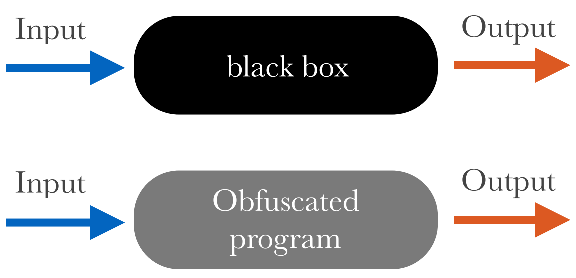
THIJMEN JESSEN CHRISTIAN SCHAFFNER MALVIN GATTINGER

Obfuscation

Presentation hand-out

Various commercial software companies have engineered techniques to manipulate computer programs in such a way that it will be harder for hackers to understand how the program works, while still having the program perform the same function. However, in practice these commercial “obfuscators" offer a mere speed bump, not nearly close to the security of the formally proven cryptographic schemes that we have seen throughout our course. Informally, an obfuscator *O* is an (efficient, probabilistic) “compiler” that takes as input a program (or circuit) *P* and produces a new program *O(P)* that has the same functionality as *P* yet is “unintelligible” in some sense[1]. On this topic, Barak, Boaz, et al. (2001) have concluded that obfuscating programs in the strongest sense of the definition is impossible. Future directions by Barak have been used by Garg (2013) to show that the weaker notions of indistinguishable obfuscation and functional encryption might be possible to construct for all circuits[2].

To understand what it means for a program to be successfully obfuscated, or unintelligible, consider the following thought experiment. Say that Alice wants to make use of a certain program (*P*) that runs from a black box. Alice can only write an input on a piece of paper, feed the paper to the black box, and receive an output on a piece of paper. In this way, Alice does not learn anything more about the functioning of the program than could be obtained from the input and output notes. Bob, in turn, has an obfuscated version of the same program *O(P)* running on his computer. We say that *O* is a successful obfuscater, if Bob learns nothing more about the program *P* from running it than Alice does. Please note that an important feature of this ‘virtual’ black box paradigm, is that the goal is not to prevent Bob from obtaining any knowledge about the program while running it, just that Bob doesn’t



[1] Barak, Boaz, et al. "On the (im) possibility of obfuscating programs." *Advances in Cryptology—CRYPTO 2001*. Springer Berlin Heidelberg, 2001.

[2] Garg, Sanjam, et al. "Candidate indistinguishability obfuscation and functional encryption for all circuits." *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013.

get more information than Alice, who has only Oracle access to the program.

In 2001, Barak, Boaz, et al. worked on a theoretical investigation of obfuscation. They came to the conclusion that obfuscation in the above mentioned 'virtual' black box paradigm is, even under very weak formalisations, impossible. They constructed a family of efficient programs $\mathcal{P}$ that were found to be unobfuscatable. They showed that, given any efficient program P' computing the same function as a program $P \in \mathcal{P}$, the "source code" of P can be efficiently reconstructed. However, given only oracle access to a program $P \in \mathcal{P}$, no efficient algorithm can reconstruct P, except with negligible probability. They went on to show that obfuscation remained impossible even when including obfuscators that (i) are not necessarily computable in polynomial time, (ii) only approximately preserve the functionality, and (iii) only need to work for very restricted models of computation. After these disappointing conclusions the paper does suggest that indistinguishability (or differing-input) obfuscators were not ruled out by the impossibility result.

This possibility was exploited by Garg (2013), constructing indeed indistinguishability obfuscators by transforming a computer program into a "multilinear jigsaw puzzle". All the pieces of the program get obfuscated by adding random elements that are chosen in such a way that running the garbled program in the intended way cancels out the randomness and computes the correct output. However, the randomness makes sure that trying to do anything else with the program makes each individual puzzle piece look meaningless. According to the paper, their scheme is unbreakable, provided that a problem concerning lattices[3] is as hard to solve as the team thinks it is. At this point it has already been proved that the most natural types of attacks on this system are guaranteed to fail[4]. Although the hard lattice problem is fairly new, it is closely related to a family of hard problems that have been around longer and have stood up to testing. The hope is that indistinguishability obfuscation in the future can be based on more conventional cryptographic assumptions. Cryptographers have already jumped on this hot-topic, exploring ways to make the scheme more efficient, construction stronger security assumptions, and further discover just many more possible applications.

[3] For further information on the topic: Damien Stehlé and Ron Steinfeld. *Making ntru as secure as worst-case problems over ideal lattices*. p. 27–47, 2011.

[4] Barak, Boaz, et al. "Protecting obfuscation against algebraic attacks." *Advances in Cryptology*. Springer Berlin Heidelberg, 2014. 221-238.