Computational problems whose input is a program are central in Cryptography, as well as Complexity, Learning, and Optimization. The nature of such problems crucially depends on the way the program is accessed -- as a black box or explicitly by its implementation.

In which settings can we exploit code to gain an advantage over black-box access? In Cryptography, we explore this question from two opposing perspectives:

Protecting Code: Can we obfuscate a program's code so that its functionality is preserved but it is otherwise unintelligible? Intuitively, such obfuscated code reveals nothing more than black-box access to the program. Obfuscation is, therefore, a powerful tool with numerous applications in software protection and Cryptography.

Exploiting Code: Most security proofs in cryptography consist of a reduction that translates any potential attacker into an algorithm solving some underlying hard problem. While most security reductions only require black-box access to the attacker, for many applications black-box reductions are provably insufficient. Can we exploit the attacker's code to prove security where black-box reductions fail?

In this talk, I will describe new techniques for protecting and exploiting code, taking advantage of the inherent tension between these two tasks. I will also demonstrate applications of these techniques in and beyond cryptography.