**An optimization.** The GQ signature scheme can be optimized in the same way as the Schnorr signature scheme. Instead of defining a signature on $m$ to be a pair $(y_{\mathrm{t}}, x_{\mathrm{z}})$ satisfying

$$x_{\mathrm{z}}^e = y_{\mathrm{t}} \cdot y^c,$$

where $c := H(m, y_{\mathrm{t}})$, we can define it to be a pair $(c, x_{\mathrm{z}})$ satisfying

$$c = H(m, y_{\mathrm{t}}),$$

where $y_{\mathrm{t}} := x_{\mathrm{z}}^e / y^c$. As a further optimization, one can store $y^{-1}$ in the public key instead of $y$, which will speed up verification.

It turns out that this same optimization can be applied to most instances of the Fiat-Shamir signature construction. See Exercise 19.19.

## 19.7 Combining Sigma protocols: AND and OR proofs

In this section we show how Sigma protocols can be combined to make new Sigma protocols. In the AND-proof construction, a prover can convince a verifier that he "knows" witnesses for a pair of statements. In the OR-proof construction, a prover can convince a verifier that he "knows" witnesses for one of two statements.

### 19.7.1 The AND-proof construction

Suppose we have a Sigma protocol $(P_0, V_0)$ for $\mathcal{R}_0 \subseteq \mathcal{X}_0 \times \mathcal{Y}_0$, and a Sigma protocol $(P_1, V_1)$ for $\mathcal{R}_1 \subseteq \mathcal{X}_1 \times \mathcal{Y}_1$. Further, let us assume that both Sigma protocols use the same challenge space $\mathcal{C}$. We can combine them to form a Sigma protocol for the relation

$$\mathcal{R}_{\mathrm{AND}} = \left\{ \big( (x_0, x_1),\ (y_0, y_1) \big) \in (\mathcal{X}_0 \times \mathcal{X}_1) \times (\mathcal{Y}_0 \times \mathcal{Y}_1) :\ (x_0, y_0) \in \mathcal{R}_0 \text{ and } (x_1, y_1) \in \mathcal{R}_1 \right\}. \quad (19.22)$$

In other words, for a given pair of statements $y_0 \in \mathcal{Y}_0$ and $y_1 \in \mathcal{Y}_1$, this **AND protocol** allows a prover to convince a skeptical verifier that he "knows" a witness for $y_0$ *and* a witness for $y_1$. The protocol $(P, V)$ runs as follows, where the prover $P$ is initialized with $((x_0, x_1), (y_0, y_1)) \in \mathcal{R}_{\mathrm{AND}}$, the verifier $V$ is initialized with $(y_0, y_1) \in \mathcal{Y}_0 \times \mathcal{Y}_1$:

1. $P$ runs $P_0(x_0, y_0)$ to get a commitment $t_0$ and runs $P_1(x_1, y_1)$ to get a commitment $t_1$, and sends the commitment pair $(t_0, t_1)$ to $V$;
2. $V$ computes $c \xleftarrow{\mathrm{R}} \mathcal{C}$, and sends the challenge $c$ to $P$;
3. $P$ feeds the challenge $c$ to both $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$, obtaining responses $z_0$ and $z_1$, and sends the response pair $(z_0, z_1)$ to $V$;
4. $V$ checks that $(t_0, c, z_0)$ is an accepting conversation for $y_0$ and that $(t_1, c, z_1)$ is an accepting conversation for $y_1$.

**Theorem 19.18.** *The AND protocol $(P, V)$ is a Sigma protocol for the relation $\mathcal{R}_{\mathrm{AND}}$ defined in (19.22). If $(P_0, V_0)$ and $(P_1, V_1)$ provide special soundness, then so does $(P, V)$. If $(P_0, V_0)$ and $(P_1, V_1)$ are special HVZK, then so is $(P, V)$.*

*Proof sketch.* Correctness is clear.

For special soundness, if $(P_0, V_0)$ has extractor $Ext_0$ and $(P_1, V_1)$ has extractor $Ext_1$, then the extractor for $(P, V)$ is

$$Ext\Big(\ (y_0, y_1), \big((t_0, t_1), c, (z_0, z_1)\big),\ \big((t_0, t_1), c', (z_0', z_1')\big)\ \Big) :=$$
$$\Big(Ext_0\big(y_0, (t_0, c, z_0), (t_0, c', z_0')\big),\ Ext_1\big(y_1, (t_1, c, z_1), (t_1, c', z_1')\big)\Big).$$

For special HVZK, if $(P_0, V_0)$ has simulator $Sim_0$ and $(P_1, V_1)$ has simulator $Sim_1$, then the simulator for $(P, V)$ is

$$Sim((y_0, y_1), c) := ((t_0, t_1), (z_0, z_1)),$$

where

$$(t_0, z_0) \xleftarrow{R} Sim_0(y_0, c) \quad \text{and} \quad (t_1, z_1) \xleftarrow{R} Sim_1(y_1, c).$$

We leave it to the reader to fill in the details. However, we point out that in our construction of $Sim$, we have exploited the fact that in our definition of special HVZK, the challenge is an input to the simulator, which we can feed to both $Sim_0$ and $Sim_1$. This is one of the main reasons for this aspect of the definition. □

### 19.7.2 The OR-proof construction

Suppose we have a Sigma protocol $(P_0, V_0)$ for $\mathcal{R}_0 \subseteq \mathcal{X}_0 \times \mathcal{Y}_0$, and a Sigma protocol $(P_1, V_1)$ for $\mathcal{R}_1 \subseteq \mathcal{X}_1 \times \mathcal{Y}_1$. We need to make some additional assumptions:

- Both Sigma protocols use the same challenge space $\mathcal{C}$, which is of the form $\mathcal{C} = \{0,1\}^n$. (Note that in the examples we have seen where challenges are numbers, we can always encode bit strings as numbers in binary notation.)
- Both protocols are special HVZK, with simulators $Sim_0$ and $Sim_1$, respectively.

