Privacy-Preserving Data Set Union

Alberto Maria Segre
alberto-segre@uiowa.edu
Department of Computer Science
The University of Iowa
Iowa City, IA 52242

Andrew Wildenberg
awildenberg@cornellcollege.edu
Department of Computer Science
Cornell College
Mount Vernon, IA 52314

Veronica Vieland
vielandv@pediatrics.ohio-state.edu
Columbus Children's Research Institute
Ohio State University
Columbus, OH 43205

Ying Zhang
yizha@math.uiowa.edu
Applied Mathematical and Computational Sciences
The University of Iowa
Iowa City, IA 52242

Abstract

This paper describes a cryptographic protocol for merging two data sets based on identifiers without divulging those identifier records; technically, the protocol computes a *blind set-theoretic union*. Applications for this protocol arise, for example, in data analysis for biomedical application areas, where identifying fields (*e.g.*, patient names) are protected by governmental privacy regulations or by institutional research board policies.

Keywords: privacy-preserving data mining, blind database union.

1. Introduction

The analysis of data collected from multiple sources presents a variety of special challenges. Since more data generally yield better and/or more meaningful results, it is often desirable to apply statistical analyses to the union of multiple data sets. Such is the case, for example, when clinical data on a particular disease and its treatment are collected independently in the context of multiple research studies.

When privacy regulations, such as HIPAA [3], prohibit the sharing of identifying patient information, it is no longer possible to openly calculate the union of the data sets. Since the replicated records cannot be identified, the usual procedure is to simply concatenate the data sets to produce an approximation of their set-theoretic union. If there are no records present in more than one data set, then the approximation is exact; but if any records are shared, the result is usually a violation of the assumptions underlying whatever statistical analysis is to be performed.

From:*Proc. of the 2006 Conference on Privacy in Statistical Databases*

Our motivating application is to apply genetic linkage analysis tools to genotypic data collected from multiple clinical sites. *Genetic linkage analysis* describes a set of statistical genetics techniques used to locate on the genome the genes responsible for an inherited trait or disease (see, for example, [11]). A characteristic of such studies is unsystematic recruitment of families with multiple members affected with the disease under investigation; and as a result, it is not uncommon for motivated families to enroll themselves in multiple studies, that is, to be represented redundantly in data sets collected at different sites. When the data are then combined for analysis across sites, the presence of duplicated individuals appearing in more than one subset of the data violates a key assumption of the statistical analysis — that each family be considered only once — and can distort the results, even if there are only a small number of duplicates. This can be particularly problematic in studying rare genetic disorders, for which data sets tend to be small.

What is needed is a mechanism that identifies the overlapping records so that they won't be used twice. The difficulty is that identifying overlapping records entails comparing identifying information, which, by law, cannot be shared. In this paper, we present cryptographic protocols that allow two parties to negotiate the true set-theoretic union of their privately-held data sets without revealing restricted identifying information.

2. Background and Related Work

Let a *data set* consist of multiple *records*, where each record corresponds to a real-world entity we wish to model. Each record consists of a number of *attributes* or *fields*, where each attribute's *value* consists of a single typed datum; for example, ''birthdate,'' ''social security number,'' or ''diagnosis'' might be attributes in a patient information database, and ''04/01/1947,'' ''484-11-1991,'' and ''asthma'' might be the respective values of these attributes for a given record, which in turn represents a real-world person. Assume that Alice and Bob each possess data sets, A and B, with identical format, consisting of $|A|$ and $|B|$ records, respectively. Assume also that the attributes divide into two disjoint attribute subsets,

denoted I and D, where I consists of the protected ‘‘identifying’’ attributes and D consists of the unprotected ‘‘data’’ attributes. In the terminology of relational databases, we expect that no combination of ‘‘data’’ attributes could serve as a candidate key, that is, could uniquely determine a real-world entity modeled by a data set record. But more to the point, the distinction between which attributes are protected ‘‘identifying’’ attributes and which are unprotected ‘‘data’’ attributes is really a matter of policy. These are fixed by the regulatory context (*e.g.*, HIPAA) in which the parties are operating.

