Kristof Verslype* and Bart De Decker

The Data Archipelago - Reconciling Privacy and Analytics on Multi-Source PII

Abstract: Data analytics, especially in the era of *Big Data*, opens huge possibilities in governmental contexts. These contexts are unfortunately very fragmented and often involve the processing of personal identifiable information (PII). Since anonymisation techniques fail at truly anonymising large citizen records, the data remain in a legal sense PII to which, hence, the privacy legislation still applies. The data owner (controller) is therefore still responsible and liable for the data. This paper presents a flexible approach consistent with the European privacy legislation which 1) enables easy linking of records, 2) maximizes the control of the data-delivering government agencies, 3) minimizes the impact in case of a data breach, and 4) allows for controlled deanonymisation.

Keywords: Privacy PII analytics big data pseudonym anonymisation

1 Introduction

A university research team would like to have access to a set of non-aggregated, pseudonimised records with PII (Personal Identifiable Information) of all married citizens of the country born in 1990 or later with a yearly income higher than €50.000. More specifically, they want to analyse some medical, financial and demographic PII of these citizens. However, these records are owned and managed by different government agencies. The logical answer to deal with such, frequently occurring, requests is a central data warehouse in which each government agency uploads its PII. However, government agencies are reluctant to do so since it implies a loss of control over this PII for which they legally remain responsible. The central question in this article, therefore, is: *"How can such PII, stemming from different sources (government agencies), efficiently and as securely as possible be linked together, whereby the European privacy legislation is respected and the involved government agencies maintain maximal control over the PII for which they are responsible?"*

An important evolution in the current digital era is the increasing collection and processing of data, including PII. The potential knowledge that can be extracted out of these data is enormous, not only for enterprises, but also for government agencies. Analysing these data can, hence, play a crucial role in fulfilling and improving the tasks of the these agencies. Examples are improving personalised health care, fighting fraud and supporting policies. Also, analytics on such data is invaluable for researchers.

Two characteristics are dominant in governmental contexts. First, governments often work with PII. Secondly, they have a high degree of fragmentation: each government agency is responsible (data controller) for a specific set of PII attributes, potentially of a specific part of the population. For many potential analytics projects, PII stemming from different sources should be linked together, which poses serious legal challenges.

Processing of PII can indeed only be allowed by the responsible government agencies if it is in correspondence to the applicable laws, mainly but not always exclusively the privacy legislation. Several principles described in the European data protection directive 95/46/EC[1] [1] apply to PII: It is not allowed to process PII in ways incompatible with the objectives for which it was initially collected (*finality*). No more data than strictly necessary is collected and processed (*proportionality*). The citizen has the right to know how his PII is processed (*transparency*). Also, the data controller will be held accountable in case of a data breach due to negligence (*information security practices*). Unfortunately, the recent past provides us with several cases of severe government data breaches, such as the OPM hack [2].

Linking together all PII from different government agencies in one big traditional data warehouse will, hence, have two major disadvantages: 1) reduction of

***Corresponding Author: Kristof Verslype:** Smals Research, Fonsny Avenue 20, 1060 Brussels, Belgium, E-mail: kristof.verslype@smals.be

Bart De Decker: Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, E-mail: bart.dedecker@cs.kuleuven.be

1 At the moment of writing, the new European privacy enactment, which is expected to be more strict, has not yet been finalised.

the control by the government agencies over the PII for which they are legally responsible and which they do not want to be used against their own interests and 2) a data breach will have outright disastrous consequences for the privacy of the involved citizens and on the reputation of the government agencies.

The reluctance of government agencies to merge their data in one big data warehouse is, hence, a mere expression of their responsibility which should indeed be respected. From a security and legal point of view, the scenario of merging all data in one centralised data warehouse is considered as a *digital data dystopia* for both government agencies and citizens. Simultaneously, there is a genuine need to link together data in order to enable advanced data analytics within the framework of the privacy legislation.

This brings us to the challenge tackled in this article: enabling efficient analytics on PII stemming from different sources, while 1) the delivering parties maintain control over the data for which they are responsible, 2) the privacy of the citizen is protected, and 3) the impact in case of a data breach is minimized. Also, the legal principle of transparency to the citizen is respected and flexibility, for example regarding key issuance and deanonymisation, is provided such that the concept can be applied in a wide variety of contexts.

The text is structured as follows. Section 2 presents related work. Section 3 presents the approach at a high level. Section 4 explains the pseudonym conversion function, an essential building block for the central protocol discussed in detail in section 5. Section 6 discusses controlled deanonymisation, which is useful for e.g. fraud detection. Section 7 focusses on key distribution and section 8 adds some extensions. The concluding remarks and future work are found in section 9.

2 Related work

Data anonymisation techniques such as *k-anonymity* [3], *l-diversity* [4] and *t-closeness* [5] have been put forward as a way to reconcile analytics and privacy. It aims at irreversibly removing the link between the data and the individual. This is achieved 1) by removing or replacing direct identifiers such as social security numbers and 2) by removing, generalizing or adding noise to pseudo-identifiers. A pseudo-identifier is a combination of attributes that together generally apply to only one or a few citizens. A typical example is the triple *ZIP code, gender, year of birth* [6]. Once data are legally anonymised, they are no longer considered as PII, the privacy legislation no longer applies, and the damage caused in case of a data breach is greatly reduced. However, applying anonymisation techniques on high dimensional data[2] such as location tracking is not working [7]: although it degrades severely the data, the effect on the uniqueness of the data records is limited. With only a few data elements, a record can be linked back to a citizen. Therefore, the White House considers anonymisation techniques, although valuable in the past, as obsolete in the era of big data [8]. Moreover, it turns out to be very hard to assess whether data is sufficiently anonymised, which is illustrated by two well-known examples. 1) AOL published in 2006 20 million 'anonymised' search queries of 650.000 users. Soon, the identity of multiple users was revealed [9]. 2) In 2007, NetFlix published 'anonymised' movie ratings of 500.000 users. By linking these data with public data on IMDB, records could be deanonymised [10]. Given the lack of information regarding future knowledge of the adversary, both in terms of algorithms and data, a formal assessment to measure anonymity seems infeasible. Paul Ohms states: *"The robust anonymisation assumption is not fundamentally incorrect but deeply flawed"* [11]. Therefore, it is unsuited in our context.

Pseudonymisation, however, is still crucial in the presented approach. Several methods exist to convert an identifier into a pseudonym that can not be linked back to the identifier without knowledge of a secret. Cryptographic approaches are typically based on symmetric encryption, pseudo-random permutations, secure hash functions or message authentication codes. A non-cryptographic approach is the generation of conversion tables. The approaches are often ad hoc, lack flexibility and require trusted intermediaries or shared secrets. Anonymous credential schemes such as *Idemix* [12] and *uProve* [13] allow a citizen to prove, anonymously or under a pseudonym, personal properties (e.g. age) without disclosing any other information. With conditional deanonymisation, a party can reveal the identity of the citizen under certain conditions such as abuse. Anonymous credential schemes provide us with interesting ideas that can be valuable in contexts other than authentication, for instance in data analytics.

2 High dimensional data are data that require a large, often increasing, amount of values (dimensions) in order to be expressed. Examples are location tracking, purchasing behavior and connections on a social network.

Record linkage [14] refers to finding records in the same or different data sets that refer to the same persons. In case of a shared identifier in a standard format this is trivial and this is what is assumed in this text. More complex forms of data linkage deal with typographical errors, false information, and the absence of standards and shared identifiers.

Multi-party computation allows calculation of functions on secret data kept by multiple parties. Each party learns the output of the function but nothing else. An overview of secure multi-party computation for privacy-preserving data mining (analytics) has been written by Lindell and Pinkas [15]. For reasons of efficiency and flexibility, in particular on bigger data sets with a wide set of potential queries, this approach is inefficient and cumbersome. Indeed, for analytics on massive amounts of data, the current paradigm is that initial collection of the required data on highly performant infrastructures is necessary. This is assumed in this paper.

Private information retrieval [16] allows *Bob* to retrieve information d from *Alice*, who owns a data set D. *Alice* does not learn what element in D *Bob* retrieved, but is sure that *Bob* only retrieved a single element. This is related with, but different to, the challenge formulated in this text where multiple domains deliver encrypted PII of many citizens to a single project. The project can only decrypt all the PII of a specific citizen if each domain has delivered some PII of that citizen.

3 Overview

Before explaining the data archipelago, three central terms are defined. An *identifier* is an alphanumerical string (or, more generally, an object) that uniquely identifies a physical person. A typical example is the social security number (SSN). A *pseudonym* is an alphanumerical string (or, more generally, an object) that refers to a single physical person but does not allow to identify this person. A person can have multiple pseudonyms in different contexts. A pseudonym allows to link PII of the same person without knowing this person's identity. A *pseudo-identifier* is a combination of attributes that together generally apply to only one or a few citizens. It allows to single out a citizen. A typical example is the triple *ZIP code, gender, year of birth.*

The data archipelago, depicted in figure 1, has two types of participants: providers (left) and researchers (right). The main terms specific for the data archipelago are now defined. *Providers* own some PII of citizens and provide PII to data analytics projects. They are typically government agencies, which may own for instance fiscal, social, economical or medical PII. *Researchers* (want to) analyse PII which result from linking together PII maintained by multiple providers. For instance, fiscal, social and economical data can be linked together for fraud detection. *Islands* are environments within the data archipelago which contain PII. An island should be maximally isolated from other islands, which means that it should be hard to link PII residing on one island to PII of the same person kept on another island. Two types of islands are distinguished: domains and projects. A *domain* is an island managed and controlled by a single provider. The provider keeps the domain up-to-date with PII that is potentially made available to data analytics projects. A domain typically contains only a subset of all the PII kept by the provider. Domains are relatively permanent and do not need fast computation. A *project* is an island that allows for data analytics by a researcher on the PII stored by this project. Therefore, a project retrieves and links together the minimal required PII from several domains. Projects are temporary and are destroyed as soon as possible (or archived when legally required). Projects require a highly efficient infrastructure for complex analytics. Access by researchers to projects is restricted and logged in order to maximally protect the PII (This is outside the scope of this paper).

