Constructive Privacy for Shared Genetic Data

Fatima-Zahra Boujdad, Mario Südholt

Constructive Privacy for Shared Genetic Data

Fatima-zahra Boujdad[1], Mario Sudholt[1]
[1]*IMT Atlantique, 4 rue Alfred Kastler, Nantes, France*
{*fatima-zahra.boujdad, mario.sudholt*}*@imt-atlantique.fr*

Keywords: genetic data, privacy, outsourcing, watermarking, fragmentation, encryption.

Abstract: The need for the sharing of genetic data, for instance, in genome-wide association studies is incessantly growing. In parallel, serious privacy concerns rise from a multi-party access to genetic information. Several techniques, such as encryption, have been proposed as solutions for the privacy-preserving sharing of genomes. However, existing programming means do not support guarantees for privacy properties and the performance optimization of genetic applications involving shared data. We propose two contributions in this context. First, we present new cloud-based architectures for cloud-based genetic applications that are motivated by the needs of geneticians. Second, we propose a model and implementation for the composition of watermarking with encryption, fragmentation, and client-side computations for the secure and privacy-preserving sharing of genetic data in the cloud.

1 INTRODUCTION

Information about the human genome has become highly valuable for development of new treatments of genetic-based diseases. Advanced sequencing technologies (NGS) (Behjati S, 2013) have made it much easier to obtain complete genetic data of human beings. However, in general genetic samples are not sufficiently available to genetic research. Indeed, genetic research has often be conducted collaboratively between several (groups of) geneticians. Doing so, more meaningful sizes of genetic cohorts can be established and allow for accurate results. For instance, Genome Wide Association Studies (GWAS) use two, if possible, large sets of genetic data (*e.g.,* in the form of positions from genomes (Cousin et al., 2006)), *case data* which belongs to subjects of the studied disease and *control data* that is obtained from healthy donors. Control data is particularly difficult to obtain because it has to be provided on a voluntary basis by healthy individuals. This is one case in which researchers want to share and work on the already available corresponding data.

Though sharing in this fashion seems to be straightforward, the very private aspect of genetic information (Erlich and Narayanan, 2014) pushes geneticians to protect their collected data and hence, hamper the practical and flexible sharing of genetic data. Instead of sharing raw data, genetic research often relied on public aggregated data, *e.g.,* allele frequency, in a false belief that this procedure was privacy-preserving. However, since the attack by Homer and *et al.* (Homer et al., 2008), the aggregated information has been retired from public access[1]. In fact, the attack shows how to infer a specific individual presence in a study based on those aggregated data only. This clearly threatens the privacy of patients. As a consequence, sharing of genetic data, notably via cloud-based services is very limited currently and performed using very restrictive queries on genetic databases.

Secure and privacy-preserving sharing of high volumes of genetic data constitute a very active research field nowadays. Data is shared using a client/server architecture where the server is often a cloud provider storing and processing, *e.g.,* homomorphically encrypted data (Lu et al., 2015; Zhang et al., 2015), which enables computations to be directly performed on encrypted data; another architecture consists in the collaboration between different biomedical sites which make use of multi-party computation protocols. However, the corresponding approaches, *e.g.,* (Liina Kamm et al., 2013; Tang et al., 2016), only handle a limited number of sites in the scenario and are yet to be extended so to handle realistic scenarios for a wider sharing of genetic data.

Some approaches have proposed the combination of different privacy-enhancing techniques, *e.g.,* homomorphic encryption, multi-party compu-

[1] http://help.gwascentral.org/data/download/

tation protocols to provide secure sharing of genetic data (McLaren et al., 2016). The combination of data fragmentation and client-side computations has been proposed in the context of an outsourcing schema (Wang et al., 2009). It locally stores identifying genome components, *i.e.,* SNPs, while computations on the common and publicly known parts of the human genome are outsourced. Combining different techniques for security purposes has also been explored for other related purposes. For instance, in (Ciriani et al., 2010), a combination of encryption and fragmentation is used to protect outsourced databases. For multimedia data, (Bouslimi et al., 2016) suggest a combination of encryption and watermarking to transmit images through an untrusted network securely.

A crucial result of the current situation is that the secure and privacy-related handling of shared data requires different technologies to be composed in order to handle realistic collaboration scenarios. *However, no such general compositional approach for the sharing of genetic data exists*. In particular, while the existing approaches focus on confidentiality properties, ownership and integrity properties have received few attention.

