Apart from speed of execution and the nondeterministic nature of **RNGCryptoServiceProvider**, the other difference between this and **System.Random** is that **RNGCryptoServiceProvider** is thread safe and **System.Random** is not.

Figure 1: Randomly generated numbers using RNGCryptoServiceProvider

In the sample code, we generate 10 sets of 32-byte random numbers by using the **RNGCryptoServiceProvider** and write the results to the console window. The **GenerateRandomNumber** method returns a byte array that is the same size as the parameter that you pass into the method. When we output the random number onto the screen, we first convert it to a Base64 string. This is common practice when displaying the results of a cryptographic operation on the screen.

Note: Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. The term Base64 originates from a specific MIME content transfer encoding.

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine("Random Number " + i + " : "
        + Convert.ToBase64String(Random.GenerateRandomNumber(32)));
}
```

Code Listing 2

If you need to convert from a Base64 encoded string back into a byte array, you call **Convert.FromBase64String**.

Chapter 3 Hashing Algorithms

Hashing Functions

A cryptographic hash function is an algorithm that takes an arbitrary block of data and returns a fixed-size string, the (cryptographic) hash value, such that any (accidental or intentional) change to the data will change the hash value. The data to be encoded is often called the “message” and the hash value is sometimes called the “message digest” or, simply, the “digest.”

The ideal cryptographic hash function has four main properties:

- It is easy to compute the hash value for any given message.
- It is infeasible to generate a message that has a given hash.
- It is infeasible to modify a message without changing the hash.
- It is infeasible to find two different messages with the same hash.

Another way of thinking of a hash function is that of creating a unique fingerprint of a piece of data. Generating a hash digest of a block of data is easy to do in .NET. There are various algorithms you can use such as MD5, SHA-1, SHA-256, and SHA-512.

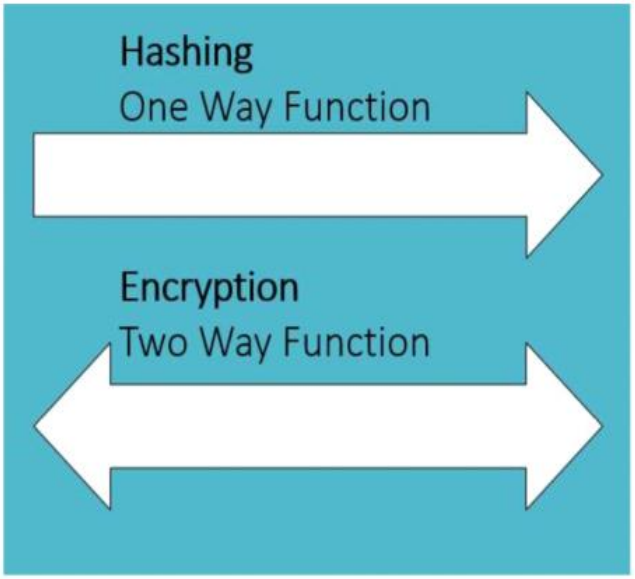

Figure 2: Hashing functions versus encryption functions

As illustrated in the preceding diagram, a hash function is a one-way function. That means once you have hashed some data, you cannot reverse it back to the original data. On the flip side to this, encryption is designed to be a two-way operation. Once you have encrypted some data by using a key, you can reverse the operation and decrypt the data by using the same key.