# 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 $\varsigma$ is the actual contact data we want to print in the *Empreinte Sociométrique*, eg. `"Cyril"`.

It is defined in the *words* space: $\omega$.

**Definition 2** (Data Type)**.** We define the data type $\tau$ 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** $d = \emptyset$ **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 $\overline{\mathcal{V}}$;
compress it and encrypt it partly to build the *Empreinte Sociométrique*:

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

**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;
- Compression;
- Encryption;
- 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 $d^\tau$.

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

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

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

- Trim and extraction of excess spaces, eg.

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

- Removal of accents, eg. $\texttt{é} \mapsto \texttt{e}$, $\texttt{ñ} \mapsto \texttt{n}$;
- Capitalization, eg. $\texttt{Cyril} \mapsto \texttt{CYRIL}$;
- Dictionary substitution eg.

$$\left.\begin{array}{r} \texttt{AVE} \\ \texttt{AVENUE} \\ \texttt{AVN} \\ \texttt{AVNU} \end{array}\right\} \mapsto \texttt{AV} \qquad \left.\begin{array}{r} \texttt{F} \\ \texttt{Madam} \\ \texttt{Ms} \\ \texttt{Mrs} \end{array}\right\} \mapsto \texttt{2} \quad \textit{etc.}$$

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

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

be two source data of the same data type:

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

then 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

Once the data will be irreversibly ciphered into the final *Empreinte Sociométrique*, we need to maximize the chance to get a match when using it, the goal of the recombination function $\rho$ is to provide as many representations as possible of the data.

Each data type may have dozens or even hundreds of possible combinations so we won't detail them here. Instead, we will use an example.

**Example 2.** Let the vector $\mathbf{d}$ be the input data:

$\{$(gender, Mr.), (firstname, John), (middle, F.), (lastname, Kennedy), (streetNumber, 1600), (streetName, Pennsylvania Avenue), (city, Washington), (zip, DC 20500)$\}$

Let $\mathcal{D} \leftarrow \nu(\mathbf{d})$ be a set of normalized data:

$\{$1, JOHN, F, KENNEDY, 1600, PENNSYLVANIA AV, WASHINGTON, DC 20500$\}$

Then $\mathcal{R} \leftarrow \rho(\mathcal{D})$ might yield the following additional values:

- 1 JOHN;
- JOHN F;
- 1 JOHN F;
- JOHN KENNEDY;
- 1 JOHN KENNEDY;
- JOHN F KENNEDY;
- 1 JOHN F KENNEDY;
- 1600 PENNSYLVANIA AV;
- WASHINGTON DC 20500;
- 1600 PENNSYLVANIA AV WASHINGTON DC 20500;
- JOHN KENNEDY 1600 PENNSYLVANIA AV WASHINGTON DC 20500;
- 1 JOHN KENNEDY 1600 PENNSYLVANIA AV WASHINGTON DC 20500;
- JOHN F KENNEDY 1600 PENNSYLVANIA AV WASHINGTON DC 20500;
- 1 JOHN F KENNEDY 1600 PENNSYLVANIA AV WASHINGTON DC 20500.

$\Longrightarrow \|\mathcal{R}\| = 22$

Once ciphered, these 22 recombined contacts along with their respective output codes $\Omega$ will form the basis of the corpus.

**Definition 6** (Output Code)**.** An output code $\Omega$ is a code defining the specific type of a recombined contact. It is either a special version of the input type code or the concatenation of its different parts.

A complete list of output types can be found in Table 9. Each code might be combined with one or several others in order to form a new output type.

### 2.3 Ciphering

As we want to take advantage of the very characteristics of cryptographic hashing, ie. making an input its unique yet irreversible fixed-length image, and as we want it to be both secure and widely spread to devices, the cipher $\mathfrak{h}()$ to apply to each item of $\mathcal{R}$ should be a well-known well-tested cryptographic hashing function:

$$\forall i \in [1..\,\|\mathcal{R}\|] : \mathcal{C}_i \leftarrow \mathfrak{h}(\mathcal{R}_i) \tag{3}$$

*Note.* Because of the subsequent steps of our process, the strength of $\mathfrak{h}()$ is actually not that important, so it could be an algorithm as trivial as `MD-5` or a more secure one such as `Blake-2` or `Keccak`[4].