We build on ideas from the PRIVGEN project[2], in which researchers in genetics and computer science study new approaches for distributed analyses over shared genetic data. In this paper, we provide a compositional approach supporting ownership and integrity properties by extending the approach by (Cherrueau et al., 2015) to genetic data scenarios. Cherrueau *et al.* allow the composition of encryption, data fragmentation and localized computations to establish confidentiality of sensitive data. Concretely, we present two main contributions:

- We present and discuss several *new architectures and scenarios for the cloud-based sharing of genetic data*.
- We add *watermarking techniques* to the approach by (Cherrueau et al., 2015) in order to support ownership and integrity properties of shared genetic data. Concretely, we present a language-based approach for building applications and servers for the composition of privacy-preserving applications manipulating shared genetic data.[3] We also present an algebraic theory that allows for the optimization of such applications and servers.

The paper is structured as follows. Sec. 2 presents basic information about the sharing of genetic data and introduces new corresponding software architectures for this purpose. In Sec. 3 we present our approach COSHED for the compositional construction of systems for the sharing of genetic data. We close with a conclusion.

[2]Privacy-preserving sharing and processing of genetic data, http://www.privgen.cominlabs.u-bretagneloire.fr

[3]The Idris implementation of our approach is available at: https://github.com/BoujdadFz/PrivGen-Rep/blob/master/coshed.idr.

2 GENETIC DATA SHARING

Together with our partners of the PRIVGEN project, we are studying more flexible architectures that support wider sharing of genetic cohorts in the Cloud while preserving data privacy properties. Existing genetic data and computation servers are limited essentially to simple client-server systems that allow geneticians to perform highly restricted stateless queries. In contrast, the partners of PRIVGEN are working on more powerful collaborative data-sharing architectures. Concretely, the geneticians in the PRIVGEN project are interested in a scalable architecture that allows for sharing between multiple owners of genetic data (research or private institutions) and researchers in genetics.

2.1 Architectures

Based on the requirements of geneticians we are proposing new architectures that allow for genetic data sets provided by different organizations to be shared. Sharing is performed using trusted parties mediating data that is stored and (partially) manipulated in federated clouds. In the following, we propose two new such architectures. These architectures differ, in particular, in the policies for genetic data sharing (GDS) they allow for. Additionally, we motivate that watermarking techniques are a crucial means to satisfy ownership and integrity properties in this context.

2.1.1 Trusted party architecture

The first architecture we propose, shown in Fig. 1, enables data to be shared among different geneticians (G in the figure) and to be transferred to a cloud infrastructure. Communications are mediated by a trusted party (TP) that can enforce privacy and security properties of the data and computations shared among the other participants. Because of the strong control provided by the mediating trusted party, this architecture is particularly suitable to cooperation scenarios between partners having different GDS policies.

Since this kind of architecture includes data to be transferred from the geneticians (or corresponding

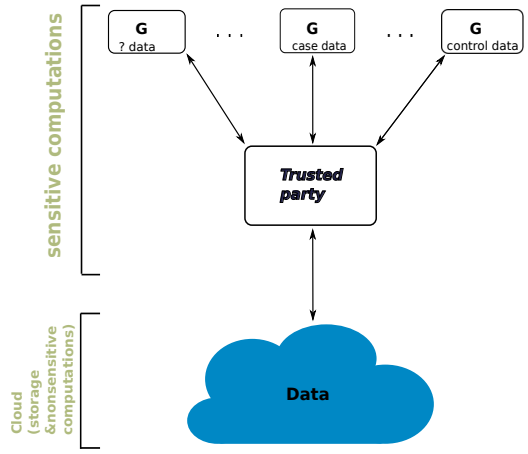

Figure 1: Genetic data sharing via a trusted party

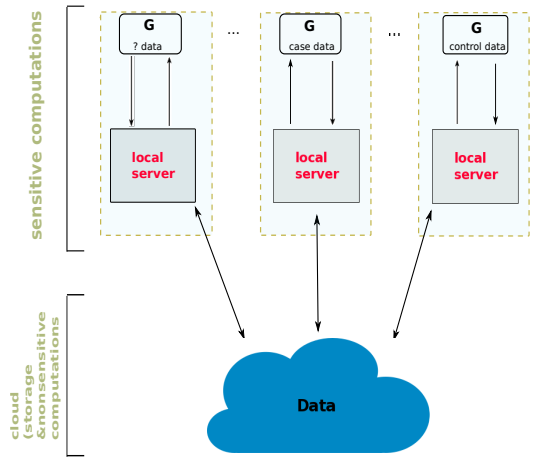

Figure 2: Genetic data sharing via local servers and cloud

owner institutions) to both the Cloud and the TP, ownership and integrity properties of shared genetic data are crucial in addition to more frequently-used confidentiality properties.

