Exercise (Security of simultaneous message exchange protocol). *Analyse security of the following simplistic protocol for simultaneous message exchange*

$\mathcal{P}_1(x_1)$		$\mathcal{P}_2(x_2)$
	$\boxed{\mathsf{pk} \leftarrow \mathsf{Gen}}$	
	$\xleftarrow{\ \ \mathsf{pk}\ \ }$ - - - $\rightarrow$	
$(c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(x_1)$		
	$\xrightarrow{\ \ \mathsf{pk}, c\ \ }$	
	$\xleftarrow{\ \ x_2\ \ }$	
	$\xrightarrow{\ \ d\ \ }$	
		$x_1 \leftarrow \mathsf{Open}_{\mathsf{pk}}(c, d)$
return (x_1, x_2)		***return*** (x_1, x_2)

where bits x_1 and x_2 are private protocol inputs and a triple of algorithms $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *is a commitment scheme $\mathcal{C}om$ with appropriate properties. The dashed line denotes sub-protocol for fixing the commitment parameters. Prove that there exist an efficient simulator for $\mathcal{P}_1$.*

Solution.
Right ideal implementation. As the first party $\mathcal{P}_1$ can refuse to open its input based on the opponents input x_2, we must consider the idealised functionality where the first party $\mathcal{P}_1$ is in the dominant position:

$\mathcal{P}_1(x_1)$		$\mathcal{T}$		$\mathcal{P}_2(x_2)$
	$\xrightarrow{\ \ x_1\ \ }$		$\xleftarrow{\ \ x_2\ \ }$	
	$\xleftarrow{\ \ x_2\ \ }$			
	$\xrightarrow{\ \textsc{Abort}/\textsc{Continue}\ }$			
			$\xrightarrow{\ \ x_1\ \ }$	
return (x_1, x_2)				**return** (x_1, x_2)

High-level descriptions for simulator constructions. Assume that the first party $\mathcal{P}_1$ is malicious. Then the corresponding simulator Sim must first provide an input $\hat{x}_1$ to the trusted third party $\mathcal{T}$ who replies x_2. After that it can still abort ideal computations by sending Abort signal. If the commitment parameters are generated by $\mathcal{P}_1$, the corresponding simulator construction can be defined as follows

$$
\begin{array}{l}
\mathsf{Sim}(\phi_1, x_1)\\
\left[\begin{array}{l}
\omega_1 \xleftarrow{u} \Omega_1\\
(\mathsf{pk}, c) \leftarrow \mathcal{P}_1^*(\phi_1, x_1; \omega_1)\\
\hat{x}_1 \leftarrow \mathcal{K}(\phi_1, x_1, \omega_1)\\
\text{Send } \hat{x}_1 \text{ to } \mathcal{T} \text{ and store the reply as } x_2.\\
d \leftarrow \mathcal{P}_1^*(x_2)\\
\mathsf{if}\ \mathsf{Open}_{\mathsf{pk}}(c, d) = \perp\ \mathsf{then}\\
\quad \left[\begin{array}{l}\text{Send } \textsc{Abort} \text{ to } \mathcal{T}\\ \textbf{return } \mathcal{P}_2^*(\perp)\end{array}\right.\\
\mathsf{else\ if}\ \mathsf{Open}_{\mathsf{pk}}(c, d) = \hat{x}_1\ \mathsf{then}\\
\quad \left[\begin{array}{l}\text{Send } \textsc{Continue} \text{ to } \mathcal{T}\\ \textbf{return } \mathcal{P}_2^*(x_2)\end{array}\right.\\
\mathsf{else}\ \textbf{return}\ \mathsf{Fail}
\end{array}\right.
\end{array}
$$

If commitment parameters are generated by $\mathcal{P}_2$ then the input extractor $\mathcal{K}$ must accept pk as an extra argument and the resulting simulator is somewhat different

$$
\begin{array}{l}
\mathsf{Sim}(\phi_1, x_1)\\
\left[\begin{array}{l}
\omega_1 \xleftarrow{u} \Omega_1\\
\mathsf{pk} \leftarrow \mathsf{Gen}\\
c \leftarrow \mathcal{P}_1^*(\phi_1, x_1, \mathsf{pk}; \omega_1)\\
\hat{x}_1 \leftarrow \mathcal{K}(\phi_1, x_1, \mathsf{pk}, \omega_1)\\
\text{Send } \hat{x}_1 \text{ to } \mathcal{T} \text{ and store the reply as } x_2.\\
d \leftarrow \mathcal{P}_1^*(x_2)\\
\mathsf{if}\ \mathsf{Open}_{\mathsf{pk}}(c, d) = \perp\ \mathsf{then}\\
\quad \left[\begin{array}{l}\text{Send } \textsc{Abort} \text{ to } \mathcal{T}\\ \textbf{return } \mathcal{P}_2^*(\perp)\end{array}\right.\\
\mathsf{else\ if}\ \mathsf{Open}_{\mathsf{pk}}(c, d) = \hat{x}_1\ \mathsf{then}\\
\quad \left[\begin{array}{l}\text{Send } \textsc{Continue} \text{ to } \mathcal{T}\\ \textbf{return } \mathcal{P}_2^*(x_2)\end{array}\right.\\
\mathsf{else}\ \textbf{return}\ \mathsf{Fail}
\end{array}\right.
\end{array}
$$

