My Behavior is my Privacy & Secure Password !

Denis Migdal, Christophe Rosenberger

My Behavior is my Privacy & Secure Password !

Denis Migdal[1] , Christophe Rosenberger[1]
[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

{denis.migdal, christophe.rosenberger}@ensicaen.fr

April 2019

Abstract

Many studies propose strong user authentication based on biometric modalities. However, they often either, assume a trusted component, are modality-dependant, use only one biometric modality, are reversible, or does not enable the service to adapt the security on-the-fly. A recent work [1] introduced the concept of Personal Identity Code Respecting Privacy (PICRP), a non-cryptographic and non-reversible signature computed from any arbitrary information. In this paper, we extend this concept with the use of Keystroke Dynamics, IP and GPS geolocation by optimizing the pre-processing and merging of collected information. We demonstrate the performance of the proposed approach through experimental results and we present an example of its usage.
Keywords: Personal information; Behavioural biometrics; Privacy protection; BioHashing; Keystroke Dynamics; Authentication; Location.

1 Introduction

When browsing the Internet, users disclose information that enable their authentication, they can be behavioral (e.g. Mouse or Keystroke Dynamics), what the user is (e.g. his/her face through the webcam), what the user knows (e.g. a password), what the user has (e.g. the browser, OS). The new European General Data Protection Regulation (GDPR) establishes rules in order to protect user privacy and ensure his/her consent. These modalities have thus to be used by the service to authenticate users, but, as possible, without knowing or enabling an attacker to know, these modalities.

1.1 State of the art

Biometric authentication is a well-studied subject in the literature, however, proposed solutions often either, assume a trusted component, are modality-dependant, use only one biometric modality, are reversible, or does not enable the service to adapt the security on-the-fly. Trusted computing using secure element or sensors often gives the best security, but requires the possession of a specific hardware that a user might not possess. Such solutions thus only protect owner of such specifics hardware, that might be lacking in desktop or laptop computers. It also assumes that such devices are trusted and cannot be attacked.

Homomorphic or Functional encryption [2] enables to compare biometric modalities in the encrypted domain, i.e. without having the knowledge of the content. However, the encrypted data often require a lot of memory space, that can be repellent for a web service. Other solutions can be mono-modal or built for a specific modality. Some of which, like Zero Knowledge Protocol [3], does not enable the change of the security level on the fly.

1.2 PICRP

A recent work [1] introduced the concept of Personal Identity Code Respecting Privacy (PICRP), a non-cryptographic and non-reversible signature computed from arbitrary personal data. PICRP is described as a cancelable non-reversible code enabling the similarity comparison of private data through the Hamming distance of two PICRP. PICRP is also assumed difficult to forge by attackers, and protecting users privacy by not disclosing users private and biometric information [1]. PICRP is computed, from the private data, as a BioCode, with the BioHashing algorithm [4]. The BioHashing algorithm transforms a real vector T into a binary model called BioCode B, by multiplying T with a Gram Schmidt matrix randomly generated from a secret used as a seed. The result is then quantified.

PICRP still needs to be integrated into a secure protocol as it is vulnerable to replay attacks. [1] proposed to protect the user-service communication channel in order to protect the PICRP. In our study, we propose a more advanced authentication scheme. Any type of personal information can be added to the PICRP as long as it can be represented as a fixed-length real vector (e.g. browser history, free-text, mouse, ...). Soft-biometrics information (s.a. age, gender) could also be computed from existing modalities and integrated to the PICRP, in order to improve performances.

However, this study, described as preliminary, aimed at demonstrating the feasibility of PICRP and did not evaluated its performance. Personal information have been simply concatenated without any appropriate pre-processing. We show the advantages of the PICRP concept, and we propose a new PICRP based on Keystroke Dynamics, IP and GPS geo-location. We optimize the pre-processing and merging of the modality, then demonstrate the performances of the proposed PICRP, then present an example of its usage.

We evaluate the performances in terms of Equal Error Rate (EER) which is the error rate when the rate of accepted imposter (False Acceptance Rate) equals the rate of rejected legitimate users (False Rejection Rate). To evaluate the performance of the proposed PICRP, the BioHashing secret is assumed to be known by the attacker (worst case). All PICRP were thus be computed using the same BioHashing secret: `0x1534FA2C4D37`. In this study, only one template is used as reference. We latter propose a PICRP usage in which the localkey (authenticating the user browser) is used to compute the BioHashing secret, instead of being part of the BioCode. In the next section, we present the PICRP pipeline we used in this study.

2 PICRP pipeline

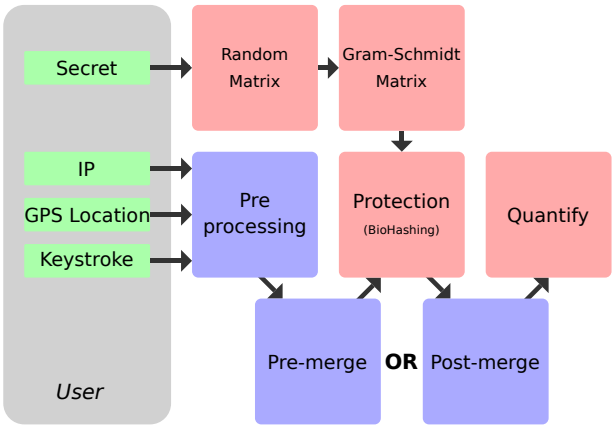


Figure 1: PICRP computation.

