Cryptography for the Cloud

David Pointcheval
ENS - CNRS - INRIA

Cyber-Sécurité - SPECIF
CNAM, Paris, France - November 7th, 2014

CryptoCloud

The Cloud


iCloud
Drive


Dropbox


SkyDrive


Google Drive


amazon cloud drive

Access from Anywhere

Available for Everything

One can

- Store documents, photos, etc
- Share them with colleagues, friends, family
- Process the data
- Ask queries on the data

With Current Solutions

The Cloud provider

- knows the content
- and claims to actually
  - identify users and apply access rights
  - safely store the data
  - securely process the data
  - protect privacy

But…

For economical reasons, by accident, or attacks

- data can get deleted
- any user can access the data
- one can log
  - all the connected users
  - all the queries

to analyze and sell/negotiate the information

Requirements

Users need more

- Storage guarantees
- Privacy guarantees
  - **confidentiality** of the data
  - **anonymity** of the users
  - **obliviousness** of the queries

Proof of storage

Encryption

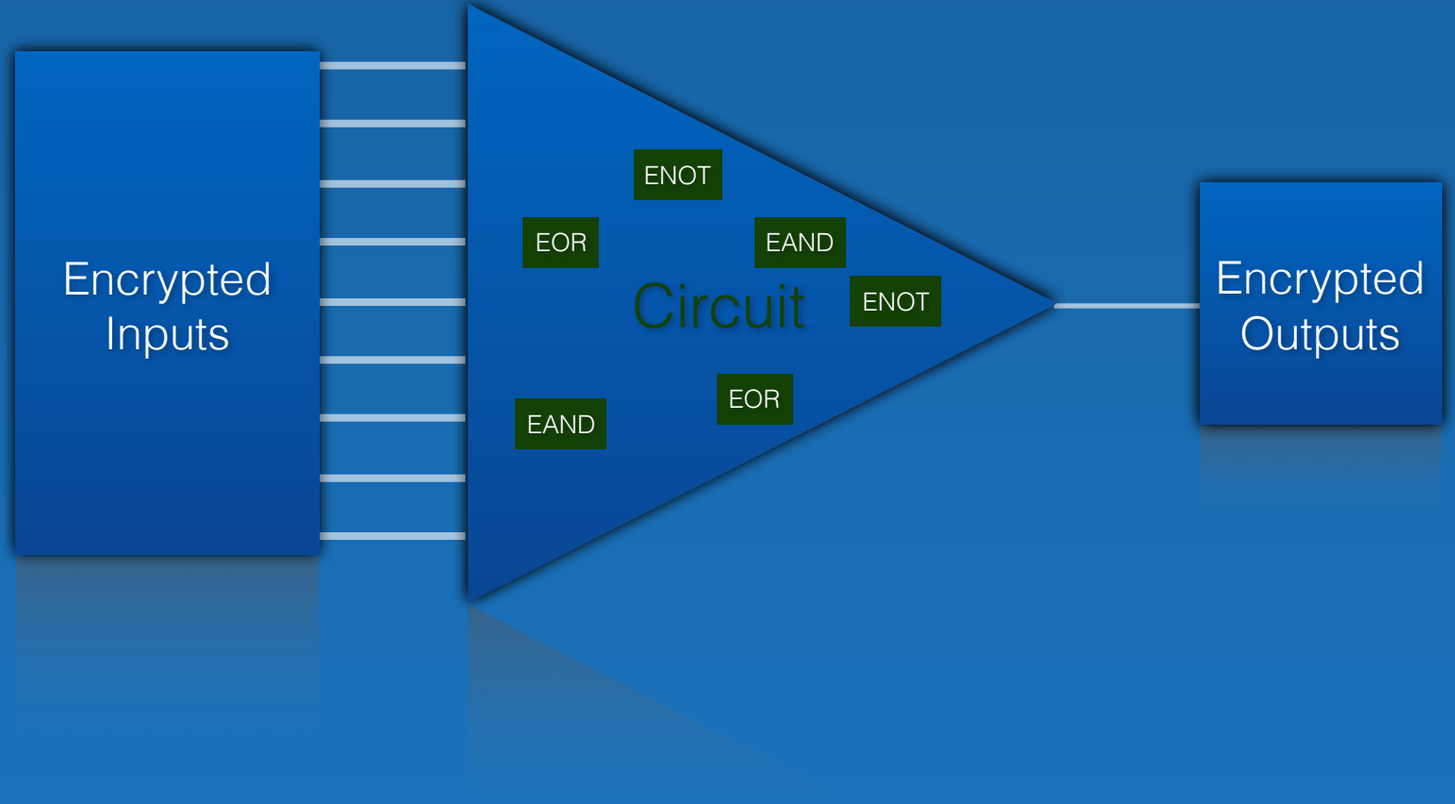
Anonymous Authentication

Private Information Retrieval

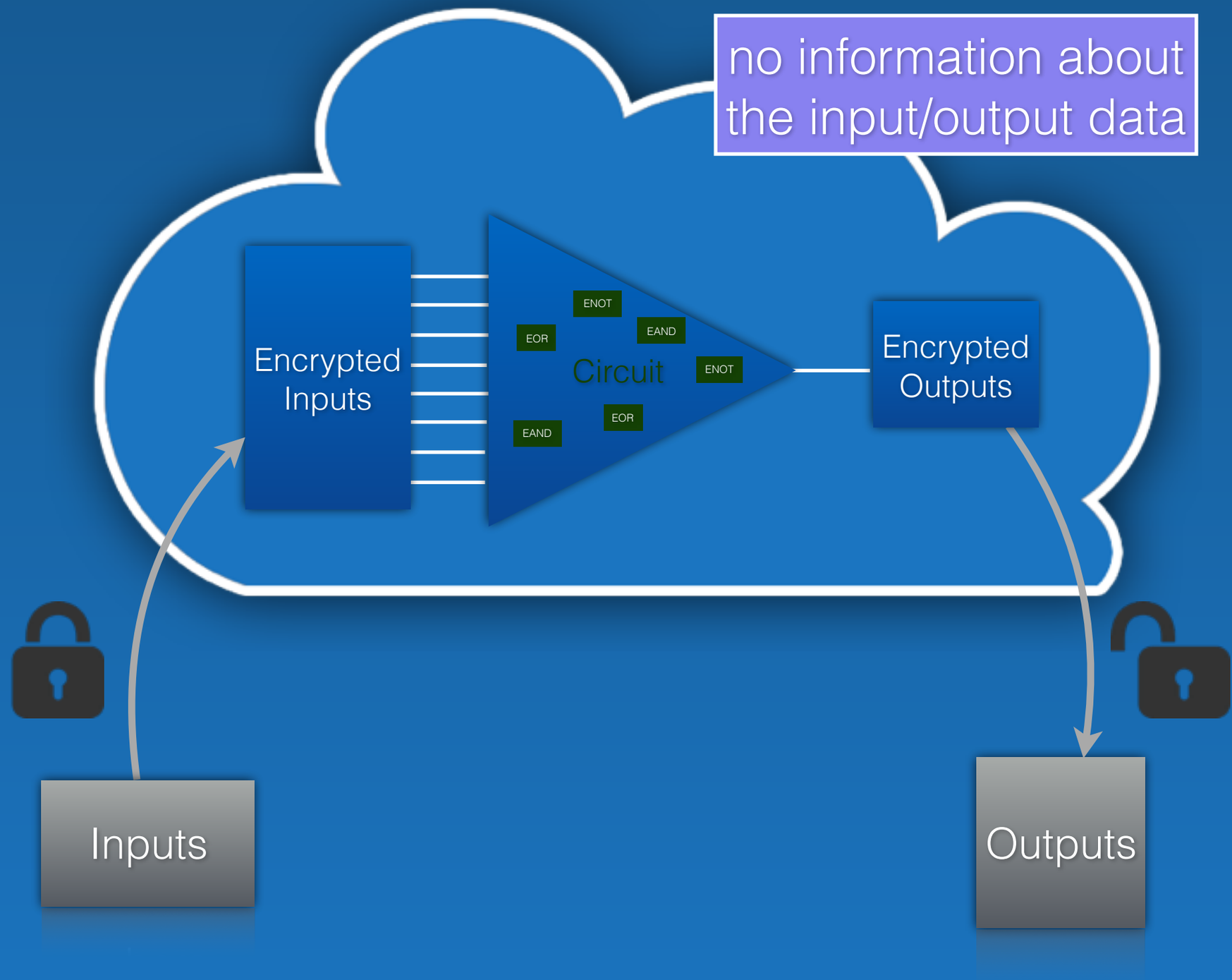
How to process users' queries?