(A) INPUT EXTRACTOR FOR $\mathcal{P}_1$. As the simulator must work universally well for all inputs (ϕ_1, x_1), the actual input $\hat{x}_1$ must be extracted form $\mathcal{P}_1$ in a black-box manner. Indeed, consider a specific adversary $\mathcal{P}_1^*$ that treats input ϕ_1 as the code and just interprets it to determine its actions. Depending on the precise computational model, such an interpretation is either linearly or quadratically slower than the dedicated attacker $\hat{\mathcal{P}}_1^*$ and we loose in efficiency. However, we gain universality – the input extractor $\mathcal{K}$ must works for all these attacks. Theoretically, the extractor $\mathcal{K}$ can use the code ϕ_1 to fine-tune its actions. However, so far nobody knows how to efficiently extract information from the program code and thus black-box execution with rewinding is the only known input extraction strategy.

The extractions itself depends who creates commitment parameters. If pk is generated inside $\mathcal{P}_1^*$ then initial inputs (ϕ_1, x_1, ω_1) together with x_2 completely fix the behaviour of $\mathcal{P}_1^*$. Consequently, we can consider the following extraction strategy

$$
\begin{array}{l}
\mathcal{K}(\phi_1, x_1, \omega_1) \\
\left[\begin{array}{l}
(\mathsf{pk}, c) \leftarrow \mathcal{P}_1^*(\phi_1, x_1, \omega_1) \\
\ \text{For } x_2 \in \mathcal{X}_2 \text{ do} \\
\ \left[\begin{array}{l}
d \leftarrow \mathcal{P}_1^*(x_2) \\
\hat{x}_1 \leftarrow \mathsf{Open}_{\mathsf{pk}}(c, d) \\
\mathsf{if}\ \hat{x}_1 \neq \bot\ \mathsf{then}\ \mathbf{return}\ \hat{x}_1
\end{array}\right. \\
\mathbf{return}\ \bot
\end{array}\right.
\end{array}
$$

where $\mathcal{X}_2$ is the set of all possible input values of the opponent $\mathcal{P}_2$. If pk is generated externally to $\mathcal{P}_1^*$ then we additionally need pk to completely fix the behaviour of $\mathcal{P}_1^*$. Thus, the plumbing between different components slightly changes

$$
\begin{array}{l}
\mathcal{K}(\phi_1, x_1, \mathsf{pk}, \omega_1) \\
\left[\begin{array}{l}
c \leftarrow \mathcal{P}_1^*(\phi_1, x_1, \mathsf{pk}, \omega_1) \\
\ \text{For } x_2 \in \mathcal{X}_2 \text{ do} \\
\ \left[\begin{array}{l}
d \leftarrow \mathcal{P}_1^*(x_2) \\
\hat{x}_1 \leftarrow \mathsf{Open}_{\mathsf{pk}}(c, d) \\
\mathsf{if}\ \hat{x}_1 \neq \bot\ \mathsf{then}\ \mathbf{return}\ \hat{x}_1
\end{array}\right. \\
\mathbf{return}\ \bot\ .
\end{array}\right.
\end{array}
$$

Prove the following facts

- If the commitment is perfectly binding then the protocol output y_2 of $\mathcal{P}_2$ is the same in the real and ideal world. Note that the output is completely determined by the values $(\phi_1, x_1, \omega_1, x_2)$ and thus can be considered as a deterministic function $y_2(\phi_1, x_1, \omega_1, x_2)$.
- Show that iteration over all possible values $\mathcal{X}_2$ is essential for black-box extraction. For that you may consider the following adversary that releases x_1 only for a specific input x_2:

$$
\begin{array}{l}
\mathcal{P}_2^*(\phi_1, x_1) \\
\left[\begin{array}{l}
(\hat{x}_1, \hat{x}_2) \leftarrow \phi_1 \\
\mathsf{pk} \leftarrow \mathsf{Gen} \\
(c, d) \leftarrow \mathsf{Open}_{\mathsf{pk}}(\hat{x}_1) \\
\text{Release } \mathsf{pk} \text{ and } c.\ \text{Store } x_2 \\
\mathsf{if}\ x_2 = \hat{x}_2\ \mathsf{then}\ \text{Release } d \\
\mathsf{else}\ \text{Abort}
\end{array}\right.
\end{array}
$$

  Show that if $\mathcal{K}$ does not use some $x_2 \in \mathcal{X}_2$ to extract input $\hat{x}_1$ then there exists inputs (ϕ_1, x_1) and (ϕ_2, x_2) such that the outcomes of $\mathcal{P}_2$ are completely different in the real and ideal world.
- What does the previous result mean in terms of size of the input domains $\mathcal{X}_2$.
- Show that the entire simulation construction is valid, i.e., the joint output distribution of $\mathcal{P}_1$ and $\mathcal{P}_2$ are identical if the commitment scheme is perfectly binding.
- Analyse what changes if we consider the setting with computationally binding commitments where pk is provided by $\mathcal{P}_2$.