T-79.514 Special course on cryptology: Cryptographic Techniques for Privacy-Preserving Data Mining

Alexey Vyskubov

Abstract

This survey is aimed at reviewing privacy preserving protocol for ID3 algorithm, described in the article "Privacy preserving data mining" by Yohuda Lindell and Benny Pinkas. As the survey is supposed to be read also by people who specialises in data mining not cryptography, the review of needed cryptographic primitives from the article "Cryptographic techniques for privacy-preserving data mining" by Benny Pinkas is included.

1 Introduction: the setting

We will consider a scenario in which two parties owning confidential databases wish to run a data mining algorithm on the union of their databases without revealing any unnecessary information, specifically ID3 for building decision tree.

We are interested in distributed scenario, i.e. all the data is processed by above-mentioned two parties, without any central server or any help of some third party.

It is obvious that if a data mining algorithm is run against the union of the databases, and its output becomes known to one or both parties, it reveals something about the contents of the other database. So if we want to build privacy preserving protocol we have to define what "privacy" is.

Let us consider *ideal scenario* in which we have some trusted third party (TTP). In this scenario both parties (we will call them Alice and Bob) send their databases to the TTP, who in turn runs data mining algorithm on them and sends results back to Alice and Bob. This scenario is ideal: both Alice and Bob got as little information as possible if they have to be given results of the algorithm. Our goal is to build a protocol which will give Alice and Bob as little information as they get in ideal scenario.

One can imagine a situation when Alice or Bob is cheating and deviates from prescribing protocol trying to get more information about other party's data. We will not consider this case, supposing that both parties are *honest but curious* (semi-honest). This means that both Alice and Bob will follow the protocol but they will also try to get as much information as possible from what they see during the protocol run. We also suppose that there is some way (outside of the protocol) to check that Alice and Bob are using their real databases during the calculations (as they can, for example, to substitute empty set instead of their data).

2 Introduction: cryptographic results

We will need some "cryptographic building blocks" for our protocol. We describe them shortly in this section.

2.1 Oblivious transfer

Oblivious transfer is a basic protocol that is the main building block of secure computation. We will need 1-out-of-2 oblivious transfer. This protocol involves two parties, *sender* and *receiver*. The sender's input is a pair (x_0, x_1) and receiver's input is $\sigma \in \{0, 1\}$. At the end of the protocol the receiver learns x_σ (and nothing else) and the sender learns nothing. In the case of semi-honest adversaries there exist simple and efficient protocols for oblivious transfer.

Idea. One straightforward approach is for the receiver to generate two random public keys, a key P_σ whose decryption key he knows, and a key $P_{1-\sigma}$ whose decryption key he does not know. The receiver then sends these two keys to the sender, who

encrypts x_0 with P_0 and x_1 with P_1, and sends the two results to the receiver. The receiver can then decrypt only x_σ.

Oblivious transfer is often the most computationally intensive operation of secure protocols, and is repeated many times. Each invocation of oblivious transfer typically requires a constant number of invocations of trapdoor permutations (i.e. public-key operations, or exponentiations).

2.2 Oblivious polynomial evaluation

Oblivious polynomial evaluation involves a sender and a receiver. The sender's input is a polynomial Q of degree k over some finite field $\mathcal{F}$ and the receiver's input is an element $z \in \mathcal{F}$. We suppose that the degree k is public.

The protocol is such that the receiver obtains $Q(z)$ without learning anything else about the polynomial Q, and the sender learns nothing. There is a protocol for semi-honest case with on overhead of $O(k)$ computation and $O(k\,|\mathcal{F}|)$ communication.

We will use another protocol, invented by Yao, which is a constant-round protocol for privately computing any probabilistic polynomial-time function (where the adversary may be either semi-honest or malicious).

Setting. Denote the parties as Alice (A) and Bob (B), and denote their respective inputs by x and y. Let f be the function that they wish to compute (let us suppose that Bob should learn the value $f(x, y)$). It is known that any polynomial-time function can be expressed as a combinatorial circuit of polynomial size. The protocol is based on expressing f as a combinatorial circuit with gates defined over some fixed base $\mathcal{B}$. For example, $\mathcal{B}$ can consist of all functions $g : \{0, 1\} \to \{0, 1\}$. The bits of the input are entered into input wires and are propagated through the gates.

