There are two main types of secure and verifiable computation: *outsourced computation* and *multi-party computation*. Our focus is on multi-party computation, but first we briefly describe outsourced computation to distinguish it from multi-party computation.

1.1 Outsourced Computation

In an outsourced computation, one party owns the data and wants to be able to obtain the result of computation on that data. The second party receives and stores the data in an encrypted form, performs computation on the encrypted data, and provides the encrypted results to the data owner, without learning anything about the input data, intermediate values, or final result. The data owner can then decrypt the returned results to obtain the output.

Homomorphic encryption allows operations on encrypted data, and is a natural primitive to implement outsourced computation. With *partially-homomorphic encryption* schemes, only certain operations can be performed. Several efficient partially-homomorphic encryption schemes are known (Paillier, 1999; Naccache and Stern, 1998; Boneh *et al.*, 2005). Systems built on them are limited to specialized problems that can be framed in terms of the supported operations.

To provide *fully homomorphic encryption* (FHE), it is necessary to support a Turing-complete set of operations (e.g., both addition and multiplication) so that any function can be computed. Although the goal of FHE was envisioned by Rivest *et al.* (1978), it took more than 30 years before the first FHE scheme was proposed by Gentry (2009), building on lattice-based cryptography. Although there has been much recent interest in implementing FHE schemes Gentry and Halevi (2011), Halevi and Shoup (2014), and Chillotti *et al.* (2016), building secure, deployable, scalable systems using FHE remains an elusive goal.

In their basic forms, FHE and MPC address different aspects of MPC, and as such shouldn't be directly compared. They do, however, provide similar functionalities, and there are ways to adapt FHE to use multiple keys that enables multi-party computation using FHE (Asharov *et al.*, 2012; López-Alt *et al.*, 2012; Mukherjee and Wichs, 2016). FHE offers an asymptotic communication improvement in comparison with MPC, but at the expense of computational efficiency. State-of-the-art FHE implementations (Chillotti *et al.*, 2017) are thousands of times slower than two-party and multi-party

secure computation in typical applications and settings considered in literature. Ultimately, the relative performance of FHE and MPC depends on the relative costs of computation and bandwidth. For high-bandwidth settings, such as where devices connected within a data center, MPC vastly outperforms FHE. As FHE techniques improve, and the relative cost of bandwidth over computation increases, FHE-based techniques may eventually become competitive with MPC for many applications.

We do not specifically consider outsourcing computation or FHE further in this book, but note that some of the techniques developed to improve multi-party computation also apply to FHE and outsourcing. Shan *et al.* (2017) provide a survey of work in the area of outsourcing.

1.2 Multi-Party Computation

The goal of secure multi-party computation (MPC) is to enable a group of independent data owners who do not trust each other or any common third party to jointly compute a function that depends on all of their private inputs. MPC differs from outsourced computation in that all of the protocol participants are data owners who participate in executing a protocol. Chapter 2 provides a more formal definition of MPC, and introduces the most commonly considered threat models.

Brief history of MPC. The idea of secure computation was introduced by Andrew Yao in the early 1980s (Yao, 1982). That paper introduced a general notion of secure computation, in which m parties want to jointly compute a function $f(x_1, x_2, \ldots, x_m)$ where x_i is the i^{th} party's private input. In a series of talks over the next few years (but not included in any formal publication), Yao introduced the Garbled Circuits Protocol which we describe in detail in Section 3.1. This protocol remains the basis for many of the most efficient MPC implementations.

Secure computation was primarily of only theoretical interest for the next twenty years; it was not until the 2000s that algorithmic improvements and computing costs had reached a point where it became realistic to think about building practical systems using general-purpose multi-party computation. Fairplay (Malkhi *et al.*, 2004) was the first notable implementation of a general-purpose secure computation system. Fairplay demonstrated the possibility that