

Implementing Privacy-Preserving Bayesian-Net Discovery for Vertically Partitioned Data

Onur Kardes*

Stevens Institute of Technology
onur@cs.stevens.edu

Raphael S. Ryger†

Yale University
ryger@cs.yale.edu

Rebecca N. Wright*

Stevens Institute of Technology
rwright@cs.stevens.edu

Joan Feigenbaum‡

Yale University
feigenbaum@cs.yale.edu

Abstract

The great potential of data mining in a networked world cannot be realized without acceptable guarantees that private information will be protected. In theory, general cryptographic protocols for secure multiparty computation enable data mining with privacy preservation that is optimal with respect to the desired end results. However, the performance expense of such general protocols is prohibitive if applying the technology naively to non-trivial databases. The gap between theory and practice in cryptographic approaches is being narrowed, in part, by the introduction of problem-specific secure computation protocols.

We describe our implementation of the recent Yang-Wright secure protocol for Bayes-net discovery in vertically partitioned data. Our development occasions the proposal of a general coordination architecture for assembly of modularly described, complex protocols from independently implemented and tested subprotocol building blocks, which should facilitate future similar implementation efforts.

1. Introduction

The discovery of Bayesian networks in large bodies of personal data—medical, racial, ethnic, educational, financial, criminal, etc.—may be the key to scientific progress of great immediate benefit, informing public policy and even leading to breakthroughs in the understanding of underlying mechanisms. As simultaneous access to such varied data

dispersed across separately maintained databases is becoming increasingly feasible technically, privacy concerns regarding this accessibility are forcing its curtailment in practice through ethical and legal hindrances, hence the efforts toward *privacy-preserving* data mining. Among the approaches in this area, the theory of secure multiparty computation [2] offers strong assurances of privacy preservation with no compromise of accuracy, although typically at a performance penalty that can only be mitigated through ingenuity in its application to particular problems.

Devising a protocol to achieve some desired accuracy, privacy, and performance characteristics is one step; implementing it is another. Implementation reveals possible theoretical gaps in the protocol design while raising new issues of software maintainability, deployability, and usability. Understandably, there is little incentive to implement a complicated protocol that will surely be impractical in its performance, which is why so little of secure multiparty computation theory has been implemented. However, available computing resources have become more powerful, bringing previously impractical protocols into new consideration and occasioning the development of software such as the *Fairplay* system [4], which implements the general two-party Yao protocol ([11], discussed in section 1.2 below). At the same time, problem-specific secure protocols promising much better performance than general approaches have been proposed—notably, in the data-mining setting, the protocols of Lindell and Pinkas [3]—further encouraging interest in the prospect of practical implementation.

Yang and Wright have recently presented just such a problem-specific secure protocol [8, 9], indeed posing, along with significant challenges, an invitation to implementors. The few points at which the protocol depends on general secure two-party computation involve very confined tasks, and the *Fairplay* system is now available to address

*Supported in part by NSF grant 0331584 and by the the Wireless Network Security Center (WiNSEC) at Stevens Institute of Technology.

†Supported in part by ONR grant N00014-01-1-0795 and by US-Israel BSF grant 2002065.

‡Supported in part by NSF grants 0219018, 0331548, and 0428422 and by ONR grants N00014-01-1-0795 and N00014-04-1-0725.

these. Our implementation of this protocol is the subject of this paper. Yang and Wright adapt the *K2* heuristic algorithm of Cooper and Herskovits for Bayes-net discovery [1] to the very relevant scenario of a logical database partitioned vertically—dividing up the (non-key) fields for the same logical records—between two parties who must not learn each other’s data beyond what follows from the output of the protocol. To accomplish their adaptation, they first transform the *K2* scoring function and then invoke several cryptographic technologies to compute and compare scores securely. We sketch the original *K2* algorithm (§1.1) and introduce the cryptographic technologies that will be brought to bear (§1.2). We then describe the synthesis of these elements in the design of the Yang-Wright protocol (§2). With this background, we turn to the issues that arise in implementing this protocol (§3). Beyond the implementation issues relating to the specific subprotocols needed by the Yang-Wright protocol (§3.3), our experience in this project leads us to some broad observations and a development approach, including a subprotocol coordination framework (discussed in §3.1), applicable to complex protocols in general.

