Empreinte Sociométrique

Cyril Dever
Edgewhere

November 14, 2016

Abstract

We describe an algorithm that takes contact data to represent the social footprint of a person in a secure and universal way. It allows recognizing an identity without sharing or exposing any personal information to any potential eavesdropper. It could evolve with time and may carry the whole contact history of a person, making it possible to verify if he or she matches with even very old data, all without needing any kind of disclosure between parties. Last update: this technique is pending patent[*]*.*

I. Introduction

Like a tyre or a shoe leaving a distinctive mark on the ground, or a finger on a glass, each one of us leaves a trace the more specific the richer our social interactions are. In particular, the history of our contact data is every day more distinct to someone else's. Even before we move from our parents house, we start leaving personal trails (a first cell phone, a pseudo we use for a game, ...) and, of course, our full civil status (names, date of birth, etc.).

Of course, we wouldn't want to share all these information to everyone. So, ensuring maximum security is obviously mandatory when it comes to manipulating personal data.

We herein describe a way to build such a safe footprint that we call *Empreinte Sociométrique*[1] and that is both totally secure, thanks to the use of strong pseudonymization techniques, and particularly effective, that is it ensures that, even though two different *Empreintes Sociométriques* don't look quite alike, they might be representing the same person, but no external stakeholder may ever know.

Embedded in a QR Code or used as is, it could become an assistant to any identification device or to further data manipulation, such as secure deduplication and anonymous enrichment, or even blind comparison of customer databases, all in full compliance with the latest regulations (GDPR, CCPA, ...).

[*]filed under registration number `FR1905778` at INPI on May 29, 2019

II. Formal Description

1. General Algorithm

An *Empreinte Sociométrique*[1] takes an input data characterized by its source and its type and operates several operations to form the final elements: its corpus and its signature concatenated into one long string.

Definition 1 (Source Data)**.** A source data ς is the actual contact data we want to print in the *Empreinte Sociométrique*, eg. `"Cyril"`.

It is defined in the *words* space: ω.

Definition 2 (Data Type)**.** We define the data type τ as a code defining which kind of source data we are dealing with, eg. `"firstname"`.

It is defined in a set of data types $\mathcal{T}$ [2].

Definition 3 (Input Data)**.** We define an input data d_i as a tuple of data type and source data:

$$d_i := [d_i^\tau, d_i^\varsigma] \text{ with } \begin{cases} d_i^\tau \in \mathcal{T}, \text{the data type} \\ d_i^\varsigma \in \omega, \text{the source data} \end{cases} \quad (1)$$

Definition 4 (Recombined Contact)**.** A recombined contact is a final contact data potentially made out of different input data.

For example, you can create a recombined `address4` by concatenating a `streetName` with a `streetNumber`.

Let $\nu : \omega \to \omega$ be a normalization function that takes an input data and returns its normalized counterpart.

[1]a literate translation being a *Sociometric Imprint*

[2]see Table 1 for available values in $\mathcal{T}$

And let $\rho : \omega^n \to \omega$ be a recombination function that takes several input data to build a missing recombined contact.

Finally, let $\mathfrak{h}()$ be a cryptographic hashing function[3], $\mathfrak{c}(msg, key)$ an encryption function and $\zeta()$ a compression algorithm.

Algorithm 1: Overall construction

Input: A vector $\mathbf{d} := \{d_1, d_2, \ldots, d_n\}$ of input data, a key K
Output: The *Empreinte Sociométrique* or an error

if $\boldsymbol{d} = \emptyset$ **then**
  throw *empty input data set*
initialize the set of normalized data $\mathcal{D} \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** n **by** 1 **do**
  if $d_i^\tau \notin \mathcal{T}$ **then**
    continue;
  normalize input data: $d_{Norm} \leftarrow \nu(d_i)$;
  if $d_{Norm} \neq \emptyset$ **then**
    $\mathcal{D} \leftarrow d_{Norm}$;
create the set of recombined contacts $\mathcal{R}$ from the normalized data: $\mathcal{R} \leftarrow \rho(\mathcal{D})$;
initialize the vector of ciphered contacts $\mathcal{C} \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $|\mathcal{R}|$ **by** 1 **do**
  $\mathcal{C} \leftarrow \mathfrak{h}(\mathcal{R}_i)$;
initialize the sets of categorized contacts: $\mathcal{V}$ the variants, and $\overline{\mathcal{V}}$ the invariants;
for $i \leftarrow 0$ **to** $|\mathcal{C}|$ **by** 1 **do**
  if $\mathcal{C}_i$ *is invariant* **then**
    $\overline{\mathcal{V}} \leftarrow \mathcal{C}_i$;
  else
    $\mathcal{V} \leftarrow \mathcal{C}_i$;
build the corpus c using $\mathcal{V}$ and $\overline{\mathcal{V}}$;
encrypt it and compress it to build the *Empreinte Sociométrique*:

$$ES \leftarrow \zeta \circ \mathfrak{c}(c, K);$$

return ES;

