Privacy-Preserving Computations

David Pointcheval
CNRS - ENS/PSL - INRIA

GDR Sécurité - Keynote
June 30th 2021

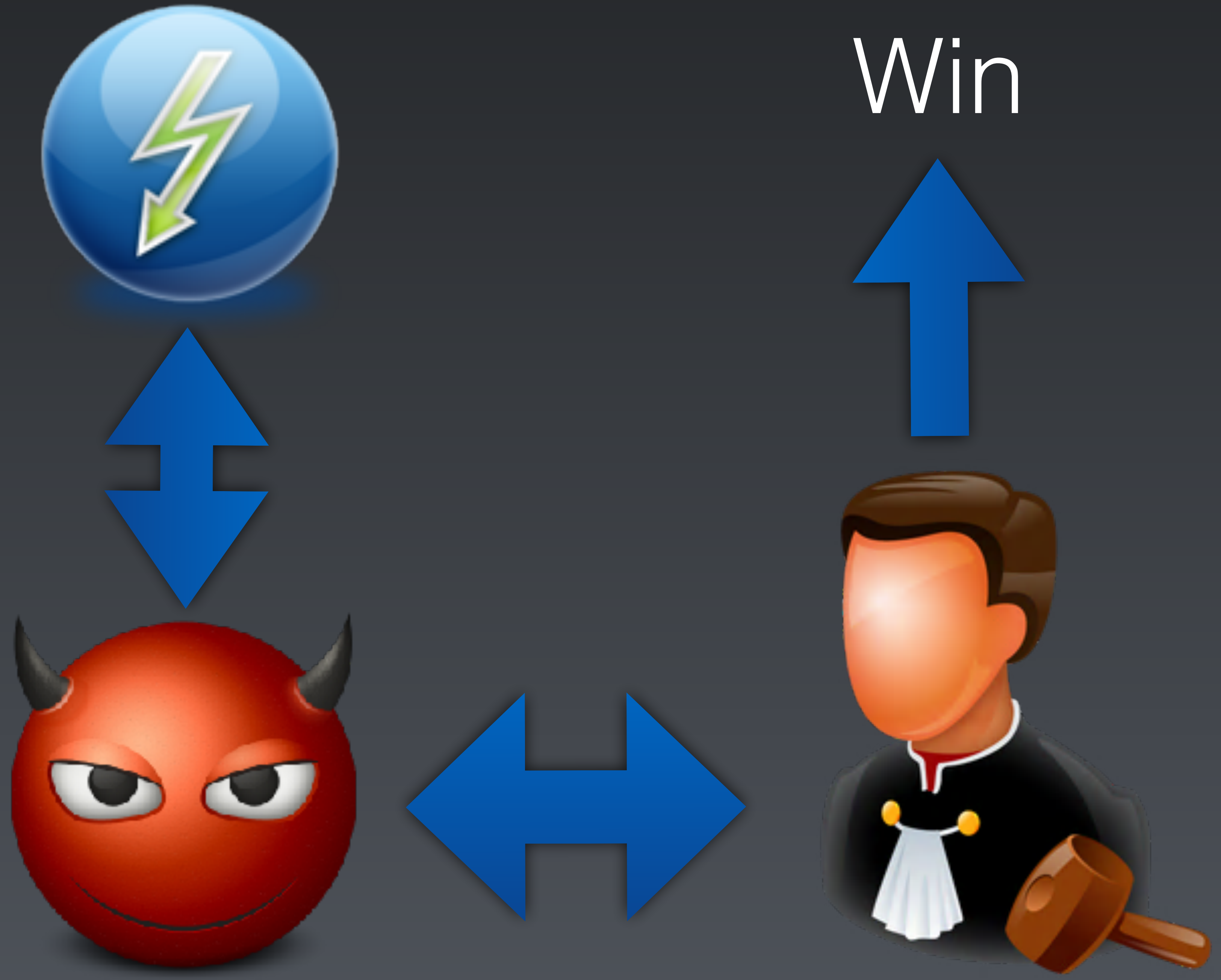
Security of Communications

- One ever wanted to exchange information securely
- With the all-digital world, security needs are even stronger: communication devices are
  - in your pocket