Trusted parties can be integrated into genetic applications in different ways: Kantarcioglu *et al.* (Kantarcioglu et al., 2008) employ a trusted entity for storing encrypted genetic data and processing (anonymous) computations on it; Xie *et al.* (Xie et al., 2014) uses a trusted component for key distribution and management. We rather allow for direct access to genetic data by the trusted party in our architecture, which is in fact conform to real-world applications (Gulcher et al., 2000).

This architecture can be generalized to allow for sharing as part of federated Clouds. In Sec. 3.1.1.1 we harness such an architecture in the context of a concrete sharing scenario.

2.1.2 Local computations architecture

Fig. 2 shows a model for computations performed by geneticians that all have their own infrastructure able to handle computations on genetic datasets of bearable sizes. In this case, sharing of data is done through the Cloud. A typical workflow may define a dataset, encrypt it for transfer in the Cloud where it may undergo some privacy-neutral processing, before finally being transmitted to its final recipient. This workflow requires fragmentation techniques, cryptographic techniques and client-side computations. In this architecture as well ownership properties have to be satisfied which can be assured by watermarking.

This architecture and corresponding workflows require closer integration between the collaborators, notably their GDS policies.

3 PRIVACY FOR SHARED GENETIC DATA

The architectures manipulating shared genetic data introduced in the previous section may easily lead to violations, for instance, of privacy properties. Such violations may stem from simple programming errors or more difficult errors in the applications' logic. Our approach is based on the C2QL approach by (Cherrueau et al., 2015) who provide composition and query languages for the secure and privacy-preserving programming of distributed applications. In the following, we first review the basic mechanisms of their approach before detailing ours.

C2QL enables the development of privacy-aware applications by the implementation of distributed algorithms composing computations involving encrypted and fragmented data, as well as client-side computations. SQL-like queries are supported on top of possibly encrypted and fragmented data. The approach essentially consists of two parts: language support for the development of secure and privacy-preserving applications and an algebraic theory supporting corresponding optimizations and correctness proofs.

Language support The C2QL language provides de/constructors for the encryption and the fragmentation of data, respectively denoted `Decrypt`, `Crypt`, `Defrag` and `Frag`. Client-side computations are initiated automatically depending on the fragmentation and encryption status of data. The constructors are used as part of a domain specific language that is embedded in the *Idris* (`www.idris-lang.org`) programming language. Idris programs are used to compose

secure and privacy-aware applications as well as the queries over genomic data. Idris' dependent type system enables the verification of basic secure properties at compile time, *e.g.,* that a given encrypted data fragment can only be used after decryption.

```
-- Basic value types
data Ty   =  BOOL | NAT | TEXT
           | CRYPT CryptTy Ty

-- Attribute: (column name, value type)

Attribute : Type
Attribute = (String, Ty)

-- Table schema: list of attributes

Schema : Type
Schema = List Attribute

-- DB environment: vector of table schemas

Env : Nat → Type
Env n = Vect (S n) Schema
```

Figure 3: Basic data types

Figure 3 shows the basic types implementing data bases. Figure 4 shows the abstract definitions of

```
-- ADT (algebraic data type) for privacy operators
data Privy : (env0 : Env n) → (env1 : Env m) →
             (Δ : Schema) → Type where

-- encrypts an attribute in an environment
Crypt : (a : Attribute) → (c : CryptTy) →
        {auto p : EnvElem a env} →
        Privy env (cryptEnv c a env) []

-- fragments an envir. at the 'most right' schema
-- @ p proof that `δ` ⊆ `(last env)`
Frag : (δ: Schema) → {auto p: Inc δ (last env)} →
        Privy env (fragEnv δ env) []

-- ADT for data recovery operators
data Query : (Δ : Schema) → Type where

  -- decrypts values of `a` using information `d`
  -- @ p1 proof that values of `a` are encrypted
  -- with schema `c`.
  Decrypt :
     (a : Attribute) → (d : Decrypt c) →
     {default Refl p1 : (CRYPT c t) = (snd a)} →
     {auto p2 : Elem a Δ} →
     Query Δ → Query (replaceOn a (fst a, t) Δ)

  -- defragments values of `q1` and `q2`
  Defrag : (q1 : Query δ) → (q2 : Query δ') →
           Query (delete Id (nub (δ ++ δ')))
```

Figure 4: Privacy-enforcing operators and queries

privacy-enforcing operators (type `Privy`) and the encryption/fragmentation constructors as well as queries (`Query`) that are constructed by, if necessary, decrypting and defragmenting shared data.