Encoding the circuit. We suppose that f is pubic, i.e. circuit wiring for it is known to both parties. Alice starts with "hardwiring" her inputs into the circuit, generating circuit computing $f(x, \cdot)$. She then assigns to each wire i two random ("garbled") values (W_i^0, W_i^1) corresponding to values 0 and 1 of the wire. These random values should be long enough to be used as a keys to a pseudo-random function (for example, 80–128 bits).

Then for every gate in the circuit Alice prepares a table T_g. Let i and j will be input wires of g, and k — output wire. Let F be some (public) pseudo-random function such that pair of garbled values for i and j can be used as a key for it. For every pair of possible (garbled) values of i and j the table should contain encryption of (garbled) value of output wire k using F and (garbled) values of i and j as a key. Every entry in table does not contain garbled input values but has some label which can be computed using these values.

The tables enables computation of the garbled output of g from the garbled inputs of g. Bob, using garbled values of i and j, may calculate the correct label in table, then get encrypted garbled value of k from table and decrypt it, using garbled values of i and j, but he cannot get any information about the output of g for any other inputs or about the actual (non-garbled) inputs.

As pseudo-random functions are usually implemented using private-key primitives such as block ciphers or hash functions, they are very efficient.

Sending garbled circuit to Bob. Then Alice sends the wiring of the original circuit, the tables T_g and tables that translate the garbled values of the output wires of the circuit to actual 0/1 values. In this form the representation reveals nothing but the wiring of the original circuit, and therefore Bob learns nothing.

Encoding Bob's input. The tables received by Bob enable him to calculate $f(x, y)$ from garbled inputs. Unfortunately, he is not able to convert his inputs to garbled form, as garbled values kept secret by Alice. He also is not able to ask Alice to convert his inputs to garbled form as he wants to keep his inputs secret. Fortunately, we already know the solution. Alice and Bob use 1-out-of-2 oblivious transfer for every Bob's input: Alice is the sender, her inputs are garbled values for the given input wire; Bob is the receiver, the value of his input for the circuit is σ. As a result of oblivious transfer Bob learns only the garbled value of his input and Alice learns nothing at all.

Computing the circuit. After previous stages are completed, Bob has enough information to calculate garbled outputs of the circuit and transfer them to non-garbled values. Overhead of the protocol involves:

1. Alice and Bob engaging in an oblivious transfer protocol for every input wire of the circuit.
2. Alice sending to Bob tables of size linear in the

size of the circuit.

3. Bob computing a pseudo-random function a constant number of times for every gate.

3 Introduction: ID3 algorithm

The aim of classification problem is to classify transactions into one of discrete set of possible categories. The input is a structured database comprised of attribute-value pairs. Each row of database is a *transaction* and each column is an *attribute* taking on different values. One of the attributes in the database is designated as the *class* attribute; the set of possible values for this attribute being the classes. We wish to predict the class of a transaction by viewing only the non-class attributes. This can thus be used to predict the class of new transactions for which the class is unknown.

The ID3 algorithm solves classification problem by building *decision tree.* A decision tree is a rooted tree containing nodes and edges. Each internal node is a test node and corresponds to an attribute; the edges leaving a node correspond to the possible values taken on by that attribute. The leaves of the tree contain the *expected* class value for transactions matching the path from root to that leaf.

We will notate a set of attributes as R, class attribute as C and a set of transactions as T.

ID3 algorithm is recursive. At the root of the tree, each attribute is tested to determine how good it alone classifies the transactions. The "best" attribute is then chosen and allows to partition the remaining transactions by it. After partitioning is done, the same scheme is applied to every partition. The algorithm stops when partition in consideration belongs to one class.

The main question is how to choose the "best" attribute. It is pretty obvious that the algorithm on every step should try to minimise the information of class attribute; so we search for minimal entropy in the following way.