Algorithm 1 describes the general steps to take that leads from a set of input data to its *Empreinte Sociométrique*.

In a nutshell, the whole process could be summed up as follows:

- Data handling;
- Normalization of contact data;
- Recombination to maximize endpoints;
- Ciphering of each endpoint;
- Organization of the corpus;
- Encryption;
- Compression;
- Signature and final packaging.

To ensure maximum security, the whole operation should take place on the data owner side.

2. Step-by-Step Process

As seen in Algorithm 1, the very first step applied to an input data d_i is to check whether it is a valid or not.

At this stage, an input data would be deemed invalid for two reasons:

- The source data is empty: $d_i^\varsigma = \emptyset$;
- The data type doesn't exist: $d_i^\tau \notin \mathcal{T}$.

Each invalid input data is discarded while each valid one is then applied the normalization function $\nu()$.

2.1 Normalization

The goal of such a normalization is to homogenize source data to make sure the subsequent operations handle equivalent data.

Definition 5 (Equivalence)**.** We speak of data equivalence when we cannot distinguish them from one another after normalization:

$$\forall x, y \in \omega : x \equiv y$$

$$\iff \begin{cases} x^\tau = y^\tau \\ |\nu(x)| = |\nu(y)| \\ n \leftarrow |\nu(x)|, \forall i := [1..n] : \\ \quad \nu(x)[i] = \nu(y)[i] \end{cases} \tag{2}$$

In other words, two input data are said equivalent if they share the same data type and their normalized versions are identical.

[3] set as a system parameter

Normalization could be applied differently depending on the data type d^τ.

The following are the transformations that might be applied to an input data source d^ς:

- Replacement of punctation and special characters by spaces, eg.

$$\left.\begin{array}{r} \texttt{-} \\ \texttt{.} \\ \texttt{;} \\ \texttt{:} \\ \texttt{*} \\ \texttt{...} \end{array}\right\} \mapsto \texttt{\textbackslash s}$$

- Trim of extra empty spaces, eg.

$$\texttt{\textbackslash sMy\textbackslash s\textbackslash sdata} \mapsto \texttt{My\textbackslash sdata} = \texttt{My data}$$

- Removal of accents, eg. `é` $\mapsto$ `e`, `ñ` $\mapsto$ `n`;
- Capitalization, eg. `Cyril` $\mapsto$ `CYRIL`;
- Mapping with a dictionary, eg.

$$\left.\begin{array}{r} \texttt{AV} \\ \texttt{AVE} \\ \texttt{AVENUE} \\ \texttt{AVN} \\ \texttt{AVNU} \\ \texttt{...} \end{array}\right\} \mapsto \texttt{AV}$$

In some case, normalization could transform the source data to a single space which in turn would lead to d_i being discarded.

Example 1. Let

$$\begin{cases} d_1^\varsigma \leftarrow \texttt{1600, Pennsylvania Avenue NW.} \\ d_2^\varsigma \leftarrow \texttt{1600 PENNSYLVANIA AV NW} \end{cases}$$

be two source data of the same data type:

$$d_1^\tau = d_2^\tau = \texttt{address4},$$

we have:

$$\nu(d_1^\varsigma) = \nu(d_2^\varsigma) = d_2^\varsigma \Longrightarrow d_1 \equiv d_2$$

As d_1 and d_2 are equivalent, only one of them would be kept in the set of normalized data $\mathcal{D}$ to pass onto the recombination step.

2.2 Recombination

Lorem ipsum ...

2.3 Ciphering

Lorem ipsum ...

2.4 Corpus

Lorem ipsum ...

2.5 Signature

Lorem ipsum ...

2.6 Output

Lorem ipsum ...

