participating players. Such channels could be achieved inexpensively through a variety of means, and are out of scope in this book.

We denote encryption and decryption of a message m under key k as $\mathsf{Enc}_k(m)$ and $\mathsf{Dec}_k(m)$. We will refer to protocol participants interchangeably also as parties or players, and will usually denote them as P_1, P_2, etc. We will denote the adversary by $\mathcal{A}$.

A negligible function $\nu : \mathbb{N} \to \mathbb{R}$ is any function that approaches zero asymptotically faster than any inverse polynomial. In other words, for any polynomial p, $\nu(n) < 1/p(n)$ for all but finitely many n.

We will denote computational and statistical security parameters by κ and σ respectively. The computational security parameter κ governs the hardness of problems that can be broken by an adversary's offline computation — e.g., breaking an encryption scheme. In practice κ is typically set to a value like 128 or 256. Even when we consider security against computationally bounded adversaries, there may be some attacks against an interactive protocol that are not made easier by offline computation. For example, the interactive nature of a protocol may give the adversary only a single opportunity to violate security (e.g., by sending a message that has a special property, like predicting the random value that an honest party will chose in the next round). The statistical security parameter σ governs the hardness of these attacks. In practice, σ is typically set to a smaller value like 40 or 80. The correct way to interpret the presence of two security parameters is that security is violated only with probability $2^{-\sigma} + \nu(\kappa)$, where ν is a negligible function *that depends on the resources of the adversary.* When we consider computationally unbounded adversaries, we omit κ and require $\nu = 0$.

We will use symbol $\in_R$ to denote uniformly random sampling from a distribution. For example we write "choose $k \in_R \{0,1\}^\kappa$" to mean that k is a uniformly chosen κ-bit long string. More generally, we write "$v \in_R D$" to denote sampling according to a probability distribution D. Often the distribution in question is the output of a randomized algorithm. We write "$v \in_R A(x)$" to denote that v is the result of running randomized algorithm A on input x.

Let D_1 and D_2 be two probability distributions indexed by a security parameter, or equivalently two algorithms that each take a security parameter as input.[1] We say that D_1 and D_2 are *indistinguishable* if, for all algorithms A

[1] In the literature, D_1 and D_2 are often referred to as an *ensemble* of distributions.

there exists a negligible function ν such that:

$$\Pr[A(D_1(n)) = 1] - \Pr[A(D_2(n)) = 1] \leq \nu(n)$$

In other words, no algorithm behaves more than negligibly differently when given inputs sampled according to D_1 vs D_2. When we consider only *non-uniform, polynomial-time* algorithms A, the definition results in *computational indistinguishability*. When we consider *all* algorithms without regard to their computational complexity, we get a definition of *statistical indistinguishability*. In that case, the probability above is bounded by the *statistical distance* (also known as total variation distance) of the two distributions, which is defined as:

$$\Delta(D_1(n), D_2(n)) = \frac{1}{2} \sum_x \left| \Pr[x = D_1(n)] - \Pr[x = D_2(n)] \right|$$

Throughout this work, we use *computational security* to refer to security against adversaries implemented by non-uniform, polynomial-time algorithms. We use *information-theoretic security* (also known as *unconditional* or *statistical* security) to mean security against arbitrary adversaries (even those with unbounded computational resources).

2.2 Basic Primitives

Here, we provide definitions of a few basic primitives we use in our presentation. Several other useful primitives are actually special cases of MPC (i.e., they are defined as MPC of specific functions). These are defined in Section 2.4.

Secret Sharing. Secret sharing is another essential primitive, which is at the core of many MPC approaches. Informally, a (t, n)-secret sharing scheme splits the secret s into n shares, such that any $t - 1$ of the shares reveal no information about s, while any t shares allow complete reconstruction of the secret s. There are many variants of possible security properties of secret sharing schemes; we provide one definition, adapted from Beimel and Chor (1993), next.

Definition 2.1. Let D be the domain of secrets and D_1 be the domain of shares. Let $\mathsf{Shr} : D \to D_1^n$ be a (possibly randomized) sharing algorithm, and $\mathsf{Rec} : D_1^k \to D$ be a reconstruction algorithm. A (t, n)-secret sharing scheme is a pair of algorithms $(\mathsf{Shr}, \mathsf{Rec})$ that satisfies these two properties: