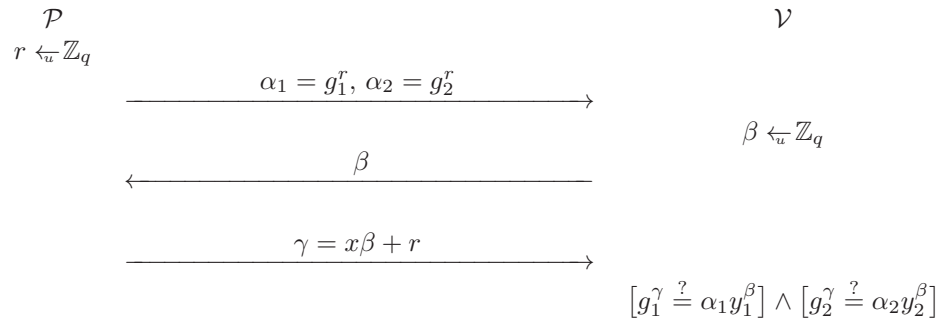
MTAT.07.003 Cryptology II
Spring 2012 / Exercise session ?? / Example Solution

Exercise (Proof of knowledge for double-exponent). *Let $\mathbb{G}$ be a cyclic group with prime number of elements q and let g_1 and g_2 be generators of the group. Now consider a sigma protocol for proving the knowledge of x such that $g_1^x = y_1$ and $g_2^x = y_2$, i.e., the public information is (g_1, g_2, y_1, y_2) and the secret knowledge is x. The protocol itself is following.*



Prove that the protocol is a sigma protocol and show how this protocol can be used to prove knowledge of plaintext m for an ElGamal encryption (c_1, c_2).

Solution. As the protocol has the right message structure, we must show only that the protocol is functional and has special soundness and zero-knowledge property.

Functionality. According to the protocol definition, we can deduce the following equalities:

$$g_1^\gamma = g_1^{x\beta+r} = g_1^r g_1^{x\beta} = \alpha_1 y_1^\beta \ ,$$
$$g_2^\gamma = g_2^{x\beta+r} = g_2^r g_2^{x\beta} = \alpha_2 y_2^\beta \ .$$

Therefore, if the prover and verifier both correctly follow the protocol, the verifier reaches accepting state at the end of the protocol, since the checks $[g_1^\gamma \stackrel{?}{=} \alpha_1 y_1^\beta]$ and $[g_2^\gamma \stackrel{?}{=} \alpha_2 y_2^\beta]$ are both true.

Special soundness property. We need to show that there exists an efficient algorithm that can extract the secret knowledge given transcripts of two successful protocol runs with same alpha message. Let us assume that $(\alpha_1, \alpha_2, \beta_1, \gamma_1)$ and $(\alpha_1, \alpha_2, \beta_2, \gamma_2)$ are different accepting transcripts. For simplicity, let us first assume that the prover is semihonest and thus replies

$$\gamma_1 = x\beta_1 + r \ ,$$
$$\gamma_2 = x\beta_2 + r \ .$$

To solve the system of linear equations for x, we can first express the difference

$$\gamma_1 - \gamma_2 = x\beta_1 + r - x\beta_2 - r = x(\beta_1 - \beta_2)$$

and then use the fact that $\beta_1 \neq \beta_2$. As a result, we obtain an expression

$$x = \frac{x(\beta_1 - \beta_2)}{\beta_1 - \beta_2} = \frac{\gamma_1 - \gamma_2}{\beta_1 - \beta_2}$$

where the right-hand side consist of quantities that are present in the transcripts.

To complete the argument, we are going to show that the same expression for x is valid even if the prover can send arbitrarily formed messages. Since we assumed that $(\alpha_1, \alpha_2, \beta_1, \gamma_1)$ and $(\alpha_1, \alpha_2, \beta_2, \gamma_2)$ are accepting transcripts the following equalities that the verifier checks must hold:

$$\begin{cases} g_1^{\gamma_1} = \alpha_1 y_1^{\beta_1} \\ g_1^{\gamma_2} = \alpha_1 y_1^{\beta_2} \end{cases} \qquad \begin{cases} g_2^{\gamma_1} = \alpha_2 y_2^{\beta_1} \\ g_2^{\gamma_2} = \alpha_2 y_2^{\beta_2} \end{cases} \ .$$