1.1. The *K2* algorithm

Given a database table—the rows viewed as records representing entities, and the columns, each defining a record field, corresponding to attributes of the entities—the statistical relationships among the attribute values generalized over the entire body of data may be partially represented by a Bayesian network. Each node of the network represents an attribute; we speak of nodes and attributes interchangeably. The directed arcs incoming to any given node come from “parent” nodes, whose values are viewed as predictive of the values of the given node. This predictiveness is specified by a conditional probability table for the values of the given node, keyed by the possible joint value assignments to its parent nodes. The Bayes-net structure is the network without the nodes’ conditional probability tables. Given the database and a Bayes-net structure, the conditional probability tables are determined. However, different choices of Bayes-net structure can produce Bayes nets, all accurate, that vary greatly in their predictive usefulness. Clearly, conditional probability tables that are sharply modal are best; those that mirror the unconditional attribute-value probabilities are unenlightening. The primary challenge, then, is to choose an optimal Bayes-net structure, i.e., an optimal identification of parent nodes for each node.

The *K2* algorithm for Bayes-net structure discovery involves two main elements: a scoring function for candidate parent-node sets; and a greedy heuristic to constrain the combinatorics of exhaustively searching the space of candidate parent sets for each node and scoring each can-

didate set. The heuristic itself has two aspects, as we will outline presently: a fixed constraint on the size of the increments to candidate parent sets between rounds of exploration of the candidate-parent-set space; and a configurable constraint on the allowable size of candidate parent sets for eligibility for consideration. (A constraint on the set size does, of course, constrain the set-increment size. Cooper and Herskovits, however, have a more stringent constraint on the set-increment size in mind.) Without the heuristic, the scoring function could be the basis of an exponential-time algorithm to find the optimal (as must be precisely defined) Bayes net for the data.

The scoring function always applies to a node and a candidate set of possible Bayes-net parents. The search for an optimally scoring parent set is conducted for each node entirely independently. The search heuristic operates as follows. We begin with a linear ordering of all the nodes such that all directed arcs in the sought Bayes net will be consistent with this linear order. (The availability of this linear ordering is a major assumption.) For each node, we build its Bayes-net parent set incrementally from the nodes that precede it in the linear order, beginning with the empty set, always adding from among the unused candidates a single node (the heuristic fixed increment) that most improves the score, and aborting the incrementing if the parent set has grown to the stipulated maximum size (the configurable heuristic parameter) or if no single added node does improve the score. Note that it is perfectly possible that two nodes added at once would improve the score even though no single node can be added to improve the score. The heuristic fixed increment size amounts to a gamble that this is not so in the case at hand. This observation suggests a natural algorithm extension that we implement, as discussed in section 3.2.

In virtue of the fixed candidate-parent-set increment size between rounds, the running time of the discovery algorithm, as measured in applications of the scoring function, goes from being exponential to polynomial in the number of nodes. Now, what of the computation of the scoring function itself? The scoring function for a node i and a candidate parent set π looks like this,

$$\prod_{j=1}^{q_\pi} \frac{(d_i - 1)!}{(d_i - 1 + \alpha_{\pi j})!} \prod_{k=1}^{d_i} \alpha_{\pi i j k}!$$

where j indexes the q_π possible value assignments to the nodes in π , k indexes the d_i possible value assignments to node i itself, $\alpha_{\pi j}$ counts the records in the database matching value assignment j to the nodes in π , and $\alpha_{\pi i j k}$ counts the records that match value assignment j to the nodes in π and additionally match value assignment k to node i itself. Noting that the outer product ranges over all value assignments to the nodes in the candidate parent set, we see that

it is here that the *K2* heuristic needs to constrain the size of parent-set candidates to be considered, to avoid worst-case growth of the outer iteration count that would be exponential in the total number of nodes.

It is suggestive and economical to grasp the operand of the outer product in the scoring function as precisely the inverse of

$$\binom{d_i - 1 + \alpha_{\pi_j}}{(d_i - 1), \alpha_{\pi_{ij1}}, \dots, \alpha_{\pi_{ijd_i}}}$$

where the notation $\binom{n}{r, s, \dots}$, with $n = r + s + \dots$, denotes the number of combinations of n things taken exhaustively in bins respectively of sizes r, s, \dots . (The usual “choose” notation, $\binom{n}{r}$, coincides with $\binom{n}{r, (n-r)}$.) It is easily seen that this expression is smallest, and its inverse biggest (but always the reciprocal of an integer!), when the bin sizes are in a sharply modal distribution. A sharply modal distribution in the “bin sizes”—the α -parameters—of the outer-product operand in the *K2* scoring expression, translates directly into a sharply modal distribution in row j of the conditional probability table for node i , which is just what we would like, as we have observed.

The α -parameters in the arguments to the factorial function represent counts of records matching partial field-value specifications, hence they may be as large as the number of records in the database. This is of little practical concern in ordinary computation. Even for a database of 100 million records, an approach as crude as looking up factorial values in a table would be feasible (if not recommended). On the other hand, in secure computation the practical options are much more limited, and so the approach to these factorials in the scoring function is at the heart of the Yang-Wright proposal.