  - at home

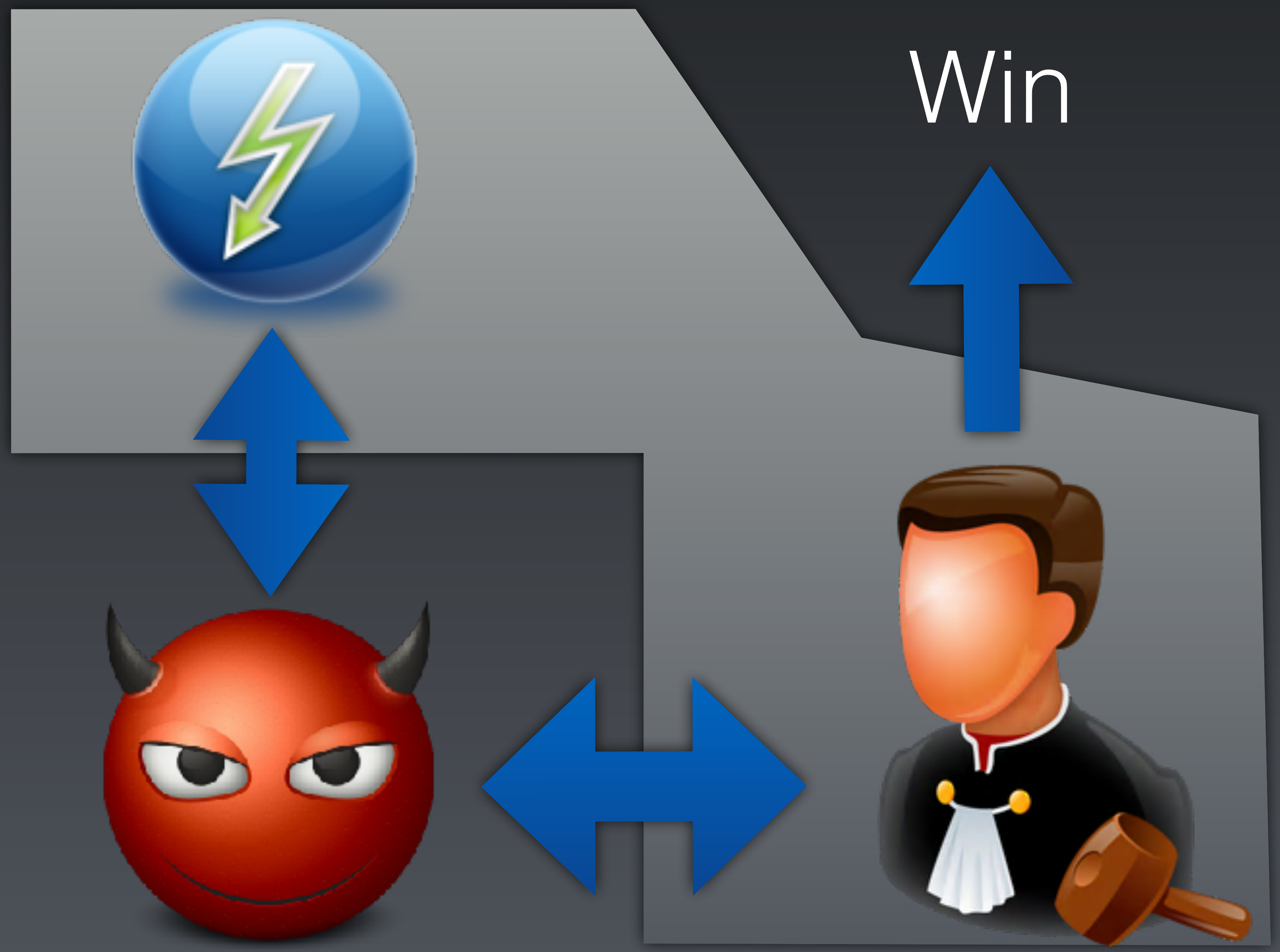
Provable Security

- If the adversary **A** can win the security game **G** within time t with probability ε



