When using the elliptic curve P256, both p and q are 256-bit primes. An ECDSA signature $\sigma = (r, s)$ is then 512 bits long.

A straightforward calculation shows that the scheme is correct: for every key pair (pk, sk) output by G, and every message $m \in \mathcal{M}$, if $\sigma \xleftarrow{R} S(sk, m)$ then $V(pk, m, \sigma)$ outputs accept. The reason is that $\hat{u}_t$ computed by V is the same as u_t computed by S.

The security of this scheme has only been established somewhat heuristically, specifically, when we model $\mathbb{G}$ as a "generic group" (see Section 16.3).

For security, it is important that the random value α_t generated during signing be a fresh uniform value in $\mathbb{Z}_q^*$. Otherwise the scheme can become insecure in a strong sense: an attacker can learn the secret signing key α. This was used in a successful attack on the Sony PlayStation 3 because α_t was the same for all issued signatures. It has also lead to attacks on some Bitcoin wallets [48]. Because generating randomness on some hardware platforms can be difficult, a common solution is to modify the signing algorithm so that α is generated deterministically using a secure PRF, as described in Exercise 13.6. This variant is called **deterministic ECDSA**. The Schnorr signature scheme suffers from the same issue and this modification applies equally well to it.

ECDSA is not strongly secure. While the Schnorr signature scheme is strongly secure (see Exercise 19.17), the ECDSA scheme is not. Given an ECDSA signature $\sigma = (r, s)$ on a message m, anyone can generate more signatures on m. For example, $\sigma' := (r, -s) \in (\mathbb{Z}_q^*)^2$ is another valid signature on m. This σ' is valid because the x-coordinate of the elliptic curve point $u_t \in \mathbb{G}$ is the same as the x-coordinate of the point $1/u_t \in \mathbb{G}$.

19.4 Sigma protocols: basic definitions

Schnorr's identification protocol is a special case of an incredibly useful class of protocols called **Sigma protocols**. In this section, we will introduce the basic concepts associated with Sigma protocols. Later, we will consider many examples of Sigma protocols and their applications:

- We will see how we can use Sigma protocols to build new secure identification schemes and signature schemes.

- We will see how to build identification schemes that we can prove (without the random oracle heuristic) are secure against *active* attacks. Recall that for Schnorr's identification protocol we could only prove security against eavesdropping attacks.

- In the next chapter, we will also see how to use Sigma protocols for other applications that have nothing to do with identification and signatures. For example, we will see how one can encrypt a message m and then "prove" to a skeptical verifier that m satisfies certain properties, without revealing to the verifier anything else about m. We will illustrate this idea with an electronic voting protocol.

Consider again Schnorr's identification protocol. Intuitively, that protocol allows a prover P to convince a skeptical verifier V that he knows a secret that satisfies some relation, without revealing any useful information to V about the secret. For Schnorr's protocol, the prover's secret was $\alpha \in \mathbb{Z}_q$ satisfying the relation $g^\alpha = u$.

We can generalize this to more general and interesting types of relations.

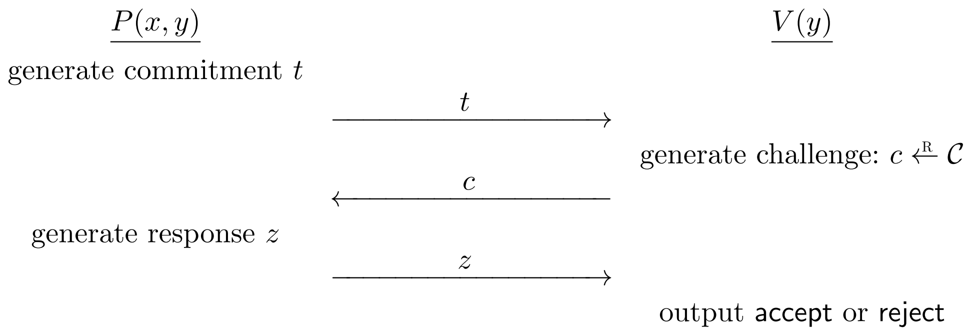

Figure 19.5: Execution of a Sigma protocol

Definition 19.2 (Effective relation). *An **effective relation** is a binary relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{R}$ are efficiently recognizable finite sets. Elements of $\mathcal{Y}$ are called **statements**. If $(x, y) \in \mathcal{R}$, then x is called a **witness for** y.*

We now define the syntax of a Sigma protocol.