1.2. Cryptographic tools

The Yao protocol for general secure two-party computation [10, 11] (see [5] for a detailed account) is the protocol that first demonstrated that general secure multiparty computation was possible. It requires the function to be computed to be represented as a Boolean circuit. A Boolean circuit is distinctive in doing the same computational work regardless of its inputs, a feature essential to the disguising of its inputs. This is wasteful when the amount of computation needed for different inputs differs significantly. Thus, the Yao protocol is inappropriate even just for the exact computation of factorials for arguments that range from very small to very large, as in the *K2* scoring function. On the other hand, the generality of the Yao protocol allows it to be a fallback option when no specialized protocol has been devised and the task is small. The Yang-Wright protocol resorts to episodes of the Yao protocol in this capacity at three points. One instance, within the Lindell-Pinkas $\ln x$ subprotocol,

is mentioned at the end of this section. The other two are described in section 2. The implementation we use is the recent *Fairplay* system [4], which provides two facilities: (1) a Boolean-circuit generator that takes a high-level algorithmic description as input; and (2), taking a Boolean circuit as input, run-time software for the two parties that will engage in the protocol. The *Fairplay* circuit generator, currently implemented in Java, is resource-hungry itself and, more important, may produce circuits that could be significantly optimized to the benefit of protocol performance, so it is often best to develop a custom circuit generator and then use *Fairplay* to run the protocol. Our implementation uses custom circuit generators for two of the three Yao-protocol episodes.

The Yang-Wright protocol requires an encryption scheme with the following additive homomorphic property, where E encrypts, and suppressing the details of the ring(s) in which the operations take place:

$$E(m_1 + m_2) = E(m_1)E(m_2)$$

More accurately, it being essential that different encryptions of the same plaintext be possible, we need the following property, where D decrypts, and r_1, r_2 are random values:

$$D(E(m_1, r_1)E(m_2, r_2)) = m_1 + m_2$$

The scheme used is one proposed by Paillier [6], as implemented using OpenSSL library functions by Subramaniam, Wright and Yang [7].

The α -parameters for the scoring, as said, represent counts of records matching partial value assignments to the database fields. With the database vertically partitioned as in the Yang-Wright setting, the logical records to be matched and counted span the local records of the parties. The fields to which values are assigned for matching are, in the general case, partitioned between the two parties. The count of matching records, then, is the scalar product of two bit vectors, the “match vectors,” each marking by 1-bits the matching local records (the local portions of the logical-records) held by one of the parties. It is supremely important to keep these scalar products, the α -parameters essential to the scoring and hence to the whole computation, secret from both parties! Revealing them can be tantamount to revealing to one party the field values held by the other party for a particular logical record. Accordingly, a secure protocol is needed to compute the scalar product of binary vectors leaving additive shares of the result, rather than the result itself, with the two parties for further computation. Yang and Wright use a simple scheme based on additive homomorphic encryption. Let Alice and Bob be the two parties, where both can encrypt but Alice alone possesses the decryption key. Alice sends Bob a bit-wise encryption of her match vector. Bob multiplies just those bit encryptions submitted by Alice that correspond to matching local

records in his own data (or 1-bits in his match vector).¹ Bob further multiplies this product by the encryption of a random r and returns the result to Alice. Alice decrypts it to get her additive share of the scalar product while Bob holds $-r$ (in the appropriate modulus) as his share.

Oblivious polynomial evaluation is a basic cryptographic protocol component we have to implement to serve within secure multiplication and secure natural-logarithm protocols, both of which are needed in the Yang-Wright version of the $K2$ scoring. In oblivious polynomial evaluation, one party has a secret polynomial and another party has a secret argument. Neither party may learn the other's secret, yet the argument holder is to learn the value of the polynomial at his argument. If the secret polynomial is $\sum_{i=0}^k a_i x^i$ and the secret argument is b then, given an additive homomorphic encryption scheme for which the argument holder has the decryption key, the argument holder may send the polynomial holder a vector of encryptions of powers of his argument $\langle E(b^i, r_i) \rangle_{i=0}^k$; the polynomial holder can then compute and return

$$\prod_{i=0}^k (E(b^i, r_i))^{a_i}$$

which the argument holder can decrypt to yield the value of the polynomial at the argument.

A secure multiplication protocol will be needed that takes additive shares of the factors and leaves additive shares of the product to the respective parties. This may be accomplished easily through two oblivious polynomial evaluations, as shown in [3].

The last major protocol building block we need is a secure protocol taking additive shares of x as party inputs and leaving the parties with additive shares of $\ln x$. Such a protocol is provided by Lindell and Pinkas [3] and is our most intricate building block. It begins with an episode of Yao-protocol computation establishing the logarithm approximately, then proceeds to an oblivious polynomial evaluation to compute some number of terms of Taylor expansion to reduce the initial error. The result emerges scaled up by a publicly known factor to retain precision while always computing in integers.