Isolation between islands is maximized 1) at the hardware level by applying (existing) techniques of containerization and 2) at the data level by minimizing linkabilities between PII records on different islands. These linkabilities occur when two or more islands share for the same citizen an *identifier*, a *pseudonym* or a *pseudo-identifier*. Identifier linkabilities can be prevented by scanning the data, sent by the providers to the data archipelago, on identifiers using regular expressions, and blocking records that contain an identifier. Clear agreements between organisations avoid that the same or related attributes are stored on multiple domains. This results in attribute partitions and is similar to normalized databases. Since no attributes are shared, pseudo-identifier linkabilities are impossible[3]. However, since projects link together information originating from several domains, attribute linkabilities will in all likelihood

3 The Belgian government uses *authentic sources* as the exclusive, accurate source of certain data. The philosophy of authentic sources fits in the proposed concept; each authentic source could be given its own domain.

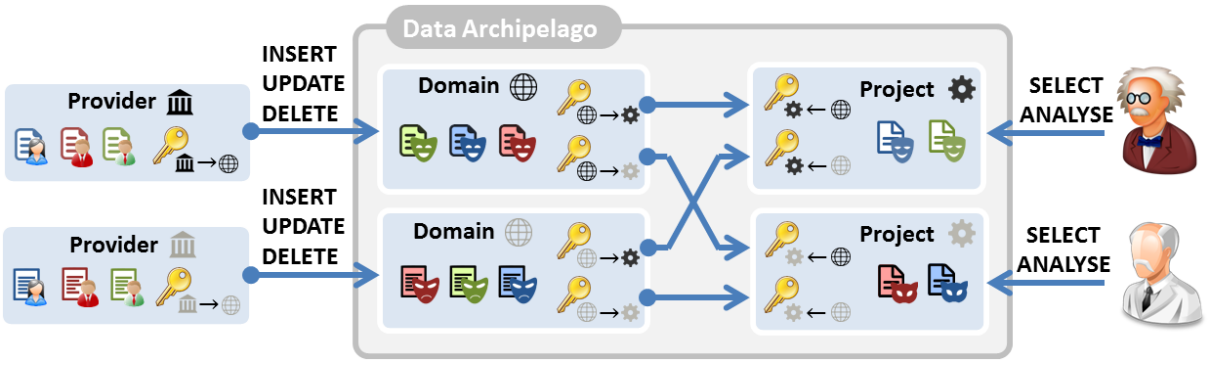

Fig. 1. Overview of the data archipelago for a country with three citizens.

arise 1) between a project and its involved domains and 2) between projects. Therefore, projects should – just as regular analytics projects – be well protected. Fortunately, projects only contain the minimum amount of data and have a limited lifespan. The more attributes about single citizens are stored in a domain, the higher the impact in case of a data breach. Therefore, in the presented solution, a domain with many sensitive attributes should be split in multiple, mutually unlinkable domains.

The remainder of this text aims at guaranteeing cryptographic unlinkability between pseudonyms of the same citizen on different islands, while linking together PII from multiple domains in a project requires the explicit cooperation of the involved providers. Simultaneously, a high degree of flexibility is offered.

4 Pseudonym conversion function

When sending data to a domain, the provider first converts every involved identifier into a pseudonym using a secret key. The PII in the domain is stored under this pseudonym. The PII of one citizen required for a project is spread over multiple domains. However, in each domain, the PII is known under different pseudonym, which is *unlinkable* to the citizen's identifier and to the citizen's pseudonyms on other islands without additional knowledge. Linking together PII based on these pseudonyms is, hence, not possible, unless the pseudonyms are converted with cryptographic keys.

When PII is sent by a domain to a project, the corresponding pseudonym is converted twice: first by the domain and then by the project. Both domain and project use for this a (different) secret key. Although the pseudonyms of the same citizen are unlinkable on the level of the domain, they become identical in the project after this double conversion. This property is called *local equality*. The required pseudonym conversion function has three other properties besides unlinkability and local equality and is discussed in this section.

4.1 Definition

Notation: k, id, n and d indicate keys, identifiers, pseudonyms and data respectively. Capitals with and without tilde indicate sets and sequences respectively. E.g. $\widetilde{S} = \{s_0, ..., s_{|S|-1}\}$, $S = \langle s_0, ..., s_{|S|-1}\rangle$. $Y \xleftarrow{rp} X$ denotes a random permutation and $y \xleftarrow{\$} X$ a random selection over a uniform distribution. For $C \leftarrow A \cap B$ and $C \leftarrow A \cup B$, no assumptions are made regarding the order of the elements in C. $A.append(a)$ adds an element at the tail of A.

The pseudonym conversion function is defined as $f : \mathcal{K}, \mathcal{N} \rightarrow \mathcal{N}$, with $\mathcal{K}$ the set of all possible keys and $\mathcal{N}$ the set of all possible pseudonyms. The set of identifiers is denoted by $\mathcal{I} \subset \mathcal{N}$. One key and one pseudonym (or identifier) are given as input, which results in a new pseudonym $\notin \mathcal{I}$. $\forall\ n \in \mathcal{N} : n \leftarrow f(\emptyset, n)$. We also define functions $\overline{f}$ and $\widehat{f}$ which take a sequence of keys $K = \langle k_0, ..., k_{m-1}\rangle$ respectively pseudonyms $N = \langle n_0, ..., n_{m-1}\rangle$ as input: $\overline{f} : \mathcal{K}^*, \mathcal{N} \rightarrow \mathcal{N} : K, n \mapsto f(k_{m-1}, (..., f(k_1, (f(k_0, n))...)$ and $\widehat{f} : \mathcal{K}, \mathcal{N}^* \rightarrow \mathcal{N}^* : k, N \mapsto \langle f(k, n_0), ..., f(k, n_{m-1})\rangle$. Furthermore, the pseudonym conversion function $f(.,.)$ needs to satisfy five properties:

- **Unlinkability.** Different pseudonyms originating from the same identifier or pseudonym cannot be linked to each other without knowledge of the corresponding keys. Formally:

$\forall\, l_0, l_1 \in \mathbb{N}_0,\, n, n_0, n_1 \xleftarrow{\$} \mathcal{N}, K^0 \xleftarrow{\$} \mathcal{K}^{l_0}, K^1 \xleftarrow{\$} \mathcal{K}^{l_1}$ s.t. $\overline{f}(K^0, n) \neq \overline{f}(K^1, n) : |P[1 \leftarrow A_U(\overline{f}(K^0, n), \overline{f}(K^1, n))] - P[1 \leftarrow A_U(n_0, n_1)]| < \varepsilon$

whereby the adversary A_U outputs 1 if and only if according to A_U the two pseudonyms given as input originate from the same pseudonym or identifier and ε negligible. This implies that for $\forall\, n \in \mathcal{N}$ and $\forall\, k, k_0, k_1 \in \mathcal{K}$ with $k_0 \neq k_1$: $f(k, n)$ cannot be linked to n (parent unlinkability) and $f(k_0, n)$ cannot be linked to $f(k_1, n)$ (sibling unlinkability).

- **Local equality.** Given one or more key sequences such that if the pseudonym conversion function is applied on them separately with the same identifier, the result is each time the same pseudonym. If the secret is known, a new such key sequence, which is independent of the existing ones, can be generated in polynomial time. Again, it maps the same identifier to the same target pseudonym. Formally:

  $\exists\, s \in \mathcal{S}$ s.t. $\forall\, m \in \mathbb{N}_0, \forall\, i \in \{0, ..., m-1\} : l_i \xleftarrow{\$} \mathbb{N}_0,$
  $K^i \xleftarrow{pol} B(l_m, s)$ s.t. $K^i \in \mathcal{K}^{l_i}$, $\overline{f}(K^0, .) = \overline{f}(K^i, .)$
  and for $i \neq 0$: $P[K^i \cap (K^0 \cup ... \cup K^{i-1}) = \emptyset] < \varepsilon$.

  with m the number of already known key sequences, l_i the length of key sequence K^i, $\mathcal{S}$ the set of all possible secrets and B an algorithm that outputs a result in polynomial time (indicated by *pol* above the arrow).
- **Collision resistance.** Applying the conversion function with the same key on two different identifiers or pseudonyms will output two different pseudonyms. Formally:

  $\forall\, l \in \mathbb{N}_0, K \xleftarrow{\$} \mathcal{K}^l : |P[(n_0, n_1) \leftarrow A_{CR}(K) :$
  $\overline{f}(K, n_0) = \overline{f}(K, n_1)]| < \varepsilon$, s.t. $n_0, n_1 \in \mathcal{N}$ and $n_0 \neq n_1$,

  whereby the adversary A_{CR} outputs two pseudonyms (or identifiers) that are converted with the given key sequence into the same pseudonym.
- **Determinism**. The conversion function generates for the same inputs always the same output.
- **Invertibility**. Given, the key and the output value of the conversion function, the initial input value can be found in polynomial time.

