MTAT.07.003 Cryptology II
Spring 2012 / Exercise session ?? / Example Solution

Exercise (Standard simulator for the QNR-ZK protocol). *Let n be a composite number with a factorisation $n = pq$ known to the prover $\mathcal{P}$. Let $v \in \mathbb{Z}_n^*$ be a number for which the prover wants to prove that it is quadratic non-residue. Show that the the following zero-knowledge protocol*

$$\begin{array}{lcl}
\mathcal{P} & & \mathcal{V} \\
 & & b \xleftarrow{u} \{0,1\}\,, \\
 & \xleftarrow{\qquad c = r^2 v^b \qquad} & r \xleftarrow{u} \mathbb{Z}_n^* \\
 & \xleftarrow{\qquad \alpha \qquad} & \alpha \leftarrow \mathcal{P}_{\mathrm{POK}} \\
\beta \leftarrow \mathcal{B} & \xrightarrow{\qquad \beta \qquad} & \\
\mathsf{Ver}(\alpha,\beta,\gamma) \overset{?}{=} 1 & \xleftarrow{\qquad \gamma \qquad} & \gamma \leftarrow \mathcal{P}_{\mathrm{POK}}(\beta) \\
\overline{b} \leftarrow \mathsf{IsQnr}_{p,q}(c) & \xrightarrow{\qquad \overline{b} \qquad} & \overline{b} \overset{?}{=} b
\end{array}$$

where the verifier $\mathcal{V}$ uses sigma protocol $\mathrm{POK}[\exists r, \beta : c = r^2 v^b]$ to prove the knowledge of b and r is simulatable. More precisely, let $\mathcal{K}^{\mathcal{V}_*}$ standard knowledge extractor for the protocol $\mathrm{POK}[\exists r, \beta : c = r^2 v^b]$. Then show that the following simulator construction

```
$\mathcal{V}_\circ(\phi)$
    $\omega \leftarrow \Omega$
    $c \leftarrow \mathcal{V}_*(\phi;\omega)$
    $(b_*, r_*) \leftarrow \mathcal{K}^{\mathcal{V}_*}(\phi;\omega)$
    $\alpha \leftarrow \mathcal{V}_*$
    $\beta \xleftarrow{u} \mathcal{B}$
    $\gamma \leftarrow \mathcal{V}_*(\beta)$
    if $\mathsf{Ver}(\alpha,\beta,\gamma) = 0$ then return $\mathcal{V}_*(\perp)$
    if $\beta_* \neq \perp$ then return $\mathcal{V}_*(b_*)$
    else return $\perp$
```

can create an output distribution $\psi_\circ$ that is statistically $\varepsilon_\circ$-distant from the output distribution of malicious verifier $\mathcal{V}_*$ that interacts with the honest prover. Also, estimate how the running-time of the simulator depends on the desired statistical distance $\varepsilon_\circ$.

Solution. For the analysis, let us contrast the simulation with the real-world execution. For that we observe when the execution of a simulator diverges form the real-world interaction between $\mathcal{V}(\phi)$ and $\mathcal{P}$ provided that the randomness of the verifier $\omega \leftarrow \Omega$ is same. Again, it is instructive to see the event trees first, see Figure 1. On the left you see the execution corresponding to the simulation. On the right you see the real protocol execution augmented with the run of the knowledge extractor. Here, the protocol run does not use knowledge-extraction for anything, it just provides the same split of events.

First, observe that the simulation and real-world execution are identical till the last verification step of $\mathrm{POK}[\exists r, \beta : c = r^2 v^b]$, since all messages are created exactly the same way in both runs. If the verification fails, then the verifier gets the same input in both executions. Hence, we must consider the remaining case. Clearly, executions diverge when the knowledge-extraction fails. In the remaining case, $\mathcal{V}$ receives $\overline{b}$ in the real execution and b_* in the simulation. Hence, we must show that $\overline{b} = b_*$.

The latter reveals a small but important detail in the knowledge proof. Namely, it is straightforward to see that any invertible element c one has only two possible decompositions: $c = r^2 v^b$ and $c = (-r)^2 v^b$ if the Jacobi symbols of v and c match. Thus, b_* must be equal to $\overline{b}$. However, if the Jacobi symbols of v and c are not both ones, the existence of the decomposition is not guaranteed and we cannot conclude that $b_* = \overline{b}$.