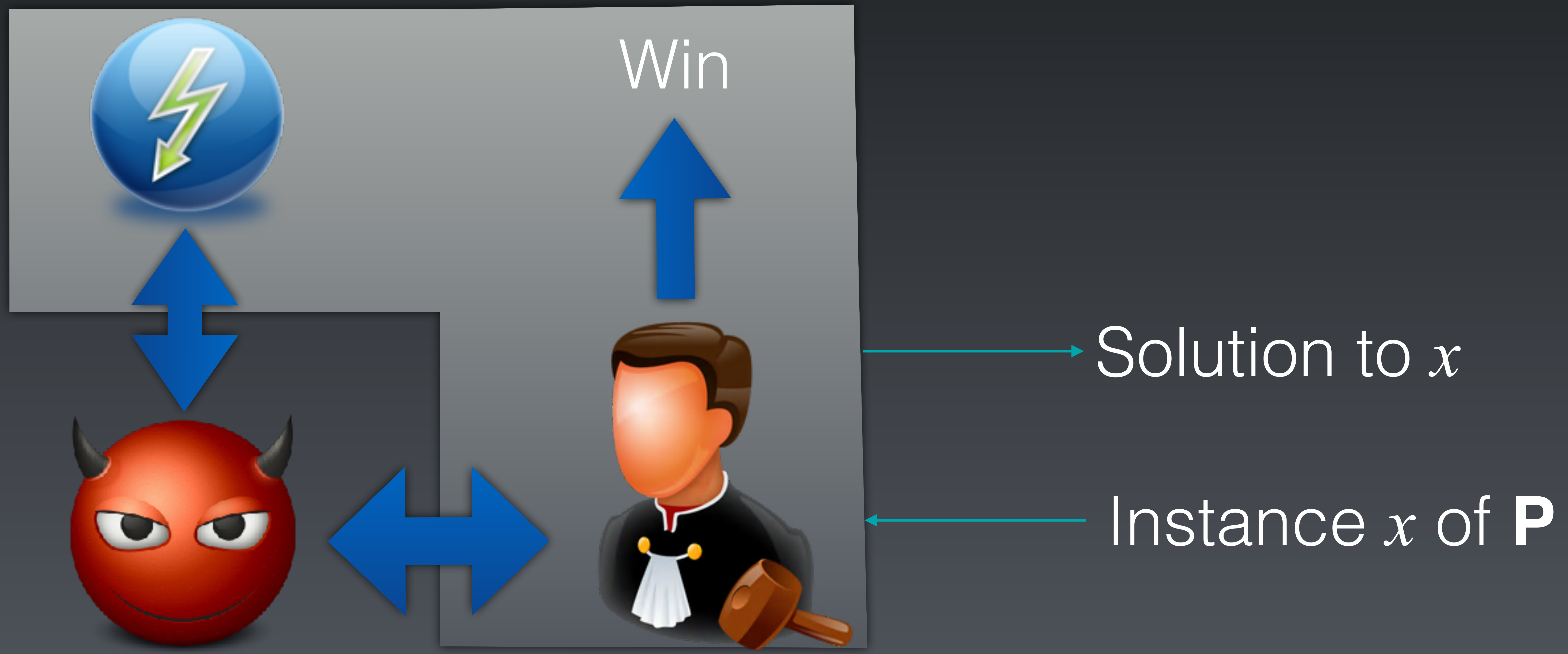
Provable Security

- If the adversary **A** can win the security game **G** within time t with probability ε



