key material among the two servers. Now, an attacker must compromise *both* S_1 and S_2 to gain access to the keys. We can run S_1 and S_2 on two different software stacks to minimize the chance that they will both be vulnerable to the exploit available to the malware, and operate them using two different sub-organizations to minimize insider threats. Of course, routine execution does need access to the keys to provide authentication service; at the same time, key should never be reconstructed as the reconstructing party will be the target of the APT attack. Instead, the three players, S_1, S_2, and the authenticating user U, will run the authentication inside MPC, *without ever reconstructing any secrets*, thus removing the singular vulnerability and hardening the defense.

1.4 Overview

Because MPC is a vibrant and active research area, it is possible to cover only a small fraction of the most important work in this book. We mainly discuss generic MPC techniques, focusing mostly on the two-party scenario, and emphasizing a setting where all but one of the parties may be corrupted. In the next chapter, we provide a formal definition of secure multi-party computation and introduce security models that are widely-used in MPC. Although we do not include formal security proofs in this book, it is essential to have clear definitions to understand the specific guarantees that MPC provides. Chapter 3 describes several fundamental MPC protocols, focusing on the most widely-used protocols that resist any number of corruptions. Chapter 4 surveys techniques that have been developed to enable efficient implementations of MPC protocols, and Chapter 5 describes methods that have been used to provide sub-linear memory abstractions for MPC.

Chapters 3–5 target the weak semi-honest adversary model for MPC (defined in Chapter 2), in which is it assumed that all parties follow the protocol as specified. In Chapter 6, we consider how MPC protocols can be hardened to provide security against active adversaries, and Chapter 7 explores some alternative threat models that enable trade-offs between security and efficiency. We conclude in Chapter 8, outlining the trajectory of MPC research and practice, and suggesting possible directions for the future.

2

Defining Multi-Party Computation

In this chapter, we introduce notations and conventions we will use throughout, define some basic cryptographic primitives, and provide a security definition for multi-party computation. Although we will not focus on formal security proofs or complete formal definitions, it is important to have clear security definitions to understand exactly what properties protocols are designed to provide. The protocols we discuss in later chapters have been proven secure based on these definitions.

2.1 Notations and Conventions

We will abbreviate *Secure Multi-Party Computation* as *MPC*, and will use it to denote secure computation among two or more participants. The term *secure function evaluation* (SFE) is often used to mean the same thing, although it can also apply to contexts where only one party provides inputs to a function that is evaluated by an outsourced server. Because two-party MPC is an important special case, which received a lot of targeted attention, and because two-party protocols are often significantly different from the general n-party case, we will use 2PC to emphasize this setting when needed.

We assume existence of direct secure channels between each pairs of