Contents

References

[1] Cyril Dever. *Système de traitement d'une base de données personnelle par un opérateur extérieur*, patent pending `FR1905778`, 2019.

Appendix

Table 1: *Input data types*

Code	Description	*Examples*
`gender`	Title or gender	*M, Mr., 1, . . .*
`firstname`	First name or given name	*John*
`middle`	Middle initials or other names	*J., John*
`birthname`	Birth name	*Kennedy*
`lastname`	Last name or married name	*Kennedy*
`suffix`	Suffix	*Jr.*
`birthdate`	Date of birth	
`birthplace`	Place of birth	
`addresses`	List of postal address	(see Table 2)
`aliases`	List of aliases	(see Table 3)
`emails`	List of e-mail addresses	(see Table 4)
`ids`	List of official IDs	(see Table 5)
`mobiles`	List of mobile phones	(see Table 6)
`phones`	List of telephone numbers	(see Table 7)
`socials`	List of social media	(see Table 8)
`updated`	Unix timestamp of collect	*1544529071*

Table 2: *Address data type*

Field	Definition	*Possible values (or examples)*
`type`	Address type	`"birth"` ∨ `"home"` ∨ `"work"`
`address2`	Additional name	*c/o Mme Dupont*
`address3`	Additional address	*Apt. 123*
`address4`	Street number and name	*1600 Pennsylvania Ave NW*
`address5`	PO Box or locality	*BP 987*
`address6`	City and ZIP code	*Washington, DC 20500*
`address7`	International destination	*U.S.A.*
`city`	City	*Washington*
`country`	Country	*United States of America*
`fullAddress`	Full address	*1600 Pennsylvania Ave NW, Washington, DC 20500*
`streetName`	Street name	*Pennsylvania Ave NW*
`streetNumber`	Street number	*1600*
`zip`	ZIP code	*DC 20500*
`updated`	Time of collect	*1544529071*

Table 3: *Alias data type*

Field	Definition	*Possible values (or examples)*
`type`	Alias type	`"commonname"` ∨ `"identity"` ∨ `"np"` ∨ `"pn"` ∨ `"pseudo"` ∨ `"tnp"` ∨ `"tpn"`
`value`	Alias value	*John John*
`updated`	Time of collect	*1544529071*

Table 4: *E-mail data type*

Field	Definition	*Possible values (or examples)*
`type`	E-mail type	`"business"` ∨ `"personal"...`
`value`	E-mail address	*john@john.com*
`isHash`	If is already hashed	`true` ∨ `false`
`engine`	Hashing algorithm	`"blake2"` ∨ `"md5"...`
`updated`	Time of collect	*1544529071*

Table 5: *ID data type*

Field	Definition	*Possible values (or examples)*
`type`	ID type	`"id"` ∨ `"cb"` ∨ `"passport"` ∨ `"registration"` ∨ `"serial"` ∨ `"ss"` ∨ `"udid"`
`value`	ID value	*1234567890abcdef*
`isHash`	If is already hashed	`true` ∨ `false`
`engine`	Hashing algorithm	`"blake2"` ∨ `"md5"...`
`updated`	Time of collect	*1544529071*

Table 6: *Mobile phone data type*

Field	Definition	*Possible values (or examples)*
`type`	Mobile phone type	`"business"` ∨ `"personal"...`
`value`	Mobile phone number	*+33 (0) 623 456 789*
`isHash`	If is already hashed	`true` ∨ `false`
`engine`	Hashing algorithm	`"blake2"` ∨ `"md5"...`
`format`	Format	`"+dd (d) ddd ddd ddd"`
`updated`	Time of collect	*1544529071*

Table 7: *Telephone data type*

Field	Definition	*Possible values (or examples)*
`type`	Telephone type	`"business"` ∨ `"personal"...`
`value`	Phone number	*+33 (0) 123 456 789*
`isHash`	If is already hashed	`true` ∨ `false`
`engine`	Hashing algorithm	`"blake2"` ∨ `"md5"...`
`format`	Format	`"+dd (d) ddd ddd ddd"`
`updated`	Time of collect	*1544529071*

Table 8: *Social media data type*

Field	Definition	*Possible values (or examples)*
`type`	Alias type	`"facebook"` ∨ `"linkedin"` ∨ `"twitter"` ∨ `"youtube"...`
`value`	Alias value	*@john*
`updated`	Time of collect	*1544529071*