Homomorphic Encryption and Secure Cloud Computing

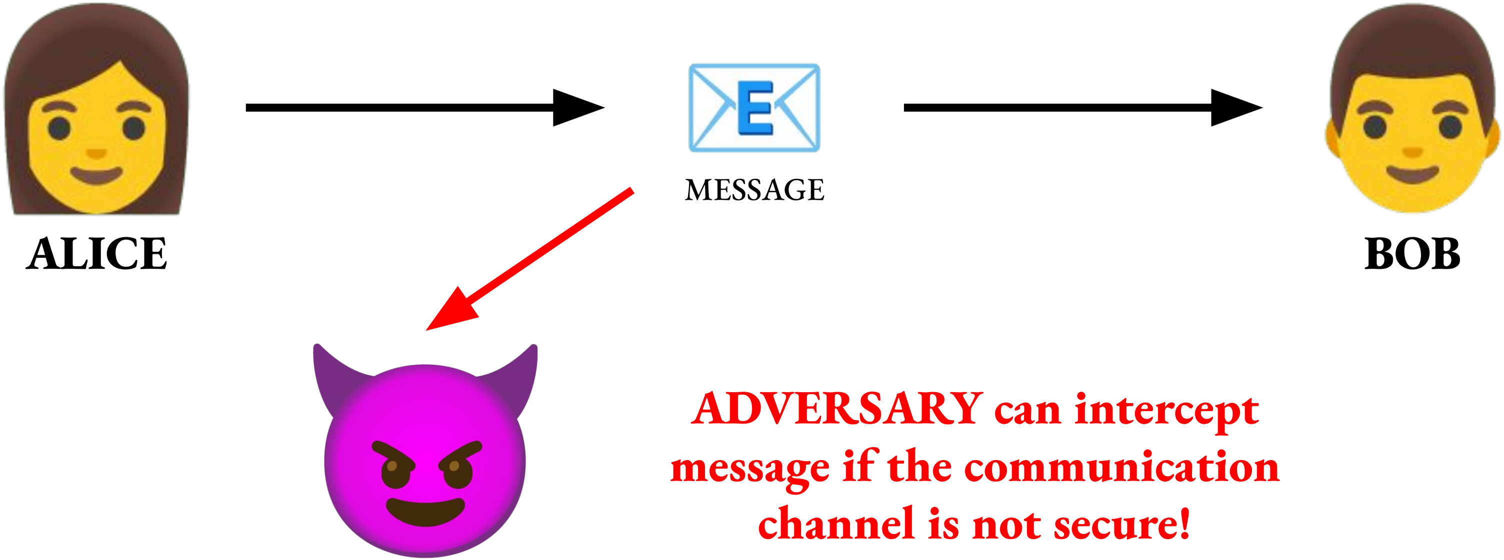
Kyle Yates - Clemson University

Cryptography

Suppose Alice wants to send a message to Bob. Alice sends her message.