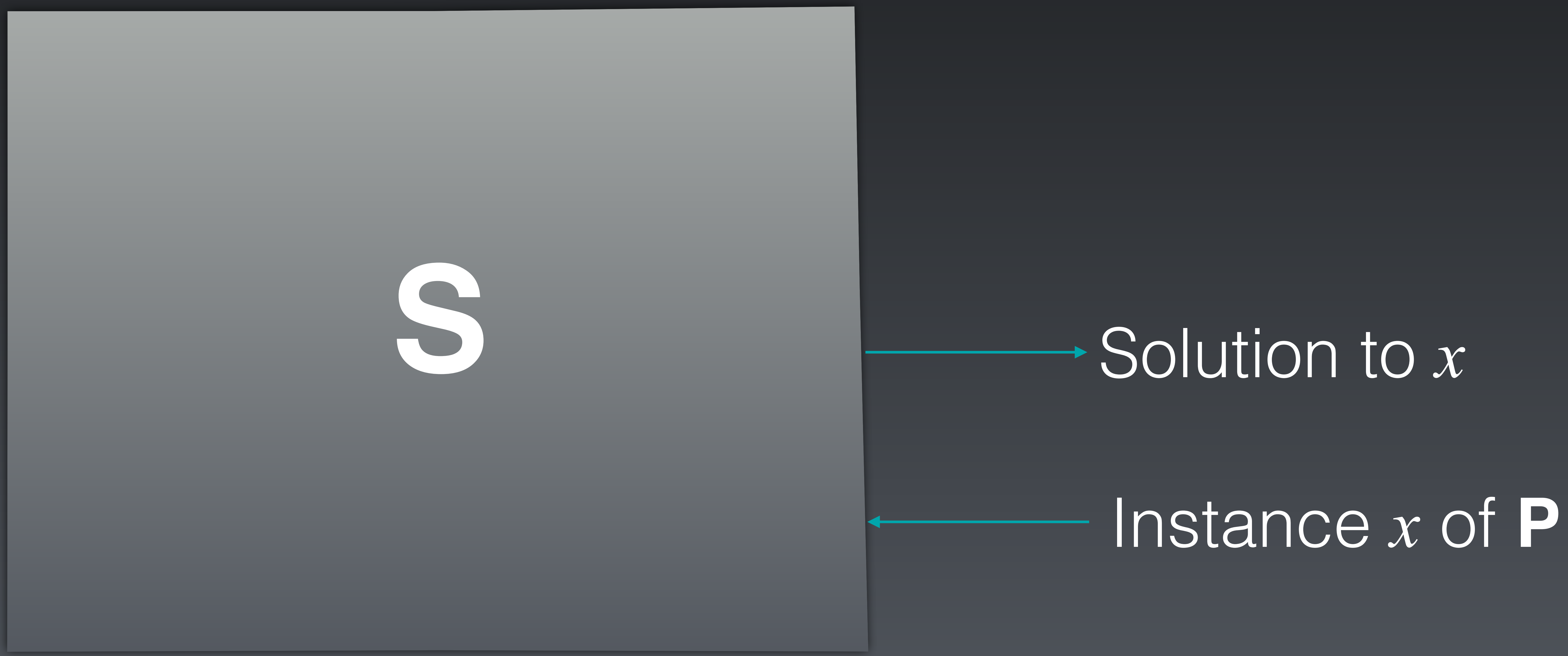
Provable Security

- If the adversary **A** can win the security game **G** within time t with probability ε



Provable Security

- If the adversary **A** can win the security game **G** within time t with probability ε
- A simulator **S** can break the problem **P** within time t' with probability ε'



Provable Security

- If the adversary **A** can win the security game **G**
  within time t with probability ε
- A simulator **S** can break the problem **P**
  within time t' with probability ε'
- Experiments give bounds on the best possible success probability ε'' within a time bound t' on the problem **P**
- We all agree on some safe assumptions:
  within a time bound t, no adversary can break **P**
  with probability greater than ε
- We eventually obtain bounds on the best possible adversary **A**

Provable Security

This methodology with a security game can be applied to any cryptographic primitive or protocol:

- Encryption: with semantic security
- Signature scheme: with unforgeability
- Authenticated key exchange: with privacy and authenticity
- etc

The Cloud: Access Anything from Anywhere

One can store

- Documents to share
- Pictures to edit
- Databases to query

and access from everywhere

Security Requirements

As from a local hard drive/server, one expects

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

How to proceed?

Confidentiality vs Sharing & Computations

Usual Encryption schemes protect data

E.g. either symmetric encryption, where $c = \mathsf{E}_{sk}(m)$ and then $m = \mathsf{D}_{sk}(c)$
or asymmetric encryption, where $c = \mathsf{E}_{pk}(m)$ and then $m = \mathsf{D}_{dk}(c)$

Only the knowledge of the decryption key (either sk or dk) allows to get m

- the provider stores the ciphertexts without any information about the messages
- nobody can access them either, except the owner/target receiver

Privacy by Design

How to outsource computations - How to share the results

without decrypting the data?

Broadcast Encryption

[Fiat-Naor - Crypto ‘94]

Broadcast Encryption

[Fiat-Naor - Crypto '94]

The sender chooses a target set

Broadcast Encryption

[Fiat-Naor - Crypto '94]

The sender chooses a target set

Broadcast Encryption

[Fiat-Naor - Crypto '94]

The sender chooses a target set

Users get **all-or-nothing** about the data

Sharing to a Target Set but No Computations!