Let $c_1, \ldots, c_l$ be the class attribute values. Let $T(c_i)$ be the set of transactions with class c_i. Then the information needed to identify the class of a transaction in T is the entropy, given by:

$$H_C(T) = \sum_{i=1}^{l} -\frac{|T(c_i)|}{|T|} \log \frac{|T(c_i)|}{|T|}$$

Let A be a non-class attribute. Then the conditional information of T given A is given by:

$$H_C(T \mid A) = \sum_{j=1}^{m} \frac{|T(a_j)|}{|T|} H_C\left(T(a_j)\right),$$

if A may have values $a_1, \ldots, a_m$.

Now we can define *information gain* for A as:

$$\mathrm{Gain}(A) = H_C(T) - H_C(T \mid A).$$

So the attribute with maximum Gain is chosen at every step.

4 The ID3$_\delta$ approximation

The privacy-preserving algorithm we are going to discuss cannot implement ID3 algorithm. Instead it implements ID3 *with some precision.* It means if at some step there are two attributes A_1 and A_2 such that

$$|\mathrm{Gain}(A_1) - \mathrm{Gain}(A_2)| < \delta$$

for some predefined small value $\delta > 0$ then any of them may be chosen as "best" one.

We will call this "approximation" ID3$_\delta$. The value of δ influences efficiency but only by a logarithmic factor.

5 Finding the attribute with maximal gain: part one

The only difficult step in privacy-preserving ID3$_\delta$ computation is finding the attribute with maximal gain. This step can be stated as: *Find the attribute A which minimises the conditional information of T given A, $H_C(T \mid A)$.* If an attribute A has m possible values $a_1, \ldots, a_m$ and a class attribute C has l possible values $c_1, \ldots, c_l$ then

$$H_C(T \mid A) = \sum_{j=1}^{m} \frac{|T(a_j)|}{|T|} H_C(T(a_j))$$

$$= \frac{1}{|T|}\sum_{j=1}^{m}|T(a_j)|\sum_{i=1}^{l}-\frac{|T(a_j,c_i)|}{|T(a_j)|}\log\frac{|T(a_j,c_i)|}{|T(a_j)|}$$

$$= -\frac{1}{|T|}\sum_{j=1}^{m}\sum_{i=1}^{l}|T(a_j,c_j)|\log|T(a_j,c_j)| + \frac{1}{|T|}\sum_{j=1}^{m}|T(a_j)|\log|T(a_j)|.$$

Now if the database T is split between Alice and Bob in parts T_1 and T_2 it is easy to see, that $T(\cdot) = T_1(\cdot)+T_2(\cdot)$ (it just a number of transactions which have given attribute value).

We do not care about actual values of the gains as we just want to compare them so it is possible to omit $1/|T|$ and use natural logarithms instead of logarithms to the base 2. Therefore the expressions that should be compared (the gains) can be written as a sum of expressions of the form $(v_1+v_2)\ln(v_1+v_2)$.

6 Computing $x \ln x$

Let Alice and Bob have values v_1 and v_2 as inputs. We suppose that there is some large enough field $\mathcal{F}$ (we will not discuss how big it should be; details are available in the original paper by Lindell and Pinkas). The goal of the protocol: Alice obtains w_1 and Bob obtains w_2 such that:

1. $w_1 + w_2 = (v_1 + v_2)\ln(v_1 + v_2) \mathrm{mod} |\mathcal{F}|$.
2. w_1 and w_2 are uniformly distributed in $\mathcal{F}$ when viewed independently of one another.

It is not possible to use directly Yao's method of garbled circuits as the size of circuit is of the order of the multiplication of the size of its inputs which is very ineffective. It is possible to build a solution which requires only a linear size circuit.

6.1 Computing shares of $\ln x$

We now show how to compute random shares u_1 and u_2 such that $u_1 + u_2 = \ln x$.

It is well-known that

$$\ln(1+\varepsilon) = \sum_{i=1}^{\infty}\frac{(-1)^{i-1}\varepsilon^i}{i}$$

for $-1 < \varepsilon < 1$. It is known that if one take only first k elements of the series the error shrinks exponentially as k grows.

