or

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1] \leq q(n) \cdot \Pr[\mathsf{Ident}_{\mathcal{A},\Pi}(n) = 1] + \mathsf{negl}(n).$$

If Π is secure then $\Pr[\mathsf{Ident}_{\mathcal{A},\Pi}(n) = 1]$ is negligible; since $q(n)$ is polynomial this implies that $\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1]$ is also negligible. Because $\mathcal{A}'$ was arbitrary, this means Π' is secure. ■

13.5.2 The Schnorr Identification/Signature Schemes

The Schnorr identification scheme is based on hardness of the discrete-logarithm problem. Let $\mathcal{G}$ be a polynomial-time algorithm that takes as input 1^n and (except possibly with negligible probability) outputs a description of a cyclic group $\mathbb{G}$, its order q (with $\|q\| = n$), and a generator g. To generate its keys, the prover runs $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, chooses a uniform $x \in \mathbb{Z}_q$, and sets $y := g^x$; the public key is $\langle \mathbb{G}, q, g, y \rangle$ and the private key is x. To execute the protocol (see Figure 13.2), the prover begins by choosing a uniform $k \in \mathbb{Z}_q$ and setting $I := g^k$; it sends I as the initial message. The verifier chooses and sends a uniform challenge $r \in \mathbb{Z}_q$; in response, the prover computes $s := [rx + k \bmod q]$. The verifier accepts if and only if $g^s \cdot y^{-r} \overset{?}{=} I$. Correctness holds because

$$g^s \cdot y^{-r} = g^{rx+k} \cdot (g^x)^{-r} = g^k = I.$$

Note that I is uniform in $\mathbb{G}$, and so the scheme is non-degenerate.

Before giving the proof, we provide some high-level intuition. A first important observation is that passive eavesdropping is of no help to the attacker. The reason is that the attacker can *simulate* transcripts of honest executions on its own, based only on the public key and *without* knowledge of the private key. To do this, the attacker just reverses the order of the steps: it first chooses uniform and independent $r, s \in \mathbb{Z}_q$ and then sets $I := g^s \cdot y^{-r}$. In an honest transcript (I, r, s), the initial message I is a uniform element of $\mathbb{G}$, the

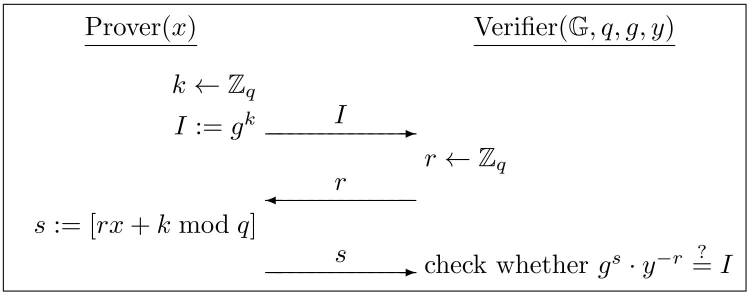
Prover(x)
Verifier$(\mathbb{G}, q, g, y)$
$k \leftarrow \mathbb{Z}_q$
$I := g^k$
I
$r \leftarrow \mathbb{Z}_q$
r
$s := [rx + k \bmod q]$
s
check whether $g^s \cdot y^{-r} \overset{?}{=} I$

FIGURE 13.2: An execution of the Schnorr identification scheme.

challenge is an independent, uniform element of $\mathbb{Z}_q$, and s is then uniquely determined as $s = \log_g(I \cdot y^r)$. Simulated transcripts constructed by an attacker have the same distribution: $r \in \mathbb{Z}_q$ is uniform and, because s is uniform in $\mathbb{Z}_q$ and independent of r, we see that I is uniform in $\mathbb{G}$ and independent of r. Finally, s is uniquely determined as satisfying the same constraint as before. Due to this, we may effectively assume that when attacking the identification scheme, an attacker does not eavesdrop on honest executions at all.

So, we have reduced to an attacker who gets a public key y, sends an initial message I, is given in response a uniform challenge r, and then must send a response s for which $g^s \cdot y^{-r} = I$. Informally, if an attacker is able to do this with high probability then it must, in particular, be able to compute correct responses s_1, s_2 to at least two different challenges $r_1, r_2 \in \mathbb{Z}_q$. Note

$$g^{s_1} \cdot y^{-r_1} = I = g^{s_2} \cdot y^{-r_2},$$

and so $g^{s_1 - s_2} = y^{r_1 - r_2}$. But this implies that the attacker (who, recall, is able to generate s_1 in response to r_1, and s_2 in response to r_2) can implicitly compute the discrete logarithm

$$\log_g y = [(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q],$$

contradicting the assumed hardness of the discrete-logarithm problem.