Computation of PICRP is shown by Figure 1. In red, the BioHashing steps that are presented below. In green the data used to compute the PICRP, and in blue, the steps added to the BioHashing. Data and additional steps are presented in the following sections.

Biohashing was used in [1] to protect personal information, it is a well-known algorithm in biometrics. It enables a biometric data T transformation by generating a binary model B called BioCode, having a size inferior or equal to the original size. This transformation is non-reversible and allows to keep input data similarity. This algorithm originally has been

proposed for face and fingerprints by Teoh *et al.* in [4]. Biohashing algorithm can be used on every biometric modality, or personal information, that can be represented by a fixed-size real vector. This transformation requires a secret linked to the user.

First, a Gram Schmidt matrix is generated from a random matrix, using the secret as a seed. T is then multiplied with the Gram Schmidt matrix and the result X is then quantified using the following formula: $B_i = (sign(X_i) + 1)/2$. BioCode comparisons is realized by the computation of their Hamming distance. The quantization process guarantees the data non-reversibility (even if $len(T) = len(B)$), as each input coordinate T is a real value, when the BioCode B is binary.

3 Datasets

3.1 Keystroke Dynamics Datasets

Fixed-text Keystroke Dynamics datasets used in this study are described in Table 1. As described in [5, 6], these datasets have been cleaned and only the first 45 entries of each user are kept. Metrics given in this paper are computed as the average value of the metric across the 4 datasets.

Name	Text	# of users (45)	Source
GREYC K	greyc laboratory	104	[7]
GREYC W1	laboratoire greyc	62	[8]
GREYC W2	sésame	46	[8]
CMU	.tie5Roanl	51	[9]

Table 1: Description of used Keystroke Dynamics datasets.

3.2 Location datasets

Location datasets are generated from the DB-IP dataset [10] where each entries describe an IP network and a GPS location. Only IPv4 entries are considered. Each user are associated to an origin place, randomly chosen among the DB-IP entries, each entry having a probability to be chosen given by the number of IP addresses the network enables. Each user entry is then generated by randomly choosing another entry from DB-IP which distance with the origin place is below an arbitrary value we name *user mobility*. The generated datasets have 100 users with 45 entries each. IP addresses are randomly chosen among the one belonging to the IP network. Two datasets are generated, *IP addresses generated from place* where each entry IP addresses are generated from its network, and *IP addresses from network*, where the IP address is generated from the origin place network.

GPS coordinates are converted in XYZ location, a coordinate in the Euclidean space represented by 3 reals (x, y, z). XYZ location enables non-biased distances, as longitudes +180 and -180 are the same longitude, and as differences in longitudes does not represents the same distances in function of the latitude. In *XYZ location, generated from places*, XYZ location are computed from the entry GPS coordinates. In *XYZ location, generated from positions*, XYZ locations are randomly picked in all possible coordinates at a distance from the origin place inferior to the user mobility.

3.3 GPS formula

This section presents formula applied to GPS/XYZ locations used in the scope of this study.

3.3.1 XYZ locations

XYZ locations (x,y,z) are computed from latitude (lat) and longitude (long) GPS coordinates with the following algorithm:

gpsToXYZ([lat, long]) : [x,y,z]
*lat *= π / 180, long *= π / 180;*
*y = 0.5 + sin(lat) * 0.5, r = cos(lat) * 0.5;*
*x = 0.5 + sin(long) * r, z = 0.5 + cos(long) * r;*

3.3.2 Distances

Distances between two places are computed as an angle a using the cosinus law. The distance between two XYZ locations A, B is computed as follow:

angle(A,B): a
$cos^{-1}(1 - 2 * (\Sigma_{i \in \{x,y,z\}}(A[i] - B[i])^2))$

Distances in meters m are converted into angle a distance with the following formula (assuming the circumference of the earth c to be 40,075,000 meters):

$a = m/c * 2 * \pi$

3.3.3 Random locations

Random locations are generated from an origin location O and a user mobility r, i.e. the distance between the random locations and O is at most r.