Algebraic theory The compositions of privacy-enforcing operators and query operations are linked by numerous algebraic laws that express, for instance, the commutativity of certain compositions of operations. Based on these laws, applications over shared data can be transformed. Such transformations are useful, for instance, in order to distribute data by fragmentation (and thus helping privacy through de-identification of data) or optimize application performance (in particular, by harnessing the cloud to handle most of the benign computations).

Henceforth, we denote a set of attributes (columns) in a given relational database as a and use $\circ$ as the symbol for composition. SQL-like projection is denoted by π_a while selection is σ_p where p is the selection predicate. The algebraic operators used are:

- $crypt_{(s,a)}$, $decrypt_{(s,a)}$: encryption and decryption operators parameterized by a schema s (*e.g.,* AES) and the target attribute a to be encrypted in the relational database.
- $frag_{\pi_a}$, $defrag_{\pi_a}$: column-oriented fragmentation and defragmentation operators

Figure 5 shows some examples of the algebraic laws that are used later.

$$\pi_{a\bar{a}} \circ defrag_{\pi_a} \equiv defrag_{\pi_a} \circ (\pi_a, \pi_{\bar{a}}) \quad (1)$$

$$\sigma_{pa \wedge p\bar{a}} \circ defrag_{\pi_a} \equiv defrag_{\pi_a} \circ (\sigma_{pa}, \sigma_{p\bar{a}}) \quad (2)$$

$$\pi_a \circ decrypt_{(s,a)} \equiv decrypt_{(s,a)} \circ \pi_a \quad (3)$$

$$if\, dom(p) \notin \wp(a)$$

$$\sigma_p \circ decrypt_{(s,a)} \equiv decrypt_{(s,a)} \circ \sigma_p \quad (4)$$

Figure 5: commutation laws

3.1 The COSHED approach

Our approach for a COnstructive SHaring of gEnetic Data (COSHED) extends the C2QL approach with watermarking functions. In fact, genetic data watermarking is a promising technique for integrity, traceability or ownership protection. Concretely, we have added the watermarking scheme in (Iftikhar et al., 2015) that supports ownership and integrity protection properties for digital genetic data. In the following, we present operators for watermarking and the detection of watermarks that can be composed with the ones for encryption and fragmentation. We also introduce the corresponding algebraic laws that govern the relationship between the different privacy-enforcing techniques.

Language support. Watermarks are represented as a type `WATERMARK` whose first argument represents the watermarking scheme (for now we have implemented only one scheme `GIG` (Iftikhar et al., 2015)). We have implemented operators wat_a and $detectw_{(a,secrets)}$,

```
1  -- @GIG stands for GenInfoGuard watermarking scheme
2   data WmTy = GIG
3   data Ty   = ...  | WATERMARK WmTy Ty
4
5  -- a function that watermarks an attribute
6  -- @ p is a proof that @ a is in 'env'
7   watEnv : (a: Attribute) → (wms: WmTy) → (env: Env n)
8            → {auto p : So (isInEnv a env)} → Env n
9   watEnv  a wms env  =
10          map (\s => if (elem a s) then replaceOn a
11                     (fst a,WATERMARK wms (snd a)) s
12                     else s) env
13
14 -- ADT with information for watermark detection
15  data ReadM : WmTy → Type where
16   RGIG : (k:Key) → ReadM GIG
17
18 -- watermark application operator
19  data Privy :
20                ...
21    Wat : (a: Attribute) →
22          {auto p1 : So (isRawType (snd a))} →
23          {auto p2 : So (isInEnv a env)}  →
24          Privy env (watEnv a GIG env) []
25
26 -- watermark detection operator
27  data Query :
28             ...
29    detectw :
30        (a : Attribute) → (info : ReadM GIG) →
31        {default Refl p1: (snd a) = (WATERMARK GIG t)}
32        → Query Δ → {auto p2 : Elem a Δ} →
33        Query ((replaceOn a (fst a, t) Δ)++[MyTattoo])
```

Figure 6: Watermarking operators

see Fig. 6, respectively for watermark application (as part of the privacy-enhancing technologies of ADT `Privy`) and watermark detection as used in queries (ADT `Query`). The wat_a operator has as parameters the attribute that indicates which columns have to be watermarked in addition to two implicit arguments for compile-time verification. More precisely, p1 serves as a (pre-defined) proof that the passed attribute is not watermarked nor encrypted; p2 ensures its membership in the targeted environment. Similarly, in the query operator $detectw_{(a,secrets)}$, p1 is a proof that the parameter attribute was previously watermarked with the right schema, *i.e.,* GIG, so that watermark detection makes sense.