As $\mathbb{G}$ is a cyclic group with prime order elements g_1 and g_2 are the generators of $\mathbb{G}$. Hence, there must exist x_1 and x_2 such that $y_1 = g_1^{x_1}$ and $y_2 = g_2^{x_2}$. Keeping this in mind, we can represent α_1 in two ways as a power of g_1 and α_1 in two ways as a power of g_2:

$$\begin{aligned}
g_1^{\gamma_1} = \alpha_1 y_1^{\beta_1} \quad &\Rightarrow \quad \alpha_1 = g_1^{\gamma_1} \cdot y_1^{-\beta_1} = g_1^{\gamma_1 - x_1\beta_1} \ , \\
g_1^{\gamma_2} = \alpha_1 y_1^{\beta_2} \quad &\Rightarrow \quad \alpha_1 = g_1^{\gamma_2} \cdot y_1^{-\beta_2} = g_1^{\gamma_2 - x_1\beta_2} \ , \\
g_2^{\gamma_1} = \alpha_2 y_2^{\beta_1} \quad &\Rightarrow \quad \alpha_2 = g_2^{\gamma_1} \cdot y_2^{-\beta_1} = g_1^{\gamma_1 - x_2\beta_1} \ , \\
g_2^{\gamma_2} = \alpha_2 y_2^{\beta_2} \quad &\Rightarrow \quad \alpha_2 = g_2^{\gamma_2} \cdot y_2^{-\beta_2} = g_2^{\gamma_2 - x_2\beta_2} \ .
\end{aligned}$$

Since the discrete logarithm is unique if we consider the exponent as a residue in $\mathbb{Z}_q$, we get two equations for determining x_1 and x_2:

$$\begin{aligned}
\gamma_1 - x_1\beta_1 &= \gamma_2 - x_1\beta_2 \\
\gamma_1 - x_2\beta_1 &= \gamma_2 - x_2\beta_2
\end{aligned}$$

which we can solve for x_1 and x_2:

$$x_1 = \frac{\gamma_1 - \gamma_2}{\beta_1 - \beta_2} = x_2 \ .$$

As result, we have shown that $x_1 = x_2$ and the knowledge extraction algorithm always computes the correct witness x corresponding to both y_1 and y_2. This concludes the proof of special soundness, since the knowledge extractor computes the secret in constant time with probability 1.

There is a more straightforward way to check whether the candidate expression for x is valid or not. Even if the prover is malicious, the knowledge extractor can still compute

$$x = \frac{\gamma_1 - \gamma_2}{\beta_1 - \beta_2} \ .$$

However, we do not know whether x satisfies the requirements $g_1^x = y_1$ and $g_2^x = y_2$. Hence, we must formally check it. Note that we can use verifier checks to simplify expressions:

$$\begin{aligned}
g_1^x = g^{\frac{\gamma_1 - \gamma_2}{\beta_1 - \beta_2}} \quad &\Leftrightarrow \quad g_1^{x(\beta_1 - \beta_2)} = g_1^{\gamma_1 - \gamma_2} = \frac{\alpha_1 y_1^{\beta_1}}{\alpha_1 y_1^{\beta_2}} = y_1^{\beta_1 - \beta_2} \ , \\
g_2^x = g^{\frac{\gamma_1 - \gamma_2}{\beta_1 - \beta_2}} \quad &\Leftrightarrow \quad g_2^{x(\beta_1 - \beta_2)} = g_2^{\gamma_1 - \gamma_2} = \frac{\alpha_2 y_2^{\beta_1}}{\alpha_2 y_2^{\beta_2}} = y_2^{\beta_1 - \beta_2} \ .
\end{aligned}$$

To continue note that the inverse of $\beta_1 - \beta_2$ exists in $\mathbb{Z}_q$ and thus we can continue derivations:

$$\begin{aligned}
g_1^{x(\beta_1 - \beta_2)} = y_1^{\beta_1 - \beta_2} \quad &\Leftrightarrow \quad g_1^x = \left(g_1^{x(\beta_1 - \beta_2)}\right)^{\frac{1}{\beta_1 - \beta_2}} = \left(y_1^{\beta_1 - \beta_2}\right)^{\frac{1}{\beta_1 - \beta_2}} = y_1 \ , \\
g_2^{x(\beta_1 - \beta_2)} = y_2^{\beta_1 - \beta_2} \quad &\Leftrightarrow \quad g_2^x = \left(g_2^{x(\beta_1 - \beta_2)}\right)^{\frac{1}{\beta_1 - \beta_2}} = \left(y_2^{\beta_1 - \beta_2}\right)^{\frac{1}{\beta_1 - \beta_2}} = y_2 \ ,
\end{aligned}$$

which finally prove the desired claim. Thes second proof has the virtue that it does not assume g_1 and g_2 are generators of the group $\mathbb{G}$ and thus we do not have to separately consider cases $g_1 = 1$ or $g_2 = 1$.