Random locations are generated by randomly picking a polar coordinate $[a,\ d]$ in a circle of radius r using *pickInCircle()*. The polar coordinate are then converted to a GPS location $GPS = [long, lat]$ using *pointInCircleToGPS()*. Random locations are generated, first assuming the origin location to be the North Pole (lat. 90, long. 0), then by rotating the space in order to move the North Pole to the origin location using *moveNorthPoleToOrigin()*.

pickInCircle(r): [a,d]
$a = rand() * 2 * \pi, \quad d = r * \sqrt{rand()};$

pointInCircleToGPS([a,d]): [long,lat]
$lat = 90 - 180 * d/\pi, \quad long = -180 * a/\pi + 180;$

moveNorthPoleToOrigin(GPS, O = [olong, olat]): GPS
*GPS = latRotation(GPS, (olat-90) / 180 * π);*
*GPS = longRotation(GPS, olong / 180 * π);*

latRotation(GPS, dx): GPS
$[x, y, z]$ *= gpsToXYZ(GPS);*
a = angle([0.5,y,z], [0.5, 0.5, 1]) + dx;
r = dist3D([x,y,z], [x,0.5,0.5]);
*y = 0.5 + r * sin(a), z = 0.5 + r * cos(a);*

longRotation(lat, long], dy): GPS
*long += dy / π * 180;*
long = (long + 360) % 360 - 360;

dist3D(A,B): d
$d = \sqrt{\Sigma_{i \in \{x,y,z\}} (A[i] - B[i])^2};$

4 Pre-processing

We intend to protect collected data with a biometric template protection scheme called BioHashing. Collected information thus have to be pre-processed in order to be represented as a vector of real values:

- *Fixed-text Keystroke Dynamics* data are pre-processed in Section 4.1. In this section, we show that the values distributions in the real vector influence BioHashing performances.
- *Locations* (GPS and IPv4 addresses) are pre-processed in Section 4.2. In this section, we reduce collisions in the final BioCode by extending small vectors of reals.

4.1 Fixed-text Keystroke Dynamics

Keystroke dynamics can be trivially represented as a concatenation of dwell (d_0/d_5) and flight times (d_3). However, such representation (raw) gives disappointing performances (EER=40%). As stated in [5], 6 duration times can be extracted from each digraph, with the last duration of a given digraph (d_5) also being the first duration (d_0) of the next digraph. However, as these duration times can be rewritten as additions of dwell and flight times, they are, by construction, not bringing any additional security or performance to the BioHashing algorithm. We thus present, in the following, several pre-processing techniques to the raw representation of Keystroke Dynamics that improve performances.

4.1.1 Standardization

A common practice in Data Sciences is to normalize variables, i.e. to center and reduce them. Assuming X_i the variable associated to the i^{th} real of the vector, with M_i and S_i its mean and standard deviation, those processes are described by the following formulas:

- center: $X'_i = X_i - M_i$;
- reduce: $X'_i = X_i / S_i$;
- standardize: $X'_i = (X_i - M_i)/S_i$;

However, in this study, we used the median value instead of the mean one. Indeed, the median is more resilient to aberrant values (e.g. hesitation times), and ensures equal numbers of positives and negatives

values after centering.

Results: We found that Standardization significantly improve the EER value (28.8%) compared to the raw (40%), reduced (38%), and centered (34.5%) cases.

4.1.2 Uniform distribution modelling

Another practice is to change the variable distribution. As the previous section shown that centered variables seem to significantly improve the EER value, we choose a target distribution that is centered. We also seek to draw closer extrema values, and to distance closed values. For these reasons, we choose the target distribution to be, in this study, a uniform distribution with support $[-1; 1]$. Change the distribution of a variable can be easily performed with the following formula: $X'_i = cdf'^{-1}_i(cdf_i(X_i))$, with $cdf_i(X_i)$ the Cumulative Density Function describing the distribution of the variable X_i, and $cdf'_i(X'i)$, the target distribution. However, while the target distribution is known ($cdf'^{-1}_i(x) = 2x-1$), the variable distribution $cdf_i(X_i)$ has to be estimated.

A naive estimation of X_i distribution is given by $cdf_i(x) = pos(x, A_i)/len(A_i)$, with x a value of X_i, A_i a sorted array of all known values of X_i, $len(A_i)$ its length, and $pos(x, A_i)$ the position of x in A_i. A more practical estimation of $cdf_i(X_i)$ is to compute the parameters of the law X_i is assumed to follow. In this study, we used the same laws (gumbel, normal, logistic, and laplace) and fitness functions (raw, R_mle, R_mge, and R_qme) used in [5]. All dwell times were assumed to follow the same law, but with different parameters, as well for the flight times. Dwell and flight times could however follow different laws. Configurations are labeled as follow: *fitness function.dwell law.flight law.*

Results: Over the 64 tested configurations, the optimal EER value (24.2%) was found with R_mle.normal.gumbel (fitting). The best raw estimation configuration, raw.gumbel.gumbel (estim), was found slightly better (24.8%) than the naive estimation (25.4%).

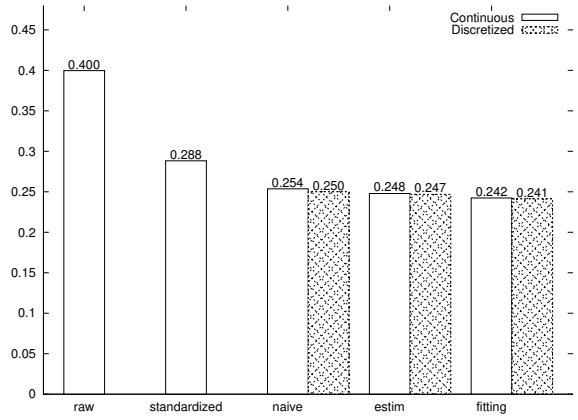


Figure 2: Keystroke Dynamics pre-processing performances (EER)