More formally, Alice owns data set $A = \{I_a, D_a\}_i$ for $0 \leq i < |A|$, Bob owns data set $B = \{I_b, D_b\}_j$ for $0 \leq j < |B|$, and we wish to compute $D_{ab} = \Pi_D(A \bigcup_I B)$, where $\bigcup_I$ is the set-theoretic union operation that considers two records equal if and only if their identifiers are equal, and Π is a multiset relational algebra project operation (here, Π_D is used to filter out identifying fields, leaving only the data fields from each record in the resulting union). Note that we fully expect that the data fields, at least, will be subject to noise or measurement error.

An obvious and convenient solution to this problem is to take advantage of a mutually trusted third party, Ugo. Since Alice and Bob both trust Ugo, it is a simple matter for them to send him A and B, so that he can compute $A \bigcup_I B$. In fact, the existence of a trusted third party is not unusual in modern multisite clinical studies, where the data analysis site may be disparate from the data collection sites. In these situations, the data analysis site can be assigned to act as the trusted third party from the outset. Indeed, using a trusted third party should be the preferred solution, by virtue of its simplicity, in situations where it is permissible to do so. Unfortunately, the more normal situation is that no *a priori* agreement to share data exists, and so legal and policy issues may actually preclude selecting and using any sort of trusted third party *a posteriori*. Thus, in this paper, we consider the more interesting question of whether the union can be negotiated after the fact, without violating privacy concerns, and without resorting to a trusted third party.

There is a significant amount of related work in this area, which generally goes by the name *privacy-preserving data mining*. This work divides roughly into two categories; approaches that rely on sharing perturbed versions of each participant's data, and those that rely on cryptographic techniques. Here, we briefly summarize these approaches and explain why they are inadequate for our problem.

Distortion approaches assume that participants can disclose appropriately perturbed versions of their data without violating whatever privacy restrictions are in force. The core idea is that appropriately randomized records will be hard to identify: the trick is to ensure that the computation on the collected distorted data yields the same outcome — or at least close enough to the same outcome — as the computation when applied to the set-theoretic union of the original data [6, 7]. There are two shortcomings with this approach that make it unsuitable for our problem. First, successful randomization strategies depend on intimate knowledge of the analysis algorithms which will be applied to the distorted data. In practice, these algorithms have typically been statistical algorithms such as sum, average, and max/min [1], or, *e.g.*, decision tree [13] construction algorithms [2], neither of which approach the complexity of genetic linkage-analysis algorithms [4, 9]. Second, and more important, distortion strategies work by camouflaging the individual and reasoning about statistical distributions: no attempt is made to remove duplicated data points or account for any bias such replicated data points —distorted or not —may introduce.

In contrast, cryptographic techniques for the privacy-preserving database union have their basis in work introduced over 20 years ago on *secure two-party computation* [14]; Yao's work was later extended to multiple parties [5], hence the more usual name *secure multiparty computation*, or SMC. The idea is to devise a protocol that governs how two or more parties can exchange information in order to allow one (or both) of the parties to compute a function based on inputs held privately by each without revealing their own inputs. The general solution is based on circuit evaluation: the specified computational problem is first represented as a combinatorial circuit, then the participating parties run a short protocol for every

gate in the circuit. The protocols generated depend on the size of the circuit, which in turn depends on the size of the input domain and on the complexity of the specified computation when expressed as a combinatorial circuit. In general, the solution to SMC is too inefficient to be useful in real applications, with the notable exception of some very simple cases, such as comparing the magnitude of two integers, where the computation remains tractable.