Cryptography

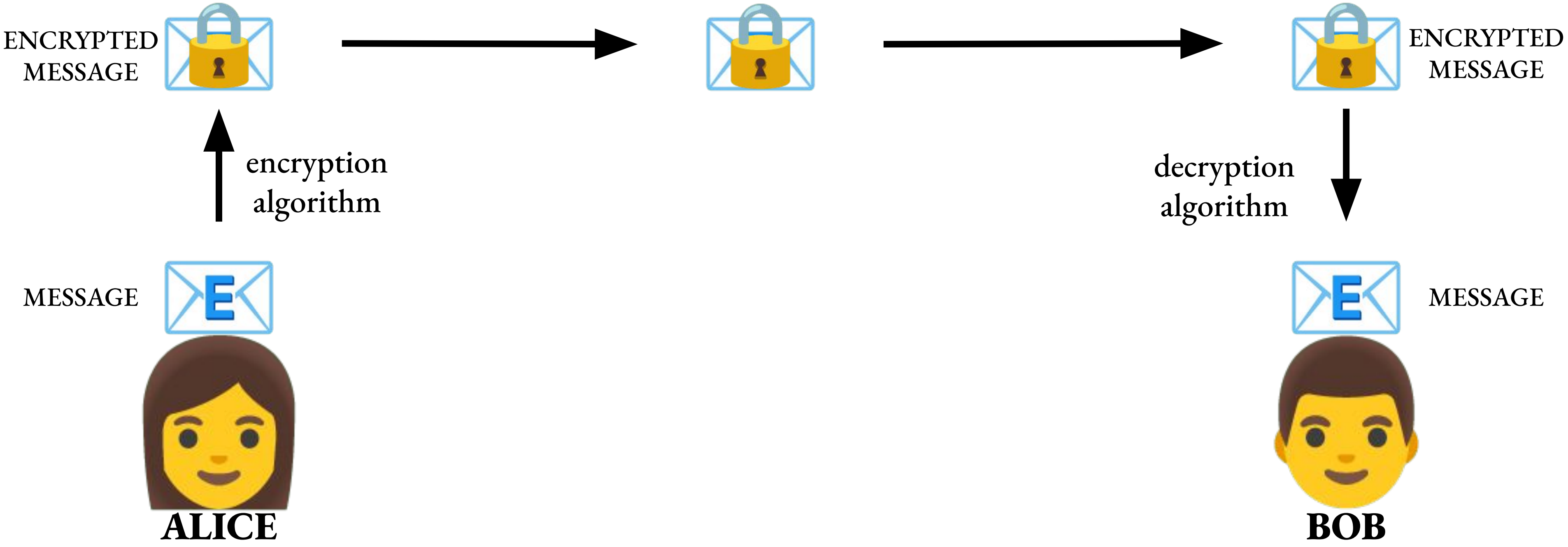
Suppose Alice wants to send a message to Bob. Alice sends her message.



ADVERSARY can intercept message if the communication channel is not secure!

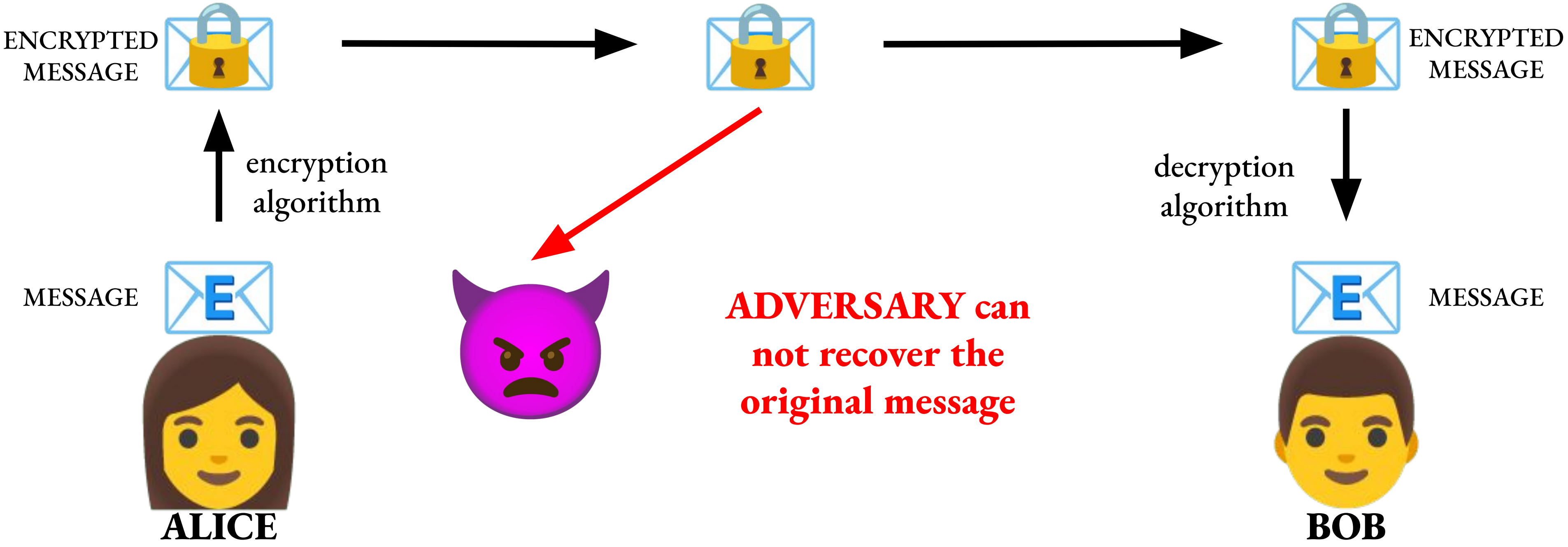
Cryptography

Suppose Alice wants to send a message to Bob. Alice first encrypts her message using some secret information that only her and Bob know.