4.1.3 Discretization

Previous studies have shown the influence of discretization on performance of Keystroke Dynamics Systems based on an Hocquet distance (up to $\simeq -0.5$ points) [6], and on SVM (up to -1.04 points) [11]. To the knowledge of the authors, none has yet study the impact of Keystroke Dynamics discretization on BioHashing-based KDS. Keystroke Dynamics data were discretized and uniformized into identical probabilities values using the following formula: $X'_i = cdf'^{-1}_i(disc_n(cdf_i(X_i)))$, with $disc_n(x) = \lfloor x * n \rfloor$ / (N-1), and n the desired number of discrete values. $\lfloor n \rfloor$ is assumed equal to $n - 1$.

Results: Uniform Keystroke Dynamics data have been discretized using 999 different values of $n \in [\![2, 1000]\!]$. The EER value is computed as the lowest EER obtained from the 999 discretization configurations. As shown in Figure 2, discretization produces negligible EER gains (-0.1 to -0.4 points).

4.1.4 Discussion

As shown in Figure 2, uniformization of Keystroke Dynamics greatly improves EER (24.2% to 25.4%) compared to raw (40%) and normalized (28.8%) values. Discretization however produces negligible EER gains (-0.1 to -0.4 points). In addition to being less efficient (+1.2 points), naive estimation of $cdf_i(X_i)$ requires headcounts of each possible values for each variable. As the BioCode is computed on the client side, this means a large sending and storing large amount of data. Assuming a fixed-text of 16 characters with 1000 possible values for each variable, this represents an increase of at least 248ko in the webpage that can be troublesome for small Internet connections. In the contrary, non-naive estimations of $cdf_i(X_i)$ only requires mean/median and standard deviations that represent less than 0.5ko, assuming a fixed-text of 16 characters. While raw estimation of $cdf_i(X_i)$ is less efficient than fitting estimation (+0.6 points), it may be more practical as mean/standard deviations can easily be computed and updated, storing, for each variable, only the number, sum, and squared sum of its known values.

4.1.5 Limits

In the previous sections, we assumed that all users are asked to type the same fixed text in order to authenticate themselves. However, in real life, they would be more likely to be asked to type an identifier, s.a. their login or e-mail address, which is a personal fixed-text known by others. However, as the pre-processing parameters depends on the content being typed, this would require the service to make hundred of users type each possible/used fixed-text in order to compute them.

Pre-processing parameters can be estimated by assuming that same parameters apply to all dwell (or flight) times, enabling to compute them from known dwell (or flight) times. Figure 3 shows that this assumption induces a significant loss of EER value (+1 to +2.5 points). The best estimation and fitting configurations under this assumption were found to be raw.normal.laplace and R_mle.normal.laplace. It is worth noticing that the naive estimation of $cdf_i(X_i)$ outperforms estim and fitting under this assumption. Pre-processing parameters might be estimated with better accuracy using additional knowledge on Keystroke Dynamics. E.g. computing parameters in function of the typed character, digraph, or tri-graph either by knowing their statistics, their position on the keyboard, or/and their frequency in the user language.

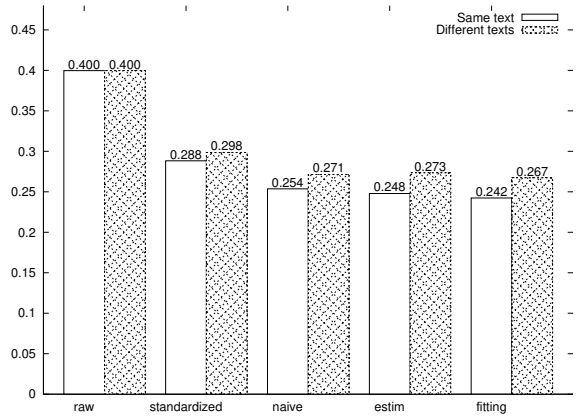


Figure 3: Keystroke Dynamics pre-processing performances (EER)

4.2 Location

IP address are represented as a set of bits some of them being more significant than others, while XYZ locations are represented as a set of 3 real values. Bitfields B on length n can easily be converted into reals r (and vice versa) thanks to the following formula: $r = \Sigma_{i=0}^{n-1} B_i * 2^{-i-1}$.

Using the real representation, IP address and XYZ locations are represented by 1 and 3 real values, thus producing BioCode of 1 and 3 bits. 4 bits, i.e. 16 possible BioCodes, is obviously not enough for both performances and security reasons. We thus present, in the following, several pre-processing techniques extending small vectors of reals in order to improve BioHashing performances. Reals will be assumed to equate to a bitfield of 32 bits, and will

be transformed as vectors of 32 real values R, while ensuring that the most significant bits have the most weight.