FHE: The Killer Tool

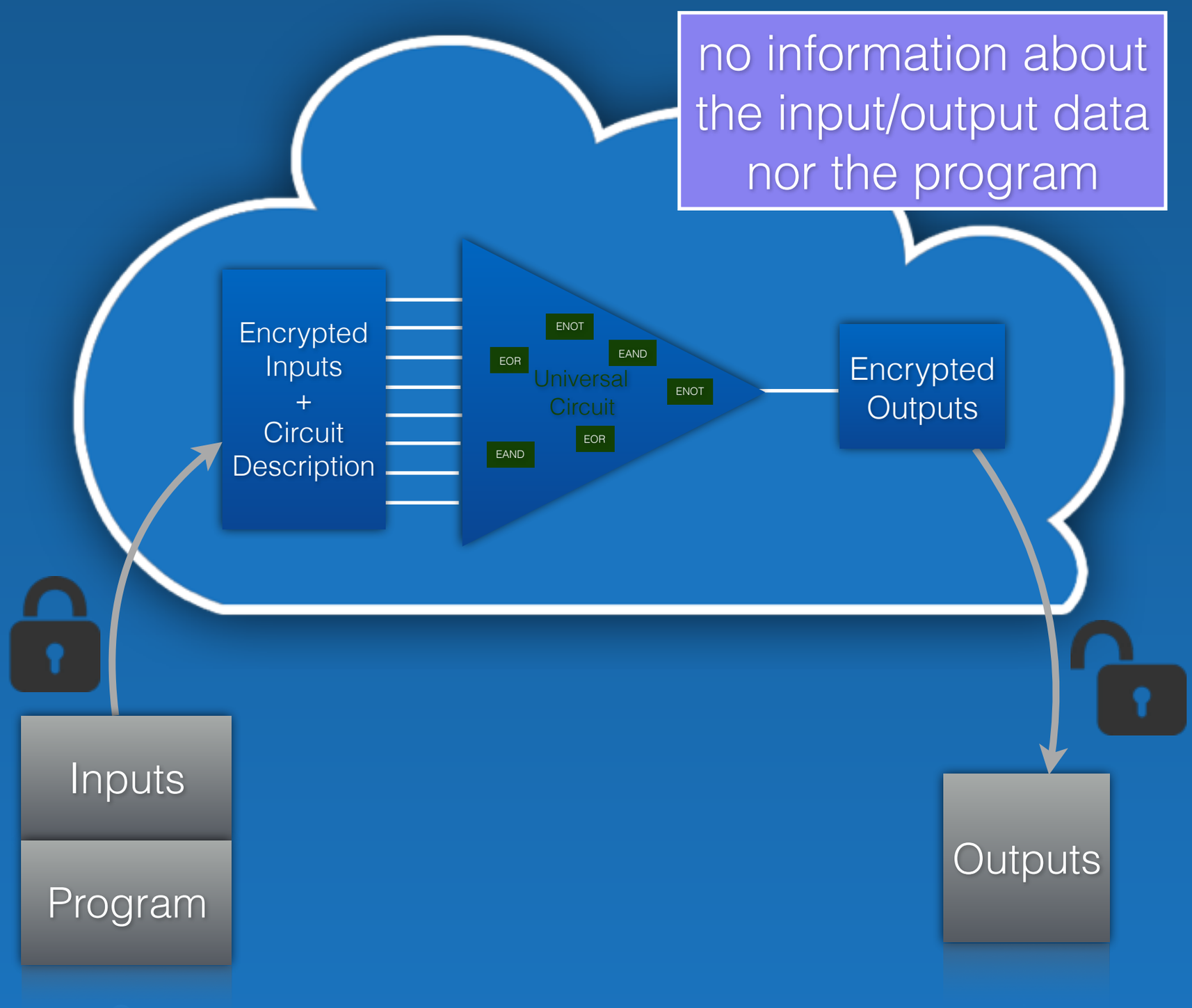
Fully Homomorphic Encryption allows to process encrypted data, and get the encrypted output



Outsourced Processing



Private Computations



FHE: Ideal Solution?