To get around the prohibitive costs associated with SMC, one approach is to combine SMC calculations with an inherently less secure but more cost-effective wrapper. This approach trades off divulging more information than the full SMC implementation for more efficiency; a good example is the ID3 decision tree construction protocol of [10]. By divulging intermediate results as they are computed, participants can construct the decision tree in tandem, at much lower cost by repeatedly using an SMC primitive to compute the information gain at each node, rather than trying to use SMC to compute the entire tree in one calculation. This approach works particularly well for ID3 tree construction, since the intermediate results divulged are explicitly part of the final solution anyway. Nonetheless, the protocol is cumbersome, and is not readily generalized to other types of data mining algorithms, and, once again, since the data mining algorithms employed are not as sensitive to replicated records as our linkage analysis algorithms, most solutions simply ignore the problem of replicated records and opt to work on the concatenation of data sets rather than their set-theoretic union.

One notable exception is the work of Kantarcioglu and Clifton on distributed data mining of association rules [8]. Since their work addresses a problem very similar to ours, it is not surprising that they explicitly consider the removal of duplicate records. Kantarcioglu and Clifton rely on a *commutative cipher* to blind records as they are exchanged among three or more participants in order compute the true set-theoretic union.[1] Ultimately, however, their solution still falls short in two ways.

[1] A commutative cipher is an encryption algorithm having the property that a message encoded twice using two different keys must also be decoded twice, once with each key, but with decoding operations occurring in either order. Several, but not all, encryption algorithms in common use have this property (such as, for example, the RSA

First, at the end of the protocol, all records are revealed to the participants; the protocol is designed only to protect the identity of the original owner of each record, rather than the identifying attributes of the individual records (this also explains why their protocol is not suitable for use by only two parties: clearly, any record Alice does not recognize must have belonged to Bob). We might try to get around this problem by using, *e.g.*, a *one-way hash function*[2] to obscure the identifying attributes prior to the union operation. But this doesn't really work as intended, because once the identifying attributes' one-way hash postimages are revealed, it is an easy matter for one participant to mount a *dictionary attack* on the other participant's data sets (*i.e.*, to ''generate and test'' candidate records to see if they are present). A second problem is that their protocol requires that matching records match completely, and does not account for noise or measurement error in at least some of the fields.

This paper presents a general solution to the two-party privacy-preserving data set union problem. Our protocol allows Alice to compute $D_{ab} = \Pi_D(A \bigcup_I B)$ with help from Bob, while Bob only learns $|A|$ and $|U_{ab}|$, the size of the resulting union $U_{ab} = A \bigcup_I B$. Neither Alice nor Bob learn which of their own records are replicated in the other participant's data set, nor are their own records' identifying attributes ever disclosed to the other party. Our protocol is robust to noise and measurement error in data fields, and, in cases where the ''data'' attributes of replicated records (*i.e.*, records in multiple data sets having identical ''identifying'' attributes) diverge, our protocol ensures that none of the divergent data values ever ''leak'' to other participants. The same protocol is easily extended to the multiparty case.

---

public-key encryption algorithm). For technical reasons having to do, at least in part, with RSA's vulnerability to chosen plaintext attacks, Kantarcioglu and Clifton rely on Pohlig-Hellman encryption [12], but the choice of cipher is not important to understanding their protocol, or, for that matter, the protocol introduced in this paper.

[2] A *one-way function* is a mathematical function that is easy to calculate in one direction (*i.e.*, calculating the output from the input) but infeasible to calculate in the other (*i.e.*, calculating the input from the output). Cryptographic systems make extensive use of a special kind of one-way function, called a *one-way hash function*, where a *hash function* maps its input, or *preimage* (which may be of arbitrary length) onto an output, or *postimage* (which is of fixed length). A hash function is also one-way if it is computationally hard to derive a preimage from the corresponding postimage.