  $f^{-1} : \mathcal{K}, \mathcal{N} \rightarrow \mathcal{N}$ s.t. $\forall\, k \in \mathcal{K}, n \in \mathcal{N} :$
  $n = f^{-1}(k, f(k, n))$ and $\mathcal{O}(f(.,.)) \approx \mathcal{O}(f^{-1}(.,.))$.

4.2 Pseudonym conversion network

The pseudonym conversion function converts an identifier into several unlinkable pseudonyms on the level of the domains. However, within a project, local equality is required, while pseudonyms of the same citizen in different projects should still be unlinkable. All this

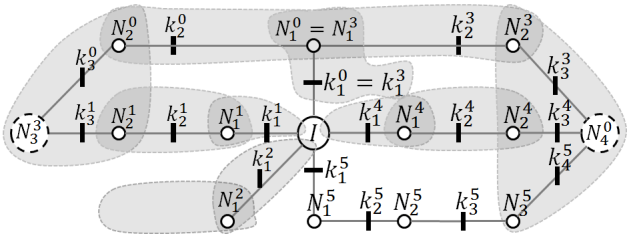

Fig. 2. Pseudonym conversion network with two join locations.

is represented in a pseudonym conversion network, of which an example is shown in figure 2.

We distinguish *locations* (circles in fig. 2) and *conversions* (bars in fig. 2). A *location* contains a sequence of identifiers or pseudonyms (maximum one per citizen). Each *conversion* requires a key and converts its input, a sequence of identifiers or pseudonyms, into a sequence of pseudonyms. Pseudonyms of the same citizen in different locations are unlinkable. We define three location types: the identity location, the disjoint location and the join location: The unique *identity location* (big circle, full line) represents a sequence of identifiers. A *disjoint location* (small circle) represents a sequence of pseudonyms of different citizens. These pseudonyms are unlinkable to other pseudonyms originating from the same identifier. A *join location* (big circle, dashed line) represents local equality. Previously unlinkable pseudonyms of the same citizen will, after conversion, coincide in a join location.

A pseudonym conversion network consists of pseudonym conversion paths. A *pseudonym conversion path* is defined by a sequence of keys $K \in \mathcal{K}^l$ with $l \in \mathbb{N}_0$. An incomplete path starts at the identity location and is followed by one or more disjoint locations (the one at the lower left in fig. 2). A complete path additionally ends in a join location (the other five paths in fig. 2).

Each gray zone in figure 2 represents the knowledge (identifiers, pseudonyms and keys) of one particular participant (provider, domain, project, ...). The four providers in the figure all contain the identity location, the four domains only contain disjoint locations and the two projects contain each a join location. (The path at the lower right is discussed in section 6.) An entity possesses per path maximum one key and can, hence, only do a single conversion in that path. A project contains the join location of multiple paths and is therefore able to link pseudonyms and their associated PII.

The *pseudonym level* of a location is the maximum number of conversions over known paths to generate pseudonyms in this location, starting from an identifier. Identifiers, pseudonyms in domains and pseudonyms

Notation: k_w^i, id_j, $n_{w,j}^i$, N_w^i indicate respectively keys, identifiers, pseudonyms and pseudonym sequences. w denotes the pseudonym level, i the index of a location or conversion for a particular pseudonym level. Hence $w||i$ is a globally unique identifier for a location, which will be written as loc_w^i. If the pseudonym level is 0, it is not written. j is an identifier index or pseudonym index.

in projects have pseudonym levels 0, 1 and at least 3 respectively. Also the key required to convert a pseudonym from level $w-1$ to pseudonym level w has pseudonym level w.

The properties of collision resistance and determinism of the pseudonym conversion function ensure that each identifier is represented by no more than one pseudonym in each location and that a pseudonym in a given location is derived from a single identifier. Since conversions can go into both directions, no arrows are drawn. This bi-directionality will be useful for deanonymising purposes explained in section 6. The necessity of unlinkability and local conversion in the network has already been argued. Hence, given a pseudonym conversion function with its five properties, pseudonym conversion networks can be defined.

4.3 Realisation in elliptic curves

Elliptic curves offer an elegant way to fulfill the previously formulated properties. In summary, the Elliptic Curve Discrete Logarithm Problem (ECDLP) states that on an elliptic curve $\mathcal{E}_q$ of order q a scalar multiplication $Q \leftarrow n \cdot P$ can be defined with $P, Q \in \mathcal{E}_q, n \in \mathbb{Z}_q$ such that it is easy to calculate Q given n and P, but extremely hard to find n given P and Q.

Identifiers and pseudonyms are points on an elliptic curve $\mathcal{E}_q$. Secrets and keys are elements in $\mathbb{Z}_q$. The key generation protocol by a central, trusted authority is given in algorithm 1 and illustrated in table 1 with two complete paths of length 3 and 4. We take q prime. Note that knowledge of a key composed of two secrets does not reveal knowledge of these secrets. Section 7 discusses alternatives to key distribution by a single trusted authority.

Notation: Points on the elliptic curve are written in bold in this subsection. The function $e(x)$ calculates y_1, y_2 such that (x, y_1) and (x, y_2) are points on the elliptic curve. $a *_q b$ is a short notation for $a * b \bmod q$. The point id_j is an identifier of which the original numerical identifier, such as the social security number, is τ_j.

Algorithm 1: Key generation by a central authority

input: $m, l_0, ..., l_{m-1} \in \mathbb{N}_0$ // # paths & path lengths

$s_{join} \xleftarrow{\$} \mathbb{Z}_q$

$for\ i : 0..m-1\{$

$\quad for\ w : 1..l_i\{$

$$\begin{cases} s_w^i \xleftarrow{r} \mathbb{Z}_q & \text{when } w \neq l_i \\ s_w^i \leftarrow s_{join} & \text{otherwise} \end{cases}$$

$$k_w^i \leftarrow \begin{cases} s_1^i & \text{when } w = 1 \\ s_w^i *_q (s_{w-1}^i)^{-1} & \text{otherwise} \end{cases}$$

$\quad \}$

$\}$

Conversion from and to identifiers is done using a key that equals a single secret, while conversion between pseudonyms is done using a key that is composed of two secrets. For each pseudonym $\boldsymbol{n_{w,j}^i} = s_w^i \cdot \boldsymbol{id_j}$. Each location loc_w^i has, hence, a unique associated secret s_w^i. Going from locations $loc_w^{i_0}$ and $loc_w^{i_1}$ to a join location loc_{w+1}^i is done using keys of the form $s_{w+1}^i *_q (s_w^{i_0})^{-1}$ and $s_{w+1}^i *_q (s_w^{i_1})^{-1}$.

	Path 4	Path 5
Secret generation	$s_{join} \xleftarrow{r} \mathbb{Z}_q$	
	$s_1^4, s_2^4 \xleftarrow{r} \mathbb{Z}_q$	$s_1^5, s_2^5, s_3^5 \xleftarrow{r} \mathbb{Z}_q$
Key generation	$k_1^4 \leftarrow s_1^4$	$k_1^5 \leftarrow s_1^5$
	$k_2^4 \leftarrow s_2^4 *_q (s_1^4)^{-1}$	$k_2^5 \leftarrow s_2^5 *_q (s_1^5)^{-1}$
		$k_3^5 \leftarrow s_3^5 *_q (s_2^5)^{-1}$
	$k_3^4 \leftarrow s_{join} *_q (s_2^4)^{-1}$	$k_4^5 \leftarrow s_{join} *_q (s_3^5)^{-1}$
Pseudonym conversion	$\boldsymbol{n_{1,j}^4} \leftarrow k_1^4 \cdot \boldsymbol{id_j}$	$\boldsymbol{n_{1,j}^5} \leftarrow k_1^5 \cdot \boldsymbol{id_j}$
	$\boldsymbol{n_{2,j}^4} \leftarrow k_{2,j}^4 \cdot \boldsymbol{n_{1,j}^4}$	$\boldsymbol{n_{2,j}^5} \leftarrow k_{2,j}^5 \cdot \boldsymbol{n_{1,j}^5}$
		$\boldsymbol{n_{3,j}^5} \leftarrow k_3^5 \cdot \boldsymbol{n_{2,j}^5}$
	$\boldsymbol{n} \leftarrow k_{3,j}^4 \cdot \boldsymbol{n_{2,j}^4}$	$\boldsymbol{n} \leftarrow k_4^5 \cdot \boldsymbol{n_{3,j}^5}$

Table 1. Example of paths of length 3 and 4. They correspond to the paths determinded by superscripts 4 and 5 in figure 2.

Algorithm 2: Conversion of numerical identifier into (base) point

input: l, τ_j

$x = \tau_j * 2^l$ // shift left

$while(!quadraticResidue(x, q)):\ x{+}{+}$

$y = max(e(x))$

$return\ (x, y)$

Algorithm 2 is a deterministic method to convert classical identifiers such as social security numbers into identifiers as points on an elliptic curve. The x-value of

the point is determined by taking the classical identifier in a numerical representation and appending a sequence of zeroes. This value is incremented until it is a quadratic residue in $\mathbb{Z}_q$. The resulting x-value has the binary form $\tau_j||0...0||nb-1$. The biggest of the maximum two corresponding y-values is calculated. The resulting (x, y) couple is a valid point on the curve and becomes the identifier. Note that it is trivial to derive a numerical identifier τ_j back from an identifier id_j. Also, since q is prime, each identifier is a base point (generator).

