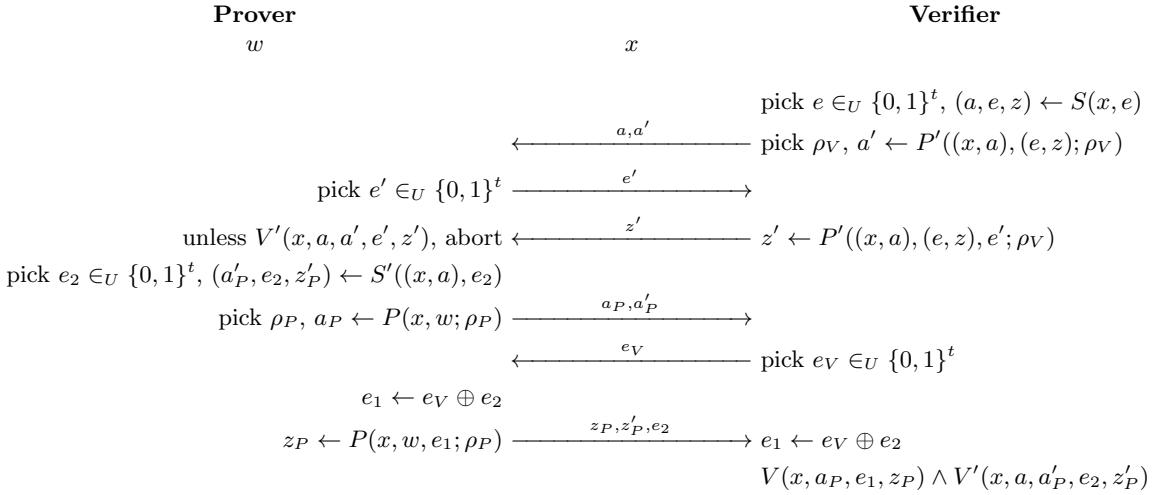
2 ZKPoK from Sigma

This exercise is inspired from Cramer-Damgård-MacKenzie, Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions, *PKC 2000, LNCS vol. 1751, Springer.*

We consider a relation $R(x, w)$ defining a language for which we have a Σ protocol (P, V) over a challenge set $\{0,1\}^t$ with accepting predicate $V(x, a, e, z)$, Σ-simulator S, and Σ-extractor E. We define a relation $R'((x, a), (e, z))$ to hold on instance (x, a) with witness (e, z) if $V(x, a, e, z)$ is accepting. We assume that R' also has a Σ protocol (P', V') over the same challenge set $\{0,1\}^t$ with accepting predicate $V'(x, a, a', e', z')$, Σ-simulator S', and Σ-extractor E'. We consider the following protocol:



Q.1 In the first part of the protocol, recognize and isolate a commitment on the value e and a proof of knowledge of a valid opening of this commitment. Fully describe the commitment scheme. Fully describe the proof of knowledge.

In the first step of the protocol, the verifier commits to some e without revealing it by using the conversion from a Σ-protocol to a commitment scheme. For that, he runs S with challenge e and sends the commit value a. He proves in a Σ protocol that he knows a valid (e, z), so that he knows how to open the commitment.
The commitment scheme was seen in the course. To commit on e, the sender runs $S(x, e) \to (a, e, z)$ and uses a as a commit value and (e, z) as the opening value. To open a commitment a with (e, z), we just check $V(x, a, e, z)$, i.e. the relation $R'((x, a), (e, z))$. Hence, the (P', V') protocol is a proof of knowledge of a valid opening of the commitment a.

Q.2 In the second part of the protocol, recognize a proof of knowledge of either w for $(R(x, w)$ or (e, z) for $R'((x, a), (e, z))$.

The second part of the protocol is a standard OR proof for R and R'. The prover proves that he knows either w for R or (e, z) for R'. He can as he actually knows w. The OR proof runs in parallel the Σ protocols for R and R' but the challenges are chosen by the prover with a XOR e_V imposed by the verifier. If we know one of the two witnesses, we can run the corresponding Σ protocol with any challenge. If we do not, we can anticipate the challenge and use the simulator of the corresponding Σ protocol. If we know one of the two challenges, we anticipate a random challenge for the ignored witness and we select the challenge for the known witness with the correct XOR.
This OR proof was the subject of a previous exam.

Q.3 Show that the protocol is complete and runs in polynomial time $\mathsf{poly}(t, |x|)$ (where $|x|$ is the length of x) for the verifier.

All protocols run by the verifier are PPT algorithm. So, the protocol runs in polynomial time for the verifier.
The first part of the protocol is complete thanks to the second Σ protocol. It is a proof that the verifier knows (e, z), a valid opening of the commitment.
The second part of the proof is an OR-proof. It completes thanks to the simulator S' and the completeness of the Σ protocol for R.

Q.4 Show that the protocol is zero-knowledge by constructing a black-box simulator.

In the first part of the protocol, we simulate a prover normally to the verifier. Then, we rewind the verifier and run the simulation again. As it is likely to produce a different challenge e', we can use the extractor E' to extract a valid $(\bar{e}, \bar{z})$ witness for R'. Then, we can simulate a prover who knows $(\bar{e}, \bar{z})$ in the OR proof: we pick e_1, run $(a_P, e_1, z_P) \leftarrow S(x, e_1)$, $a'_P \leftarrow P'(x, a, \bar{e}, \bar{z}; \rho_P)$, send (a_P, a'_P), get e_V, set $e_2 = e_V \oplus e_1$, run $z'_P \leftarrow P'(x, a, \bar{e}, \bar{z}, e_e; \rho_P)$, and send (z_P, z'_P, e_2). We can easily see that the distribution is correct.

Q.5 Construct a knowledge extractor for this protocol to prove that it is a zero-knowledge proof of knowledge for R.

The extractor runs the prover twice with the same random coins and simulate the verifier the same way in both executions, except for issuing e_V where they may fork. Like in the proof which was seen in the course, we obtain two executions with the same transcript until e_V then two transcripts $e_V^1, z_P^1, {z'}_P^1, e_2^1$ and $e_V^2, z_P^2, {z'}_P^2, e_2^2$. We assume $e_V^1 \neq e_V^2$.
Let $e_1^i = e_V^i \oplus e_2^i$. If $e_1^1 \neq e_1^2$, we have two transcripts a_P, e_1^1, z_P^1 and a_P, e_1^2, z_P^2 with different challenges so E extracts a witness w for R.
If now $e_1^1 = e_1^2$, we must have $e_2^1 \neq e_2^2$, so we get two transcripts $a'_P, e_2^1, {z'}_P^1$ and $a'_P, e_2^2, {z'}_P^2$ with different challenges so E' extracts a witness $(\bar{e}, \bar{z})$ for R'. Since the extractor simulated the verifier, he already has some witness (e, z) for R'. If $e \neq \bar{e}$, we can use E again to obtain a witness w for R.
What remains to show is that $e = \bar{e}$ occurs with probability 2^{-t}. This is due to the extractor revealing no information about e to the prover so that $\bar{e}$ and e are statistically independent. As e is uniformly distributed, this concludes the proof.