- Allows private storage
- Allows private computations
  - Private queries in an encrypted database
  - Private « googling »
- The provider does not learn
  - the content
  - the queries
  - the answers

Privacy by design…

… But each gate requires a few minutes…

Confidentiality & Sharing

Encryption allows to protect data

- the provider stores them without knowing them
- nobody can access them either, except the owner

How to share them with friends?

- Specific people have full access to some data:
  with public-key encryption with multiple recipients
- Specific people have specific access
  such as partial information, statistics or agregation

Broadcast Encryption

But a **key manager** generates decryption keys…

Requirement to avoid trusted authority:

- either a broadcast system per sender
- or a **decentralized (*ad-hoc*) broadcast encryption scheme**

Functional Encryption

The sender generates sub-keys $\mathbf{K}_f$:

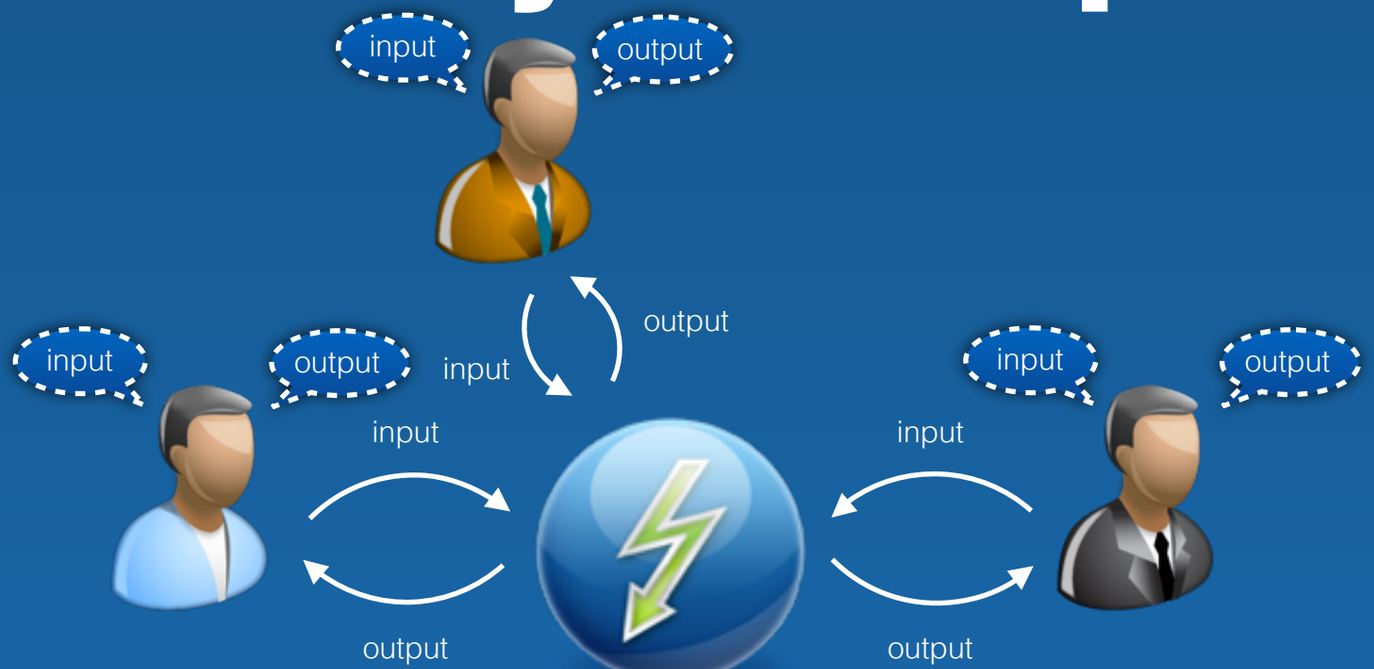
- From C = **Encrypt**(m), **Decrypt**($\mathbf{K}_f$, C) outputs $f(m)$
- This allows controlled sharing of data
- But receivers' keys are **specific** to the sender: many secrets!

