Chapter 6
Sigma Protocols and Efficient Zero-Knowledge

A zero-knowledge proof is an interactive proof with the additional property that the verifier learns nothing beyond the correctness of the statement being proved. The theory of zero-knowledge proofs is beautiful and rich, and is a cornerstone of the foundations of cryptography. In the context of cryptographic protocols, zero-knowledge proofs can be used to enforce “good behavior” by having parties prove that they indeed followed the protocol correctly. These proofs must reveal nothing about the parties’ private inputs, and as such must be zero knowledge. Zero-knowledge proofs are often considered an expensive (and somewhat naive) way of enforcing honest behavior, and those who view them in this way consider them to be not very useful when constructing efficient protocols. Although this is true for arbitrary zero-knowledge proofs of $\mathcal{NP}$ statements, there are many languages of interest for which there are extraordinarily efficient zero-knowledge proofs. Indeed, in many cases an efficient zero-knowledge proof is the best way to ensure that malicious parties do not cheat. We will see some good examples of this in Chapter 7 where efficient zero-knowledge proofs are used to achieve oblivious transfer with very low cost.

In this chapter, we assume familiarity with the notions of zero-knowledge and honest verifier zero-knowledge, and refer readers to [30, Chapter 4] for the necessary theoretical background, and for definitions and examples. As everywhere else in this book, here we are interested in achieving efficiency.

6.1 An Example

We begin by motivating the notion of a $\Sigma$-protocol through an example. Let $p$ be a prime, $q$ a prime divisor of $p - 1$, and $g$ an element of order $q$ in $\mathbb{Z}_p^*$. Suppose that a prover $P$ has chosen a random $w \leftarrow \mathcal{R} \mathbb{Z}_q$ and has published $h = g^w \bmod p$. A verifier $V$ who receives $(p, q, g, h)$ can efficiently

---

1 Much of this chapter is based on a survey paper by Ivan Damgård [20]. We thank him for graciously allowing us to use his text.
verify that \( p, q \) are prime, and that \( g, h \) both have order \( q \). Since there is only one subgroup of order \( q \) in \( Z_p^* \), this automatically means that \( h \in \langle g \rangle \) and thus there always exists a value \( w \) such that \( h = g^w \) (this holds because in a group of prime order all elements apart from the identity are generators and so \( g \) is a generator). However, this does not necessarily mean that \( P \) knows \( w \).

Protocol 6.1.1 by Schnorr provides a very efficient way for \( P \) to convince \( V \) that it knows the unique value \( w \in \mathbb{Z}_q \) such that \( h = g^w \mod p \):

\[
\text{PROTOCOL 6.1.1 (Schnorr’s Protocol for Discrete Log)}
\]

- **Common input:** The prover \( P \) and verifier \( V \) both have \((p, q, g, h)\).
- **Private input:** \( P \) has a value \( w \in \mathbb{Z}_q \) such that \( h = g^w \mod p \).
- **The protocol:**
  1. The prover \( P \) chooses a random \( r \leftarrow R \mathbb{Z}_q \) and sends \( a = g^r \mod p \) to \( V \).
  2. \( V \) chooses a random challenge \( e \leftarrow R \{0, 1\}^t \) and sends it to \( P \), where \( t \) is fixed and it holds that \( 2^t < q \).
  3. \( P \) sends \( z = r + ew \mod q \) to \( V \), which checks that \( g^z = ah^e \mod p \), that \( p, q \) are prime and that \( g, h \) have order \( q \), and accepts if and only if this is the case.

Intuitively, this is a proof of knowledge because if some \( P^* \), having sent \( a \), can answer two different challenges \( e, e' \) correctly, then this means that it could produce \( z, z' \) such that \( g^z = ah^e \mod p \) and also \( g^{z'} = ah^{e'} \mod p \). Dividing one equation by the other, we have that \( g^{z-z'} = h^{e-e'} \mod p \). Now, by the assumption, \( e - e' \neq 0 \mod q \) (otherwise \( e \) and \( e' \) are not different challenges), and so it has a multiplicative inverse modulo \( q \). Since \( g, h \) have order \( q \), by raising both sides to this power, we get \( h = g^{(z-z')(e-e')^{-1}} \mod p \), and so \( w = (z - z')(e - e')^{-1} \mod q \). Observing that \( z, z', e, e' \) are all known to the prover, we have that the prover itself can compute \( w \) and thus knows the required value (except with probability \( 2^{-t} \), which is the probability of answering correctly with a random guess). Thus, Protocol 6.1.1 is a proof of knowledge; we prove this formally below in Section 6.3.

In contrast, Protocol 6.1.1 is not known to be zero-knowledge. In order to see this, observe that in order for the problem of finding \( w \) to be non-trivial in the first place, \( q \) must be (exponentially) large. Furthermore, to achieve negligible error in a single run of the protocol, \( 2^t \) must be exponentially large too. In this case, standard rewinding techniques for zero-knowledge simulation will fail because it becomes too hard for a simulator to guess the value of \( e \) in advance. It is therefore not known if there exists some efficient malicious strategy that the verifier may follow which, perhaps after many executions of the protocol, enables it to obtain \( w \) (or learn more than is possible from \((p, q, g, h)\) alone).

On the positive side, the protocol is honest verifier zero-knowledge. To simulate the view of an honest verifier \( V \), simply choose a random \( z \leftarrow R \mathbb{Z}_q \).