3. A Two-Party Privacy-Preserving Database Union Protocol

Our two-party protocol relies on several underlying assumptions. First, we assume that communication between parties is secure; that is, that no one but the originator and intended recipient can read any message sent as part of the protocol. This assumption is easily met, either by traditional means (*e.g.*, a trusted courier service) or by electronic/cryptographic means (*e.g.*, secure sockets, Open SSH, etc.). Second, we assume that the parties have been properly authenticated, that is, no one can pass themselves off as somebody else. This second assumption is typically met in a network environment by using a password-based authentication mechanism, and in a traditional environment by using a token-based authentication mechanism (*e.g.*, having the courier check the recipient's ID card). Third, we assume that all participants are basically honest and cooperative, yet still curious, an assumption commonly referred to as the *semihonest model*. Thus all participants will follow the rules of the protocol, but are still expected to try to discover what we are intent on keeping secret.

Like Kantarcioglu and Clifton, our protocol uses a commutative cipher, but, in addition, it uses a keyed commutative one-way hash function. We'll let $K_a(M)$ denote encryption of message M with Alice's key K_a, and let $\bar{K}_a$ represent the decryption operator (so that $\bar{K}_a K_a(M) = M$). Further, let $H_a(M)$ denote the postimage of Alice's keyed commutative one-way hash function H_a applied to message M. Similarly defining K_b and H_b for Bob's cipher and hash, respectively, we can describe our protocol as follows.

(1) Alice provides Bob with copies of her records, where the identifier is hashed using Alice's one-way keyed hash function and a random hash key H_a of her own choosing, and the data fields are encrypted using a commutative cipher with a random encryption key K_a of Alice's own choosing.

Alice $\rightarrow$ *Bob*: $\{H_a(I_a), K_a(D_a)\}_i \quad 0 \le i < |A|$

The hash and cipher keys H_a and K_a are Alice's secrets, and should never be revealed.

(2) Bob retains a copy of Alice's hashed identifiers and their associated encrypted data fields "in escrow" for later use. He then rehashes copies of Alice's hashed identifiers using the same commutative one-way hash function with a new random hash key H_b of his choosing and shuffles the result, returning the now doubly-hashed identifiers to Alice in some random order.

$$Bob \rightarrow Alice: \quad \{H_b H_a(I_a)\}_i \quad 0 \leq i < |A|$$

The hash key H_b is Bob's secret, and should never be revealed.

(3) Bob next provides Alice with copies of his records, where the identifiers are hashed using the same commutative one-way hash key H_b of the previous step and the data fields are encrypted using a commutative cipher with random key K_b of Bob's own choosing.

$$Bob \rightarrow Alice: \quad \{H_b(I_b), K_b(D_b)\}_j \quad 0 \leq j < |B|$$

The encryption key K_b is also Bob's secret, and should never be revealed.

(4) Alice uses her one-way hash function key H_a to rehash Bob's hashed identifiers, and her encryption key K_a to encrypt the associated data fields. At this point, Alice knows both

$$\begin{array}{ll} Alice: & \{H_b H_a(I_a)\}_i \quad 0 \leq i < |A| \\ & \{H_a H_b(I_b), K_a K_b(D_b)\}_j \quad 0 \leq j < |B| \end{array}$$

and recall that, because we are using a commutative cipher and a commutative one-way hash function, $K_b K_a(x) = K_a K_b(x)$ and $H_b H_a(x) = H_a H_b(x)$.

(5) Alice now computes the union of the doubly-hashed identifiers and their associated data, if any, and then fills out the missing data fields with random bit strings, R. Each R must be identical in size to the encrypted data fields, so that all the records in the union have like format and size. She then shuffles the result, and returns it to Bob in some random order.

$$Alice \rightarrow Bob: \quad \{H_b H_a(I_{ab}), \{K_a K_b(D_b) \vee R\}\}_l \quad 0 \leq l < |U_{ab}|$$

where $I_{ab} = I_a \bigcup I_b$ and each doubly-hashed identifier $H_b H_a(I_{ab})$ is paired with either its doubly-

encrypted data field $K_a K_b(D_b)$, if available, or else some random pattern R.