2. The Yang-Wright protocol

Returning to the $K2$ scoring function, how can a factorial be computed securely from secret shares of the argument? We have already observed that a Yao-style computation of a Boolean circuit, whether it carries out the multiplication or looks up the result in a large table, would not be practical, because too large a circuit would be required, necessarily to be traversed entirely for every factorial invocation.

¹Bob may need to pad his response time to disguise his computation time, which will be proportional to his 1-bit count.

The Yang-Wright protocol addresses this problem by replacing each factorial in the scoring function with a Stirling approximation. For $n \geq 1$,

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

Next, since scores are important only in their ordering, the natural logarithm of the entire Stirling-approximated expression is taken as the scoring function to implement securely. The transformed function is then amenable to secure computation from additive shares of the α -parameters using the Lindell-Pinkas protocols just mentioned.

The protocol has a superstructure that involves no private information and tracks the $K2$ algorithm almost precisely. The one difference is that scores cannot appear in the clear for comparison at the top level, as they are too revealing of private data. Instead, the top level is aware only of the chosen best score-improving increment, if there is one, to the parent set being grown. All else that is known at the top level of the protocol is considered safely disclosable to both parties: the parent set that has already been established for a node; which additional node is being scored; and which of all the additional nodes tried, if any, has been chosen for inclusion in the parent set. This information is considered disclosable on the premise that the parties could reconstruct these stages of the progress of the algorithm from the end result anyway.²

The parties are particularly *not* to know the α -parameters that feed the scoring. This means that the α -parameters must be computed cooperatively by the parties so as not only to hide each other's inputs, which are the identities of their respective matching private records, but also to disguise the α -parameter outcomes, which are the counts of matching logical (cross-party) records, as well. This is accomplished by the secure scalar product protocol described above.

The transformation of the scoring as described—Stirling approximation, then natural log—leaves a specification gap in failing to attend to the cases where α_{π_j} or $\alpha_{\pi_{ijk}}$ is 0. The Stirling approximation formula does not apply in this case. The 0 value here means simply that no record matches a partial field-value configuration being considered, which is a perfectly normal state of affairs. Clearly, this case must be handled differently. The challenge is that the parties must not realize that different handling has been triggered. A tip that, in some instance, the value of $\alpha_{\pi_{ijk}}$ is 0 can be tantamount to specific information regarding the value of a field held by the other party for a specific record.

²Strictly, the parties could not necessarily reconstruct in which round of consideration each parent node was included, so the Yang-Wright protocol, by revealing the progress of the growth of the parent sets, even without revealing complete score-based orderings (let alone scores themselves) *within* the rounds of parent-set-increment consideration, presumably does reveal some non-result-implied private information. This leak should be small. Remedying it would be extremely expensive.

Two resolutions for this problem come to mind, both involving interpolation of an additional Yao-protocol episode and adjustments to the algebraic manipulations of the transformed scoring formula. The first resolution invokes a Yao episode immediately after returning from the scalar product computation that yields $\alpha_{\pi_{ijk}}$. Observing that $0! = 1!$, securely check whether the shares of the scalar product are shares of 0; if they are, replace them with shares of 1; if not, reshare the sum of the shares. The weakness of this resolution is that in the common 0 case it proceeds to calculate the Stirling approximation of $1!$, which is the least accurate instance of this approximation, undershooting by 8%. Whereas a value configuration not instantiated in the candidate parent set would, in original *K2* scoring, either not be considered at all or put a 1 to no effect in the product—which should translate to a clean 0 in the summation of the logarithms—the promotion of 0 to 1 in this secure version introduces additional error. The alternative resolution is to invoke a similar Yao episode, but to do so late, after all the protocol for securely computing (an approximation of) the \ln of (the Stirling approximation of) $\alpha_{\pi_{ijk}}$ has run, possibly quite inappropriately. We feed the corrective Yao episode both the newly obtained shares of the result and the original shares of $\alpha_{\pi_{ijk}}$, as well as random values from the two parties. If the protocol finds that we started with $\alpha_{\pi_{ijk}}$ shares that are shares of 0, it returns new shares of 0; otherwise, it returns new shares of the elaborately computed result.

A Yao protocol is again used when deciding which candidate parent node most improves the score of the parent-node set. The inputs are vectors of score shares from the two parties. The output is the index of the (first) best score.

3. Implementation

The original presentation [8] of the Yang-Wright protocol addressed only binary data in the database fields. We currently implement that version, although almost no change is necessary to implement the more recent, general version [9].