Homomorphic Encryption

Encryption of a bit b: $c = \mathsf{E}(b) := b + 2r + qp$, for random integers q, r

$k = 2r + qp$ can be seen as a random mask, even for a fixed secret p

- $\mathsf{D}(c) := (c \bmod p) \bmod 2 = b + 2r \bmod 2 = b$
- $\mathsf{E}(b) + \mathsf{E}(b') = (b + 2r + qp) + (b' + 2r' + q'p)$

  $= (b \oplus b') + 2(r + r' + b \cdot b') + q''p$

  $= \mathsf{E}(b \oplus b')$ if $r + r' + 1 < p/2$
- Noise: $r'' = r + r' + b \cdot b'$ grows slowly (sum)
- Secret key: large integer p
- Additively homomorphic

Homomorphic Encryption

[DGHV - Eurocrypt '10]

$$c = \mathsf{E}(b) := b + 2r + qp$$

- $\mathsf{E}(b) \times \mathsf{E}(b') = (b + 2r + qp) \times (b' + 2r' + q'p)$
  $= (b \cdot b') + 2(rb' + r'b + 2rr') + q''p$
  $= \mathsf{E}(b \cdot b')$ if $r + r' + 2rr' < p/2$
- Noise: $r'' = rb' + r'b + 2rr'$ grows very fast (product)
- Encryption: small random noise r, large random q
- Multiplicatively homomorphic

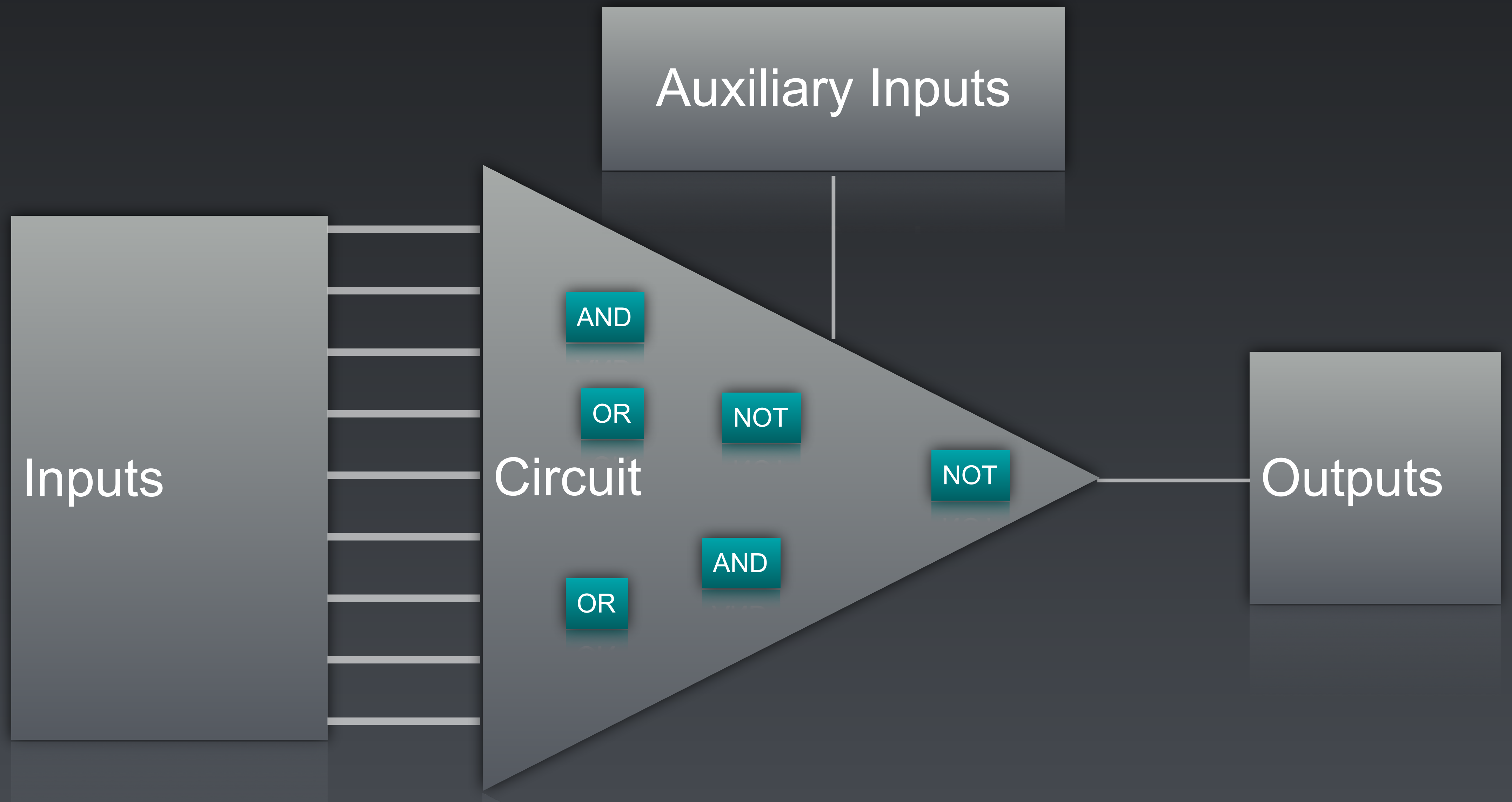
$\mathsf{E}(b) + \mathsf{E}(b') = \mathsf{E}(b \text{ XOR } b')$ $\quad$ $\mathsf{E}(b) \times \mathsf{E}(b') = \mathsf{E}(b \text{ AND } b')$