Laws. The watermarking schema `GIG` is reversible, we can hence define the identity law

$$id \equiv detectw_a \circ wat_a \tag{5}$$

Unlike for the encryption/decryption operators, the watermarking schema is not a parameter because the entire schema has already been defined (Iftikhar et al., 2015). Actually, most watermarking laws will be specific to the watermarking schema, contrary to encryption which essentially is a more general operation.

The second law stipulates that watermark detection can be delayed to after decryption provided that the watermark application took place before encryption.

$$\begin{aligned} decrypt_{(s,a)} \circ crypt_{(s,a)} \circ detectw_a \circ wat_a \equiv \\ detectw_a \circ decrypt_{(s,a)} \circ crypt_{(s,a)} \circ wat_a \end{aligned} \tag{6}$$

Furthermore, watermark detection commutes with projection:

$$\pi_a \circ detectw_a \equiv detectw_a \circ \pi_a \tag{7}$$

Provided that selection is not performed on watermarked attributes, watermark detection commutes with selection: $if\, dom(p) \cap a = \emptyset$

$$detectw_a \circ \sigma_p = \sigma_p \circ detectw_a \tag{8}$$

3.1.1 Cloud-based association studies

We are now ready to present our definition of the advanced architectures for sharing of genetic data. Considering a scenario for large genomic-wide association studies (GWAS), a federated cloud architecture as proposed in Sec. 2.1 is considered that satisfies the characteristics required by geneticians from the PRIVGEN project:

- The public cloud provider should not be able to get direct access to identifying data
- Geneticians/researchers should not be able to get direct access to external identifying data, *e.g.,* genomes.
- Ownership and integrity properties of the data have to be satisfied.

Explanation of how these requirements are satisfied in our architecture is given later in this section.

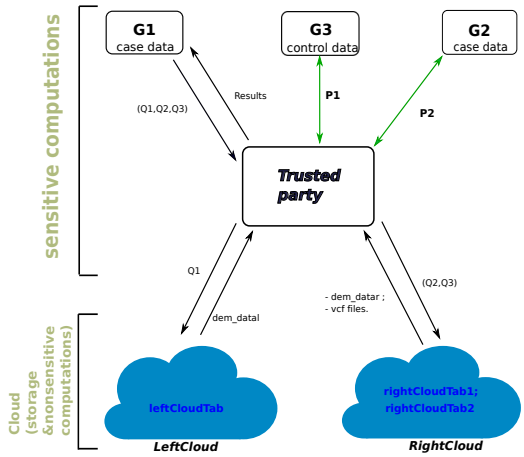

Figure 7: Genetic data sharing scenario via a trusted party

3.1.1.1 Scenario architecture. We illustrate an implementation of a variant of the sharing through the trusted-party architecture presented in Sec. 2.1, variant shown in Fig. 7. This architecture variant is used as part of a scenario for sharing genetic cohorts in GWAS. In this scenario, genetic data is of two sorts: *case data* represents the set of vcf files containing variants belonging to patients holding the studied disease, while *control data* concerns healthy participants. As is often the case in genetics, *control data* are not in sufficient supply in research centers: data sharing with external researchers that possess interesting control data is therefore of high interest.

We present participants with their zip code, day of birth (DoB), gender, information of whether the subject is contributing `case` or `control` data and their corresponding vcf file. Had the database been local, the representation would be in two tables as follows:

```
Subject (SubjectId,ZIP,DoB,Gender,CaseCtrl)

SubjectVcf (recordId,Variant,TypeVariant,
            position,SubjectId)
```

For privacy-preserving outsourcing in the cloud that fits the aforementioned requirements defined by geneticians, four security techniques are applied in the scenario:

- *Fragmentation* for confidentiality: the triplet *(zip code,gender,DoB)* forms a quasi-identifier (Sweenay, 2000): fragmentation is therefore used to forbid any re-identification attempts. Therefore, we store the pair (*zip code*, *gender*) and *DoB* in different fragments in different non-communicating Clouds (denoted as LeftCloud and RightCloud in Fig. 7);
- *Encryption* for confidentiality: used for any data that can not be fragmented nor it can be kept plain at cloud level; in our scenario case, the vcf file is symmetrically encrypted (field-wise) except for the *position* field;
- *Watermarking* for ownership and integrity protection of genomes. In fact, genomes will be accessed in clear format by the trusted party and any unintended disclosure of data can also threaten the ownership of data and its integrity.
- *Client-side computations* are used for TP computations. This means that the geneticians need to share (only) their decryption keys with the TP which satisfies the first requirement of geneticians as any access to identifying data is performed through the trusted party.

Querying a database is more efficient when data is plain. Therefore, increasing performance can be achieved by decreasing the application of costly security techniques. One essential idea behind the composition of security and privacy-enhancing techniques is foremost using any security method that will keep data in plain format and does not threaten privacy. A typical corresponding application case consists in information whose sensitivity results from it being associated to other data. The triplet *(zip code,gender,DoB)* is an example.

As part of the scenario, G1, G2 are genetic research sites that initially each possess *case data* files; G3 holds *control data*. We first assume that every researcher and genetic center already outsourced their data to a cloud provider as introduced above. In the scenario, G1 wishes to process an association study over a disease X. For this purpose, G3's control sets are needed. The scenario then proceeds as follows:

1. G1 starts by requesting TP to perform an association study over indicated variants `(Q1,Q2,Q3)`.
2. TP asks G3 for authorization (A1) to use its control data in the cloud for G1's research.
3. Assuming G3 provided its authorization, TP can apply distributed queries to the corresponding cloud databases and thus get the necessary data (demographic data and parts of vcf files).
4. After computations are done, TP communicates the inferred results to G1 (in a secure manner *e.g.,* an SSL connection).

The architecture can also accommodate special cases of sharing. For instance, it may happen that G1's datasets are not sufficient for some study: TP then can search complementary data from other geneticians which obviously requires their authorization (A2).

To prepare for the implementation of the scenario, we need to write correct queries, a process which is performed using the laws about privacy-enhancing compositions. We consider two queries, one for retrieving demographic data and another query to recuperate genetic data. The new tables of the new distributed database after applying the aforementioned techniques become (cf. Fig. 7):

```
leftCloudTab   (SubjectId,ZIP,Gender)

rightCloudTab1 (recordId,VariantWE,
                TypeVarE,position,
                SubjectId,)

rightCloudTab2 (SubjectId,DoB,CaseCtrl)
```

As the genetic application represented in the scenario is an (abstract) genetic association computation performed by the trusted party, both demographic and genetic data should be provided by the

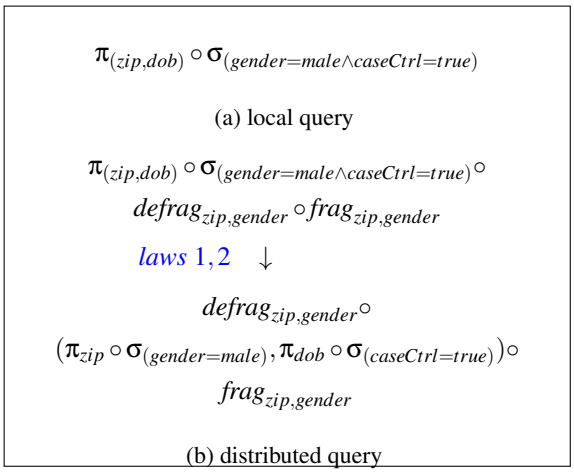

Figure 8: Query for demographic case data recovery

cloud providers. The first query over the distributed `Subject` table is meant to return the *zip code* and the *DoB* of male subjects holding the disease. In order to retrieve this data from the distributed environment the query should be split. Fig. 8a shows a suitable local version of the query. In a distributed setting, we use commutation laws to obtain a distributed query (Fig. 8b) from a local one. As for the genetic data recovery query, the goal is to retrieve some specific *position*s in the genomes of the previously selected males demographic data (which are the results of the first query, referred to as *mdd* in the figure). Therefore, the right *Variant* and *TypeVariant* elements are returned. Similarly, this second query is transformed from a local formula (Fig. 9a) to a distributed one (Fig. 9b) using the identity and commutation laws of encryption and watermarking described earlier in Secs. 3 and 3.1, respectively. The last distributed query obtained shows, after law-driven transformations, that to have access to both columns, a decryption step and a watermark detection operation should be executed over the data, a step that 'deconstructs' the previous encryption and watermarking introduction that were necessary for a secure and privacy-preserving outsourcing process.