3.1. The coordination architecture

Our implementation takes the unusual course of positing a seemingly extraneous role of *coordinator* for distributed computations. When considering security in distributed computations, we often do imagine, at least for theoretical comparison, an added role of “trusted party.” By definition, such an added party can be depended upon by the principal parties to compute and communicate as required, and particularly to refrain from communicating any more than is required (and so it can be resorted to straightforwardly

for a benchmark, “ideal” solution to any secure-multiparty-computation problem). In contrast, the coordinator we are envisioning is a party that, at least in this role, assumes absolutely no responsibility toward the principal parties whose protocol activity it coordinates, whether in computation, communication, or discretion with confidential information. On the contrary, the principal parties who may have privacy concerns should think that information that has reached the coordinator has become public thereby; in a sense, the coordinator may be taken to *represent* the public at large to them. The advantage in having a trusted party, if one can be found, is clear. Why would we add a party that is not to be trusted?

There is a definite change in orientation here as to who is responsible for what, and it turns on appreciating, in the first place, that as we move from theorizing about a large, intricate protocol to implementing it, we are moving squarely into the realm of software engineering. We need modularity not only in the design but also in the coding and testing for all the reasons that apply in developing any software. We need the modules to know as little as possible of the world outside themselves, interfacing with each other minimally. The more outside awareness an individual code module has, the more difficult it is to keep it up to date and deployed to the agents running it as changes occur elsewhere in the code, whether in the way of enhancements or bug fixes.

Now, consider what it takes to run the Yang-Wright protocol. The protocol involves several subprotocols, each of which runs many times in the course of a single run of the overall protocol. As it happens, this protocol is completely synchronous. There is no indeterminacy in the sequence of the communication, so two non-faulty parties cannot have doubts as to who is waiting for whose message and where they stand in the protocol. This means that if the database owners both know the entire protocol, with no version discrepancies, and if they know exactly when a run is to commence, and if no additional messages could possibly appear on the channel between them (for whatever reason), and if no messages between the parties get dropped, then they should be able to step through the whole protocol and ultimately both output the computed Bayes-net structure. On one hand, this is the manner of execution envisioned during the design of the protocol at the theoretical level. On the other hand, each of the run-time assumptions just enumerated introduces fragility that is unacceptable in real deployed software.

Instead of requiring that the database owners themselves know the whole protocol, we envision them as being willing and able to run the particular needed subprotocols on cue as discrete services available to certain requestors. Note that this notion of service is more elaborate than the one referred to in common client/server terminology. We are imagining a service provided not by a single party but by multiple parties

who are expected, when cued, to engage each other in some protocol in order to return the sought result, each party, or at least some quorum, reporting back to the “client.” This is not directly supported in our networking infrastructure, and so it must be built on top of the common client/single-server model.

Given that all the subprotocols in question are acceptable to all the principal parties with respect to their privacy concerns, this leaves it up to any interested party authorized to request their services to invoke any of those subprotocols in any order to achieve whatever larger end—or fail to achieve it. That shifts the onus of getting the overall protocol right to some one party, the *coordinator*, that is cuing the principal parties to engage in subprotocols and is then somehow processing the results. We can imagine that the coordinator is the party that has the primary interest in the result, or may have a secondary interest in virtue of receiving payment for provision of the result to the party with primary interest, or may have a farther removed interest still, of course.

If the database owners are to offer discrete lower-level services, they should be able to do so in support of multiple concurrent runs of larger protocols. Otherwise, a long-running higher-level protocol will either be subject to interference by other requests to the database owners or else, if a locking scheme is used, the long-running protocol would monopolize services that should really be a general resource. Interleaving multiple provisions of the discrete services, possibly involving interaction with some of the same peer parties and possibly even on behalf of the same client party, entails keeping the messages associated with the different concurrent conversations properly sorted out. (Note that this is not a concern over possible information leakage across multiple concurrent conversations carried on by the same parties, but rather the more basic requirement that all the communication involved, in the first place, be attributable to the distinct conversations.)

The challenge here is entirely familiar from basic networking, wherein order is maintained in the cloud of message exchanges through reliance on metainformation in successive layers of message wrappers. Our coordination protocol requires similar metainformation. Parties need to know who their peers are (practically, domain names or IP addresses and port numbers) for each requested multiparty computation episode. Parties need to know which type of computation to engage in and they need the inputs to the computation. The individual episode of the computation needs an identifier assigned to it and passed around throughout. The coordination protocol must accommodate error propagation or non-propagation. In the broader realm of multiparty computation, the coordinator for a subprotocol may need to implement timeouts for individual parties and appropriate quorum and consistency threshold checks for the available party responses to afford promised correctness

and robustness to the overall protocol. This suggests that, as we bring more of distributed-computation theory into practice, enhancement of a general coordination protocol will be an ongoing development project.