(6) Bob decrypts, using $\bar{K}_b$, the data fields, producing:

$$Bob: \quad \{H_b H_a(I_{ab}), \{K_a(D_b) \vee \bar{K}_b(R)\}\}_l \qquad 0 \le l < |U_{ab}|$$

Since the random fillers R are just random bit sequences, $\bar{K}_b(R)$ is also a random bit sequence, and remains indistinguishable, to Bob, from $\bar{K}_b K_b K_a(D_b) = K_a(D_b)$.

(7) Next, Bob applies his hash H_b to the identifiers in Alice's original records (which he had previously, in step 2, retained ''in escrow''), and then merges the resulting records with those just received from Alice in the previous step, overwriting any existing data fields in the latter.

$$Bob: \quad \{H_b H_a(I_{ab}), K_a(D_{ab})\}_l \qquad 0 \le l < |U_{ab}|$$

where D_{ab} denotes the data attributes associated each record in I_{ab} with D_a taking precedence over D_b for records in the intersection of the two data sets. Note that the overwritten fields are either just random bit strings or encrypted, and therefore unrecognizable, versions of Bob's own data for records that appear in both data sets.

(8) Bob discards the doubly-hashed identifiers, then shuffles and returns the remaining now singly-encrypted data fields to Alice:

$$Bob \rightarrow Alice: \quad \{K_a(D_{ab})\}_l \qquad 0 \le l < |U_{ab}|$$

(9) Alice decrypts, using $\bar{K}_a$, the data fields received from Bob, to produce the set-theoretic union of the two original data sets:

$$Alice: \quad \{D_{ab}\}_l \qquad 0 \le l < |U_{ab}|$$

As we shall soon see, Alice should *not* share this result with Bob in any form; the result is for Alice and Alice alone.

Having established that Alice is indeed able to compute $\Pi_D(A \bigcup_I B)$, we next consider whether the protocol permits either Bob or Alice to acquire additional information which should instead have been protected.

4. Discussion

Recall that, while we are primarily interested in safeguarding identifying information (*i.e.*, I_a and I_b), we must also safeguard other aspects of the union, such as the number and/or identity of any records in the intersection of A and B. Clearly, both Alice and Bob learn $|U_{ab}|$, the size of the union. Alice learns $|B|$, the size of Bob's data set, in step 3, so it is a simple matter to compute $|A \bigcap B| = |A| + |B| - |U_{ab}|$ once she computes the union in step 5. Similarly, Bob can also compute the size of the intersection in step 5 when Alice sends the shuffled union back to him for decryption (he learns $|A|$ in step 1).

It is possible, with some effort, to obscure the exact value of $|A|$ from Bob and $|B|$ from Alice by having both parties introduce extraneous records (selected from a predetermined illegal data attribute distribution) which are then filtered out by Alice at the end of the protocol. But such inflationary masking still reveals loose bounds on $|A|$ and $|B|$ and, moreover, is not a critical element of our protocol: in our application, the data set sizes *per se* are never confidential information. In the end, our data analysis will lead to publication in the medical/scientific literature, where the sizes of the data sets are generally revealed.

Leaving aside the size of the intersection, the whole point of the protocol is not to reveal any I_a to Bob or any I_b to Alice. Alice obtains $H_b(I_b)$ in step 3, while Bob obtains $H_a(I_a)$ in step 1. Of course, these values are only as secure as the chosen keyed one-way hash function; since one-way hash functions cannot easily be inverted (indeed, that's the definition of a one-way hash function), neither Alice nor Bob can reconstruct an identifier from its postimage. Furthermore, as long as Alice cannot determine Bob's hash key H_b and as long as Bob can not determine Alice's hash key H_a, it is also impossible for either

party to engage in a dictionary attack to reveal the other party's identifiers.