3.1.1.2 Scenario implementation.

The scenario implementation is given in Fig. 10: it initially sends query requests from the genetician `G1` to the trusted party TP that forwards them to the two clouds where they are executed. The resulting data is sent back to TP where the genetic computation is performed. The corresponding results are then communicated to G1.

Fig. 11 shows the entity definitions for geneticians, trusted parties and clouds, a query and the main part of the ADT for genetic queries.

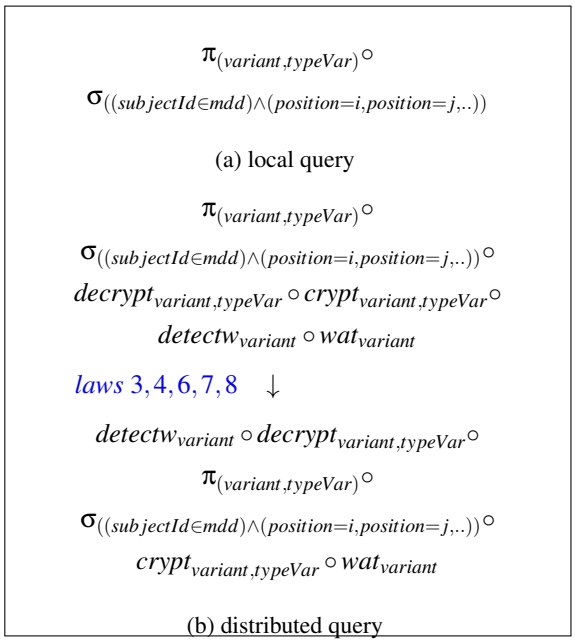

Figure 9: Query for genetic case data recovery

```
1  scenario : GeneticQuery [SubjectId,ZIP,Gender,DoB,
2                           Variant,TypeVar,MyTattoo]
3  scenario =  do
4
5   G1  `SendRequest` (TP,[Q1])
6   G1  `SendRequest` (TP,[Q2,Q2'])
7   G1  `SendRequest` (TP,[Q3,Q3'])
8
9   TP  `SendRequest` (LeftCloud,[Q1])
10  TP  `SendRequest` (RightCloud,[Q2,Q2'])
11  TP  `SendRequest` (RightCloud,[Q3,Q3'])
12
13  let q1 = LeftCloud  `executeRequest` [Q1];
14  let q2 = RightCloud `executeRequest` [Q2,Q2'];
15  let q3 = RightCloud `executeRequest` [Q3,Q3'];
16
17  demDatal       ← LeftCloud  `SendData` (TP,q1)
18  demDatar       ← RightCloud `SendData` (TP,q2)
19  vcfFiles       ← RightCloud `SendData` (TP,q3)
20
21  let r1 = decrypt VariantWE (AESD "key2") vcfFiles;
22  let r2 = decrypt TypeVarE (AESD "key1") r1;
23  let vcfFiles = detectw VariantW (RGIG "wkey1") r2;
24  let Data = defrag (defrag demDatal demDatar) vcfFiles
25
26  TP `ReturnResults` (G1, TP `Compute` Data)
```

Figure 10: Scenario implementation

Ensuring privacy properties. Our approach, being based on Idris, allows for proofs of certain safety properties (that entail security and privacy properties) to be passed as arguments to operators that are part of correctly built queries. For instance, if we try to watermark a genetic data that is already encrypted, type checking will not pass because of the proof p1 of the *wat* operator that verifies data has not been transformed yet. Similarly, trying to detect a watermark from data that has not been decrypted yet will give rise to a type checking error, see Fig. 12.

```
-- Entities
data Entity = Genetician | TrustedP | Cloud

G1, TP, LeftCloud, RightCloud : Entity
G1          = Genetician; TP          = TrustedP
LeftCloud   = Cloud;      RightCloud = Cloud

leftCloudTab,rightCloudTab1,rightCloudTab2 : Schema
leftCloudTab   = index 1 SafeTPEnv ...

-- Queries
Q1 : Query [SubjectId,ZIP,Gender]
Q1 = π [SubjectId,ZIP,Gender]
   $ σ (Gender == "male") (toQuery leftCloudTab);

-- ADT which parameter indicates the expected results
-- of an exchange or computation involving genetic data

data GeneticQuery : Schema → Type where

 SendRequest :
   Entity → (Entity,List (Query Δ)) → GeneticQuery Δ
 SendData : ...
 Compute  : ...
```

