Secure two party computation, an example

Burton Rosenberg

9 June 2003

Introduction

Two parties, $\mathcal{A}$ and $\mathcal{B}$, respectively and privately holding values a and b wish to compute $f(a,b)$ in a manner which reveals as little as possible to the other party. In particular, given the known result $w = f(a,b)$, $\mathcal{A}$ should assign equal likelihood to each element of the set $\{\, y \,|\, f(a,y) = w \,\}$ and $\mathcal{B}$ should likewise assign equal likelihood to each element of the set $\{\, x \,|\, f(x,b) = w \,\}$. To do so, the two parties agree to engage in a protocol. In the *honest party model*, the parties carry out the protocol in accordance with the protocol. In the *adversarial model* the two parties are not assumed to follow the protocol. We are prepared to accept that the two parties might deviate from the protocol in order to gain some advantage.

Privacy is defined with respect to a *simulation paradigm*. Each party as a view consisting of its private input, its randomness, and the messages received from the other party. The messages sent to the other party and the final value are entirely determined by this data. The view is a random variable depending on the other party's randomness and private input. *Privacy for* $\mathcal{A}$ is achieved if the random variable,

$$(x, r, m_1, \ldots, m_k)$$

is indistinguishable, perfectly, statistically or computationally, from the output of a probabilistic polynomial time simulator $\mathcal{S}(x, f(x,y))$ which is provided with the private input and the result $f(x,y)$ of the joint computation. Privacy for $\mathcal{B}$ is defined analogously.

Since x and r are given in the view, and $f(x,y)$ is given to the simulator, indistinguishability is respect to each x, y pair over the sample space of the contra-party's randomness. This gives two facts: that for y and y' such that $f(x,y) = f(x,y')$, views are indistinguishable; and that any of the common views are polynomial time computable. Since $\mathcal{S}$ cannot simulate the other party's randomness without an assumption about the distribution, we assume that the honest parties generate randomness by flipping fair coins.

Protocol

As example, we consider two party computation of logical or. We will carry out our boolean operations as arithmetic in F_2. Logical or is given by the formula:

$$\vee(a, b) = (a \times b) + (a + b)$$

We therefore have a circuit (of depth 2) which evaluates logical or. Each party has a copy of this circuit, and will distribute shares of their inputs. The parties will each evaluate the circuit wire by wire, gate by gate, finally revealing to each other the output shares. The intuitive description of privacy is that If one of the parties, say $\mathcal{A}$ holds a 1 then the result is 1, and party $\mathcal{A}$ should learn nothing of the value held by party $\mathcal{B}$.

The evaluation protocol is given by Oded Goldreich:

`http://www.wisdom.weizmann.ac.il/~oded/pp.html.`

To evaluate a sum, one uses that the share of a sum is the sum of the shares. To evaluate a multiplication, however, is more difficult. The parties engage in an exchange in which one party, say $\mathcal{A}$, proposes four different values, calculated based on the shares it holds, from which $\mathcal{B}$ will select the appropriate value given the shares it holds. Using 1 out of 4 Oblivious Transfer $\mathcal{A}$ learns nothing of the shares held by $\mathcal{B}$ and by blinding the calculation by a random value, $\mathcal{A}$ prevents $\mathcal{B}$ from learning its shares.

Summary for honest parties:

1. Distribute the circuit;
2. Distribute inputs as shares;
3. Evaluation circuit: for addition, each party evaluates individually on share;
4. Evaluate circuit: for multiplication, a 4,1–OT transfer from $\mathcal{A}$ to $\mathcal{B}$ with a random masking bit chosen by party $\mathcal{A}$;
5. Combine output shares.

Privacy

We will justify in concrete terms why the protocol evaluates privately for the example circuit. The following subscript notation is used for shares: $x = x_a + x_b$, for a generic variable x, where $\mathcal{A}$ holds x_a and $\mathcal{B}$ holds x_b. The steps in the computation are given notation:

$$\begin{aligned} s &= a \times b \\ t &= a + b \\ y &= s + t \end{aligned}$$

That is, $\mathcal{A}$ evaluates the circuit on inputs a_a, b_a, assigning values to s_a, t_a, and output y_a. Likewise, $\mathcal{B}$ evaluates the circuit on inputs a_b, b_b, assigning values to s_b, t_b, and output y_b. The result is $y = y_a + y_b$.

Party $\mathcal{A}$ flips two coins, r_{a1}, r_{a2}, the first to split a and the second is required for the 4,1–OT. Party $\mathcal{B}$ flips a single coin, r_{b1}, to split b. W.L.O.G., the inputs are split with the communicated share being the coin value:

$$a_a = a + r_{a1} \quad a_b = r_{a1},$$

likewise for b. The result of the OT is to transfer $a \cdot b + r_{a2}$ to $\mathcal{B}$,

$$s_a = r_{a2} \quad s_b = a \cdot b + r_{a2}$$

The evaluations of t and y are done by each party separately, $t_a = a_a + b_a$, and so on. The result the following dialog between parties:

$\mathcal{A}$		$\mathcal{B}$
$a_a = a + r_{a1}$	$r_{a1} \rightarrow$	$a_b = r_{a1}$
$b_a = r_{b1}$	$\leftarrow r_{b1}$	$b_b = b + r_{b1}$
$t_a = a_a + b_a$		$t_b = a_b + b_b$
$s_a = r_{a2}$	$a \cdot b + r_{a2} \rightarrow$	s_b
y_b	$\leftarrow a \cdot b + b + r_{a1} + r_{a2} + r_{b1}$	$y_b = s_b + t_b$
$y_a = s_a + t_a$	$a + r_{a1} + r_{a2} + r_{b1} \rightarrow$	y_a
$y = y_a + y_b$		$y = y_a + y_b$

The view for $\mathcal{A}$ is then,

$$a, (r_{a1}, r_{a2}), r_{b1}, (a \cdot b + b + r_{a1} + r_{a2} + r_{b1})$$

and for $\mathcal{B}$,

$$b, r_{b1}, r_{a1}, (a \cdot b + r_{a2}), (a + r_{a1} + r_{a2} + r_{b1})$$

If $a = 0$ then there can be no privacy for $\mathcal{B}$. However, if $a = 1$ then the view of $\mathcal{A}$ reduces to:

$$1, (r_{a1}, r_{a2}), r_{b1}, (r_{a1} + r_{a2} + r_{b1})$$

which obviously discloses nothing about b, and is easily simulated.

If $b = 0$ then there can ben no privacy for $\mathcal{A}$. However, if $b = 1$ then the view of $\mathcal{B}$ reduces to:

$$\begin{aligned} &1, r_{b1}, r_{a1}, (a + r_{a2}), ((a + r_{a2}) + r_{a1} + r_{b1}) \\ &= 1, r_{b1}, r_{a1}, r', (r' + r_{a1} + r_{b1}) \end{aligned}$$

which obviously discloses nothing about a and is easily simulated.