Zero-knowledge property. For zero-knowledge, we must to show that we are able to simulate protocol transcripts between the honest prover and honest verifier. Differently form the real protocol execution, we can generate the messages in different order than they appear in the protocol. Let us first construct a simulator that correctly creates the first and last message if the challenge β is known ahead. The corresponding simulator construction for is following:

$\mathcal{S}(\beta)$

$\gamma \xleftarrow{u} \mathbb{Z}_q$

$\alpha_1 \leftarrow g_1^{\gamma} \cdot y_1^{-\beta}$

$\alpha_2 \leftarrow g_2^{\gamma} \cdot y_2^{-\beta}$

return $(\alpha_1, \alpha_2, \beta, \gamma)$

Note that even if we fix β in the protocol runs and consider corresponding protocol transcripts of real protocol runs, the γ value still varies. More precisely, we can conclude that $\gamma = x\beta + r$ is uniformly random, since r is generated uniformly randomly from $\mathbb{Z}_q$. Thus $\mathcal{S}$ generates γ with correct distribution. By the protocol construction, the β and γ uniquely determine the values of α_1 and α_2:

$$\begin{aligned} g_1^\gamma = \alpha_1 y_1^\beta \qquad &\Leftrightarrow \qquad \alpha_1 = g_1^\gamma \cdot y_1^{-\beta} \ , \\ g_2^\gamma = \alpha_2 y_2^\beta \qquad &\Leftrightarrow \qquad \alpha_2 = g_2^\gamma \cdot y_2^{-\beta} \ . \end{aligned}$$

Since β is uniformly distributed in the real protocol runs, we can fully simulate the transcript between honest prover and honest verifier as follows:

$$\begin{array}{l} \mathsf{Sim} \\ \left\lfloor \begin{array}{l} \beta \xleftarrow{u} \mathbb{Z}_q \\ (\alpha_1, \alpha_2, \beta, \gamma) \leftarrow \mathcal{S}(\beta) \\ \textbf{return } (\alpha_1, \alpha_2, \beta, \gamma) \ . \end{array} \right. \end{array}$$

This construction can be further extended to simulate the output of semihonest verifier $\mathcal{V}_*$ without access to the prover. Let (g_1, g_2, y_1, y_2) be the verifier's input of the sigma protocol and let ϕ be some auxiliary information gathered by $\mathcal{V}_*$ so far and $\omega \in \Omega$ be the randomness used by $\mathcal{V}_*$. As $\mathcal{V}_*$ is semihonest, its challenge β does not depend on α_1 and α_2. More formally, we can express this as follows:

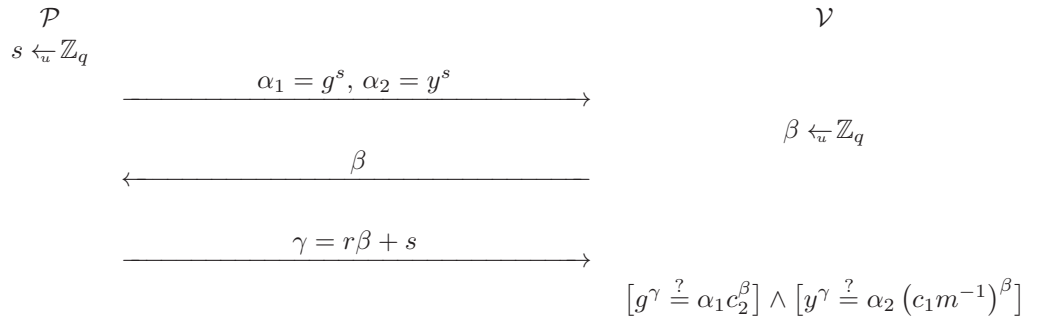
$$\forall \phi \in \{0,1\}^* \, , \, \forall \omega \in \Omega, \, \forall \alpha_1, \alpha_2 \in \mathbb{G} : \exists \beta \in \mathbb{Z}_q : \ \mathcal{V}_*(\phi, g_1, g_2, y_1, y_2, \alpha_1, \alpha_2; \omega) = \beta \ .$$