Figure 11: Entities, queries and ADT for scenario building

```
... let r1  =
       detectw VariantWE (AESD "key2") vcfFiles -- error
    let r2  = decrypt TypeVarE (AESD "key1") r1; ...
```

Figure 12: Ill typed query

4 CONCLUSION

In this paper we have pointed to the lack of programming support for privacy-preserving applications that manipulate shared genetic data. We have presented two contributions: (i) new cloud-based architectures for such applications that are motivated by concrete requirements from researchers in genetics and (ii) a model and corresponding security- and privacy-enhancing techniques for the development of such applications, notably using watermarking for the preservation of ownership and integrity properties.

As future work, we are striving for the integration of other privacy-enhancing techniques, an efficient implementation of a general Java library for biomedical analyses using shared genetic data, and its application to real-world genetic analyses.

Acknowledgements. We thank our partners from the PRIVGEN project[2], in particular, D. Niyitegeka, G. Coatrieux and E. Genin, for valuable discussions on watermarking and genetic data sharing.

REFERENCES

Behjati S, T. P. (2013). What is next generation sequencing? *Archives of Disease in Childhood Education and Practice Edition*.

Bouslimi, D., Coatrieux, G., et al. (2016). Data hiding in encrypted images based on predefined watermark embedding before encryption process. *Signal Processing: Image Communication*, 47.

Cherrueau, R.-A., Douence, R., and Südholt, M. (2015). A Language for the Composition of Privacy-Enforcement Techniques. In *IEEE RATSP*, pages 1037 – 1044, Helsinki, Finland.

Ciriani, V., Vimercati, S. D. C. D., Foresti, S., et al. (2010). Combining fragmentation and encryption to protect privacy in data storage. *ACM Trans. Inf. Syst. Secur.*, 13(3):22:1–22:33.

Cousin, E., Deleuze, J.-F., and Génin, E. (2006). Selection of SNP subsets for association studies in candidate genes: .... *BMC Genetics*, 7:20.

Erlich, Y. and Narayanan, A. (2014). Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, pages 409–421.

Gulcher, J. R., Kristjansson, K., Gudbjartsson, H., and Stefnsson, K. (2000). Protection of privacy by third-party encryption in genetic research in iceland. *European Journal Of Human Genetics*.

Homer, N., Szelinger, S., Redman, M., et al. (2008). Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLOS Genetics*, 4(8):1–9.

Iftikhar, S., Khan, S., Anwar, Z., et al. (2015). Geninfo-guarda robust and distortion-free watermarking technique for genetic data. *PLOS ONE*, 10(2):1–22.

Kantarcioglu, M., Jiang, W., Liu, Y., and Malin, B. (2008). A cryptographic approach to securely share and query genomic sequences. *Trans. Info. Tech. Biomed.*, 12(5):606–617.

Liina Kamm, Dan Bogdanov, S. L. et al. (2013). A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics (Oxford, England)*.

Lu, W.-J., Yamada, Y., and Sakuma, J. (2015). Privacy-preserving genome-wide association studies on clouds using fully homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(5):S1.

McLaren, P. J., Raisaro, J. L., Aouri, M., et al. (2016). Privacy-preserving genomic testing in the clinic: a model using hiv treatment. *Genetics in Medicine*, 18(8):814–822.

Sweenay, L. (2000). Simple demographics often identify people uniquely. Carnegie Mellon. Data Privacy Working Paper 3.

Tang, H., Jiang, X., Wang, X., et al. (2016). Protecting genomic data analytics in the cloud: state of the art and opportunities. *BMC Medical Genomics*, 9(1):63.

Wang, R., Wang, X., Li, Z., Tang, H., et al. (2009). Privacy-preserving genomic computation through program specialization. In *ACM CCS*, pages 338–347, New York, NY, USA. ACM.

Xie, W., Kantarcioglu, M., Bush, W. S., et al. (2014). Securema: protecting participant privacy in genetic association meta-analysis. *Bioinformatics*, 30(23):3334.

Zhang, Y., Dai, W., Jiang, X., et al. (2015). Foresee: Fully outsourced secure genome study based on homomorphic encryption. *"BMC Medical Informatics and Decision Making*, 15(5):S5.