DOI:10.1145/3632568

To view the accompanying paper, visit doi.acm.org/10.1145/3611095

Technical Perspective
Hiding Secrets in Programs

By Daniel Wichs

CAN WE CREATE computer programs that do not reveal anything about their inner workings? This is the goal of program obfuscation. An obfuscator is a compiler that transforms a program into an obfuscated version that, when executed, has the same functionality as the original but is completely inscrutable otherwise, hiding all internal aspects of the original implementation.

Program obfuscation holds immense promise. It can protect intellectual property by preventing others from reverse engineering software and stealing or modifying the underlying ideas. It also allows us to securely hide secrets inside programs. To see how this could work, consider a program that contains a treasure map, but only outputs it when given a formal proof of the Riemann hypothesis as an input. The treasure map is hard-coded inside the program and may be easy to extract from the original code, but by releasing an obfuscated version of this program, we would ensure nobody can recover the treasure map *unless* they have a proof of the Riemann hypothesis. It turns out a variant of this idea yields public-key encryption: Alice can encrypt a secret message to Bob by obfuscating the program that only outputs the message when given Bob's correct secret key as an input.

In fact, program obfuscation has the potential to revolutionize cryptography. Not only would it give us a unified way to realize essentially all the cryptographic tools, such as public-key encryption, which were painstakingly developed over the last 50 years, but it would enable many amazing new applications as well. For example, we could give an email server the ability to recognize whether encrypted messages are spam without giving it the ability to read them, by giving it an obfuscated program that decrypts the email, checks if it is spam and only outputs the result, but does not reveal anything else.

Considering that program obfuscation is so powerful, can we achieve it? There is a long history of ad-hoc attempts to make reverse engineering more difficult, but most can be broken by a sufficiently clever attacker. The first rigorous treatment of program obfuscation was given by Barak et al. in 2001.[1] They proposed a security notion called *virtual black box* (VBB), which ensures an obfuscated program does not reveal anything more than black-box executions of the program. Unfortunately, they showed that VBB security is unachievable in general for all programs. The result led to widespread pessimism and research on obfuscation largely stalled for the next decade.

In the same paper, Barak et al. also discussed an alternate security notion called *indistinguishability obfuscation* (iO). It guarantees that the obfuscations of any two programs with the same functionality are indistinguishable from each other, meaning that the original implementation is hidden from among all possible implementations of the same functionality. Their impossibility result does not extend to iO and they left it as an open problem whether iO is achievable. It also remained unclear whether iO is meaningful and sufficient for interesting applications. Without any evidence of either feasibility or usefulness, iO did not initially receive much attention.

This changed over a decade later with the work of Garg et al.,[2] who gave the first candidate construction of iO. Together with the work of Sahai and Waters,[3] they also showed how to use iO to realize several advanced cryptographic applications, which then opened the floodgates for more applications in subsequent works. Suddenly iO emerged as the most powerful tool in cryptography. But questions still lingered whether iO is possible.

The following paper finally places indistinguishability obfuscation on firm foundations.

Cryptography generally relies on unproven assumptions that certain problems (for example, factoring large integers) cannot be solved efficiently. But the candidate iO construction of Garg et al., as well as later candidates, all relied on new assumptions in which there was much less confidence. In fact, many of these ended up being broken, sowing serious doubts on whether the entire endeavor is doomed.

The following paper largely dispels these doubts and finally places iO on firm foundations. It gives a new construction of iO that is provably secure under assumptions that have been extensively studied and withstood the test of time. While we cannot rule out the possibility that these assumptions could be broken in the future, such a break would constitute a major surprising development in the field. The paper builds on prior work showing how to construct iO from simpler components, but these still appeared out of reach. The authors find the right component to target (a special type of pseudorandom generator) and provide a novel approach to realizing it. By giving the first strong evidence that iO is achievable, this work paves the way for future research toward efficient and practical constructions.

References

1. Barak, B. et al. On the (im)possibility of obfuscating programs. *CRYPTO 2001, 2139*. LNCS, Springer, Heidelberg, 1–18.
2. Garg, S. et al. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 54th FOCS*. IEEE Computer Society Press, Oct. 2013, 40–49.
3. Sahai, A. and Waters, B. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th STOC*. ACM Press, May/June 2014, 475–484.

Daniel Wichs is an associate professor in the Khoury College of Computer Sciences at Northeastern University, Boston, MA, USA.