$\mathsf{E}(b) + 1 = \mathsf{E}(\text{NOT } b)$ $\quad$ $\Longrightarrow$ any Boolean circuit

Somewhat Homomorphic Encryption

[Gentry - STOC ’09]

Additive + Multiplicative Homomorphisms allow any Bolean operation

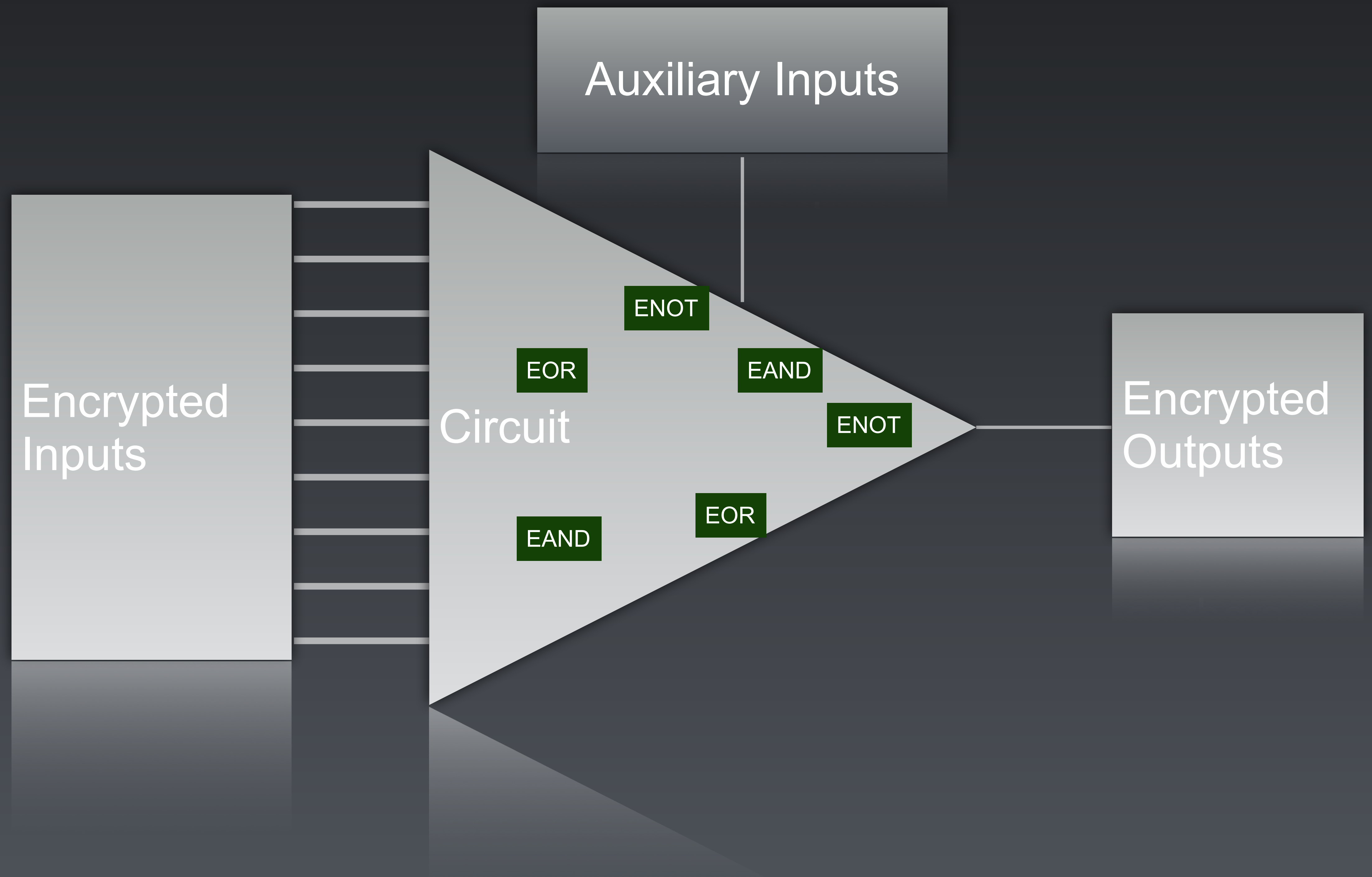


Somewhat Homomorphic Encryption

[Gentry - STOC '09]