Fortunaltely, in the first step of the proof of knowledge the verifying party checks that v and c are in the suitable form and the same is done in the knowledge extractor. Hence, the claim $b_* = \overline{b}$ holds.

$$\begin{tikzcd}
 & \mathcal{V}(\phi;\omega) \arrow[d, squiggly] & & \\
 & \mathsf{Ver}(\alpha,\beta,\gamma) \arrow[dl, "0"'] \arrow[dr, "1"] & & \\
\mathcal{V}(\bot) \arrow[dd] & & \mathcal{K}^{\mathcal{V}}(\phi;\omega) \arrow[dl, "\bot"'] \arrow[dr, "b_*, r_*"] & \\
 & \mathcal{V}(\bot) \arrow[d] & & \mathcal{V}(b_*) \arrow[d] \\
\phi & \phi & & \phi
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathcal{V}(\phi;\omega) \arrow[d, squiggly] & & \\
 & \mathsf{Ver}(\alpha,\beta,\gamma) \arrow[dl, "0"'] \arrow[dr, "1"] & & \\
\mathcal{V}(\bot) \arrow[dd] & & \mathcal{K}^{\mathcal{V}}(\phi;\omega) \arrow[dl, "\bot"'] \arrow[dr, "b_*, r_*"] & \\
 & \mathcal{V}(\overline{b}) \arrow[d] & & \mathcal{V}(\overline{b}) \arrow[d] \\
\phi & \phi & & \phi
\end{tikzcd}$$

Figure 1: Event trees for the protocol simulation and for real protocol execution.

These observations combined with the the following lemma allow us to estimate the statistical distance by computing the probability of the event that verification succeeds while knowledge error fails.

Lemma. *If two strategies start to diverge only after the event* div *and and are identical before that the the statistical distance between the output distributions is at most* $\Pr[\mathsf{div}]$.

Proof. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be the strategies in question and ψ_1 and ψ_2 corresponding output distributions. Then by the definition

$$\begin{aligned}
\mathsf{sd}(\psi_1,\psi_2) &= \frac{1}{2}\cdot\sum_{\phi}\left|\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \phi_1 = \phi\right] - \Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \phi_2 = \phi\right]\right| \\
&= \frac{1}{2}\cdot\sum_{\phi}\left|\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \phi_1 = \phi \wedge \mathsf{div}\right] - \Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \phi_2 = \phi \wedge \mathsf{div}\right]\right| \\
&\quad + \frac{1}{2}\cdot\sum_{\phi}\left|\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \phi_1 = \phi \wedge \neg\mathsf{div}\right] - \Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \phi_2 = \phi \wedge \neg\mathsf{div}\right]\right| \\
&= \frac{1}{2}\cdot\sum_{\phi}\left|\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \phi_1 = \phi \wedge \mathsf{div}\right] - \Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \phi_2 = \phi \wedge \mathsf{div}\right]\right| \\
&\leq \frac{1}{2}\cdot\sum_{\phi}\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \phi_1 = \phi \wedge \mathsf{div}\right] + \frac{1}{2}\cdot\sum_{\phi}\Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \phi_2 = \phi \wedge \mathsf{div}\right] \\
&\leq \frac{1}{2}\cdot\Pr\left[\phi_1 \leftarrow \mathcal{S}_1 : \mathsf{div}\right] + \frac{1}{2}\cdot\Pr\left[\phi_2 \leftarrow \mathcal{S}_2 : \mathsf{div}\right] \leq \Pr\left[\mathsf{div}\right]
\end{aligned}$$

Note that by the construction the event div has the same probability for both strategies since up to this procedure both systems are identical and thus $\Pr[\mathsf{div}]$ is well defined. □

For brevity, let $\mathsf{SimFail}$ denote the event that verification succeeds while knowledge error fails. As the probability of this event may depend on the randomness ω, we must configure the knowledge extractor so that the corresponding probability is negligible on average over all possible $\omega \in \Omega$. First all all the failure probability can be high for fixed ω only if verification succeeds and knowledge error fails with non-negligible probability. Intuitively, achieving such a bad case is a careful balancing act, since higher success probability on verification also increases the success of $\mathcal{K}^{\mathcal{V}}(\phi;\omega)$. Figure 2 illustrates the corresponding behaviour.