The present Yang-Wright implementation relies on a basic coordination protocol implemented in a library of Perl functions. Thus, the highest level of our implementation is expressed in a relatively small amount of Perl code in the coordinator. Most coordination housekeeping is done inside the coordination-library functions, allowing the code to read much like the pseudo-code for the original *K2* protocol. In fact, the coordinator code can be used unchanged to run the original *K2* against a single physical database, the original *K2* against a logical database vertically partitioned among any number of parties with no privacy concern, or the Yang-Wright privacy-preserving *K2* variant. (To make the coordinator code this general, we do need it to deviate slightly from the *K2* pseudo-code. We keep parent-set scores and any intermediate values from their computation down at the principal-party code level. Because in the privacy-preserving case these values must not be public, we keep them from the coordinator in all cases for uniformity. The parties, on cue, determine best-score parent sets between themselves, either securely or not, and report them to the coordinator.) The principal parties similarly run a thin layer of Perl code which interacts with the coordinator through invocation of a coordination-library function. It is at this level that the Yang-Wright security proposals are brought in. The parties provide the various requested services to the coordinator by engaging in secure subprotocols between themselves, whereas they could satisfy the coordinator equally by engaging in the simpler insecure subprotocols. This is appropriate. We imagine that the database owners are the ones that have the interest, intrinsic or however induced, in privacy—else privacy is pretty hopeless! The coordinator just wants the answers.

The principal parties run their ends of each subprotocol as command-line-invokable processes. (The overhead of process invocation is not significant, especially when running secure subprotocols, as it is completely dominated by the computation to be done by at least one of the parties in each subprotocol episode.) This means that all the subprotocol code is available for direct testing and use from the command line just as is the overall protocol.

Nothing in the separation of roles in the coordination architecture precludes an agency acting both as coordinator of a multiparty computation and as one (or more) of the parties whose interactive episodes are being coordinated. In the case of the Yang-Wright protocol, one of the private-data owners may also run the coordinator process, on the same machine that runs the party process or on another machine. We have said that the coordinator is not to be entrusted with private data. The data owners must in any case, wherever

the coordinator is to run, assure themselves that the code they run as parties proper, in communicating with the coordinator process, not pass it revealing information. (This is in addition to having faith in the security of the subprotocols that they run with each other without the coordinator’s intervention.) A data owner assuming the coordinator role as well must also be assured that the coordinator code not somehow access the local private data without mediation of the party process. From this perspective, there is an advantage to running the coordinator process physically removed from the data-accessing party process. On the other hand, scrutiny of the coordinator code—which should generally, as in this case, be relatively simple—should allay any concern over possible rogue data access.

3.2. An extension to $K2$

Our coordinator code, which we said is not specific to the Yang-Wright security-oriented transformation of $K2$, actually implements an extension to $K2$ itself, allowing further tuning of its complexity-controlling heuristic. The desirability of the enhancement became evident in the course of our experimentation.

If, for example, two binary fields in the database are in themselves uniformly randomly distributed and the two fields are independent of each other, and if a third field is completely determined by the first two fields, being 1 if the first two fields match and 0 if not, then the $K2$ algorithm is likely not to discover any Bayesian structure in the three data fields. This is because the first two fields are tried only one at a time as members of a prospective Bayes parent set for the third field, but neither field is kept because, in itself, it is completely unpredictable of the value of the third field.

We add a parameter which we call “interactivity.” It controls the maximum number of nodes to be tried together in incrementing the parent set for a node. The idea is that it may be the interaction of several antecedent nodes, rather than any one in itself, that is predictive of the consequent node in question. The original $K2$ algorithm is obtained as the special case of interactivity-parameter 1. Raising this parameter improves the analytical capability of the algorithm at the expense of raising the degree of its polynomial time complexity.

This enhancement is mentioned here as a feature of the present implementation. It is orthogonal to the Yang-Wright security enhancements to $K2$ and their special implementation issues which are our main focus.

3.3. Subprotocol issues

The coordination architecture addresses only the generic needs of subprotocol marshaling. Specific difficulties arise in ensuring that subprotocols nest and tile properly with

respect to their data interchange. For instance, choices of moduli and bit lengths in different subprotocols must match up acceptably.

Within the Lindell-Pinkas computation of shares of $\ln x$ from shares of x , there is an oblivious polynomial evaluation for which we use a Paillier homomorphic encryption scheme. The homomorphism entails that additions and multiplications of plaintext numbers modulo some product of primes pq are carried out, respectively, as multiplications and exponentiations modulo $(pq)^2$ of their encryptions. For Lindell-Pinkas, however, the polynomial is to be computed obliviously modulo $|\mathcal{F}|$, the size of \mathcal{F} , the latter prescribed to be a field. The requirement for a Paillier plaintext modulus and the requirement for a Lindell-Pinkas modulus, then, would appear to be incompatible. Fortunately, examining the Lindell-Pinkas polynomial specification, we see that we need multiplicative inverses only of powers of 2. These inverses would exist modulo the Paillier pq , so we simply proceed letting the Paillier plaintext ring serve for the Lindell-Pinkas computation, dropping the field requirement.