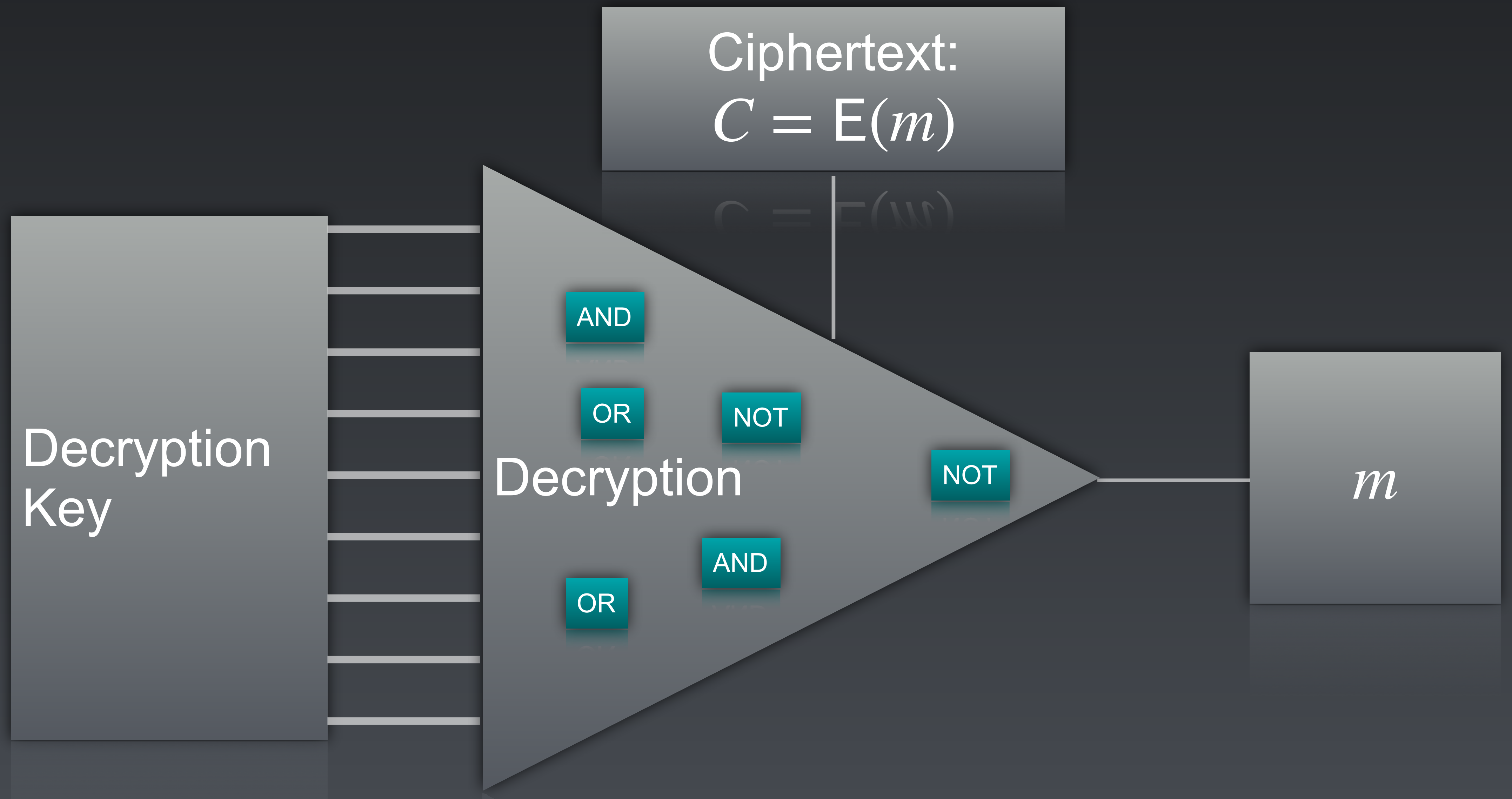
Additive + Multiplicative Homomorphisms allow any Bolean operation