4.2.1 LogDist

LogDist does not ensure the bits significance. It associates each real value R_i to a bit B_i that determines its sign. In order to ensure that, e.g. bitfields 0111 (=0.4375) and 1000 (=0.5) have similar representations, the amplitude is computed from the following bits (viewed as a real). The amplitude is computed so that, the more such real is close to 0 or 1, the more the amplitude is close to 0, and the more it is close to 0.5, the more the amplitude is close to 1: $R_i = (-1)^{B_i} * (3^{B_{i+1}} - 1 + (-1)^{B_{i+1}} \Sigma_{j=1}^{n-i-2} B_{j+i} * 2^{-j})$.

4.2.2 PrefixDist

PrefixDist associates to each real R_i a bit B_i whose sign is determined by B_i as well as the previous computed real: $R_i = (-1)^{B_i} * R_{i-1}$. In this way, two bitfields sharing their n most significant bits produce vectors sharing at least n real values.

4.2.3 PrefixHash

PrefixHash is a variant of PrefixDist. It associates each real R_i to an hash computed from all bits $B_{\{j \leq i\}}$: $R_i = H_i * 2^{-31} - 1$, with H_i a 32-bit hash. The hash is computed as $H_i = \mathcal{H}(H_{i-1}, b_i)$, with $\mathcal{H}$ the hashing function. As the hashing function is not used for its security properties, but to diversify the output, Java `hashCode()` algorithm is used in this study: $\mathcal{H}(IV, c) = IV << 5 - IV + c$. The first IV (i.e. H_{-1}) is computed from the BioHashing secret.

4.2.4 PartitionDist

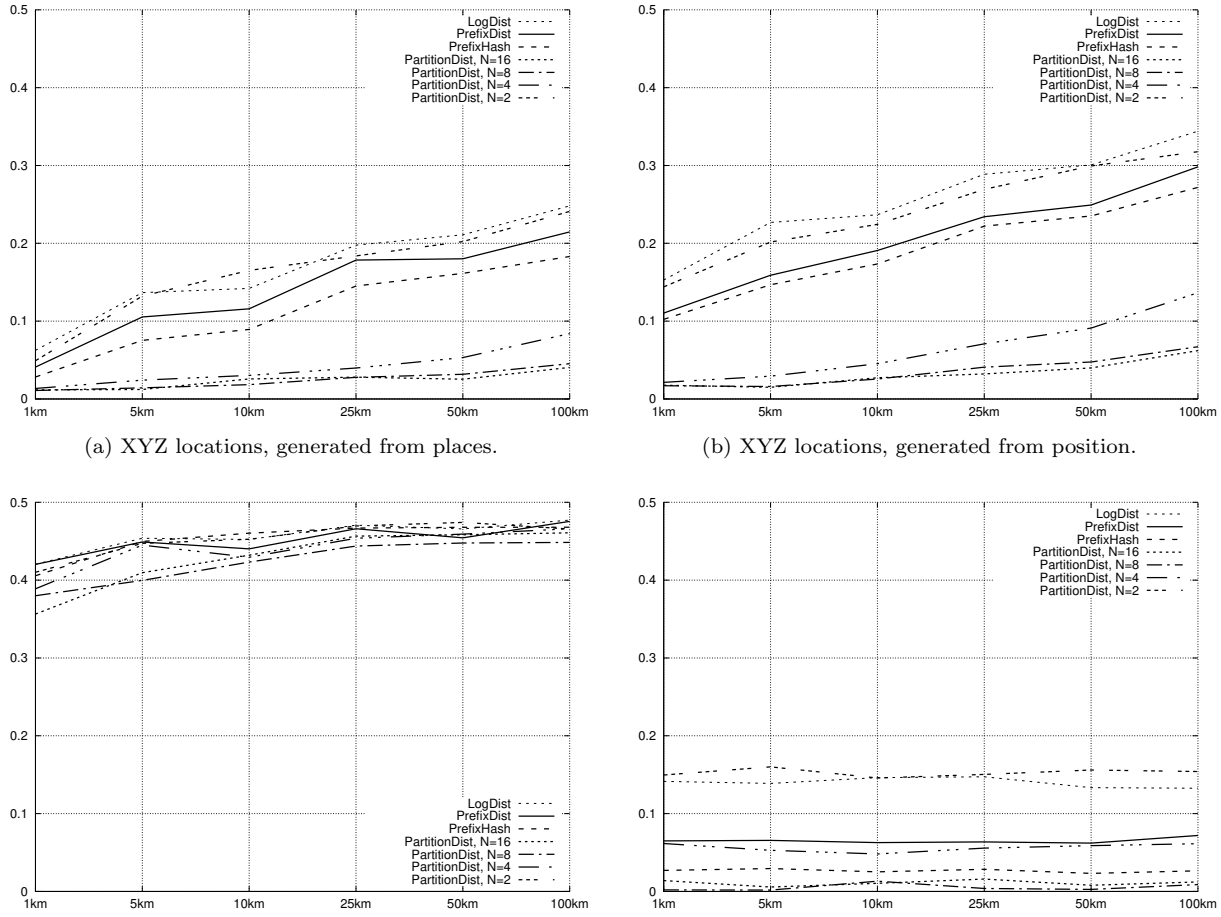
PartitionDist generates 32 reals R_i by combining n bits from 32 bits B_i while preserving their significance: $R_i = (-1)^{f_i(B,0)} * \Sigma_{j=1}^{n-1} f_i(B,j) * 2^{-j}$. $f_i(B,j)$ equals to B_r with r chosen between $j * 32/n$ and $(j+1) * 32/n - 1$. All combinations of the $n' = \lceil log(32)/log(32/n) \rceil$ first bits are listed. 32 of them are randomly selected, each determining the $f_i(B,j)$ bits for $j < n'$. The other $f_i(B,j)$ bits are randomly selected. Random engines are initialized from the BioHashing secret.