The only thing mandatory is to set it once and for all to be sure any future utilization will use the same cipher.

Once the ciphering stage finished, we have all the necessary components to build the first part of the *Empreinte Sociométrique*: its corpus.

### 2.4 Corpus

To build the corpus, we first need to create the set of all imprints $\mathcal{E}$ from $\mathcal{C}$.

**Definition 7** (Imprint)**.** We define an imprint[5] $\varepsilon$ by the following tuple of information:

- Its general type code, eg. $\varepsilon^{\tau} =$ `ad` for an address type;
- The ciphered recombined normalized item $\varepsilon^{h}$, called its *hash*;
- A timestamp of collect (or insertion) in the system, eg. $\varepsilon^{t} =$ `1544529071`;
- Its specific short code $\varepsilon^{\bar{s}}$.

The complete list of type codes are given in Table 10.

It is easy to prove that each imprint is as safe as the functions used to build $\varepsilon^{h}$. And when we say "safe" it is by means of recovering the original data source (which doesn't exist anymore at this point).

[4]Short list available on Wikipedia: `https://bit.ly/3bDpGxd`

[5]Empreinte in French

**Definition 8** (Short Code)**.** A short code $\bar{s}$ is a special 4-characters hash defining the specific type of imprint.

Algorithm 2 describes the construction of a short code using the output field type $\Omega$. It uses a 62-characters basis $\mathcal{B}$ using the three following sequences in that order:

- The 10 figures: 1234567890;
- The 26 lower-case alphabet;
- The 26 upper-case alphabet.

Let $idx(char)$ be a function that returns the index number (starting at 1) of the passed character in $\mathcal{B}$.

**Algorithm 2:** Short code $\bar{s}$

**Input:** $\Omega$
**Output:** The corresponding short code

1 initialize the sum of $\Omega$ characters on that basis: $s \leftarrow 0$;
2 **for** $i \leftarrow 1$ **to** $\|\Omega\|$ **do**
3 $\quad s \leftarrow s + idx(\Omega_i)$;
4 make it an hexadecimal string representation: $s_{16} := (s)_{16}$;
5 hash $\Omega$ with `MD-5` hashing function: $h := \texttt{md5}(\Omega)$;
6 extract and concatenate the first two characters of $s_{16}$ and $h$:

$$\bar{s} \leftarrow s_{16}[0] +\!\!+\, s_{16}[1] +\!\!+\, h[0] +\!\!+\, h[1]$$

7 **return** $\bar{s}$;

For example, for $\Omega =$ `email`, we have: $\bar{s}(\Omega) :=$ `0c5a`

**Definition 9** (Variance)**.** A invariant contact data is a data that is not meant to change in a lifetime. For example, the date of birth is invariant. There is a limited set of variant contact data.

On the other hand, a variant data may change in time (for example, a postal address where one lives). There is an unlimited list of invariant contact data. An *Empreinte Sociométrique* may even carry multiple of the same types (for example, if one has moved a lot, he should have as many invariant addresses as places he is been living).

**Definition 10** (Corpus)**.** The corpus of an *Empreinte Sociométrique* is a special concatenation of all the imprints built from the input data using a secret token $T_k$. It is made of variant and invariant imprints.

Algorithm 3 describes its construction.

Let `BLOCK_SEP` and `SEC_SEP` be two special string separators [6], `PREFIX` a system parameter setting a cutting index, and $cut(str, at)$ the cutting string utility.

Finally, let `V_HASH` be the hashing algorithm used by the system.

### 2.5 Signature

Sometimes, you wouldn't want to take a chance to share the actual corpus, but rather an even safer version of it, which won't allow anyone (including yourself) to rebuild the imprints. That is where the signature comes in handy.

**Definition 11** (Registrar)**.** A registrar is a special invariant imprint. Built from their equivalent imprints, it includes the following items:

- Birth country (with code: `birthcountry`);
- Birth day (`birthday`) with default format `dd/MM/yyyy`;
- Birth place (`birthplace`);
- Given name (`givenname`);
- Full identity (`identity`) taking into account country of origin;
- Middle names (`middlenames`);
- Surname (`surname`).