Although the theoretical maximum sequence of (non-)quadratic residues is $\mathcal{O}(q^{\frac{1}{4}+\delta})$ [17], a quadratic residue is found quickly in practice. Simulations were done using the standard SEC parameters $secp256k1$ [18] with $l = 64$ on the complete set of 36.5 million numbers that are well-formatted (and potentially valid) Belgian social security numbers. This showed that the quadratic residues are distributed randomly over a uniform distribution and that, hence, the probability to find a quadratic residue after exactly nb attempts is 2^{-nb}. The probability to find a quadratic residue in maximum nb attempts is, hence, $\sum_{x=1}^{nb} 2^{-x} = 1 - 2^{-nb}$. The highest number of required attempts was 27 (for SSN 64.01.28-018.89 and $l = 64$).

4.4 Proofs of correctness

Unlinkability. Since $\mathcal{E}_q$ has prime order (q), each point and, hence, each identifier and pseudonym, is a base point. We therefore have: $\forall\ \boldsymbol{n_1}, \boldsymbol{n_2}, \boldsymbol{n} \in \mathcal{E}_q : \exists\ k_1, k_2 \in \mathbb{Z}_q$ s.t. $\boldsymbol{n_1} \leftarrow k_1 \cdot \boldsymbol{n}$ and $\boldsymbol{n_2} \leftarrow k_2 \cdot \boldsymbol{n}$. Therefore, without knowledge of the keys, each pseudonym or identifier has the same probability to be the common parent. Since a sequence of keys $K \in \mathcal{K}$ can be merged into a single key $k_{merged} \leftarrow \prod_{k \in K} k \bmod q$, the result can be generalised to $\overline{f}$: $\overline{f}(K, \boldsymbol{n}) = f(\prod_{k \in K} k \bmod q, \boldsymbol{n})$.

Secondly, we should ensure that no $\boldsymbol{g}$ exists such that each identifier $\boldsymbol{id_j}$ can be written as $\tau_j \cdot \boldsymbol{g}$, since this would cause a vulnerability. It would suffice to take two random pseudonyms $\boldsymbol{n}^i_{w,0}$ and $\boldsymbol{n}^i_{w,1}$ in location loc^i_w, which have the form $\boldsymbol{n}^i_{w,0} \leftarrow (s^i_w * \tau_0) \cdot \boldsymbol{g}$ and $\boldsymbol{n}^i_{w,1} \leftarrow (s^i_w * \tau_1) \cdot \boldsymbol{g}$. Then, find a couple of numerical identifiers (τ_0, τ_1) such that $\tau_1 \cdot \boldsymbol{n}^i_{w,0} = \tau_0 \cdot \boldsymbol{n}^i_{w,1}$. This could be found relatively quickly due to the relatively small set of potential numerical identifiers. This would allow to calculate for each τ_j the pseudonym for location loc^i_w: $\boldsymbol{n}^i_{w,j} \leftarrow (\tau_j * \tau_0^{-1}) \cdot \boldsymbol{n}^i_{w,0}$. This is prevented in algorithm 2.

Local equality. We have $K^i = \langle k^i_1, ..., k^i_{l_i} \rangle$ and $k^i_1 = s^i_1$, $k^i_2 = s^i_2 *_q (s^i_1)^{-1}$, ..., $k^i_{l_i} = s_{join} *_q (s^i_{l_i-1})^{-1}$. Hence, $\prod_{k \in K^l} k \bmod q = s_{join}$, independent of the previous secrets. The only two requirements to create a new key sequence (= a path) of length l defined by $K = \langle k_1, ..., k_l \rangle$ are 1) the secrets s_1, ..., s_{l-1} are generated randomly and 2) the final key k_l has the form $s_{join} * (s_{l-1})^{-1}$.

Collision resistance. This follows immediately from the fact that each point is a base point in $\mathcal{E}_q$ with q prime. The probability of a collision in a set of m pseudonyms is $1 - \frac{q-1}{q} * \frac{q-2}{q} * ... * \frac{q-m}{q}$. According to the birthday paradox, the probability exceeds 0.5 when $m \geq q/2$. For sufficiently large q (e.g. 256 bits), the probability to have a collision is therefore negligible, even for large population sets.

Deterministic. Trivial.

Invertible. Since the group has prime order, the inverse of each key can easily be computed using the extended Euclidean algorithm.

5 Protocols

This section discusses three protocols, with increasing complexity, to combine PII from domains into a project. The assumptions and requirements are summed up and are followed by the discussion of the protocols.

5.1 Assumptions and requirements

Infrastructure (I), participant (P) and citizen (C) assumptions are distinguished:

- I1 *Secured keys.* Keys or secrets are never disclosed. This can be achieved using HSMs (hardware security modules).
- I2 *Secure communication.* Communication between participants and islands and between islands mutually guarantees confidentiality and integrity.
- I3 *Isolation.* Each island is located in a completely isolated environment, even when these environments are running on the same physical machine.
- P1 *Covert adversaries.* Participants are curious but do not want that their abuse/curiosity is detected.
- P2 *Domain control.* A domain island is under the control of and managed by a single provider.

P3 *Collusion.* Two types of collusion are possible. Compromised domains collude mutually or projects collude mutually, both to obtain more PII. No other collusions are considered. This is natural given the colliding interests of participants. Contractual/legal means can help at enforcing this separation of duties.

C1 *Identifier.* Each citizen has a unique identifier used by all government agencies (providers). This is for instance the case in Belgium.

Under these assumptions, the following requirements should be realised:

CL *Collaborative linkability.* The project can link PII of the same citizen originating from different domains, but only if the involved domains agree.

MKP *Minimum knowledge for projects.* The project does not learn more information than what is strictly necessary. This implies that it does not learn 1) anything about uninvolved records and 2) more than the minimum required PII per involved record.

ZKD *Zero-knowledge for domains.* A protocol execution does not leak PII to domains; domains cannot link extra PII to their records than what they knew before.

The first protocol (section 5.2) realises only the CL requirement, the second (section 5.3) also realises the MKP requirement and the third and most complex protocol (section 5.4) realises all three requirements with the help of an extra entity. For each deployment of the data archipelago, a trade-off can result in another protocol choice, based on the concrete needs and resources.

5.2 Collaborative linkability

Three types of pseudonyms are defined – domain, (temporary) transfer and project pseudonyms. They are shown in figure 2 and reoccur in algorithm 3.

Notation: i, z and j are the domain identifier, the project identifier and the index of the citizen or pseudonym. $n^i_{1,j} \in N^i_1$, $n^i_{2,j} \in N^i_2$, and $n^z_{3,j} \in N^z_3$ denote domain, transfer and project pseudonyms. The corresponding PII are denoted as $d^i_{1,j} \in D^i_1$, $d^i_{2,j} \in D^i_2$ and $d^z_{3,j} \in D^z_3$. $d^i_{w,j}$ does not contain identifiers or pseudonyms. m is the number of involved domains and hence $i \in \{0, ..., m-1\}$. The query ζ defines 1) the selection of involved citizens, based on properties such as age, and 2) the required PII of these citizens.

In the first phase (1. pseudonyms), the project learns

Algorithm 3: Collaborative linkability (CL)

Domain i, $i\|_0^{m-1}$		Project z
input: ζ, i, N^i_1, D^i_1, k^i_2		input: ζ, m, $\langle k^0_3, ..., k^{m-1}_3 \rangle$
1. Pseudonyms		
$\zeta_i \leftarrow getQuery(\zeta, i)$		$for\ i : 0...m-1\{$
$N^{i\diamond}_1 \leftarrow getNyms(N^i_1, \zeta_i)$		
$N^i_2 \xleftarrow{rp} \widehat{f}(k^i_2, N^{i\diamond}_1)$	$\xrightarrow{N^i_2}$	
		$\}$
		$N^z_3 \leftarrow \bigcap_{i=0}^{m-1} \widehat{f}(k^i_3, N^i_2)$
2. PII		
		$for\ i : 0...m-1\{$
$N^{i\star}_1 \leftarrow \widehat{f}^{-1}(k^i_2, N^{i\star}_2)$	$\xleftarrow{N^{i\star}_2}$	$N^{i\star}_2 \xleftarrow{rp} \widehat{f}^{-1}(k^i_3, N^z_3)$
$D^i_2 \leftarrow getPii(D^i_1, N^{i\star}_1, \zeta_i)$	$\xrightarrow{D^i_2}$	$D^i_2 \leftarrow map(D^i_2, N^{i\star}_2, N^z_3)$
		$\}$
		$for\ j : 0...\|N^z_3\| - 1\{$
		$d^z_{3,j} \leftarrow \{d^0_{2,j}, ..., d^{m-1}_{2,j}\}$
		$rec^z_{3,j} \leftarrow \{n^z_{3,j}, d^z_{3,j}\}$
		$\}$

the project pseudonyms of the involved citizens, but not yet any other PII. The *selection of citizens* is part of an approved and publicly known query ζ and is defined by a set of personal *properties*. For instance, a project may need data about all citizens who 1) are born after 1990, 2) earn yearly more than €50.000 and 3) are married. However, not all the involved *attributes* (DoB = *Date-of-Birth*, *income* and *status* in our example) are necessarily stored by the same domain. Therefore, the query ζ will be split in separate queries ζ_0, ζ_1, ..., ζ_{m-1}. Query ζ_i can be answered by (and only by) domain i, since it is the only domain that knows the attributes used in ζ_i. Let's assume in our example that the DoB, *income* and *status* are respectively stored by domain 0, domain 1 and domain 2. Then $\zeta_0.selection$ = "$DoB \geq 1990$", $\zeta_1.selection$ = "$income \geq$ €50.000" and $\zeta_2.selection$ = "$status = married$". Each involved domain i selects the sequence of domain pseudonyms $N^{i\diamond}_1 (\subset N^i_1)$ that matches the citizen selection in ζ_i and converts them into the sequence of transfer pseudonyms N^i_2. All domains send their sequence of transfer pseudonyms to project z. The latter converts these m received sequences again and takes the intersection. This results in the sequence of project pseudonyms denoted by N^z_3.