4.2.5 Results

Figure 4 shows the performance of the previously described location pre-processing methods. In all 4 tested datasets, PartitionDist, n=16, and PartitionDist, n=8 outperform other pre-processing methods. IP addresses generated from places provides an unacceptable EER value ($\geq 40\%$ for a user mobility $\geq$ 5 km) whereas IP addresses generated from networks provides a great EER value ($0.17\% < EER < 1.6\%$). PartitionDist, n=8 is the best setting for IP addresses pre-processing methods. In real-life, users do not use all networks from a place, like in IP addresses generated from places, but might still use several networks. This suggests the need of several templates, e.g. one template per network the user connects to. Further studies should be conducted with real-life data. XYZ locations provide great EER values both generated from places ($1.08\% < EER < 4.53\%$) and from positions ($1.49\% < EER < 6.71\%$). PartitionDist, n=16 is the best of XYZ locations pre-processing methods. Contrary to IP addresses, XYZ locations does not need several templates.

5 Merging of pre-processed data

A naive way to merge personal data is to compute a BioCode for each and concatenating them. However, by doing so, some BioCode might be too small to be protected against brute force attacks. For example, an IP address has less than 2^{32} possibilities and its resulting BioCode cannot have more than 32 bits, i.e. 2^{32} possibilities. We present in this section new merging methods, applied on the 3 previously pre-processed modalities:



(a) XYZ locations, generated from places.

(b) XYZ locations, generated from position.

(c) IP addresses, generated from places.

(d) IP addresses, generated from networks.

Figure 4: Location pre-processing performances (EER) in function on users mobility.

- XYZ locations, generated from places: PartitionDist, n=16;
- IP addresses, generated from networks: PartitionDist, n=8;
- Keystroke: R_mle.normal.gumbel.

In both merging methods, each modality vector $\mathcal{M}_i$ is associated to, and multiplied by, a positive weight W_i. In the first method, vectors are weighted and concatenated before applying the BioHashing algorithm. In the second method (post), a BioCode is computed on each modality. Before the BioHashing quantification step, a new vector V is computed as a weighted mean of the 3 non-quantification BioCode $\mathcal{M}_{\{0,1,2\}}$ with the following formula: $V_i = 1/(\Sigma_{j=0}^{2} W_j) \Sigma_{j=0}^{2} W_j \mathcal{M}_j[i\%len(\mathcal{M}_j)]$. V length is computed as the length of the longer BioCode. BioHashing quantification step is then applied on the resulting vector V.

	Pre	EER (%)	Post	EER (%)
min(0)	pre.len.1.98	0.069	post.raw.14.85	0.077
min(K)	pre.len.19.80	0.102	post.raw.15.84	0.203
min(X.IP)	pre.raw.10.6	20.00	post.raw.3.36	21.78
min(K,X.IP)	pre.raw.13.25	20.39	post.raw.3.38	21.92
min(IP.K,X.K,X.IP)	pre.len.47.25	21.42	post.raw.34.26	24.14

Table 2: Best merging configuration performances.

Merging methods are evaluated through 7 scenarios, labelled as the concatenation of modalities known/stolen by the attacker among the Keystroke (K), XYZ location (X), and IP addresses (IP). The scenario in which the attacker has no knowledge is 0. All possible combinations of weights have been tested. Weights were chosen from 1% to 98% per step of 1 point so that their sum is 100%, then multiplied by the number of modalities (here 3). As modalities vectors have different lengths and amplitudes, weight were multiplied to 3 modifiers: raw (no modification), len (correct the influence of modalities length), alen (correct the influence of both modalities length and mean amplitude). Modifier are computed as follows:

- raw: $RAW_i = 1$;
- len: $L_i = 1/(len(\mathcal{M}) * len(\mathcal{M}_i)) * \Sigma_{j=0}^{len(\mathcal{M})-1} len(\mathcal{M}_j)$.
- alen: $LA_i = L_i/ampl(\mathcal{M}_i)$, with $ampl(\mathcal{M}_i) = 1/len(\mathcal{M}_i) * \Sigma_{j=0}^{len(\mathcal{M}_i)-1} |\mathcal{M}_i(j)|$.

Merging configuration are labeled as: {pre|post}.{raw|len|alen}.W_0.W_1.

