CMSC 858K — Introduction to Secure Computation November 6, 2013

Lecture 26

Lecturer: Guest Lecture from Aseem Rastogi *Scribe(s): Xiao Wang*

1 Summary

This is a guest lecture about knowledge inference for optimizing secure multiparty computation[1]

Given an Secure Multiparty Computation program $\mathcal{S}$, we want to know for each party P, which program variables they "know".

Motivation: Given this knowledge, we can optimize Secure Multiparty computation to be more efficient.

- Knowledge Inference Algorithm: For each Party, it outputs which variable they know.
- constructive knowledge Inference Algorithm: For each Party, it outputs which variable they know, and a program that can generate it using party's input and output.

2 Example: 2 party 2 input median

We assume that party A and B owns x_1, x_2, y_1, y_2 respectively, s.t. $x_1 < x_2, y_1 < y_2$. The binary-search-like code is as follows:

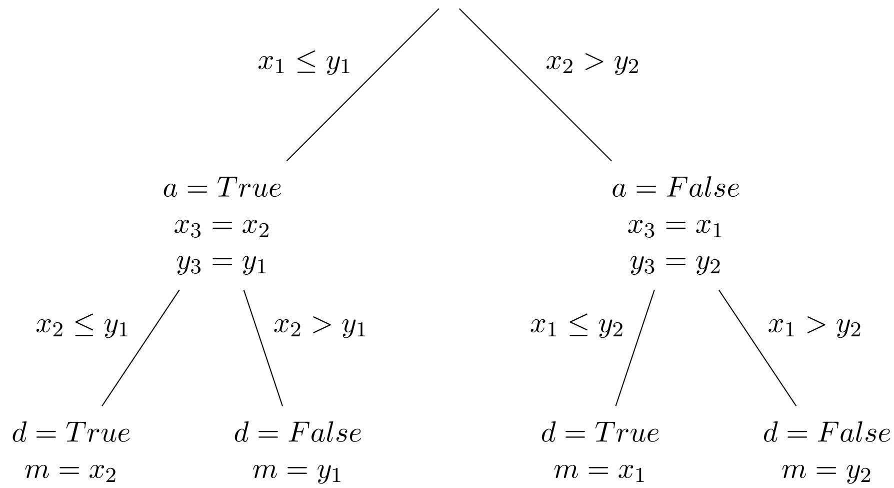
Algorithm 1: $median(x_1, x_2, y_1, y_2)$

bool $a = x_1 \leq y_1$
int $x_3 = a?x_2 : x_1$
int $y_3 = a?y_1 : y_2$
bool $d = x_3 \leq y_x$
int $m = d?x_3 : y_3$
return m

Traditional way of doing Secure Computation will transform the whole piece of code into Garbled-Circuits or GMW, which is quiet large. However, in this particular case we do not need to do secure computation for all code.

Claim 1 *Given x_1, x_2, m, Alice can always infer values of a and d independent of y_1, y_2. Similarly, given y_1, y_2, m, Bob can always infer values of a and d independent of x_1, x_2.*

This can be verified in the following tree:



Each run of the program will take one of the path of the tree. we can see that for bob, $d = (m \neq y_1) \wedge (m \neq y_2)$, $a = m \leq y_1$

3 In general

Let $\mathcal{S}$ be a program, y be a variable. Party A knows y if value of y only depends on its inputs and outputs. In other word, for any two program runs R_1, R_2, the coincidence on A's input and output implies the coincidence of y.

We say a is knows if:

$$(x_1 = x_1') \wedge (x_2 = x_2') \wedge (m = m') \rightarrow (a = a')$$

We feed

$$\Phi_{pre} \wedge (\bigvee_i \Phi_i) \wedge (x_1 = x_1') \wedge (x_2 = x_2') \wedge (m = m') \rightarrow (a = a')$$

into SMT to see if this is valid, where Φ_{pre} are some pre-conditions, Φ_i is one of the path condition, which is a set of predicates relating the program variables.

References

[1] Rastogi, Aseem, Piotr Mardziel, Michael Hicks, and Matthew A. Hammer. "Knowledge inference for optimizing secure multi-party computation." *In Proceedings of the Eighth ACM SIGPLAN workshop on Programming languages and analysis for security*, pp. 3-14. ACM, 2013.