We can combine them to form a Sigma protocol for the relation

$$\mathcal{R}_{\text{OR}} = \Big\{\ \big(\ (b, x),\ (y_0, y_1)\big) \in \big(\{0,1\} \times (\mathcal{X}_0 \cup \mathcal{X}_1)\big) \times (\mathcal{Y}_0 \times \mathcal{Y}_1) :\ (x, y_b) \in \mathcal{R}_b \Big\}. \tag{19.23}$$

In other words, for a given pair of statements $y_0 \in \mathcal{Y}_0$ and $y_1 \in \mathcal{Y}_1$, this **OR protocol** allows a prover to convince a skeptical verifier that he "knows" a witness for $y_0$ *or* a witness for $y_1$. Nothing else should be revealed. In particular the protocol should not reveal if the prover has a witness for $y_0$ or for $y_1$.

The protocol $(P, V)$ runs as follows, where the prover $P$ is initialized with $((b, x), (y_0, y_1)) \in \mathcal{R}_{\text{OR}}$, the verifier $V$ is initialized with $(y_0, y_1) \in \mathcal{Y}_0 \times \mathcal{Y}_1$, and $d := 1 - b$:

1. $P$ computes $c_d \xleftarrow{R} \mathcal{C}$, $(t_d, z_d) \xleftarrow{R} Sim_d(y_d, c_d)$.

   $P$ also runs $P_b(x, y_b)$ to get a commitment $t_b$, and sends the commitment pair $(t_0, t_1)$ to $V$;

2. $V$ computes $c \xleftarrow{R} \mathcal{C}$, and sends the challenge $c$ to $P$;

3. $P$ computes $c_b \leftarrow c \oplus c_d$

   $P$ feeds the challenge $c_b$ to $P_b(x, y_b)$, obtaining a response $z_b$, and sends $(c_0, z_0, z_1)$ to $V$;

4. $V$ computes $c_1 \leftarrow c \oplus c_0$, and checks that $(t_0, c_0, z_0)$ is an accepting conversation for $y_0$, and that $(t_1, c_1, z_1)$ is an accepting conversation for $y_1$.

**Theorem 19.19.** *The OR protocol $(P, V)$ is a special HVZK Sigma protocol for the relation $\mathcal{R}_{\mathrm{OR}}$ defined in (19.23). If $(P_0, V_0)$ and $(P_1, V_1)$ provide special soundness, then so does $(P, V)$.*

*Proof sketch.* Correctness is clear.

For special soundness, if $(P_0, V_0)$ has extractor $Ext_0$ and $(P_1, V_1)$ has extractor $Ext_1$, then the extractor $Ext$ for $(P, V)$ takes as input $(y_0, y_1)$ and a pair of accepting conversations

$$\big((t_0, t_1),\ c,\ (c_0, z_0, z_1)\big) \quad \text{and} \quad \big((t_0, t_1),\ c',\ (c'_0, z'_0, z'_1)\big).$$

Let $c_1 := c \oplus c_0$ and $c'_1 := c' \oplus c'_0$. The key observation is that if $c \neq c'$, then we must have either $c_0 \neq c'_0$ or $c_1 \neq c'_1$. So $Ext$ works as follows:

if $c_0 \neq c'_0$
  then output $\big(\ 0,\ Ext_0(y_0, (t_0, c_0, z_0), (t_0, c'_0, z'_0))\ \big)$
  else output $\big(\ 1,\ Ext_1(y_1, (t_1, c_1, z_1), (t_1, c'_1, z'_1))\ \big)$

For special HVZK, the simulator for $(P, V)$ is

$$Sim((y_0, y_1), c) := ((t_0, t_1), (c_0, z_0, z_1)),$$

where

$$c_0 \xleftarrow{\mathrm{R}} \mathcal{C},\ c_1 \leftarrow c \oplus c_0,\ (t_0, z_0) \xleftarrow{\mathrm{R}} Sim_0(y_0, c_0),\ (t_1, z_1) \xleftarrow{\mathrm{R}} Sim_1(y_1, c_1).$$

We leave it to the reader to fill in the details. However, we point out that to guarantee correctness, we have exploited the fact that in our definition of special HVZK, the simulator always outputs an accepting conversation. This is one of the main reasons for this aspect of the definition. □

## 19.8 Witness independence and applications

We next study a useful property of Sigma protocols called **witness independence**.

For a given statement there may be several witnesses. Roughly speaking, witness independence means the following: if a "cheating" verifier $V^*$ (one that need not follow the protocol) interacts with an honest prover $P$, then $V^*$ cannot tell *which* witness $P$ is using. In particular, even if $V^*$ is very powerful and/or very clever and is able to compute a witness after interacting with $P$, this witness will be unrelated to $P$'s witness. Of course, this property is only interesting if a given statement has more than one witness.

First, we define this property more precisely. Second, we show that special HVZK implies witness independence. This is perhaps a bit surprising, as HVZK is a property about *honest* verifiers, while witness independence applies to *all* verifiers (even computationally unbounded cheating verifiers). Finally, as an application, we show how to use witness independence to design identification protocols that are secure against *active* attacks, rather than just eavesdropping attacks. These identification protocols are simple and efficient, and their security can be based on either the DL or RSA assumptions (and without relying on the random oracle heuristic).