Results: Table 2 shows the best pre and post merging methods configurations that minimize the EER value in the scenario indicated in the first column, if several scenarios are indicated, the configuration minimizes the maximal EER value of each scenario. Figures 5 and 6 show the performances of each configuration under each scenario. As shown in Figures 5 and 6, configurations that minimize EER under (0) scenario produce really great EER ($< 1\%$), however such scenarios poorly perform (EER $> 40\%$) if IP addresses and XYZ location are known by attackers. Other configurations ensure an EER value

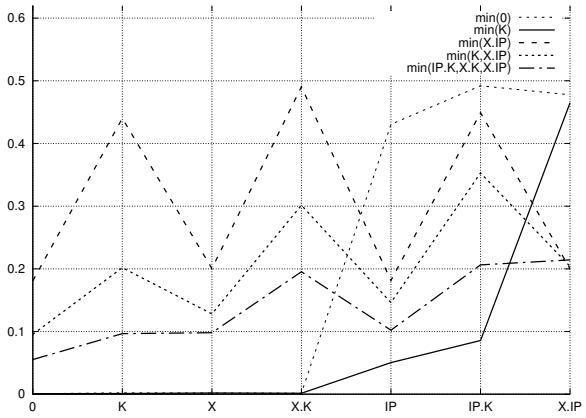


Figure 5: Best pre merging configuration performances under 7 scenarios.

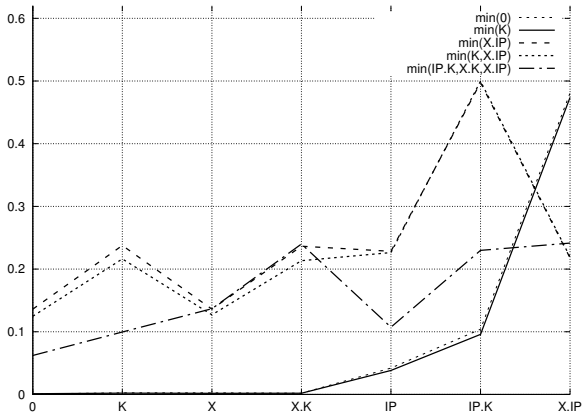


Figure 6: Best post merging configuration performances under 7 scenarios.

$\lesssim 20\%$ under all scenarios at the cost of a worst EER under (0) scenario. A solution would be to use several configuration to benefit from the best EER under (0) scenario while minimizing EER under (X.IP) scenario. It is worth noting that if attackers has knowledge of the BioCode (and the BioHashing secret), and some modalities, attackers could use such knowledge to invert the BioCode, mainly for pre merging configurations. A possible countermeasure would to to reduce the BioCode size.

6 PICRP usage case

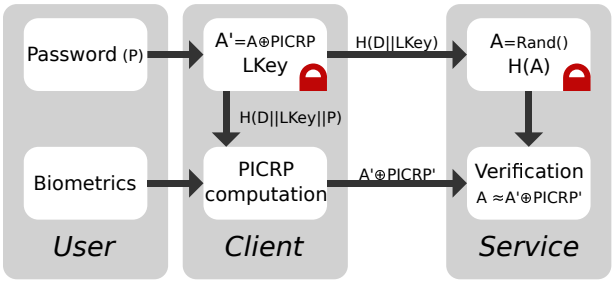


Figure 7: Proposed PICRP Authentication Scheme.

In this section, we propose a PICRP authentication scheme illustrated by Figure 7. We assume that the client and the service communicate using a secure channel, s.a. TLS, with authentication of the service, e.g. using TLS certificates. To enrol, a localkey (LKey) and a random binary vector of length equals to the PICRP (A) is generated by the client. The hash (H, e.g. using SHA2/SHA3) of the service domain name (D), LKey, and the user password (P) is used to compute the PICRP with the user biometrics. A' is computed from the PICRP and A as $A' = PICRP \oplus A$. A' and LKey are stored in the client, encrypted (e.g. with AES256) by the password. A is transmitted to the service, and is encrypted, with its hash (H(A)) using the hash of the Domain and LKey.

To authenticate, the client retrieves LKey and A' with the user password, thus enabling the client to compute PICRP' and thus $A \oplus PICRP'$. By sending H(D||LKey) and $A \oplus PICRP'$ to the server, the server is able to retrieve A, verify its integrity, and compare it to $A' \oplus PICRP'$. If the Hamming distance between A and $A' \oplus PICRP'$ is below a given threshold the user is authenticated.

In this system, an attacker getting into the client cannot gain knowledge as the only stored information are encrypted using a password, without integrity checks. If the attacker knows the password, he/she would only be able to know LKey and A' that are useless without the knowledge of the Biometrics, or A. The system is obviously vulnerable if the attacker gets into the client system while the user discloses his/her biometrics on the client. In the same way, attacker cannot gain knowledge by getting into the service as information are encrypted using a long random key. If the attacker gets into the service as the user authenticate, he would be able to obtain A that transport no information as randomly generated. He would however be able to authenticate (that we can mitigate by adding a 0-Knowledge proof of LKey to the authentication process).

Thus, to retrieve the biometric data, the attacker has to get into the service, the client, and guess the user password, or to collect directly the biometric data on the client during its usage by the user. As LKey is client-dependent, if the user has many devices, he/she will either need to get a reference per device, or to compute a new A' for each devices from A and the device-dependant PICRP. The latter solution has the advantage to not disclose the devices used by the user to the server. As the password is only used to encrypt the data on the client-side, it can also be easily changed without impacting the existing references.

