49	90,41	92,0	142,53	125,12	0	0	0
50	81,83	76,37	138,09	102,0	0	0	0
51	88,32	76,25	104,33	71,5	0	0,25	0
52	226,64	106,75	146,19	108,62	0	2	0
53	156,40	97,0	102,85	96,0	0	1,25	0
54	91,74	93,37	86,08	89,62	0	0	0
55	93,71	91,12	217,08	115,87	0	0	0
56	94,13	77,25	161,32	111,12	0	1,75	0
57	82,15	77,87	124,65	85,87	0	0,75	0
58	68,21	64,25	108,01	75,75	0	0,25	0
59	133,79	120,62	151,93	119,87	0	2,5	0
60	94,26	91,0	238,88	130,62	0	0	0
61	120,51	79,25	396,39	238,75	0	1	0
62	65,37	64,0	352,95	214,25	0	0,5	0
63	89,40	75,75	121,44	98,37	0,25	0,5	0
64	115,09	102,37	115,49	94,75	0	0,25	0
65	124,02	133,0	241,47	163,87	0	1	0