Interactivity

- Fully Homomorphic Encryption is non-interactive
  - all the computations to be done in the Cloud
  - without any additional hint from the user
- Decentralized Broadcast Encryption
  - impossible without interactions between the players
- Some interactivity can help
  - additional hints from the user
  - but… latency issues

Efficient & Secure *two/multi-party* Computation

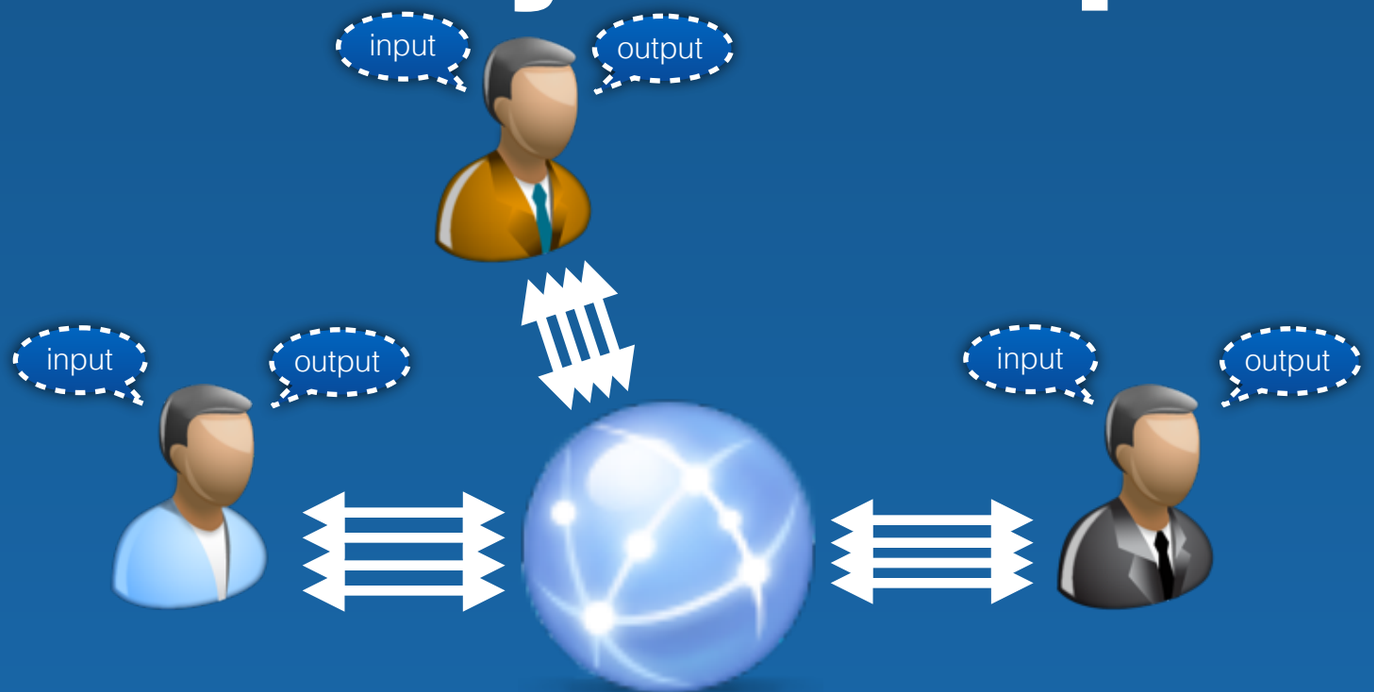
Multi-Party Computation



- **Secure** Multi-Party Computation
  - Ideally: each party gives its input and just learns its output