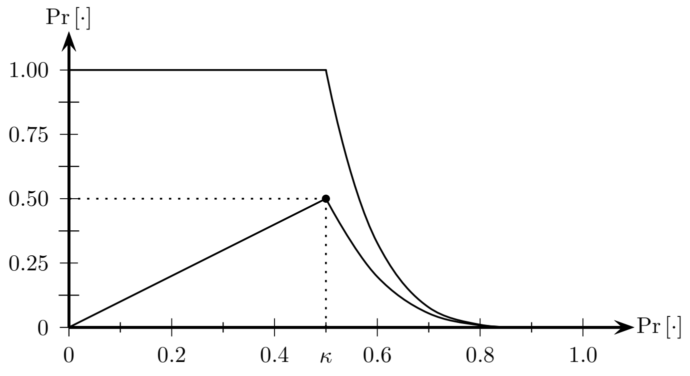

Figure 2: Knowledge-extraction failure as function of verification success. The x-axis corresponds to the verification success and y-axis corresponds to the failure probability. The upper graph corresponds to the knowledge extraction failure and the lower graph corresponds to the simulation failure. The plot is drawn for the knowledge extractor that does 10 probes. In reasonable simulation construections $\ell \gg 10$ and the drop after the $\frac{1}{2}$ is extremely sharp and the optimum does not change for fixed running-time simulators.

Figure 2 clearly shows that the worst configuration is such that $\Pr[\mathsf{Ver}(\alpha,\beta,\gamma)=1]=\frac{1}{2}$ for which the simulation failure $\mathsf{SimFail}$ occurs with probability $\frac{1}{2}$. The latter holds for fixed randomness $\omega \in \Omega$. Thus, the worst adversarial configuration for the simulation is such that it will succeed the proof of knowledge with probability $\frac{1}{2}$. As a result,

$$\Pr[\mathsf{SimFail}] = \sum_{\omega\in\Omega} \Pr[\omega]\cdot\Pr[\mathsf{SimFail}|\omega] = \frac{1}{2}$$

and the lemma provides a poor bound on the statistical distance:

$$\mathsf{sd}(\psi_1,\psi_2) \leq \Pr[\mathsf{SimFail}] = \frac{1}{2} \enspace .$$

The bound is sharp. If the malicious verifier chooses a random challenge or a challenges obtained prior to the beginning of the protocol. Then the verifier can cheat in the proof of knowledge by first guessing β and then simulating the triple using the canonical simulator $\mathcal{S}$. As a result, the verifier will learn the quadratic residuosity of c with probability $\frac{1}{2}$. Under the standard assumptions detecting quadratic residuosity is intractable problem and thus no efficient simulator that achieves $\mathsf{sd}(\psi_1,\psi_2) \ll \frac{1}{2}$ can exist.

Naive protocol upgrade. The large discrepancy in the simulation is doused by the high knowledge error. The simplest solution is to repeat the proof-of-knowledge protocol k times. As a result, the knowledge-error κ decreases form $\frac{1}{2}$ to 2^{-k}. However, the lower knowledge error alone does not guarantee a small simulation failure, the the knowledge-extraction must be successful enough when $\varepsilon > \kappa$. To understand the logic behind the simulation of multi-stage proof-knowledge, it is instructive to look at Figure 3.

By the construction the events on each stage are independent form the previous stages. The stage fails if the verification succeeds but knowledge extraction fails. This stage failure depends on the adversaries behaviour. However, as the stage failure is identical to the simulation failure of the knowledge proof, it is easy to guarantee that the stage failure probability is below $\frac{1}{2}$. Thus, the statistical distance between verifiers outputs in the simulation and real protocol execution is well bounded:

$$\mathsf{sd}(\psi_1,\psi_2) \leq \Pr[\mathsf{SimFail}]\cdot\ldots\cdot\Pr[\mathsf{SimFail}] \leq 2^{-k} \enspace .$$

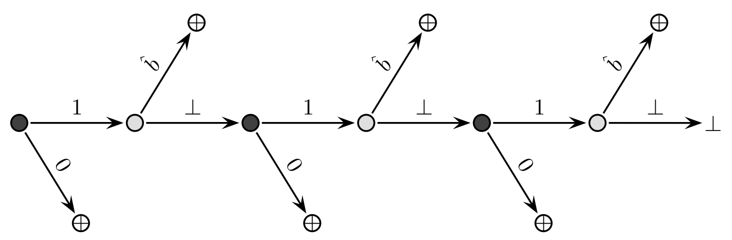

Figure 3: Combined simulation and knowledge extraction in case of a multi-stage proof of knowledge. Black dots represent verification stages doting the simulation. Grey dots represent knowledge-extraction phases following successful verification stages. Dots with crosses denote states where perfect simulation if achievable either because a verification stage failed or because a knowledge-extraction yielded the desired witness. In these cases, perfect simulation of the protocol becomes trivial.