Cryptography

Suppose Alice wants to send a message to Bob. Alice first <u>encrypts</u> her message using some secret information that only her and Bob know.



Homomorphic Encryption

Homomorphic encryption describes encryption schemes in which addition and multiplication can be performed on encrypted messages (ciphertexts). That is, for messages m_0, m_1 and key k

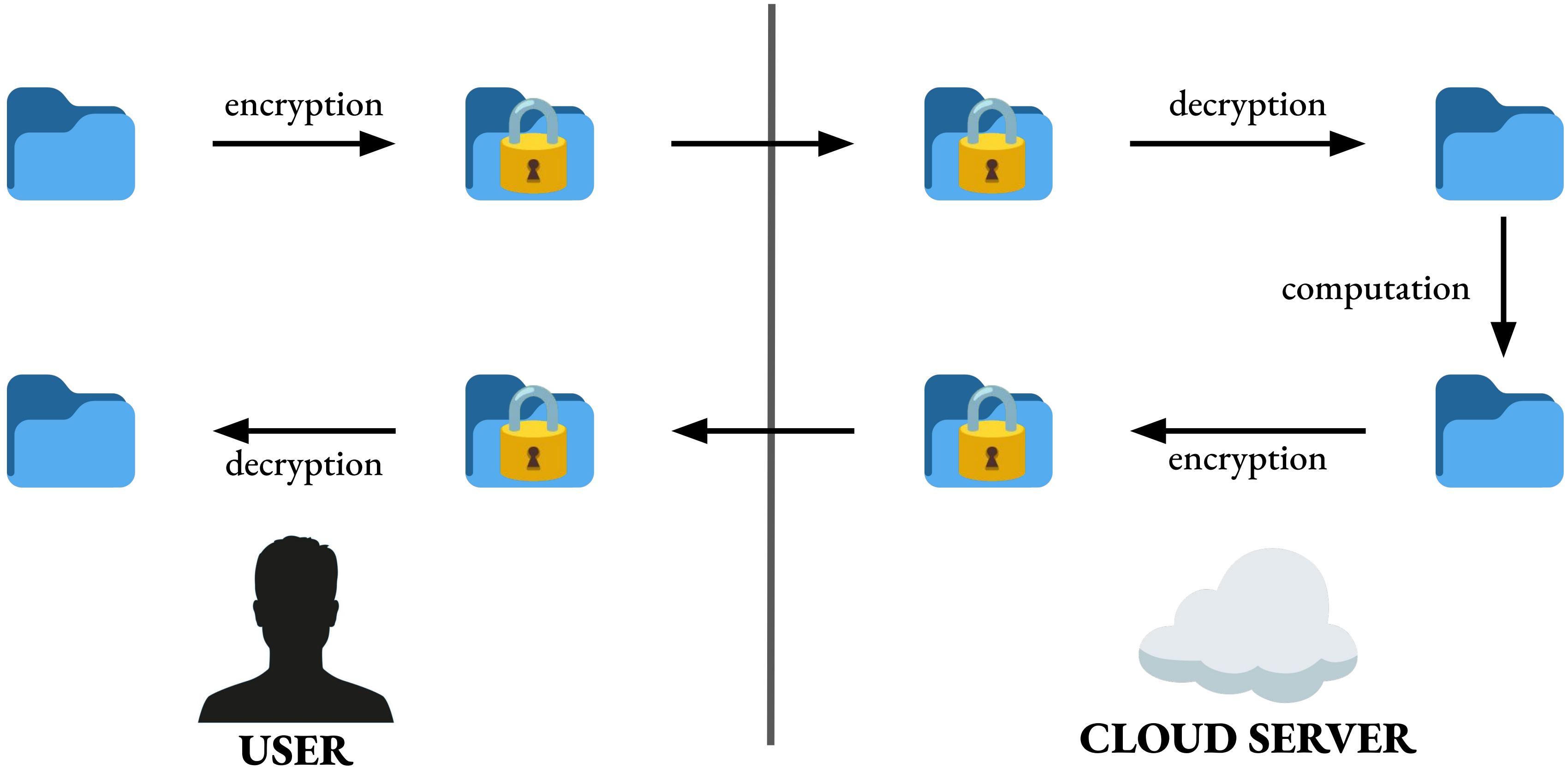
$$\texttt{Enc}(m_0, \mathrm{k}) + \texttt{Enc}(m_1, \mathrm{k}) = \texttt{Enc}(m_0 + m_1, \mathrm{k})$$