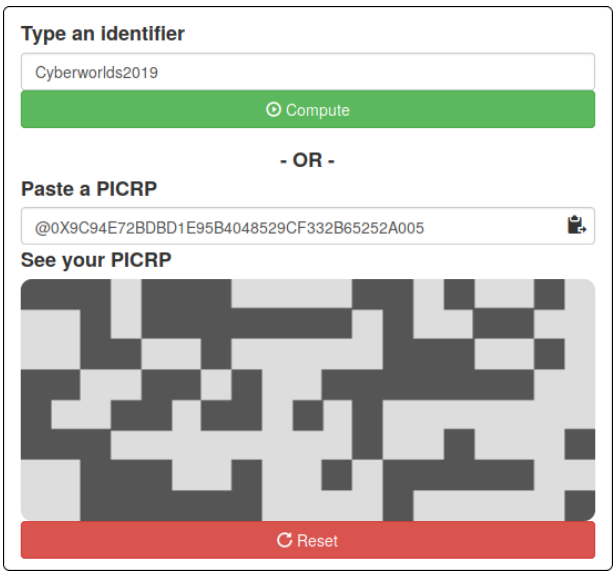
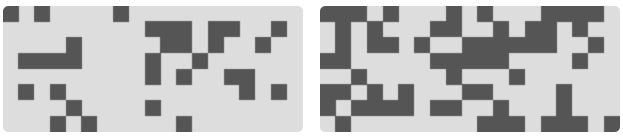


Figure 8: PICRP demonstration interface.



(a) Same user, same location. (b) Different users, different location.

Figure 9: Examples of differences (in black) between two PICRP (using the same secret) computed with pre.len.47.25 under scenario (0).

7 Discussion and Conclusion

In this paper, we extended the concept of PICRP [1] with the use of Keystroke Dynamics, IP, and GPS geo-location. The pre-processing of Keystroke Dynamics permits to significantly increase performances (EER from 40% before pre-processing to 24.2% after). geo-location has been found to produce great performances (EER $\lesssim 5\%$). Performances obtained after merging of theses modalities produce satisfactory performances (EER $< 1\%$). A PICPR authentication scheme has been introduced as a possible use case. Other usage could be found s.a. generation of keys from PICRP in order to sign, encrypt, or hash.

The interface of a PICRP demonstration is shown in Figure 8. Figure 9 shows differences in PICRP from the same user and from different users.

Future works could focus on improving pre-processings and merging methods. Other modalities could also be integrated to the PICRP, s.a. Keystroke Dynamics on Free-text, mouse, or even soft-biometrics computed from modalities. In this study, users geo-location have been synthetically generated. Further study should be conduct with real-life data. Only one template has been used as reference in this study, template-update techniques with user-dependant threshold could also be explored s.a. in [12].

References

[1] D. Migdal and C. Rosenberger, "Towards a Personal Identity Code Respecting Privacy," in *International Conference on Information Systems Security and Privacy (ICISSP)*, Madeira, Portugal, Jan. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01620972

[2] Y. Tian, Y. Li, X. Liu, R. H. Deng, and B. Sengupta, "Pribioauth: Privacy-preserving biometric-based remote user authentication," in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2018, pp. 1–8.

[3] P. Saini and A. K. Singh, "Biometric-based authentication in cloud computing," *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*, p. 147, 2018.

[4] A. Teoh, D. Ngo, and A. Goh, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pattern recognition*, vol. 40, 2004.

[5] D. Migdal and C. Rosenberger, "Statistical Modeling of Keystroke Dynamics Samples For the Generation of Synthetic Datasets," *Future Generation Computer Systems*, 2019.

[6] ——, "Keystroke Dynamics Anonymization System," in *SeCrypt*, Prague, Czech Republic, Jul. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02126985

[7] R. Giot, M. El-Abed, and C. Rosenberger, "Greyc keystroke: a benchmark for keystroke dynamics biometric systems," in *IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, 2009, pp. 1–6.

[8] R. Giot, M. E. Abed, and C. Rosenberger, "Web-based benchmark for keystroke dynamics biometric systems: a statistical analysis," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*. IEEE, 2012, pp. 11–15.

[9] K. S. Killourhy and R. A. Maxion, "Comparing anomaly detectors for keystroke dynamics," in *Proc. of the 39th Ann. Int. Conf. on Dependable Systems and Networks*, 2009, pp. 125–134.

[10] DB-IP, "Ip geolocation api and database," https://db-ip.com/.

[11] R. Giot, M. El-Abed, B. Hemery, and C. Rosenberger, "Unconstrained keystroke dynamics authentication with shared secret," *Computers & Security*, vol. 30, no. 6-7, pp. 427–445, Sep. 2011.

[12] A. Mhenni, E. Cherrier, C. Rosenberger, and N. Essoukri Ben Amara, "Analysis of Doddington Zoo Classification for User Dependent Template Update: Application to Keystroke Dynamics Recognition," *Future Generation Computer Systems*, Feb. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02050173