The polynomial-computation space needed within the Lindell-Pinkas natural-log protocol, hence the Paillier pq for the homomorphic-encryption plaintext space, must be much larger than the space of allowable inputs for the protocol to avoid loss of information in the polynomial evaluation. The space of inputs, in turn, must accommodate all α -parameters, which may be as large as the number of records in the database. Collecting the various requirements, then, where s is the size in records of the database being mined, we need an integer N , a count of Taylor terms k for Lindell-Pinkas, and primes p, q for Paillier that will determine also the ring (not field) \mathcal{F} for Lindell-Pinkas, such that

$$s < 2^N$$

$$2^{(N+2)k} \leq |\mathcal{F}| = pq$$

We give two examples of the interplay of these conditions. To accommodate databases of up to 8,000 records and go to four Taylor terms, we can use a Lindell-Pinkas ring of size approximately 2^{60} . To accommodate databases of up to 8,000,000 records and go to five Taylor terms, we need a Lindell-Pinkas ring of size 2^{125} . As usual, more bits strengthen the cryptography, but incur a performance penalty, particularly evident in our case in the episodes of *Fairplay*-implemented Yao protocol.

As mentioned, the natural logarithm delivered in shared fashion by the Lindell-Pinkas protocol is scaled up by a large factor in order to preserve precision in the integer output. It is not practical to have the parties engage in a secure protocol episode, for each logarithm computed, to replace their shares with new shares, scaled back down, to the correct logarithm value. The Yang-Wright score expression involves many privately shared terms each of which

involves a Lindell-Pinkas-computed logarithm and would thus be scaled up, and terms that are public with no scale-up. Since scores are important only in their comparison to other scores, we need only multiply the public terms in each score by the scaling factor to scale up the entire system of scores, preserving their comparisons.

4. Some experimental results

The running time of the Yang-Wright protocol depends on the number of fields in the database, the number of records in the database, and the size of the computational spaces chosen to accommodate the database size, as reflected in the bit lengths of Paillier keys and computation values and in the input-wire and gate counts of Yao Boolean circuits.

Figures 1 and 2 show how the program modules behave with different key lengths and numbers of records.

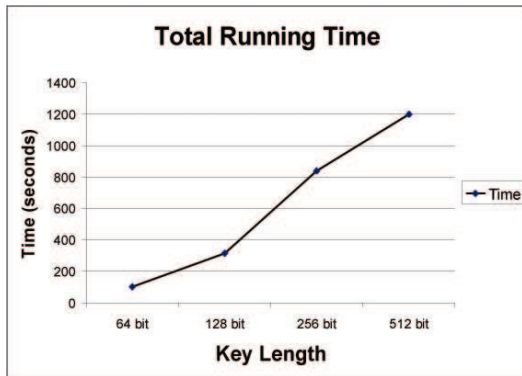


Figure 1. Key length and total running time

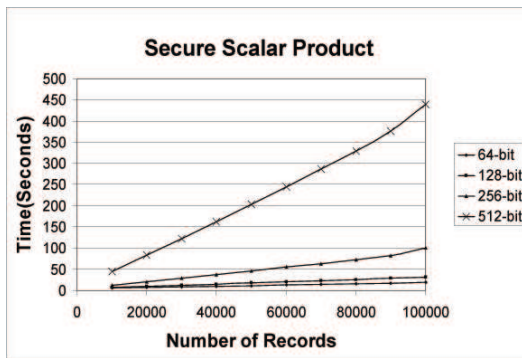


Figure 2. Database size and scalar product time

The total running time is dominated by the Lindell-Pinkas secure computation of $\ln x$ (Figure 3).

The secure $\ln x$ computation itself is dominated by its initial *Fairplay* Yao episode, which is far more expensive

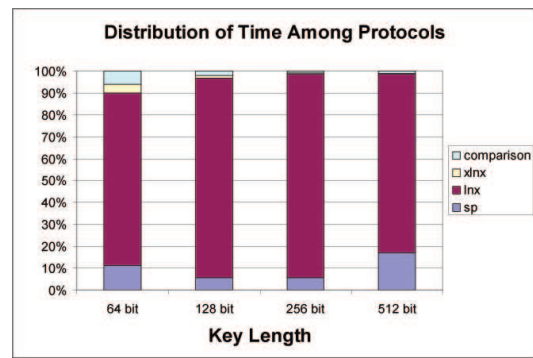


Figure 3. Distribution of overall running time

than the oblivious polynomial evaluation (Figure 4). Not surprisingly then, increasing the number of Taylor terms computed in the oblivious polynomial evaluation has little impact on the $\ln x$ running time (Figure 5).