for **any** ideal functionality

Multi-Party Computation

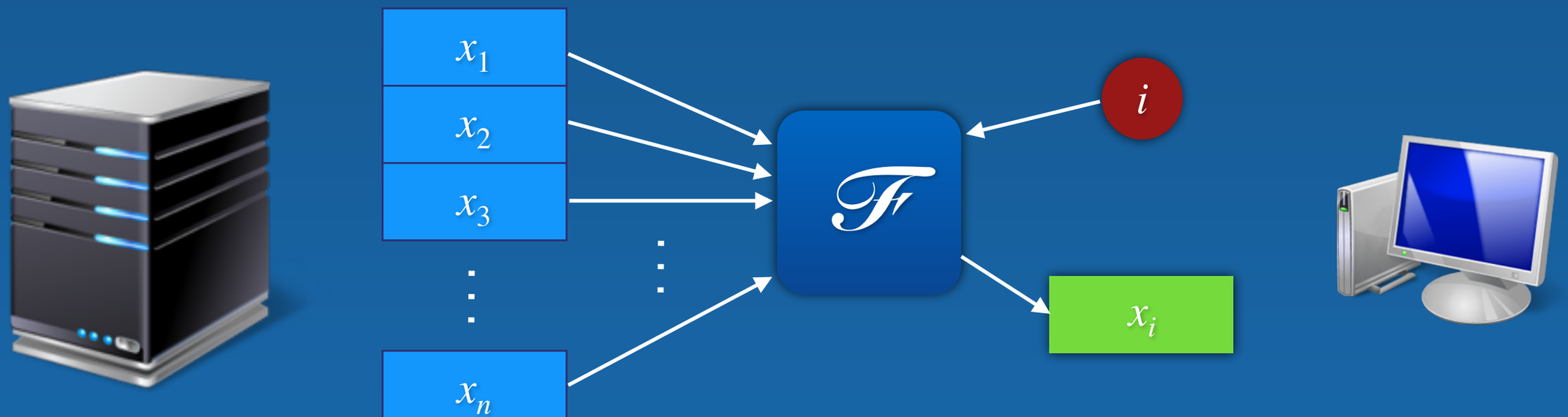


- **Secure** Multi-Party Computation
  - Ideally: each party gives its input and just learns its output

    for **any** ideal functionality
  - In practice: **many interactions** between the parties