THEOREM 13.11 *If the discrete-logarithm problem is hard relative to $\mathcal{G}$, then the Schnorr identification scheme is secure.*

PROOF Let Π denote the Schnorr identification scheme, and let $\mathcal{A}$ be a PPT adversary attacking the scheme. We construct the following PPT algorithm $\mathcal{A}'$ solving the discrete-logarithm problem relative to $\mathcal{G}$:

Algorithm $\mathcal{A}'$:
The algorithm is given $\mathbb{G}, q, g, y$ as input.

1. Run $\mathcal{A}(pk)$, answering all its queries to Trans_{sk} as described in the intuition given previously.
2. When $\mathcal{A}$ outputs I, choose a uniform $r_1 \in \mathbb{Z}_q$ as the challenge. Give r_1 to $\mathcal{A}$, who responds with s_1.
3. Run $\mathcal{A}(pk)$ a second time (from the beginning), using the same randomness as before except for uniform and independent $r_2 \in \mathbb{Z}_q$. Eventually, $\mathcal{A}$ responds with s_2.
4. If $g^{s_1} \cdot h^{-r_1} = I$ and $g^{s_2} \cdot h^{-r_2} = I$ and $r_1 \neq r_2$ then output $[(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q]$. Else, output nothing.

Considering a single run of $\mathcal{A}$ as a subroutine of $\mathcal{A}'$, let ω denote the randomness used in that execution except for the challenge itself. So, ω comprises any randomness used by $\mathcal{G}$, the choice of (unknown) private key x, any randomness used by $\mathcal{A}$ itself, and the randomness used by $\mathcal{A}'$ when answering queries to Trans_{sk}. Define $V(\omega, r)$ to be equal to 1 if and only if $\mathcal{A}$ correctly responds to challenge r when randomness ω is used in the rest of the execution. For any fixed ω, define $\delta_\omega \overset{\text{def}}{=} \Pr_r[V(\omega, r) = 1]$; having fixed ω, this is the probability over choice of the challenge r that $\mathcal{A}$ responds correctly.

Define $\delta(n) \overset{\text{def}}{=} \Pr[\mathsf{Ident}_{\mathcal{A},\Pi}(n) = 1]$. Since the simulation of the Trans_{sk} oracle is perfect, we have

$$\delta(n) = \Pr_{\omega,r}[V(\omega, r) = 1] = \textstyle\sum_\omega \Pr[\omega] \cdot \delta_\omega.$$

Moreover, the intuition preceding the proof shows that $\mathcal{A}'$ correctly computes the discrete logarithm of y whenever $\mathcal{A}$ succeeds twice and $r_1 \neq r_2$. Thus:

$$\begin{aligned}
\Pr[\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(n) = 1] &= \Pr_{\omega,r_1,r_2}[V(\omega, r_1) \ \wedge \ V(\omega, r_2) \ \wedge \ r_1 \neq r_2] \\
&\geq \Pr_{\omega,r_1,r_2}[V(\omega, r_1) \ \wedge \ V(\omega, r_2)] - \Pr_{\omega,r_1,r_2}[r_1 = r_2] \\
&= \textstyle\sum_\omega \Pr[\omega] \cdot (\delta_\omega)^2 - 1/q \\
&\geq \left(\textstyle\sum_\omega \Pr[\omega] \cdot \delta_\omega\right)^2 - 1/q \\
&= \delta(n)^2 - 1/q,
\end{aligned}$$

using Jensen's inequality in the second-to-last step. (Jensen's inequality says that $\sum_i a_i \cdot b_i^2 \geq \left(\sum_i a_i\right)^{-1} \cdot \left(\sum_i a_i \cdot b_i\right)^2$ for positive $\{a_i\}$.) If the discrete-logarithm problem is hard relative to $\mathcal{G}$ then $\Pr[\mathsf{DLog}_{\mathcal{A}',\mathcal{G}}(n) = 1]$ is negligible. Since $1/q$ is negligible (because $\|q\| = n$), this implies that $\delta(n)$ is also negligible, and so Π is a secure identification scheme. ■

The Schnorr signature scheme is obtained by applying the Fiat–Shamir transform to the Schnorr identification scheme. See Construction 13.12.

CONSTRUCTION 13.12

Let $\mathcal{G}$ be as described in the text.

- Gen: run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$. Choose a uniform $x \in \mathbb{Z}_q$ and set $y := g^x$. The private key is x and the public key is $(\mathbb{G}, q, g, y)$. As part of key generation, a function $H : \{0,1\}^* \to \mathbb{Z}_q$ is specified, but we leave this implicit.
- Sign: on input a private key x and a message $m \in \{0,1\}^*$, choose uniform $k \in \mathbb{Z}_q$ and set $I := g^k$. Then compute $r := H(I, m)$, followed by $s := [rx + k \bmod q]$. Output the signature (r, s).
- Vrfy: on input a public key $(\mathbb{G}, q, g, y)$, a message m, and a signature (r, s), compute $I := g^s \cdot y^{-r}$ and output 1 if $H(I, m) \overset{?}{=} r$.

The Schnorr signature scheme.