All transfer pseudonyms will (with a very high probability) be different, even if they correspond to the same citizen, as illustrated in figure 3. However, if different transfer pseudonyms, sent by different domains, are converted by the project, they will result in the same project pseudonym if (and only if) they correspond to the same citizen. Taking the intersection of all these project pseudonyms (in gray in figure 3) results, hence,

in the required set of project pseudonyms. In our example this intersection contains exactly the pseudonyms corresponding to married citizens born after 1990 with a yearly income higher than €50.000.

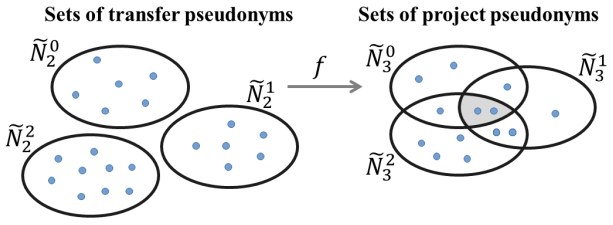

Fig. 3. Disjoint sets of transfer pseudonyms originating from three different domains are converted into intersecting sets of project pseudonyms.

In the second phase (2. PII), the project maps for each domain i the project pseudonyms in the intersection back to transfer pseudonyms ($N_2^{i\star} \subset N_2^i$). The domain receives this and maps it back to domain pseudonyms ($N_1^{i\star} \subset N_1^{i\diamond}$) and selects now for these (permanent) domain pseudonyms the PII required by the project ($D_2^i = \langle d_{2,0}^i, ..., d_{2,|N_1^{i\star}|-1}^i \rangle$) and returns them. The order in D_2^i and $N_1^{i\star}$ should correspond such that the project can map the PII in D_1^i to the pseudonyms in N_3^z. Once the project has this result from each of the involved domains, it can trivially link the PII of the same citizen on the basis of the identical project pseudonyms. The project can now analyse the combined PII set.

Performance. Domain i does $|N_2^i|$ pseudonym conversions and the project $\sum_{i=0}^{m-1} |N_2^i|$. We assume that values are cached instead of recalculated. Domain i sends $|N_2^i| * |q|$ bits plus the minimal required data, contained in D_2^i to the project. The amount of data sent by the project to each individual domain is $|N_3^z| * |q|$.

Requirements. *Collaborative linkability (CL)* is fulfilled. As explained, the protocol allows the project to link PII stemming from different domains, but this requires the collaboration of the involved domains. However, the protocol fails at fulfilling MKP and ZKD.

Minimum knowledge for project (MKP) is not met. The project can indeed cheat by asking PII corresponding to a project pseudonym that was not in the intersection since the domain does not know which of the sent transfer pseudonyms ended up – after conversion to project pseudonyms – in this intersection. In our example, project z could e.g. ask domain 1 PII about unmarried citizens born before 1990 who earn more than €50.000. Although the project can this way request too many records, it cannot ask too much PII per record since this is described in the public query ζ.

Zero-knowledge for domains (ZKD) is not met either. Each domain i learns new information about both $\widetilde{N_1^{i\star}}$ and $\widetilde{N_1^{i\diamond}} \setminus \widetilde{N_1^{i\star}}$ (unless the project is cheating). For example, domain 1 learns that $\widetilde{N_1^{1\star}}$ corresponds to married citizens born after 1990, while $\widetilde{N_1^{1\diamond}} \setminus \widetilde{N_1^{1\star}}$ corresponds to citizens who do not have this property combination. This leakage is incremental over projects.

Fortunately, since the order of the transfer pseudonyms in each $N_2^{i\star}$ is different, colluding (compromised) domains cannot link data on the basis of the order of the received pseudonyms in step 2. Also, since different projects have different, unlinkable join locations, colluding projects cannot link information on the basis of project pseudonyms or transfer pseudonyms.

5.3 Minimisation knowledge for projects

To meet the MKP requirement, each domain i adds a number of fake transfer pseudonyms to the sequence N_2^i, which are indistinguishable by the project from real transfer pseudonyms. Both fake and real transfer pseudonyms are upon receipt by the project converted into indistinguishable fake and real project pseudonyms.

Just as real project pseudonyms, fake project pseudonyms should appear in significant amounts in each region of the Venn diagram at the right side of figure 3, except – and this is different from real project pseudonyms – in the intersection $\bigcap_{i=0}^{m-1} \widetilde{N_3^i} = \widetilde{N_3^z}$. This will deter the project to request more data than strictly necessary due to the high risk of detection.

An extension of the previous protocol is presented in which each region in the Venn diagram, except the intersection (in gray), contains an equal amount of fake pseudonyms.

Algorithm 4 shows how *total* fake identifiers are initially generated. This can be done by each provider separately. Alternatively, one entity can do this once and publish the result. The latter is more efficient and guarantees that all providers use the same fake identifier given the same index. To generate the fake identifier on the elliptic curve, j is shifted $l/2$ positions to the left. The first quadratic residue equal or bigger than this value is taken. Algorithm 2 shifted a numerical identifier l positions to obtain an identifier point. If l is chosen sufficiently large (e.g. 64 bits as shown in section 4.3),

a sufficiently large amount of fake pseudonyms can be generated without risking overlap with real identifiers.

The second step is shown in algorithm 5 and is performed once per domain: the public fake identifiers are converted into a fake domain pseudonyms which are sent to the domain, together with the index from which the fake identifier was initially derived.

Before proceeding, we mention that all the regions in the Venn diagram (figure 3, right) can be enumerated from 1 to $2^m - 1$. This becomes clear when the index r of the region is written in binary with m digits; one for each pseudonym set (or domain). The i^{th} bit from the right (starting at 0) indicates if domain i is involved in this region. Hence, for $m = 5$, $r = 00101$ indicates $(\widetilde{N_3^0} \cap \widetilde{N_3^2}) \setminus (\widetilde{N_3^1} \cup \widetilde{N_3^3} \cup \widetilde{N_3^4})$ and $r = 11111$ the central intersection $\sum_{i=0}^{4} \cap \widetilde{N_3^i}$.

The next step is now shown in algorithm 6 and is illustrated in figure 4. nb is the amount of fake pseudonyms in each region. The domains agree on this value beforehand, e.g. using a secure multi-party protocol (to prevent leakage of information). Each region r is assigned a sequence of nb indices, going from $(r-1)*nb$ to $r * nb - 1$ (precondition: $total \geq nb * (2^m - 2)$). Each index corresponds to a fake identifier. If a domain is involved in a region it creates nb fake transfer pseudonyms based on these indices. Determining if domain i is involved in a region is done by checking if the i^{th} bit of r is 1. Once a domain added all the fake transfer pseudonyms to the sequence of transfer pseudonyms N_2^i, it sends this sequence to the project, which will convert them all into project pseudonyms. If multiple domains choose the same index (fake identifiers), the resulting fake project pseudonyms coincide.

Performance. Since $\forall\ i \in \{0, ..., m-1\}$ there exist exactly $2^{m-1} - 1$ binary numbers $\in \{1, ..., 2^m - 2\}$ with the i^{th} bit 1, each domain converts and transfers $(2^{m-1} - 1) * nb$ extra pseudonyms (of size $|q|$) and the project receives and converts $m * (2^{m-1} - 1) * nb$ extra pseudonyms.

Requirements. This scheme is an extension of the first, which already fulfilled the CL requirement. By construction, we have now nb fake pseudonyms in each region except the central intersection. This will, even for small nb values, deter the project from requesting more data than needed, given the indistinguishability between real and fake pseudonyms (see further). It can, hence, no longer actively intervene to get hold of more information than strictly necessary. However, if the project is able to get hold of the value of nb, it learns the real size of each region, which we consider as acceptable. The ZKD requirement is still not met as shown in the previous subsection.