Definition 19.3 (Sigma protocol). *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be an effective relation. A **Sigma protocol** for $\mathcal{R}$ is a pair (P, V).*

- *P is an interactive protocol algorithm called the **prover**, which takes as input a witness-statement pair $(x, y) \in \mathcal{R}$.*
- *V is an interactive protocol algorithm called the **verifier**, which takes as input a statement $y \in \mathcal{Y}$, and which outputs* accept *or* reject.
- *P and V are structured so that an interaction between them always works as follows:*
  - *To start the protocol, P computes a message t, called the **commitment**, and sends t to V;*
  - *Upon receiving P's commitment t, V chooses a **challenge** c at random from a finite **challenge space** $\mathcal{C}$, and sends c to P;*
  - *Upon receiving V's challenge c, P computes a **response** z, and sends z to V;*
  - *Upon receiving P's response z, V outputs either* accept *or* reject*, which must be computed strictly as a function of the statement y and the **conversation** (t, c, z). In particular, V does not make any random choices other than the selection of the challenge — all other computations are completely deterministic.*

We require that for all $(x, y) \in \mathcal{R}$, when $P(x, y)$ and $V(y)$ interact with each other, $V(y)$ always outputs accept.

See Fig. 19.5, which illustrates the execution of a Sigma protocol. The name Sigma protocol comes from the fact that the "shape" of the message flows in such a protocol is vaguely reminiscent of the shape of the Greek letter Σ.

As stated in the definition, we require that the verifier computes its output as a function of the statement y and its conversation (t, c, z) with the prover. If the output is accept we call the conversation (t, c, z) an **accepting conversation for** y. Of course, interactions between the verifier and an honest prover only produce accepting conversations; non-accepting conversation can arise, for example, if the verifier interacts with a "dishonest" prover that is not following the protocol.

In most applications of Sigma protocols, we will require that the size of the challenge space is super-poly. To state this requirement more succinctly, we will simply say that the protocol has a **large challenge space**.

Example 19.1. It should be clear that for Schnorr's identification protocol (G, P, V), the pair (P, V) is an example of a Sigma protocol for the relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$, where

$$\mathcal{X} = \mathbb{Z}_q, \quad \mathcal{Y} = \mathbb{G}, \quad \text{and} \quad \mathcal{R} = \{ \ (\alpha, u) \in \mathbb{Z}_q \times \mathbb{G} : \ g^\alpha = u \ \}.$$

The challenge space $\mathcal{C}$ is a subset of $\mathbb{Z}_q$. We call (P, V) **Schnorr's Sigma protocol**.

The reader should observe that unlike an identification protocol, a Sigma protocol itself does not specify an algorithm for generating elements of $\mathcal{R}$.

Note also that the relation $\mathcal{R}$ in this case is parameterized by a description of the group $\mathbb{G}$ (which includes its order q and the generator $g \in \mathbb{G}$). In general, we allow effective relations that are defined in terms of such "system parameters," which are assumed to be generated at system setup time, and publicly known to all parties.

A statement for Schnorr's Sigma protocol is a group element $u \in \mathbb{G}$, and a witness for u is $\alpha \in \mathbb{Z}_q$ such that $g^\alpha = u$. Thus, every statement has a unique witness. An accepting conversation for u is a triple of the form $(u_{\rm t}, c, \alpha_{\rm z})$, with $u_{\rm t} \in \mathbb{G}$, $c \in \mathcal{C}$, and $\alpha_{\rm z} \in \mathbb{Z}_q$, that satisfies the equation

$$g^{\alpha_{\rm z}} = u_{\rm t} \cdot u^c.$$

The reader may have noticed that, as we have defined it, the prover P from Schnorr's identification protocol takes as input just the witness α, rather than the witness/statement pair (α, u), as formally required in our definition of a Sigma protocol. In fact, in this and many other examples of Sigma protocols, the prover does not actually use the statement explicitly in its computation. □

19.4.1 Special soundness

We next define a critical security property for Sigma protocols, which is called **special soundness**.