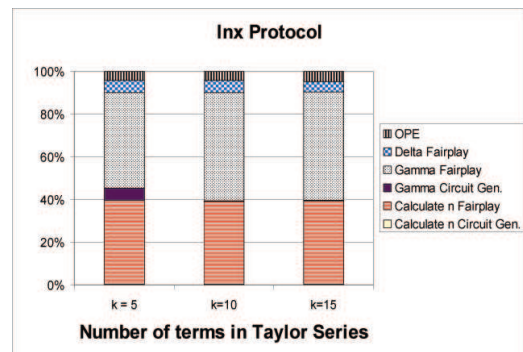


Figure 4. Distribution of running time in secure $\ln x$ computation

4.1. Visualization

The implementation provides a graphical user interface allowing the user to view the progress of the algorithm. The partitioning of the nodes between the two parties is indicated by a color code, Alice's nodes appearing in pink, Bob's in blue. The nodes carry text labels derived from a configuration table. At any time, the node whose parent set is currently being determined is highlighted. A parent-child arrow appears in red while it is being considered; the arrow disappears if the candidate parent node is rejected (in that iteration); it turns blue and becomes permanent if the candidate is accepted. For development purposes, presumably against test data, the user may click on a node to display its conditional probability table—data not properly revealed in the structure-learning-only version of the Yang-Wright protocol implemented here.

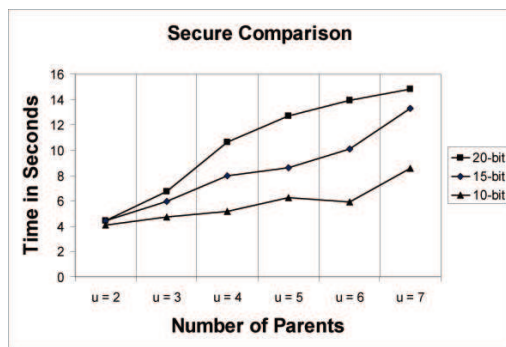


Figure 5. Terms and oblivious polynomial evaluation time

5. Conclusion

This project, implementing the Yang-Wright secure Bayes-net discovery protocol, attempts to negotiate a range of issues involved in turning a theoretical cryptographic protocol into usable privacy-preserving data-mining software. Considerable success has been achieved in understanding how such software might be deployed and invoked on demand. We have identified various specific and general pitfalls, easily missed in early design, when assembling a complex cryptographic protocol out of building blocks. We come away with a methodology for building complex protocols—particularly privacy-preserving protocols—and software implementing a coordination framework that will be reusable in future projects translating complex theoretical protocols to practice. The coordination framework itself is an arena for further development.

With respect to privacy-preserving discovery of Bayes-net structure via secure multiparty computation, performance remains a central concern. Our implementation currently takes over two hours to determine the Bayes-net structure in a six-node database of 1,000 records partitioned between two parties, three nodes each, computing with a key length of 512 bits on a Pentium 4 processor with 1 GB of RAM. On one hand, this seems very slow. On the other hand, the present technology already can render research of Bayes-net structure feasible against private data. What can be learned thereby may well justify the computational resources and the wait for the results, even at this level of performance. This is an initial report on the implementation work. There is every reason to believe that further work will yield optimizations at various levels and that performance will improve.

References

- [1] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.
- [2] O. Goldreich. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge, 2004. Chapter 7, General Cryptographic Protocols.
- [3] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.
- [4] D. Malkhi, N. Nissan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *Proc. of the 13th Symposium on Security*, pages 287–302. Usenix, 2004.
- [5] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. of the 1st Conference on Electronic Commerce (EC)*, pages 129–139. ACM, 1999.
- [6] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT ’99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.
- [7] H. Subramaniam, R. N. Wright, and Z. Yang. Experimental analysis of privacy-preserving statistics computation. In *Proc. of the VLDB Workshop on Secure Data Management (SDM)*, volume 3178 of *Lecture Notes in Computer Science*, pages 55–66. Springer-Verlag, 2004.
- [8] Z. Yang and R. N. Wright. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proc. of the 10th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–718. ACM, 2004.
- [9] Z. Yang and R. N. Wright. Privacy-preserving computation of bayesian networks on vertically partitioned data. Manuscript, <http://www.cs.stevens.edu/~rwright/Publications/bayes.ps>, 2005.
- [10] A. C. Yao. Protocols for secure computation. In *Proc. of the 23rd Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE, 1982.
- [11] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.