Now given an input x, let 2^n be the power of 2 which is closest to x. Then $x = 2^n(1+\varepsilon)$ and

$$\ln x = n\ln 2 + \varepsilon - \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3} - \frac{\varepsilon^4}{4} + \ldots$$

For our purpose x will be the number of transactions so $n < \log|T|$. Let us choose some large $N > \log|T|$. Then it is possible to built small circuit that receives v_1 and v_2 as inputs and output shares of $2^N n\ln 2$ and $2^N\varepsilon$. To create shares the first party inputs in circuit random values α_1 and β_1; the circuit actually calculates $\alpha_2 = 2^N n\ln 2 - \alpha_1$ and $\beta_2 = 2^N\varepsilon - \beta_1$.

The parties therefore have shares $\alpha_1, \beta_1, \alpha_2, \beta_2$ such that

$$\alpha_1 + \alpha_2 = \varepsilon 2^N$$

and

$$\beta_1 + \beta_2 = 2^N n\ln 2.$$

The first party constructs the polynomial

$$Q(x) = \mathrm{lcm}(2,\ldots,k)\sum_{i=1}^{k}\frac{(-1)^{i-1}}{2^{(N(i-1))}}\frac{(\alpha_1+x)^i}{i} - w_1,$$

where w_1 is a random value. Oblivious polynomial evaluation allows the second party to obtain $w_2 = Q(\alpha_2)$ and $w_1 + w_2$ is an approximation of $\ln\varepsilon$ up to multiplicative factor $\mathrm{lcm}(2,\ldots,k)2^N$.

Let $u_i = w_i + \mathrm{lcm}(2,\ldots,k)\beta_i$. Then

$$u_1+u_2 \approx \mathrm{lcm}(2,\ldots,k)2^N\ln\varepsilon + \mathrm{lcm}(2,\ldots,k)2^N n\ln 2 = \mathrm{lcm}(2,\ldots,k)2^N\ln x.$$

So we calculated shares $\ln(v_1+v_2)$ up to multiplicative factor which is public and can be removed (but it is not important as we are interested only in comparing values).

6.2 Back to $x \ln x$

So now Alice and Bob have v_1, v_2, u_1, u_2 such that $v_1 + v_2 = x$ and $u_1 + u_2 \approx \ln x$ and they are interested in getting shares of $(v_1 + v_2)(u_1 + u_2)$. To achieve this Alice can define two linear polynomials $P_1(y) = v_1 y + r_1$ and $P_2(y) = u_1 y + r_2$ where r_1, r_2 are random. Bob runs oblivious polynomial

evaluation to obtain $P_1(u_2)$ and $P_2(v_2)$ and sets his share to

$$P_1(u_2) + P_2(v_2) + u_2v_2 =$$

$$v_1u_2 + r_1 + u_1v_2 + r_2 + u_2v_2 =$$

$$(v_1 + v_2)(u_1 + u_2) - v_1u_1 - r_1 - r_2.$$

Alice sets her share to be $v_1u_1 + r_1 + r_2$.

7 Finding the attribute with maximal gain: part two

Given the above protocol for privately computing shares of $x \ln x$, the attribute with maximum information gain can be determined. For this after getting shares for information gains for every attribute A, the shares are input into a small circuit which outputs the appropriate attribute.

8 Conclusion

In this survey we shortly described privacy-preserving algorithm for computing ID3 (with some predefined precision δ) and cryptographic primitives used to construct it. We omitted the following points (discussion of which can be found in the original paper by Lindell and Pinkas):

- Formal definition of privacy.
- All calculations because of cryptographic reasons should be done in "large enough" field $\mathcal{F}$. We do not discuss here how large is it.
- We do not provide formal definitions of described protocols as they can be found in the original paper.
- We do not provide formal proofs of privacy; in the most cases it is quite intuitive that described protocols reveal no additional information during their execution.
- We do not calculate how large should be parameter k to achieve the given precision δ.

9 Used articles

Yehuda Lindell and Benny Pinkas, "Privacy preserving data mining".

Benny Pinkas, "Cryptographic techniques for privacy-preserving data mining".