Cryptography-B coursework: Paper review

Manan Vaswani

January 29, 2019

1 Introduction

At the CRYPTO2012 conference, Mike Rosulek raised an interesting question regarding black-box constructions in Cryptography, which he discussed in his paper "Must you know the code of f to securely compute f?" [1]. This paper looks at the possibility of using functions without having to compute the actual code of such a function in Multi-Party Communications.

In their 1989 paper [2], Impagliazzo and Rudich asked the important question "When do black-box constructions actually exist?". Their results showed that for random permutations ...

Black-box constructions in cryptography are those that rely only on the input and output behaviour of their components without actually knowing the details and construction of the components. They are widely used in cryptography due to the fact that they are highly practical, efficient and modular. Secure Multi-Party Computation(MPC) allows mutually distrusting parties to compute a function f on its shared inputs. One non-black box step that is used in all secure MPC communications is the evaluation of this function f. The function is first expressed as a low-level circuit, and then evaluated gate by gate on the inputs provided. This means that the complexity of the communication protocol is directly dependent on the circuit complexity of the function. However, this step is unavoidable for most general purpose MPC, but the paper looks at exploring for which special purpose secure communication tasks it would be possible to have a true black box construction. In particular, the author defines the necessary conditions for these special MPC tasks and gives a proof to show why these use true black-box evaluations without actually computing the function in the blackbox.

Interestingly, this topic has not been as widely researched as one would expect due to ??

2 Main Results

2.1 Functionally Black Box Protocols

For a general-purpose MPC with a fixed functionality f, the protocol directly depends on f anyway, so the protocol could simply have the circuit for f hard-coded and use that every time. Instead, the author models a protocol as a pair of oracle machines that is instantiated with any functionality f that is taken from a much larger class of functionalities $\mathcal{C}$, and then emulates a functionality related to f. If this class of functionalities is particularly large, the protocol cannot construct the circuit representations of all the related functionalities. Hence, these protocols "do not know" the code of their target function. With this goal, he introduces the definition of a functionally-black box protocol.

Let $\mathcal{C}$ be a class of functions. $\mathcal{F}$ is an ideal functionality implemented as an oracle machine. A **functionally-black-box (FBB)** protocol for $\mathcal{F}^{\mathcal{C}}$ (i.e $\mathcal{F}$ instantiated with $\mathcal{C}$) is a pair of interactive oracle machines (π_A, π_B) if for all $f \in \mathcal{C}$, the protocol (π_A^f, π_B^f) is a secure protocol for the ideal functionality $\mathcal{F}^f$. Here, the protocol $\mathcal{F}^f$ *must* treat the function it uses as a black-box, as it could really be any function out of the large class of functions $\mathcal{C}$.

In the definition, the security condition is guaranteed separately for each instantiation of the protocol with the different fs. Hence, for an FBB protocol, the adversaries may have access to an explicit representation of the function f that the protocol is instantiated with, so there is no compromise on the security condition being observed, but the honest parties only use a black-box definition of the function. The intent of the definition above is to characterize the efficiency of the honest parties, without affecting any security conditions.

An observation is that the set $\mathcal{C}$ must not be learnable in the sense that the circuit representations of $f \in \mathcal{C}$ can be obtained by repeated interactions between the honest parties, or with an external oracle. Additionally, for all $f \in \mathcal{C}$, the domain size must be infinite, as for a constant-sized domain, $\mathcal{C}$ would be learnable by exhaustively querying the functions. If the class of functions $\mathcal{C}$ was learnable, the protocol could simply obtain the code for the functions and construct their circuits to carry out the task in a non-blackbox manner. This is precisely what we are trying to avoid, hence this condition of non-learnability is necessary.

2.2 2-Hiding Autoreducible

Now the class of functions $\mathcal{C}$ in the definition of FBB protocols above is first spoken about rather arbitrarily. If the definition for every FBB protocol was feasible and secure for any and every $\mathcal{C}$, the paper would almost be trivial and this would mean that every single MPC protocol is truly blackbox which is rather absurd! The author describes a

classification of such classes of functions, using the notion of autoreducibility and then shows that this classification is a sufficient condition to characterise the feasibility of MPC FBB protocols.

A class of functions is said to be autoreducible if there exists some oracle machine M such that $M^f(x) = f(x)$, and the oracle queries made by the machine are independent of x cite here. Rosulek introduces a variant of this called 2-hiding autoreducible. The notion of **2-hiding autoreducibility** requires that there exists an oracle machine M such that $M^f(x, y) = f(x, y)$ and additionally, half the oracle queries are independent of y (called type-1 queries), and the other half are independent of x (type-2 queries). This additional condition in the definition is necessary as it prevents the machine from simply querying the oracle on its input (x, y) and returning the result of this query as its output.

2.3 Theorem

3 Proof Outline

4 Positive Example

5 Negative Example

6 Results for Malicious Security

7 Zero-Knowledge Proofs

8 Related Works

[3]

References

[1] Mike Rosulek. Must you know the code of f to securely compute f? In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 87–104. Springer, Heidelberg, August 2012.

[2] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[3] Yuval Ishai, Eyal Kushilevitz, Manoj Prabhakaran, Amit Sahai, and Ching-Hua Yu. Secure protocol transformations. In Matthew Robshaw and Jonathan Katz, editors,

CRYPTO 2016, Part II, volume 9815 of *LNCS*, pages 430–458. Springer, Heidelberg, August 2016.