**Definition 12** (Signature)**.** The signature $\Sigma$ uses the corpus of the *Empreinte Sociométrique* to build a unique yet irreversible identifier to freely share with no risk whatsoever to go back to the corpus, let alone the original data, from it.

Algorithm 4 describes how to build it from the corpus.

Let $xt(corpus)$ be a function that transforms existing imprints in the corpus to registrars, $hash()$ the `Blake2b` hashing function returning its hexadecimal string representation, and $pub(shortCode, shortenedHash)$ the

**Algorithm 3:** Corpus construction

**Input:** The set of imprints $\mathcal{E}$, a secret token $T_k$
**Output:** The corpus string

split the variant from the invariant imprints: $\mathcal{V}, \overline{\mathcal{V}} := \mathcal{E}$;
initialize the invariants string: $\overline{v} \leftarrow 0$;
**for** $i \leftarrow 1$ **to** $\left\|\overline{\mathcal{V}}\right\|$ **by** 1 **do**
    $\overline{v} \leftarrow \overline{v} + \overline{\mathcal{V}}_i$;
    **if** $i < \left\|\overline{\mathcal{V}}\right\|$ **then**
        $\overline{v} \leftarrow \overline{v} +\!\!+ \texttt{BLOCK\_SEP}$;
initialize the variants string: $v \leftarrow 0$;
**for** $i \leftarrow 1$ **to** $\|\mathcal{V}\|$ **by** 1 **do**
    $v \leftarrow v + \mathcal{V}_i$;
    **if** $i < \|\mathcal{V}\|$ **then**
        $v \leftarrow v +\!\!+ \texttt{BLOCK\_SEP}$;
concat the core corpus: $c \leftarrow \overline{v} +\!\!+ \texttt{SEC\_SEP} +\!\!+ v$;
compress it: $C \leftarrow \zeta(c)$;
separate prefix $p^-$ from suffix $s^+$ at `PREFIX`: $p^-, s^+ := cut(C, \texttt{PREFIX})$;
encrypt the prefix with the token: $\mathfrak{e} \leftarrow \mathfrak{c}(p^-, T_k)$;
get the length of the encrypted prefix: $l \leftarrow \|\mathfrak{e}\|$;
set the version short code: $v \leftarrow \overline{s}(\texttt{V\_HASH})$;
build the full corpus $\chi$:

$$\chi \leftarrow s^+ +\!\!+ \mathfrak{e} +\!\!+ \texttt{SEC\_SEP} +\!\!+ l +\!\!+ v \quad (4)$$

hash the corpus with `MD-5` to get a 32-characters pseudo-random head string: $H \leftarrow \texttt{md5}(\chi)$;
**return** $H +\!\!+ \chi$;

[6] In our current impmlementation, we use:

- The pipe character (`|`) as `BLOCK_SEP`;
- The percent character (`%`) as `SEC_SEP`.

function building a public imprint where the *shortenedHash* is the integer value of the imprint's *hash* $\varepsilon^h$ with its first character stripped off[7].

**Algorithm 4:** Signature construction

**Input:** A corpus $\chi$, the token $T_k$
**Output:** The signature

extract the set of registrars from the corpus (if any); $\mathcal{R} \leftarrow xt(\chi)$
get existing gender $g$ and/or social security number $ss$: $\mathcal{R} \leftarrow \{g, ss\}$;
build the prefix $P$ by concatenating the registrars, the gender and the social security *hashes*;

$$P := \mathbin{+\!\!+}_{i=1}^{\|\mathcal{R}\|} \varepsilon_i^h$$

set the hashed prefix: $H_P \leftarrow hash(P)$;
encrypt the hashed prefix with the token: $E_P \leftarrow \rfloor(H_P, T_k)$;
finalize the prefix: $P \leftarrow H_P[0] \mathbin{+\!\!+} E_P$;
initialize the empty string $s^\iota$;
**for** $i \leftarrow 1$ **to** $\|\mathcal{R}\|$ **by** 1 **do**
    build its public imprint $\varepsilon_i^\pi$:

$$\varepsilon_i^\pi \leftarrow pub(\varepsilon^{\bar{s}}, shortenHash(\mathcal{R}_i))$$