Latency too high over Internet……

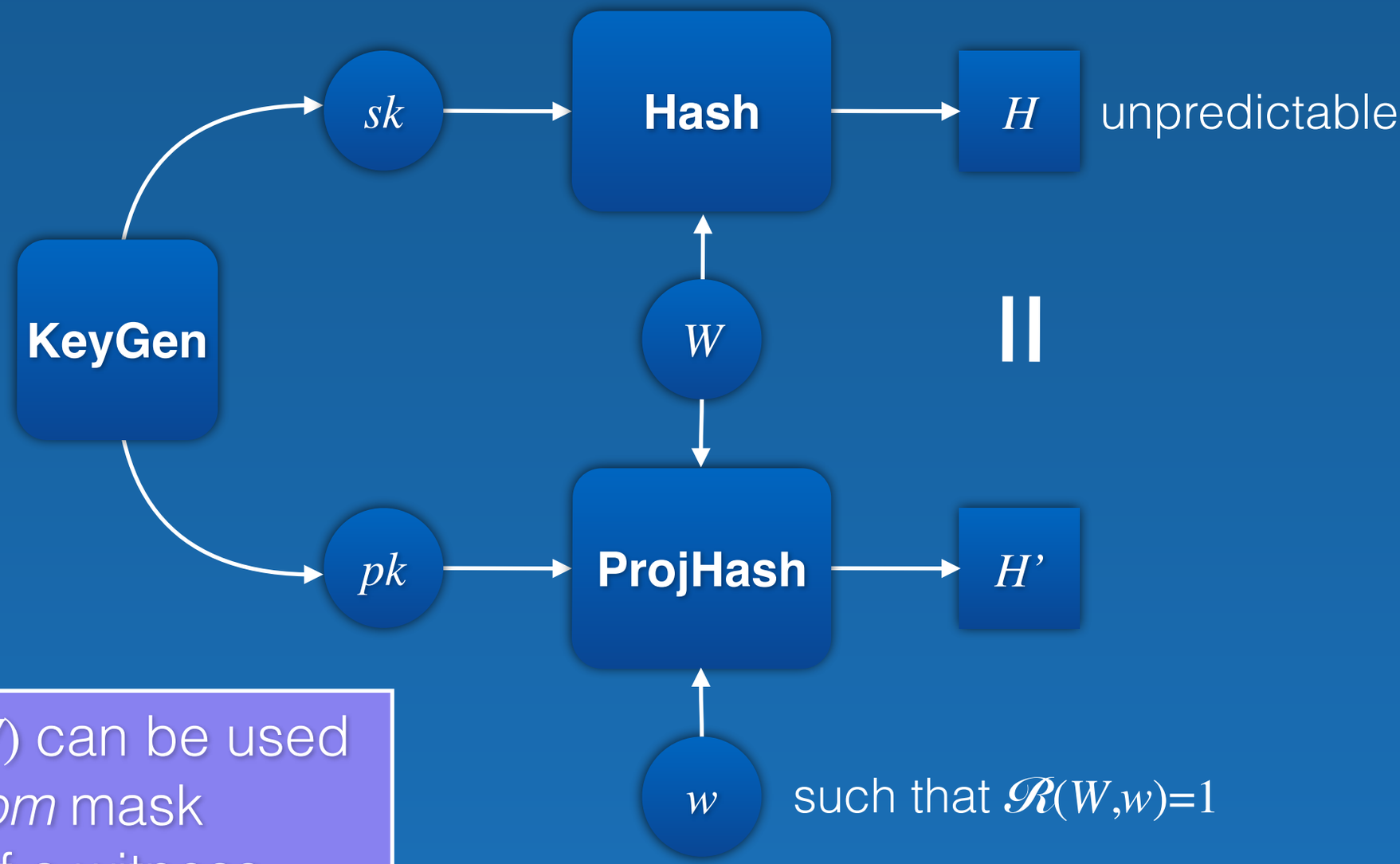
Ex: Private DB Queries



Obliviousness of the queries

- The client learns x_i and nothing else
- The database does not learn anything

Projective Hashing



H=**Hash**(sk,W) can be used as a *random* mask
Knowledge of a witness w required to unmask

PH and Private Queries

- $\mathcal{L}_1$ **and** $\mathcal{L}_2$ two distinct **NP**-hard languages in $\mathcal{X}$
- A word W_i is in $\mathcal{L}_i$ if and only if there exists a witness w_i such that $\mathcal{R}_i(W_i,w_i)=1$

$(sk_i,pk_i)_i \leftarrow$ **KeyGen**

$H_i =$ **Hash**(sk_i,W) $\xleftarrow{W}$ $W \leftarrow \mathcal{L}_b$

$C_i = M_i \oplus H_i$ $\xrightarrow{(pk_i,C_i)_i}$ $H_b' =$ **ProjHash**(pk_b,W,w)

$M_b = C_b \oplus H_b'$

Secure Oblivious Transfer

Diffie-Hellman

One can efficiently instantiate Projective Hashing with Diffie-Hellman

- $\mathcal{L} = \{(g^r, h^r)\} \subset \mathcal{X} = \{(g^r, h^s)\} = \mathbf{G}^2$
- $sk = (a,b)$ and $pk = g^a h^b$

$$H = \textbf{Hash}(sk,(u,v)) = u^a v^b = pk^r = \textbf{ProjHash}(pk,(u,v),r) = H'$$

if $(u,v) = (g^r, h^r) \in \mathcal{L}$

if $(u,v) \notin \mathcal{L}$, H is random, perfectly independent of pk

Evaluations and communications are quite efficient

Conclusion

- **Threat**

  However strong the trustfulness of the Cloud provider may be, any system or human vulnerability can be exploited against privacy

- **Cryptography for the Cloud**

  Fully-safe access to the Cloud for everybody whatever the behavior of the provider

  - The provider is just trusted to
    - store the data (*can be controlled*)
    - process and answer any request (*or DoS*)

A privacy breach cannot be checked