Algorithm 4: Fake identifier generation

```
Input: l, total
F ← ∅
for j : 0...total − 1{
    x ← j * 2^{l/2}
    while(!quadraticResidue(x)) : x++
    y ← max(e(x))
    fakeId_j ← (x, y)
    F.append(⟨j, fakeId_j⟩)
}
```

Algorithm 5: Fake domain pseudonym generation

```
Input: F, k_1^i
for each ⟨j, fakeId_j⟩ ∈ F {
    ν_{1,j}^i ← f(k_1^i, fakeId_j)
    send (j, ν_{1,j}^i) to domain i
}
```

Algorithm 6: Fake transfer pseudonym generation

```
Input: m, nb, i, k_2^i, N_2^i
for r : 1...2^m − 2{
    if(binary(r)[i] = 1){
        for x : 0...nb − 1{
            j ← (r − 1) * nb + x
            N_2^i.append(f(k_2^i, ν_{1,j}^i))
        }
    }
}
```

Indistinguishability real and fake pseudonyms. Let $\mathcal{I}$ and $\mathcal{F}$ be the sets of all possible real and fake identifiers respectively. Indistinguishability formally means: $\forall\ K^0, K^1 \in \mathcal{K}^*, id_0 \in \mathcal{F}, id_1 \in \mathcal{I} : |P[1 \leftarrow A_f(\overline{f}(K^0, id_0))] - P[1 \leftarrow A_f(\overline{f}(K^1 \cdot id_1))]| < \varepsilon$, whereby A_f outputs 1 if the input is considered as a pseudonym derived from a real identifier and 0 otherwise. For the EC-based construction in section 4.3, this can be written as: $\forall\ s_0, s_1 \in \mathbb{Z}_q, id_0 \in \mathcal{F}, id_1 \in \mathcal{I} : |P[1 \leftarrow A_f(s_0 \cdot id_0)] - P[1 \leftarrow A_f(s_1 \cdot id_1)]| < \varepsilon$.

Each fake or real identifier is a base point. Therefore: $\forall\ id_0 \in \mathcal{F}, id_1 \in \mathcal{I}, s_0, s_1 \in \mathbb{Z}_q, \exists\ s_0', s_1' \in \mathbb{Z}_q$ s.t. $s_0' \cdot id_0 = s_1 \cdot id_1$ and $s_1' \cdot id_1 = s_0 \cdot id_0$. Therefore, indistinguishability holds as long as the s-values are unknown to A_f (the project).

5.4 Zero-knowledge for domains

The approach to fulfill the ZKD requirement is as follows. Together with the transfer pseudonym, each do-

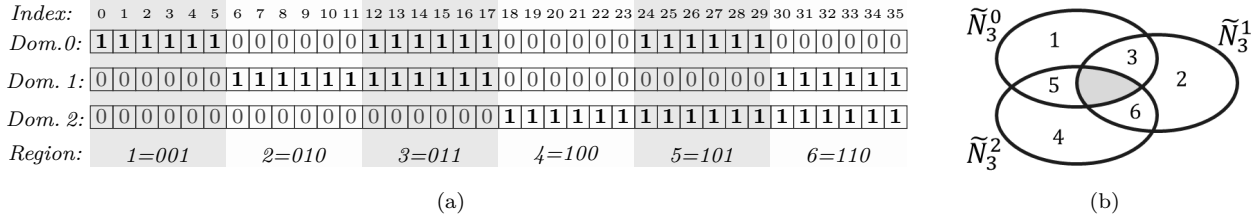

Fig. 4. (a) Selection of fake indices by the different domains for $m = 3$ and $nb = 6$. If a cell contains 1, a fake transfer pseudonym is generated from a domain pseudonym with the given index. (b) The indices of the regions in the Venn diagram.

Existing **secure multi-party modular multiplication** protocols [19] have the form $P_0, ..., P_{m-1} : (s; ...; s) \leftarrow \pi(s_0; ...; s_{m-1})$ with $s = \prod_{i=0}^{m-1} s_i \bmod q$, which is shared by all parties P_i after protocol execution. We define a variant $\tilde{\pi}$. If we have $m+1$ parties, m of them $(P_1, ..., P_m)$ have each a secret $s_i \in \mathbb{Z}_q$. As a result, the remaining party P_0 learns $\prod_{i=1}^{m} s_i \bmod q$. No other information is revealed to any of the $m+1$ participants. This protocol is denoted as $P_0, P_1, ...P_m : (s; \emptyset; ...; \emptyset) \leftarrow \tilde{\pi}(\emptyset; s_1; ...; s_m)$ with $s = \prod_{i=1}^{m} s_i \bmod q$. This can be easily derived from π by $P_0, P_1, ..., P_m : (s'; s'; ...; s') \leftarrow \pi(s_0; s_1; ...; s_m)$, whereby P_0 takes as input $s_0 \xleftarrow{\$} \mathbb{Z}_q$ a fresh random value and at the end calculates $s \leftarrow s' *_q s_0^{-1}$.

Algorithm 7: ZKD subprotocol 1; the setup

Domain i, $i\|_0^{m-1}$	Project z	S. Gen.
input: k_2^i	input: $m, \{k_3^0, ..., k_3^{m-1}\}$	
$b_2^i \xleftarrow{\$} \mathbb{Z}_q$	$b_3^z \xleftarrow{\$} \mathbb{Z}_q$	
$bk_2^i \leftarrow k_2^i *_q b_2^i$	$for\ i : 0...m-1 :$	
$\tilde{\pi}(.; (b_2^i)^{-1}; .)$	$\leftrightarrow \quad \tilde{\pi}(.; .; b_3^z *_q k_3^i) \quad \leftrightarrow$	$bk_3^i \leftarrow \tilde{\pi}(\emptyset; .; .)$

main sends to the project the encrypted PII and some key data. A project is only able to decrypt every encrypted PII that belongs to a specific project pseudonym if it received the corresponding key data from each of the m involved domains. Otherwise, it cannot decrypt any of this PII. These key data are distributed to the domains by a new entity; the *secret generator*. The protocol consists of three separately discussed subprotocols: 1) setup, 2) obtaining the secrets from the secret generator, and 3) sending the required data to the project.

The first subprotocol, **setup**, is shown in algorithm 7. It generates blinded keys bk_2^i and bk_3^i with the following two properties:

- **Key hiding.** Knowledge of bk_3^i with $i \in \{0, ..., m-1)$ does not reveal information about k_3^i. The secret generator, hence, does not learn anything about the keys k_3^i. Formally: $\forall\ i \in \{0, ..., m-1) : P[k_3^i \leftarrow A_K(bk_3^0, ..., bk_3^{m-1})] < \varepsilon$, where A_K tries to output an unblinded key, given all the blinded keys.
- **Blind isomorphism.** Two transfer pseudonyms are converted into coinciding project pseudonyms if and only if the corresponding blinded transfer pseudonyms are converted into coinciding blinded project pseudonyms. Formally:
$\forall\ id_j \in \mathcal{I}, i \in \{0, 1\}, k_1^i, k_2^i, k_3^i \in \mathcal{K}$,
$n_{1,j}^i \leftarrow f(k_1^i, id_j)$,
$n_{2,j}^i \leftarrow f(k_2^i, n_{1,j}^i), n_{3,j}^i \leftarrow f(k_3^i, n_{2,j}^i)$,
$bn_{2,j}^i \leftarrow f(bk_2^i, n_{1,j}^i), bn_{3,j}^i \leftarrow f(bk_{3,j}^i, bn_{2,j}^i)$:
$bn_{3,j}^0 = bn_{3,j}^1 \Leftrightarrow n_{3,j}^0 = n_{3,j}^1$

Proof key hiding. We work in the cyclic group $\mathbb{Z}_q, *$. $bk_3^i = (b_2^i)^{-1} *_q b_3^z *_q k_3^i$. For each bk_3^i, b_2^i is randomly generated over a uniform distribution in $\mathbb{Z}_q$. Therefore, the set of all possible bk_3^i has a random distribution over $\mathbb{Z}_q$ and bk_3^i is independent of k_3^i.

Proof blind isomorphism.

$$
\begin{aligned}
n_{3,j}^0 = n_{3,j}^1 &\Leftrightarrow (k_3^0 *_q k_2^0) \cdot n_{1,j}^0 = (k_3^1 *_q k_2^1) \cdot n_{1,j}^1 \\
&\Leftrightarrow ((b_2^0)^{-1} *_q b_3^z *_q k_3^0) *_q (b_2^0 *_q k_2^0)) \cdot n_{1,j}^0 \\
&\quad = ((b_2^1)^{-1} *_q b_3^z * k_3^1) *_q (b_2^1 *_q k_2^1)) \cdot n_{1,j}^1 \\
&\Leftrightarrow (bk_3^0 *_q bk_2^0) \cdot n_{1,j}^0 = (bk_3^1 *_q bk_2^1) \cdot n_{1,j}^1 \\
&\Leftrightarrow bn_{3,j}^0 = bn_{3,j}^1
\end{aligned}
$$

Algorithm 8 shows the second subprotocol, **obtaining the secrets from the secret generator**. Each domain i sends blinded transfer pseudonyms BN_2^i to the secret generator, which converts them into blinded project pseudonyms BN_3^i. Domain i receives for each blinded transfer pseudonym $bn_{3,j}^i$ a vector t_j^i of length $m+1$. If the corresponding blinded project pseudonym is in the central intersection, an $m \times m$ matrix of random values is generated by the secret generator, and each domain i will receive as vector t_j^i the i^{th} row and the seed $seed_j^i$, which is the modular multiplication of the i^{th} column of the matrix. If the blinded project pseudonym is not in the intersection, the vector will just contain random values. The blind isomorphism property guarantees that the intersection of blinded project

Algorithm 8: ZKD subprotocol 2: obtaining the secrets

Domain i, $i|_0^{m-1}$
input: ζ, i, N_1^i, bk_2^i
$\zeta_i \leftarrow getQuery(\zeta, i)$
$N_1^{i\diamond} \leftarrow getNyms(N_1^i, \zeta_i)$
$BN_2^i \xleftarrow{rp} \widehat{f}(bk_2^i, N_1^{i\diamond})$