$$\texttt{Enc}(m_0, \mathrm{k}) \times \texttt{Enc}(m_1, \mathrm{k}) = \texttt{Enc}(m_0 \times m_1, \mathrm{k})$$

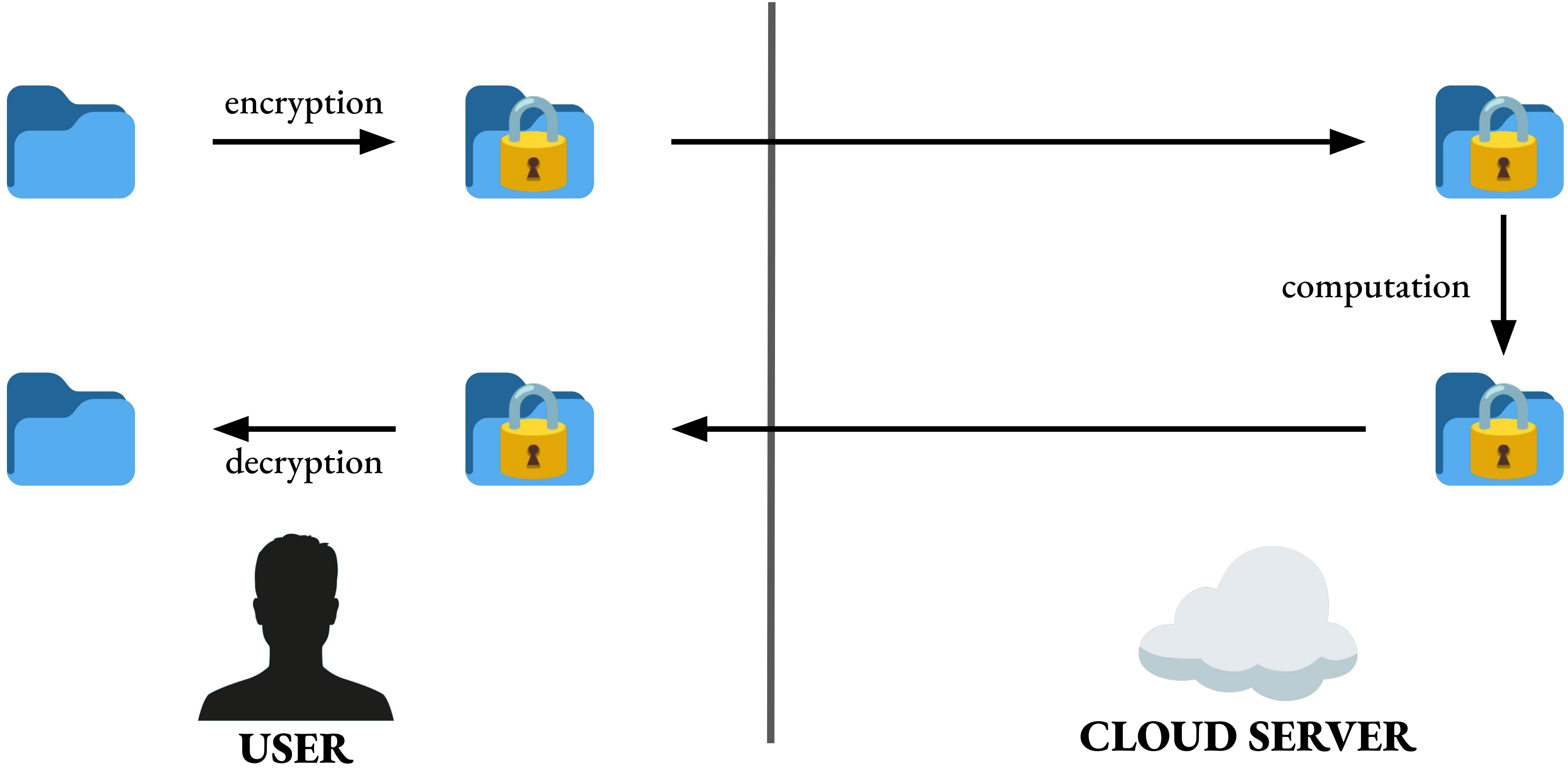
Furthermore, we can compute on ciphertexts without revealing any private information, which includes the secret key and messages.