Note that even if both Alice and Bob learn how many of their original records were replicated in each others' data set, they can not know *which* of their records were actually replicated.[3] From Alice's perspective, this is because the end product of the algorithm is $\Pi_D(A) + \Pi_D(B \setminus_I A)$ where + is the concatenation operator and $\setminus_I$ is the set difference operator that considers two records equal if and only if their identifiers are equal. In other words, all of Alice's original data fields are present in the result, and the only time she could have matched her own identifiers to Bob's identifiers (step 4) they were blinded by H_b. From Bob's perspective, while he was ultimately responsible for overwriting his own replicated data attributes while restoring Alice's data from escrow (step 7), he was not able to recognize them or their identifiers because they were blinded by H_a. Note further that it is clearly in Bob's interest to use Alice's data from escrow and not deviate from the protocol described, since doing so could only be to Alice's, and not Bob's, advantage.

The protocol also precludes the sort of dictionary attack alluded to in Section 2, since neither party has access to the other party's hash function key. Even if Alice adds additional ''probe'' records to her data set *a priori*, by overwriting his own data in step 7, Bob ensures that Alice will only ever see the ''probe'' data attributes she provides for records in the intersection. And since Bob does not return the doubly hashed identifiers to Alice, there is no means for Alice to probe the intersection after the fact; all probe records need to be added before the protocol starts (for Alice and Bob to collude here makes no sense, since if they're willing to collude they may as well share data openly). There is still one relatively weak form of dictionary attack available to Alice: by submitting only ''probe'' records (along with,

[3] More precisely, $\text{Prob}(\{I_b, D_b\} \in A \bigcap B) = \frac{|A \bigcap B|}{|B|} = \frac{|A| + |B| - |U_{ab}|}{|B|}$, which goes to 1 when $|U_{ab}| = |A|$ and goes to 0 when $|U_{ab}| = |A| + |B|$ (a symmetric formulation holds for $\text{Prob}(\{I_a, D_a\} \in A \bigcap B)$). So in the special cases where all or none of one participant's records are replicated in the other participant's data set, the extent of the overlap is clear once $|U_{ab}|$, $|A|$, and $|B|$ are known. In other cases, random guessing about whether a particular record is in the intersection of the data sets is the best anyone can hope to do.

optionally, some randomly-generated ‘‘distractor’’ records) as A, Alice can determine whether all of her probe records are also in Bob’s database once she learns $|B|$ and $|U_{ab}|$. Of course, this will only work once, since Bob would have to be stupid not to wonder why Alice is repeatedly asking to engage in computing the union of her data sets with his: also, it violates our previously assumed semihonest model since Alice’s actions violate the rules of the protocol.

It is important to note that Alice should never share her computed union U_{ab} with Bob. This is because the D_{ab} attributes in U_{ab} may not necessarily be the same as the D_{ba} attributes in U_{ba}, the union computed if Bob and Alice replay the protocol with roles reversed. To see why this is so, consider what happens when Alice and Bob share a record with identifiers $I_a = I_b$, but with $D_a \neq D_b$. Since the union computed by Alice contains D_a, and not D_b, Bob would immediately recognize the missing D_b and deduce that that particular record is replicated in both data sets. If, on the other hand, the protocol is replayed with roles reversed, Bob would see his own data attributes for any record in the intersection, and not those originally belonging to Alice.

5. Conclusion

We have presented a solution to the two-party privacy-preserving database union problem. Our solution allows an initiating participant to compute the true set-theoretic union (*i.e.*, without duplicated subjects) of a collection of data sets without obtaining identifying information about records belonging to other participants or divulging identifying information about one’s own records. Our solution operates under the standard semihonest model, which assumes cooperating, yet still curious, participants, and precludes a participant from learning exactly which of his or her own records are present in the other participant’s data set. Although in this paper we have assumed a horizontal partitioning of the data (*i.e.*, all parties have data sets with an identical collection of attributes), our protocol is equally suitable for vertically partitioned data or even mixed data, where some attributes are present in only some of the participants’ data sets (in such cases, the escrow process ensures that values for unshared attributes are not

acquired in the case of a record in the intersection of the data sets). Moreover, unlike the existing work in secure multiparty computation, which is notoriously inefficient in practice, our protocol is quite efficient. Since each participant encrypts and decrypts a record at most once with each key in their possession, and since each participant hashes each identifier at most once, the cost of the protocol is $O(n)$ where $n = |A| + |B|$.