BN_2^i

Secret generator
input: $m, \langle bk_2^0, ..., bk_2^{m-1}\rangle$
$for\ i : 0...m-1$
  $BN_3^i \leftarrow \widehat{f}(bk_3^i, BN_2^i)$
$\langle bn_{3,0}^z, ..., bn_{3,|BN_3^z|-1}^z\rangle = BN_3^z \leftarrow \bigcap_{i=0}^{m-1} BN_3^i$
$for\ y : 0...|BN_3^z| - 1\{$

$$\begin{bmatrix} R_y^0 \\ \vdots \\ R_y^{m-1} \end{bmatrix} = \begin{bmatrix} r_{0,0} & \cdots & r_{0,m-1} \\ \vdots & \ddots & \vdots \\ r_{m-1,0} & \cdots & r_{m-1,m-1} \end{bmatrix} \xleftarrow{\$} \mathbb{Z}_q^{m\times m}$$

  $for\ i : 0...m-1 : seed_y^i \leftarrow \prod_{x=0}^{m-1} r_{x,i} \bmod q$
$\}$
$for\ i : 0...m-1\{$
  $for\ j : 0...|BN_2^i| - 1\{$
    $y \leftarrow BN_3^z.getIndex(f(bk_3^i, bn_{j,2}^i))$

$$\begin{cases} t_j^i \leftarrow \langle R_y^i, seed_y^i\rangle & \text{when } y \text{ valid index} \\ t_j^i \xleftarrow{\$} \mathbb{Z}_q^{m+1} & \text{otherwise} \end{cases}$$

  $\}$
  $T^i \leftarrow \langle t_0^i, ..., t_{|BN_2^i|-1}^i\rangle$

T^i

$\}$

Algorithm 9: ZKD subprotocol 3; sending and decrypting PII

Domain i, $i|_0^{m-1}$
input: ζ_i, D_1^i, k_2^i, $N_1^{i\diamond}, T^i = \langle t_0^i, ...t_{|T^i|-1}^i\rangle$
$U^i \leftarrow \emptyset$
$for\ j : 0...|N_1^{i\diamond}| - 1\{$
  $key_j^i \leftarrow genKey(t_j^i.seed_j^i)$
  $d_{2,j}^i \leftarrow getPII(D_1^i, n_{1,j}^{i\diamond}, \zeta_i)$ // $n_{1,j}^{i\diamond} \in N_1^{i\diamond}$
  $c_j^i \leftarrow enc(key_j^i, d_{2,j}^i)$
  $n_{2,j}^i \leftarrow f(k_2^i, n_{1,j}^{i\diamond})$
  $R_j^i \leftarrow t_j^i.R_j^i$ // omit seed
  $U^i \xleftarrow{rp} U^i \cup \langle n_{2,j}^i, c_j^i, R_j^i\rangle$
$\}$

U^i

Project z
input: m, $\langle k_3^0, ..., k_3^{m-1}\rangle$
$for\ i : 0...m-1 : N_2^i \leftarrow getNymSeq(U^i)$
$N_3^z \leftarrow \bigcap_{i=0}^{m-1} \widehat{f}(k_3^i, N_2^i)$
$for\ y : 0...|N_3^z| - 1\{$
  $for\ i : 0...|m| - 1\{$
    $n_{2,y}^i \leftarrow f^{-1}(k_3^i, n_{3,y}^z)$
    $\langle c_y^i, R_y^i\rangle \leftarrow get(U^i, n_{2,y}^i)$
  $\}$
  $for\ i : 0...|m|-1 : \langle r_{i,0}, ..., r_{i,m-1}\rangle \leftarrow R_y^i$
  $for\ i : 0...|m| - 1\{$
    $seed_y^i \leftarrow \prod_{x=0}^{m-1} r_{x,i} \bmod q$
    $key_y^i \leftarrow genKey(seed_y^i)$
    $d_{1,y}^i \leftarrow dec(key_y^i, c_y^i)$
  $\}$
  $d_{3,y}^z \leftarrow \langle d_{1,y}^0, ..., d_{1,y}^{m-1}\rangle$
  $rec_{3,y}^z \leftarrow \langle n_{3,y}^z, d_{3,y}^z\rangle$
$\}$

pseudonyms BN_3^z corresponds to the intersection of unblinded project pseudonyms N_3^z in the next subprotocol. t-Indistinguishability (see below) guarantees that the domain cannot distinguish t-vectors generated on the basis of a matrix from the ones that are completely random. As a consequence, the domain has to treat each involved record the same way.

t-Indistinguishability. In subprotocol 2, the t-vectors generated as $t \leftarrow \langle R_y^i, seed_y^i\rangle$ ($\mathcal{T}_1$) cannot be distinguished by a domain from the ones generated as $t \xleftarrow{\$} \mathbb{Z}_q^{m+1}$ ($\mathcal{T}_0$). Formally: $\forall\ m \in \mathbb{N}_0, t_0 \in \mathcal{T}_0, t_1 \in \mathcal{T}_1 : |P[1 \leftarrow A_t(t_0)] - P[1 \leftarrow A_t(t_1)]| < \varepsilon$, where 1 is output by A_t if, according to A_t, its input belongs to $\mathcal{T}_1$ and 0 otherwise.

This is trivial given that 1) for both vector types, the first m elements are selected randomly and independently over a uniform distribution and 2) the final element is either selected randomly over a uniform distribution ($\mathcal{T}_0$) or it is the modular multiplication of one known value and one or more unknown random numbers over a uniform distribution ($\mathcal{T}_1$), which is again a random number selected over a uniform distribution.

Algorithm 9 presents subprotocol 3, **sending the required data to the project**. The seed is used to deterministically generate a symmetric key with which the PII is encrypted. The transfer pseudonym will be sent to the project together with the encrypted PII and the random values received from the secret generator. Each project pseudonym in the intersection will have its own full $m \times m$ matrix. Only if a project pseudonym is in the intersection, the project will have received each row from a different domain, enabling the project to recompose the full matrix. This allows the project to generate the corresponding keys, which are based on the columns of the matrix, to decrypt all the corresponding PII, originating from the different domains.

Performance. The computational complexity is given in table 2 and the amount of transferred data in table 3. Values are cached instead of recalculated.

Requirements. The CL requirement is fulfilled as argued in this section. The property of local equality guarantees that records of the same citizen can be linked

	Domain i	Project	Secret gen.
Pseudonym convers.	$2 * \|N_2^i\|$	$\sum_{i=0}^{m-1} \|N_2^i\|$	$\sum_{i=0}^{m-1} \|N_2^i\|$
Encrypt/decrypt	$\|N_2^i\|$	$m * \|N_3^z\|$	

Table 2. Computational complexity of the ZKD scheme.

Domain $i \rightarrow$ Sec. gen.	$\|N_2^i\| * \|q\|$
Domain $i \leftarrow$ Sec. gen.	$\|N_2^i\| * (m+1) * \|q\|$
Domain $i \rightarrow$ Project	$\|N_2^i\| * (m+1) * \|q\| + \sum_{j=0}^{\|N_2^i\|-1} \|d_{2,j}^i\|$

Table 3. Amount of data transferred in the ZKD scheme

together. If a project pseudonym is in the intersection, the project is able to decrypt all the corresponding PII, thanks to the blind isomorphism property.

The PDM requirement is met. The project is in this protocol a passive entity. Each active intervention by the projects will therefore be detected and be suspicious. Just as in the previous protocol, it will learn the size of each region. Since the order of each BN_2^i and U^i is changed, no information or linkabilities can be derived on the basis of the order of the elements.

The ZKD requirement is fulfilled thanks to t-indistinguishability. As a consequence, the domain has to treat each involved record in exactly the same way.

In addition, the key hiding property guarantees that the secret generator does not learn information about the keys k_3^i.

The domains can only recompose the matrices for each pseudonym if they are all compromised and collude. They do so by searching m t-vectors, one from each domain, that result in a valid combination of a matrix and m seeds. This way they can link transfer pseudonyms of the same citizen together and, hence, also the corresponding domain pseudonyms. However, the complexity thereof is high: $\mathcal{O}(\prod_{i=0}^{m-1} |N_2^i|)$.

6 Deanonymisation

Researchers analyse combined data in projects. In specific situations, deanonymisation (re-identification) of a pseudonym is required, e.g. in the context of fraud detection. However, not the researchers but only a responsible authority should be able to do this. Additionally, sometimes involvement of one or more providers and external parties (e.g. a supervisor) is required.

In figure 2, deanonymisation is possible by following from a join location to the identity location one of the paths that have superscripts 0, 1, 3 or 4. As a result, one of the involved providers learns the identity of the citizen and can take further actions.

This is not always desired. Therefore, an alternative path, for example the ones determined by superscript 5 in figure 2, can be taken. The key with the highest index in the path (k_4^5) is a secret key of the project, while the key with the lowest index is a secret key of the authority that can deanonymise the citizen in order to take further actions. The intermediate keys (k_2^5, k_3^5) are known by other entities. This could for instance be one of the involved data-delivering providers and a privacy watchdog.

The length of deanonymisation path and the entities that possess the keys in this path are typically determined at the project setup. The minimum path length for deanonymisation is two. In that case, the project does the first conversion and the authority the second, implying that no other entities are involved in the deanonymisation process.