    $s^\iota \leftarrow s^\iota \mathbin{+\!\!+} \varepsilon_i^\pi$;
    **if** $i < norm\mathcal{R}$ **then**
        $s^\iota \leftarrow s^\iota \mathbin{+\!\!+}$ `SEC_SEP`;
keep the 8 first characters of the `MD-5` hash of $s^\iota$: $H_{s^\iota} \leftarrow \texttt{md5}(s^\iota)[1..8]$;
encrypt $s^\iota$ with the token: $E_{s^\iota} \leftarrow \mathfrak{c}(s^\iota, T_k)$;
finalize the signature $\Sigma$:

$$\Sigma \leftarrow P \mathbin{+\!\!+} \texttt{\%} \mathbin{+\!\!+} H_{s^\iota} \mathbin{+\!\!+} E_{s^\iota} \qquad (5)$$

**return** $\Sigma$;

The $pub()$ function is where the alteration happens and therefore where lies the irreversibility of the signature and its public imprints. But it can be proven that it is highly unlikely that this alteration, albeit coarse, might lead to false positive when matching signatures, all the more so if the imprints are numerous in the initial corpusÓ[8].

### 2.6 Output

As you might have noticed, there are two different output we can get from the *Empreinte Sociométrique* algorithms:

- The full corpus $\chi$ which allows for further processing (like computing its signature) or enhancement (by adding new data to it);
- The signature $\Sigma$ which is an altered version of the corpus for safe public sharing.

Indeed, each person should build its own *Empreinte Sociométrique* step-by-step with his own secret token. Then, he may rebuild specific ones with others he wished to share contact information. For this last situation, he should probably only share the signature. That way, each person has his personal database of contact data secured, and can interact with others safely, both parties being able to ensure the other's identity.

## References

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

[7] For example, for a hash value of `ab12`, you strip off the heading `a` to get `b12` and recover the latter as its integer value `2834`: $shortenHash(\texttt{ab12}) = \texttt{2834}$

[8] and you shouldn't be using too small corpuses when relying on signatures

# 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* |

**Table 9:** *List of base output codes*

| Code | Description |
|---|---|
| `adresse2` | Postal address line 2 |
| `adresse3` | Postal address line 3 |
| `adresse4` | Postal address line 4 |
| `adresse5` | Postal address line 5 |
| `adresse` | Full postal address |
| `civ` | Title |
| `cp` | Postal (or ZIP) code |
| `ddn` | Date of birth |
| `dpt` | Department, eg.`75` |
| `dv` | Department and city, eg. `75 PARIS` |
| `email` | E-mail address |
| `facebook` | Facebook ID |
| `instagram` | Instagram ID |
| `linkedin` | LinkedIn ID |
| `pinterest` | Pinterest ID |
| `snapchat` | Snapchat ID |
| `twitter` | Twitter ID or screen name |
| `youtube` | Youtube ID |
| `in` | Initials and last name |
| `ini` | Initials |
| `middle_ini` | Middle initial(s) |
| `mob` | Mobile phone number |
| `ni` | Last name and initials |
| `nom` | Last (or birth) name |
| `np` | Last and first names |
| `no` | Street number |
| `pays` | Country |
| `pn` | First and last names |
| `prenom` | First (or given) name |
| `middle` | Middle names |
| `raisonsociale` | Corporate name |
| `tel` | Telephone / landline number |
| `tin` | Title + Initials + Last name |
| `tn` | Title and last name |
| `tni` | Title + Last name + Initials |
| `tnp` | Title + Last name + First name |
| `tpn` | Title + First name + Last name |
| `ville` | City |
| `voie` | Street name |

**Table 10:** *List of imprint type codes*

| Type | Code |
|---|---|
| Address | `ad` |
| Alias | `al` |
| Birth date | `4d` |
| Birth name | `2n` |
| Birth place | `5p` |
| Country | `8c` |
| E-mail | `ml` |
| Facebook | `fb` |
| First name | `3n` |
| Gender | `1x` |
| Instagram | `ig` |
| LinkedIn | `li` |
| Marital name | `mn` |
| Middle names | `7m` |
| Mobile phone | `mb` |
| Pinterest | `pi` |
| Registration | `rg` |
| Snapchat | `sn` |
| Social security number | `6s` |
| Telephone number | `tl` |
| Twitter | `tw` |
| Youtube | `yt` |