This bound leads to an interesting observation about knowledge-extraction stategy. The knowledge extraction does not have to be very successful to get a simulation profile that is bounded above by $\frac{1}{2}$. By simple analysis we can show that the failure probability of a standard knowledge extractor is

$$\Pr\left[\mathcal{K}^{\mathcal{V}}(\phi;\omega)=\perp\right]=\begin{cases}1, & \text{if } \varepsilon\le\frac{1}{2}\ ,\\ (2-2\varepsilon)^{\ell}, & \text{if } \varepsilon>\frac{1}{2}\ ,\end{cases}$$

where the number of rewindings is 2ℓ. As the success probability in the verification stage is ε, the simulation failure can be expressed as follows

$$\Pr\left[\mathsf{SimFail}\right]=\begin{cases}\varepsilon, & \text{if } \varepsilon\le\frac{1}{2}\ ,\\ \varepsilon(2-2\varepsilon)^{\ell}, & \text{if } \varepsilon>\frac{1}{2}\ .\end{cases}$$

The corresponding graph illustrated in Figure 4 shows that the optimal number of rewindings is 2.

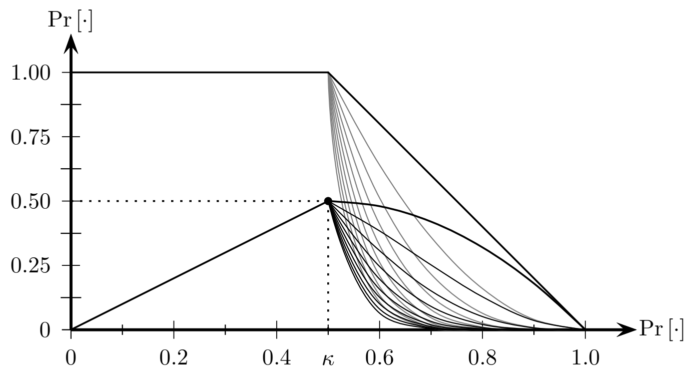

Figure 4: Knowledge-extraction failure as function of verification success. The x-axis corresponds to the verification success and y-axis corresponds to the failure probability. The upper collection of graphs corresponds to the knowledge extraction failure for $\ell\in\{1,\dots,10\}$ and the upper collection of graphs lower graph corresponds to the simulation failure for $\ell\in\{1,\dots,10\}$. The optimal configuration where $\ell=1$ is drawn in bold. By increasing the value of ℓ, the simulation failure drops faster when $\varepsilon>\kappa$ but this does not reduce the worst-case bound on the simulation failure.

Round-efficient solution. More efficient way to reduce the knowledge error is to run several protocols in parallel. The latter does not change the soundness of the protocol, since the latter depends on witness indistinguishability that is preserved under parallel composition. It is also straightforward to see that the knowledge error indeed drops 2^{-k}. There are 2^k potential challenges and if the verifier is able to answer at least two of them correctly, then at least two sub-protocol runs are successful with different challenges and we can extract the witness. In brief, the knowledge extractor has to find two successful verification runs in order to extract the secret. As the simulation is already successful, it is enough to use random challenges in order to reveal the second successful verification run with the same randomness.

If we probe the verifier with random challenges, the probability that we get a successful run that is different from the one revealed in simulation is $(\varepsilon - \kappa)$ where ε is the unknown success probability for verification. Thus, the probability of continuos stream of failures after ℓ tries is

$$\Pr\left[\mathcal{K}^{\mathcal{V}}(\phi,\kappa)=\bot\right]=\begin{cases}1, & \text{if } \varepsilon\leq\kappa\ \ ,\\ (1+\kappa-\varepsilon)^{\ell} & \text{if } \varepsilon>\kappa\ \ .\end{cases}$$

Hence, the probability of simulation failure if

$$\Pr\left[\mathsf{SimFail}\right]=\begin{cases}\varepsilon, & \text{if } \varepsilon\leq\kappa\ \ ,\\ \varepsilon(1+\kappa-\varepsilon)^{\ell} & \text{if } \varepsilon>\kappa\ \ .\end{cases}$$

Figure 5 shows that this setting offers different tradeoffs in terms of rewindings.