In short, our protocol is cheap to execute, easy to implement, and does not require a trusted third party. The protocol is easily extended to multiple parties with some minor modifications. Our protocol is therefore suitable for use in the analysis of data obtained from multisite clinical studies, where prior arrangements for sharing data have not been made, and, therefore, the sharing of identifying record information is precluded by law. This problem is also encountered in practice when data mining in a commercial setting, where data set owners are willing to cooperate to a certain extent, yet are keen to protect identifying values of their own records.

Acknowledgements

The authors are grateful to Robert Hansen, Eunjin Jung, Ramon Lawrence, Meredith Patterson, and Alessio Signorini for their helpful comments. Support for this research was provided in part by the National Science Foundation through grants ITR/ACI0218491 (AMS) and CCLI/A&I0511391 (APW), the National Alliance for Autism Research through the Autism Genome Project grant (VJV), and the National Institutes of Health through grant R01/NS042165 (VJV).

References

[1] N. Adam and J. Wortman, "Security-Control Methods for Statistical Databases: A Comparative Study," *Association for Computing Machinery Computing Surveys* **21**:4 (December 1989), pp. 515-556.

[2] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," *Association for Computing Machinery SIGMOD Workshop on Management of Data* (2000).

[3] G.J. Annas, "HIPAA Regulations —a New Era of Medical-Record Privacy?," *The New England Journal of Medicine* **348**:13 (April 10, 2003), pp. 1486-1490.

[4] R.C. Elston and J. Stewart, "A General Model for the Genetic Analysis of Pedigree Data," *Human Heredity* **21** (1971), pp. 523-542.

[5] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game: A Completeness Theorem for Protocols with Honest Majority," *Annual ACM Symposium on Theory of Computing* (1987).

[6] A. Hundepoool, A. vanDeWetering, R. Ramaswamy, P.P. deWolf, S. Giessing, M. Fischietti, J.J. Salazar, J. Castro, and P. Lowthian, "T-Argus User's Manual, Version 3.1," CASC 4.2-D6, Statistics

Netherlands (February 2005).

[7] A. Hundepoool, A. vanDeWetering, R. Ramaswamy, L. Franconi, S. Polettini, A. Capobianchi, P.P. deWolf, J. Domingo, V. Torra, R. Brand, and S. Giessing, "M-ARgus User's Manual, Version 4.0," CASC 2-D6, Statistics Netherlands (November 2004).

[8] M. Kantarcioglu and C. Clifton, "Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data," *IEEE Transactions on Knowledge and Data Engineering* **16**:9, IEEE Computer Society (September 2004), pp. 1026-1037.

[9] E. Lander and P. Green, "Construction of Multilocus Genetic Linkage Maps in Humans," *Proceedings of the National Academy of Sciences* **84** (April 1987), pp. 2363-2367.

[10] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," *Journal of Cryptology* **15**:3 (June 2002), pp. 177-206.

[11] J. Ott, "Analysis of Human Genetic Linkage," *Johns Hopkins University Press* (1999).

[12] S.C. Pohlig and M.E.Hellman, "An Improved Algorithm for Computing Logarithm of GF(p) and its Cryptographic Signifi cance," *IEEE Transactions on Information Theory* **IT24**, IEEE Computer Society (1978), pp. 106-110.

[13] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning* **1**:1, Kluwer Academic (1986), pp. 81-106.

[14] A.C. Yao, "Protocols for Secure Computations," *Annual Symposium on Foundations of Computer Science* (1982).