Consequently, the following simulator

$$\begin{array}{l} \mathsf{Sim}(\phi, g_1, g_2, y_1, y_2) \\ \left\lfloor \begin{array}{l} \omega \leftarrow \Omega \\ \alpha_1^* \xleftarrow{u} \mathbb{G}, \alpha_2^* \xleftarrow{u} \mathbb{G} \\ \beta \leftarrow \mathcal{V}_*(\phi, g_1, g_2, y_1, y_2, \alpha_1^*, \alpha_2^*; \omega) \\ (\alpha_1, \alpha_2, \beta, \gamma) \leftarrow \mathcal{S}(\beta) \\ \beta_* \leftarrow \mathcal{V}_*(\phi, g_1, g_2, y_1, y_2, \alpha_1, \alpha_2; \omega) \\ \textbf{return } \mathcal{V}_*(\gamma) \end{array} \right. \end{array}$$

provides a perfect simulation of real protocol runs between honest prover and semihonest verifier $\mathcal{V}_*$. Indeed, note that the semihonest behaviour assures that β_* coincides with β. Since $\mathcal{S}$ provides a valid response γ for the challenge β and there is only one valid response, the final output $\mathcal{V}_*(\gamma)$ would be the same as for interacting with the honest prover who has previously output commitment messages α_1 and α_2. Since we already proved that the simulator $\mathcal{S}$ generates α_1 and α_2 with the same distribution as the honest prover, we have the informal justification that the output distributions of the semihonest verifier is the same as in the real protocol runs. The formal proof is left as a tedious exercise for the reader.

Knowledge of plaintext. Note that ElGamal encryption is defined over a cyclic group $\mathbb{G}$ with prime number of elements as required by the sigma protocol analysed above and recall that a valid ElGamal ciphertext is in the form (g^r, my^r). If we do not want to hide the message m then we can prove that the ciphertext (c_1, c_2) corresponds to a message m by proving that we know r such that $c_1 = g^r$ and $c_2 \cdot m^{-1} = y^r$. Thus, we should take g_1 as the public generator g and g_2 as the public key $y = g^x$ in the sigma protocol. As a result, the prover proves knowledge of secret $r \in \mathbb{Z}_q$ such that $c_1 = g^r$ and $c_2 \cdot m^{-1} = y^r$, where the public information is (g, y, c_1, c_2, m). The resulting protocol itself is the following.



This protocol is structurally similar to the double-exponent sigma protocol discussed above and the corresponding security proofs for special-soundness and zero-knowledge properties are analogous. It is easy to see that the protocol is functional due to following equalities:

$$\begin{aligned} g^\gamma &= g^{r\beta+s} = g^s g^{r\beta} = \alpha_1 c_2^\beta \ , \\ y^\gamma &= y^{r\beta+s} = y^s y^{r\beta} = \alpha_2 \left(\frac{my^r}{m}\right)^\beta = \alpha_2 \left(\frac{c_1}{m}\right)^\beta \ . \end{aligned}$$

To prove knowledge of a plaintext corresponding to an ElGamal ciphertext without revealing the message, we can first assume that m is from some limited set $m \in \{m_1, m_2\}$. Then we can use the previous protocol to construct a standard disjunctive composition sigma protocols to prove knowledge of the plaintext without actually revealing it. Formally we will construct a sigma protocol for proving

$$\textsc{pok}[r \in \mathbb{Z}_q : \mathsf{Enc}(m_1, r) = (c_1, c_2) \vee \mathsf{Enc}(m_2, r) = (c_1, c_2)] \ ,$$

which is obviously equivalent to

$$\textsc{pok}[m \in \{m_1, m_2\}, r \in \mathbb{Z}_q : \mathsf{Enc}(m, r) = (c_1, c_2)] \ .$$

This construction can also be generalized to a larger set $\mathcal{M} \subseteq \mathbb{G}$, by constructing nested disjunctive proofs, but the communication complexity will be linear to the size of $\mathcal{M}$. Alternatively, every bit of a message $m \in \mathbb{G}$ could be encrypted separately, and the above protocol could be run in parallel for all the encrypted bits. That way communication complexity is much better: $\mathrm{O}(\log |\mathcal{M}|)$.