The conversion done by the project prevents leaking the project pseudonym, which is only known inside the project. This prevents undesired linkabilities and even deanonymisations in case of a project data breach.

Note that each complete pseudonym conversion path can be used for deanonymisation, including the ones that were set up to deliver data to the project.

The presented deanonymisation approach is flexible and compatible with the legal requirement of proportionality. The involved organisations can maintain control by requiring their cooperation for each deanonymisation while they do not learn the identity of the affected citizen. The unlinkability property of f guarantees that none of the intermediate parties does learn anything about the affected citizen. Also, a single project can have multiple deanonymisation paths.

7 Key generation

The relationships between the keys require coordination during the key distribution process. Three approaches are discussed in this section.

The most straightforward approach requires a single central authority which knows all the secrets needed to generate the pseudonym conversion keys. The central authority generates the keys as shown in algorithm 1 and delivers each key to the right entity. The disadvantage is obviously the large amount of required trust by the other participants in this central authority.

Algorithm 10: Distributed generation of k_w^i for participant P_0

$\forall\ P_y \in \{P_1, ..., P_{m+1}\}$

Shared input: $w, i, id_{project}$ Private input: σ_y

$salt_w^i \leftarrow H(w||i||id_{project})$

$\sigma_w^{i,y} \leftarrow KDF(\sigma_y, salt_w^i)$

$if\ w \neq 1\{$

$\quad salt_{w-1}^i \leftarrow H(w-1||i||id_{project})$

$\quad \sigma_{w-1}^{i,y} \leftarrow KDF(\sigma_y, salt_{w-1}^i)$

$\}$

$$\kappa_w^{i,y} \leftarrow \begin{cases} \sigma_w^{i,y} & \text{if } w = 1 \\ \sigma_w^{i,y} *_q (\sigma_{w-1}^{i,y})^{-1} & \text{otherwise} \end{cases}$$

$P_0, P_1, ...P_{m+1}$

$(k_w^i; \emptyset; ...; \emptyset) \leftarrow \tilde{\pi}(\emptyset; \kappa_w^{i,1}; ...; \kappa_w^{i,m+1})$

The second approach involves multiple trusted authorities instead of one. The entity that needs a key contacts each authority separately and each authority executes independently of the others the key generation protocol presented in algorithm 1. The resulting partial keys are securely delivered to the right entity. This entity needs all these partial keys to derive its final key. More formally, a key k_i^x is derived as follows: $k_i^x \leftarrow \prod_{y=0}^{|Auth|-1} \kappa_i^{x,y} \bmod q$ where $|Auth|$ is the number of trusted authorities and $\kappa_i^{x,y}$ is the partial key for k_i^x generated by authority y. The secrets and keys can only be derived when all authorities collude. The cost of more distributed trust is a heavier infrastructure.

The final approach relies on a higher degree of trust distribution and requires the execution of the multi-party protocol $\tilde{\pi}$ (see section 5.4) with $m+2$ participants $P_0, ..., P_{m+1}$. These participants are the project (P_{m+1}), the involved providers ($P_1, ..., P_m$) and an entity to which the key will be issued (P_0). If the key-receiving participant P_0 is a provider, it controls an identity location, if it is a domain it controls a disjoint location, and if it is a project, it controls a join location. Algorithm 10 shows how a particular key k_i^x for P_0 can be generated in this multi-party approach. Each participant $P_y \neq P_0$ possesses a master secret. For all participants $P_y \neq P_0$, the same salt $salt_i^x$ is used. These salts are input, together with the master secret σ_y of the participant to a key derivation function KDF to generate a partial key $\kappa_w^{x,y}$. This partial key is given as input to $\tilde{\pi}$, which results in knowledge of the key k_w^i by – and only by – P_0. The keys in algorithm 1 were composed of one or two secrets s_w^i. If we take $s_w^i = \prod_{y=1}^{m+1} \sigma_w^{i,y}$, the previously defined relationships between keys and secrets hold: $k_w^i = \prod_{y=1}^{m+1} \kappa_w^{i,y} = \prod_{y=1}^{m+1} \sigma_w^{i,y} = s_w^i$ if $w = 1$ and $k_w^i = \prod_{y=1}^{m+1} \kappa_w^{i,y} = \prod_{y=1}^{m+1} \sigma_w^{i,y} *_q (\sigma_{w-1}^{i,y})^{-1} = s_w^i *_q (s_{w-1}^i)^{-1}$ otherwise.

Table 4 presents the main differences between the proposed approaches.

# Trusted authorities	1	>1	0
Cost infrastructure	medium	high	low
Trust in individual entity	high	medium	low
Probability key compromised	low	medium	high
Impact compromised key	high	medium	low

Table 4. Comparison of the three key distribution approaches.

8 Extensions

Some possible extensions are presented at a high level.

Although efficient during project creation, the default setup shown in figure 1 implies data replication which has negative repercussions on security and consistency. A subset of the data managed by the provider is indeed duplicated in the domain. The PII (and potentially also the pseudonyms and keys) can, therefore, be removed from the domain (and keys moved to the provider). If data that now only virtually remains in the domain is needed by a project, the required data is generated by the provider and sent to the domain, which forwards it to the project. This is called *data virtualisation*. The cost is a performance decrease during data collection by the project.

The legal requirement of *transparency* for citizens is fulfilled by publishing the structured descriptions of all (old and active) projects. After download, the citizen learns in which projects (s)he is involved by locally querying these descriptions with personal properties such as ZIP code, gender and salary. Simultaneously, no government agency has such an overview for all citizens.

The data archipelago can be run on one or multiple physical machines. Extra analytics tools can be foreseen in the data archipelago such that a researcher does not (always) need access to the full PII dataset in the project, which improves the security significantly. Monitoring and control mechanisms can be implemented to prevent or detect abuse by the researcher.

9 Conclusions & future work

9.1 Conclusions

European privacy legislation poses serious challenges to analytics projects on PII originating from different

legal entities like government agencies. This text presented a technical approach which is compatible with this legislation and which alleviates this seemingly unsolvable paradox by giving control back to the data controllers (the government agencies responsible for the PII) who have to check legal requirements such as proportionality and finality before providing PII for an analytics project. It increases security compared to traditional centralised approaches which, hence, indirectly improves the privacy of the citizens. And from a technical point of view, combining data can be done easily. Also the legal requirement of transparency towards the citizens can be met.

Since the scheme only runs during the data combining process, it does not result in a performance penalty during the actual analytics phase, when performance becomes crucial.

The presented approach is flexible. Deanonymisation requires, besides the project and the authority, the collaboration of zero, one or more chosen parties. Key generation and distribution can be done by a single authority, by multiple authorities or completely distributed. Furthermore, lower or higher degrees of virtualisation are possible and abstraction is made of the underlying physical infrastructure. Different islands can be hosted on different, geographically dispersed servers. Sending updates to the project is possible by rerunning the protocol but instead of sending all the data, only the changes are sent.

A proof-of-concept has been made in Java. The pseudonym conversion functions are simple scalar multiplications on an elliptic curve and are very efficient compared to older technologies such as RSA.

9.2 Future work

The first challenge is obtaining zero-knowledge for domains without extra party. Secondly, sometimes a new attribute for the project can only be generated by combining data from multiple sources. A project could for instance need data about all citizens of which $domain_0.wage + domain_1.benefits > constant$. Or a project might want financial data about households, which requires combining fiscal and family data that are managed by different providers. Other challenges involve key replacement (versioning), formal definition of the project queries and a better pseudonym conversion function.

References

[1] European Parliament and the Council of the European Union. Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, L 281:0031–0050, 1995.

[2] Mark Hosenball. U.s. has yet to notify 21.5 million data breach victims: officials. *Reuters*, 14 July 2015.

[3] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.

[4] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[5] Ninghui Li and Tiancheng Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07)*, 2007.

[6] Latanya Sweeney. Simple demographics often identify people uniquely. *Carnegie Mellon University, Data Privacy*, 2000.

[7] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3, March 2013.

[8] President's Council of Advisors on Science and Technology. Big data and privacy - a technological perpective. May 2014.

[9] Michael Barbaro and Tom Zeller, Jr. A face is exposed for AOL searcher no. 4417749. The New York Times, 9 August 2006.

[10] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[11] Paul Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review, Vol. 57, p. 1701, 2010*, 2009.

[12] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 21–30, New York, NY, USA, 2002. ACM.

[13] S. Brands. A technical overview of digital credentials, 1999.

[14] Dinusha Vatsalan, Peter Christen, and Vassilios S. Verykios. A taxonomy of privacy-preserving record linkage techniques. *Inf. Syst.*, 38(6):946–969, September 2013.

[15] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, pages 36–54, London, UK, UK, 2000. Springer-Verlag.

[16] Rafail Ostrovsky and William E. Skeith, III. A survey of single-database private information retrieval: Techniques and applications. In *Proceedings of the 10th International Conference on Practice and Theory in Public-key Cryptography*, PKC'07, pages 393–411, Berlin, Heidelberg, 2007. Springer-Verlag.

[17] D. A. Burgess. The distribution of quadratic residues and non-residues. *Mathematika*, 4:106–112, 12 1957.

[18] Daniel R. L. Brown. Sec 2: Recommended elliptic curve domain parameters. *Certicom Research*, 2010.

[19] Ronald Cramer and Robbert De Haan. Atomic secure multi-party multiplication with low communication. In *Proceedings of EUROCRYPT 2007*.