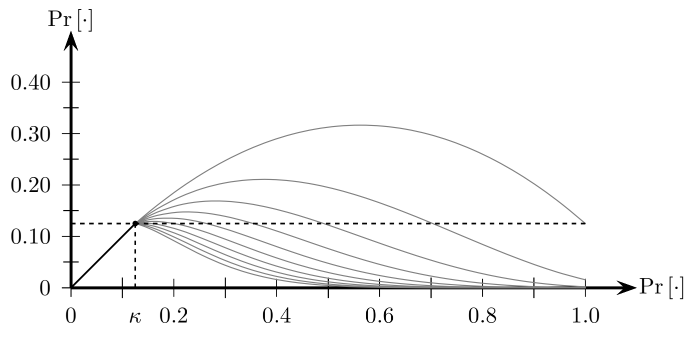

Figure 5: Simulation failure as function of verification success when $\kappa = \frac{1}{8}$. The x-axis corresponds to the verification success and y-axis corresponds to the the simulation failure probability. The collection of graphs corresponds to $\ell \in \{1, \ldots, 10\}$. The choice of the optimal value of ℓ is not a clear cut. The maximal value for simulation failure is above $\frac{1}{8}$ when $\ell \in \{1, \ldots, 9\}$ but there is no rational reasons why the number of rewindings should be large enough to guarantee that the simulaten failure is equal to κ.

To find the maximal value on the simulation failure profile note that logarithm is a monotone function and thus we can minimise $\mathcal{L} = \log \Pr\left[\mathsf{SimFail}\right]$ instead. It is straightforward to see that

$$\frac{\partial\mathcal{L}}{\partial\varepsilon}=\frac{1+\kappa-\varepsilon(\ell+1)}{\varepsilon(1+\kappa-\varepsilon)}\qquad\text{for}\qquad\varepsilon\geq\kappa$$

and thus the function is bound to have a single maximum in the location

$$\varepsilon_*=\frac{1+\kappa}{1+\ell}$$

father which the function starts to decrease. Consequently, if the number of rewindings

$$\ell\geq\frac{1}{\kappa}\ \ ,$$

the simulation failure is quarantined to decrease after $\varepsilon \geq \kappa$ and simulation failure coincides with knowledge error. For other choices of ℓ the maximal simulation failure is

$$p_* = \max_{\varepsilon\in[0,1]} \Pr\left[\mathsf{SimFail}\right] = \frac{(1+\kappa)^{\ell+1}}{2+\ell}\cdot\left(1+\frac{1}{\ell+1}\right)^{\ell+1} \approx \frac{(1+\kappa)^{\ell+1}}{2+\ell}\cdot e$$

To illustrate the asymptotic behaviour in the process $\kappa \to 0$, we can try different ways to increase ℓ. By choosing $\ell = \Theta(\frac{1}{\kappa})$ we get

$$p_* \approx \frac{(1+\kappa)^{\frac{1}{c\kappa}+1}}{2+\frac{1}{c\kappa}}\cdot e \approx e^{1+\frac{1}{c}}\cdot\frac{(1+\kappa)c\kappa}{2c\kappa+1} \approx \alpha\kappa \enspace .$$

In other words, if $\ell = \Theta(\frac{1}{\kappa})$ then the simulation failure is of order $\Theta(\kappa)$. The latter is not very good for getting tight simulation bounds. In order to get statistical difference 2^{-80} we have to do around 2^{80} rewindings, which makes the simulator 2^{80} times slower than the original attack code.

The simulation failure does not have be of same order as the knowledge error κ. As the knowledge error decreases exponentially in terms of the protocol efficiency, we could in principle allow different tradeoffs between p_*, κ, ℓ. Ideally, we would like that p_* would decrease exponentially wrt ℓ. As the second term in the approximation

$$\log p_* \approx 1 - \log(2+\ell) + (\ell+1)\log(1+\kappa) \approx 1 - \log(2+\ell) + (\ell+1)\kappa$$

cannot be of order ℓ, the exponential decay of simulation failure is impossible. In fact the simulation failure can decrease as a function of ℓ only if

$$\kappa \leq \frac{\log(2+\ell)}{\ell+1} \enspace .$$

Note that even if the third term remains constant and does not grow, the simulation failure decreases at most linearly in $\log(\ell)$ and thus for any choice of κ the simulation failure is still $\Omega(\frac{1}{\ell})$. In that sense, inversely proportional slowdown cannot be escaped with conventional methods.