Definition 19.4 (Special soundness). *Let (P, V) be a Sigma protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. We say that (P, V) provides **special soundness** if there is an efficient deterministic algorithm Ext, called a **witness extractor**, with the following property: whenever Ext is given as input a statement $y \in \mathcal{Y}$, and two accepting conversations (t, c, z) and (t, c', z'), with $c \neq c'$, algorithm Ext always outputs $x \in \mathcal{X}$ such that $(x, y) \in \mathcal{R}$ (i.e., x is a witness for y).*

Example 19.2. Continuing with Example 19.1, we can easily verify that Schnorr's Sigma protocol provides special soundness. The witness extractor takes as input the statement $u \in \mathbb{G}$, along with two accepting conversations $(u_{\rm t}, c, \alpha_{\rm z})$ and $(u_{\rm t}, c', \alpha'_{\rm z})$ for u, with $c \neq c'$. Just as we did in the proof of Theorem 19.1, we can compute the corresponding witness $\alpha = \mathsf{Dlog}_g u$ from these two conversations as $\Delta\alpha / \Delta c \in \mathbb{Z}_q$, where $\Delta\alpha := \alpha_{\rm z} - \alpha'_{\rm z}$ and $\Delta c := c - c'$. □

Suppose (P, V) is a Sigma protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. Moreover, suppose (P, V) provides special soundness and has a large challenge space. Then in a certain sense, (P, V) acts as a "proof of knowledge." Indeed, consider an arbitrary prover P^* (even a potentially "cheating" one) that makes V accept a statement y with non-negligible probability. Then P^* must "know" a witness for y, in the following sense: just as in the proof of Theorem 19.1, we can rewind P^* to get two accepting conversations (t, c, z) and (t, c', z') for y, with $c \neq c'$, and then use the witness extractor to compute the witness x.

More generally, when a cryptographer says that P^* must "know" a witness for a statement y, what she means is that the witness can be extracted from P^* using rewinding. Although we will not formally define the notion of a "proof of knowledge," we will apply special soundness in several applications.

19.4.2 Special honest verifier zero knowledge

We introduced the notion of honest verifier zero knowledge (HVZK) in Section 19.1.1 for identification protocols. We can easily adapt this notion to the context of Sigma protocols.

Let (P, V) be a Sigma protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. Intuitively, what we want to say is that for $(x, y) \in \mathcal{R}$, a conversation between $P(x, y)$ and $V(y)$ should not reveal anything about the witness x. Just as in Section 19.1.1, we will formalize this intuition by saying that we can efficiently simulate conversations between $P(x, y)$ and $V(y)$ without knowing the witness x. However, we will add a few extra requirements, which will streamline some constructions and applications.

Definition 19.5 (Special HVZK). *Let (P, V) be a Sigma protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ with challenge space $\mathcal{C}$. We say that (P, V) is **special honest verifier zero knowledge**, or **special HVZK**, if there exists an efficient probabilistic algorithm Sim (called a **simulator**) that takes as input $(y, c) \in \mathcal{Y} \times \mathcal{C}$, and satisfies the following properties:*

(i) for all inputs $(y, c) \in \mathcal{Y} \times \mathcal{C}$, algorithm Sim always outputs a pair (t, z) such that (t, c, z) is an accepting conversation for y;

(ii) for all $(x, y) \in \mathcal{R}$, if we compute

$$c \xleftarrow{R} \mathcal{C}, \;\; (t, z) \xleftarrow{R} Sim(y, c),$$

then (t, c, z) has the same distribution as that of a transcript of a conversation between $P(x, y)$ and $V(y)$.

The reader should take note of a couple of features of this definition. First, the simulator takes the challenge c as an additional input. Second, it is required that the simulator produce an accepting conversation even when the statement y does not have a witness. These two properties are the reason for the word "special" in "special HVZK."

Example 19.3. Continuing with Example 19.2, we can easily verify that Schnorr's Sigma protocol is special HVZK. Indeed, the simulator in the proof of Theorem 19.4 is easily adapted to the present setting. On input $u \in \mathbb{G}$ and $c \in \mathcal{C}$, the simulator computes

$$\alpha_z \xleftarrow{R} \mathbb{Z}_q, \;\; u_t \leftarrow g^{\alpha_z}/u^c,$$

and outputs the pair (u_t, α_z). We leave it to the reader to verify that this simulator satisfies all the requirements of Definition 19.5. $\Box$