This allows us to outsource computations to third parties. These third parties can compute on data, while at the same time learning nothing about the data!

Traditional Cloud Computing Model



Secure Cloud Computing Model



How Does Homomorphic Encryption Work?

Homomorphic encryptions schemes are based on the hardness of the ring learning with errors (RLWE) problem. Define the following polynomial rings.

$$R_n := \mathbb{Z}[x]/(\Phi(x))$$

$$R_{n,q} := \mathbb{Z}_q[x]/(\Phi(x)) \cong \mathbb{Z}[x]/(\Phi(x), q)$$

Choose $s \in R_{n,q}$ secret and sample a polynomial $e \leftarrow \chi(R_n)$ such that $\|e\|_\infty \leq \rho$. Sample $a \leftarrow U(R_{n,q})$ and compute $b \in R_{n,q}$ via $b = [-as + e]_{\Phi(x),q}$. The ordered pair $(a, b) \in R_{n,q}^2$ is called an *RLWE sample*.

Given many samples $(a_i, b_i) \in R_{n,q}^2$, the *RLWE problem* is to find s. This problem is known to be at least as hard as many worst-case lattice problems.

$\texttt{BFV.Encrypt}(m_0, \texttt{pk})$	
Input:	$m_0 \in R_{n,t}$ message, $\texttt{pk} = (a, b) \in R_{n,Q}^2$ public key.
Output:	$\texttt{ct}_0' \in R_{n,q}^2$.
Step 1.	Generate a random $u \in R_{n,3}$.
Step 2.	Sample $e_0', e_0'' \leftarrow \chi(R_n)$ such that $\|e_0'\|_\infty, \|e_0''\|_\infty \leq \delta_R$.
Step 3.	Compute $\texttt{ct}_0 = (a_0, b_0) \in R_{n,Q}^2$ where $a_0 = [au + e_0']_{\Phi(x),Q}$ and $b_0 = [bu + Dm_0 + e_0'']_{\Phi(x),Q}$.
Step 4.	Compute $(a_0', b_0') = \texttt{BFV.Modreduce}(Q, q, \texttt{ct}_0)$.
Step 5.	Return $\texttt{ct}_0' = (a_0', b_0') \in R_{n,q}^2$.

Algorithm 3: $\texttt{BFV}$ Encryption

	$\texttt{BFV.Multiply}(\texttt{ct}_0, \texttt{ct}_1)$
Input:	$\texttt{ct}_0 = (a_0, b_0), \texttt{ct}_1 = (a_1, b_1) \in R^2_{n,q}$ BFV ciphertexts.
Output:	$(c'_0, c'_1, c'_2) \in R^3_{n,q}$.
Step 1.	Compute $c_0 = [b_0 b_1]_{\Phi(x)}, c_1 = [b_1 a_0 + b_0 a_1]_{\Phi(x)}$, and $c_2 = [a_0 a_1]_{\Phi(x)}$.
Step 2.	Compute $c'_0 = \lfloor t c_0 / q \rceil, c'_1 = \lfloor t c_1 / q \rceil$, and $c'_2 = \lfloor t c_2 / q \rceil$.
Step 3.	Return (c'_0, c'_1, c'_2).

Algorithm 7: BFV Multiplication

Research Topics and Results

There are many active research topics in homomorphic encryption. I currently study noise growth during homomorphic operations and how it relates to number of computations allowed and precision.

Resources on Homomorphic Encryption

Interested in learning more about homomorphic encryption? Check out FHE.org!