But the depth of the circuit increases the noise: limited computations



Bootstrapping: Fully Homomorphic Encryption

[Gentry - STOC '09]

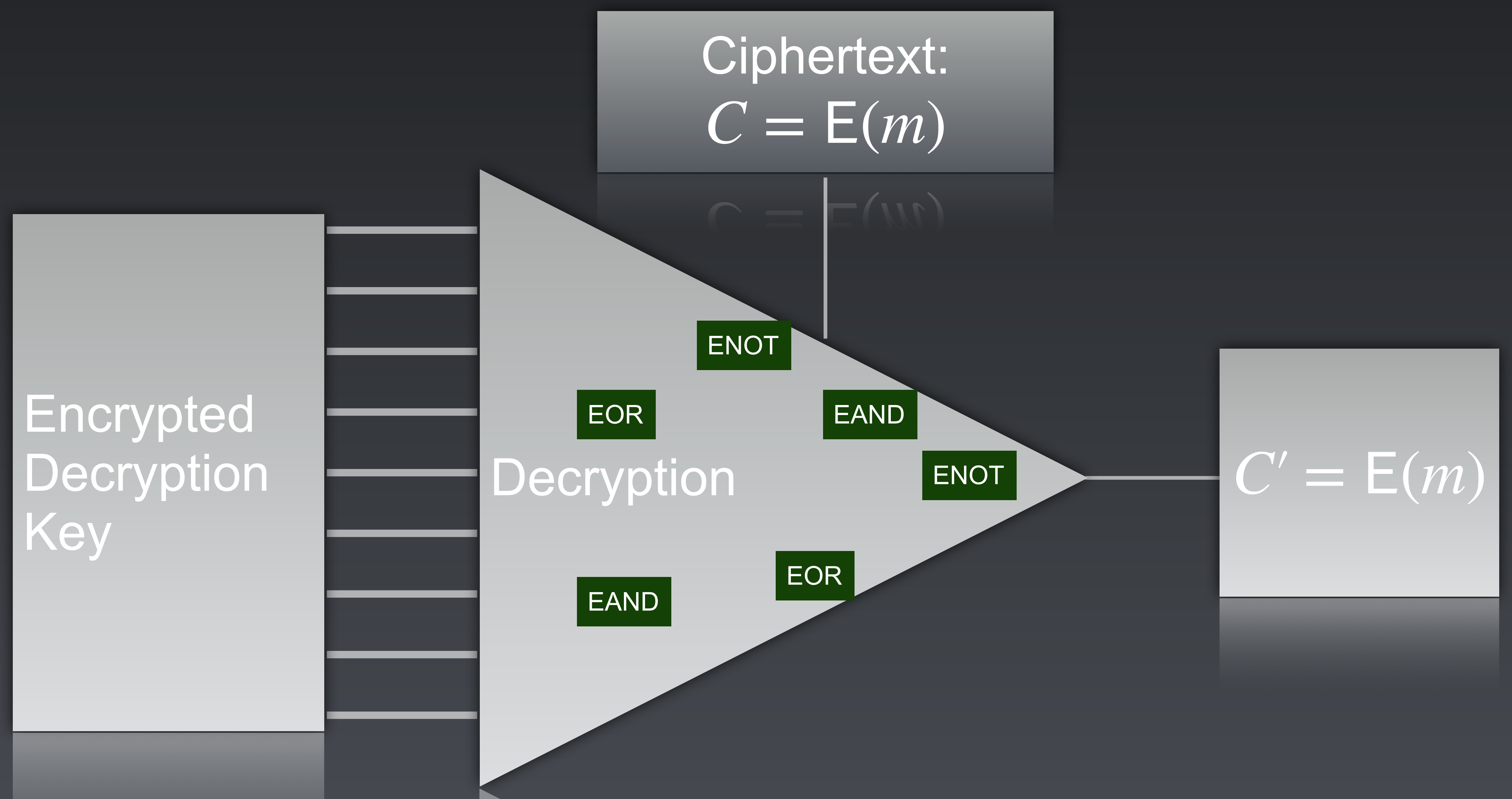


Bootstrapping: Fully Homomorphic Encryption

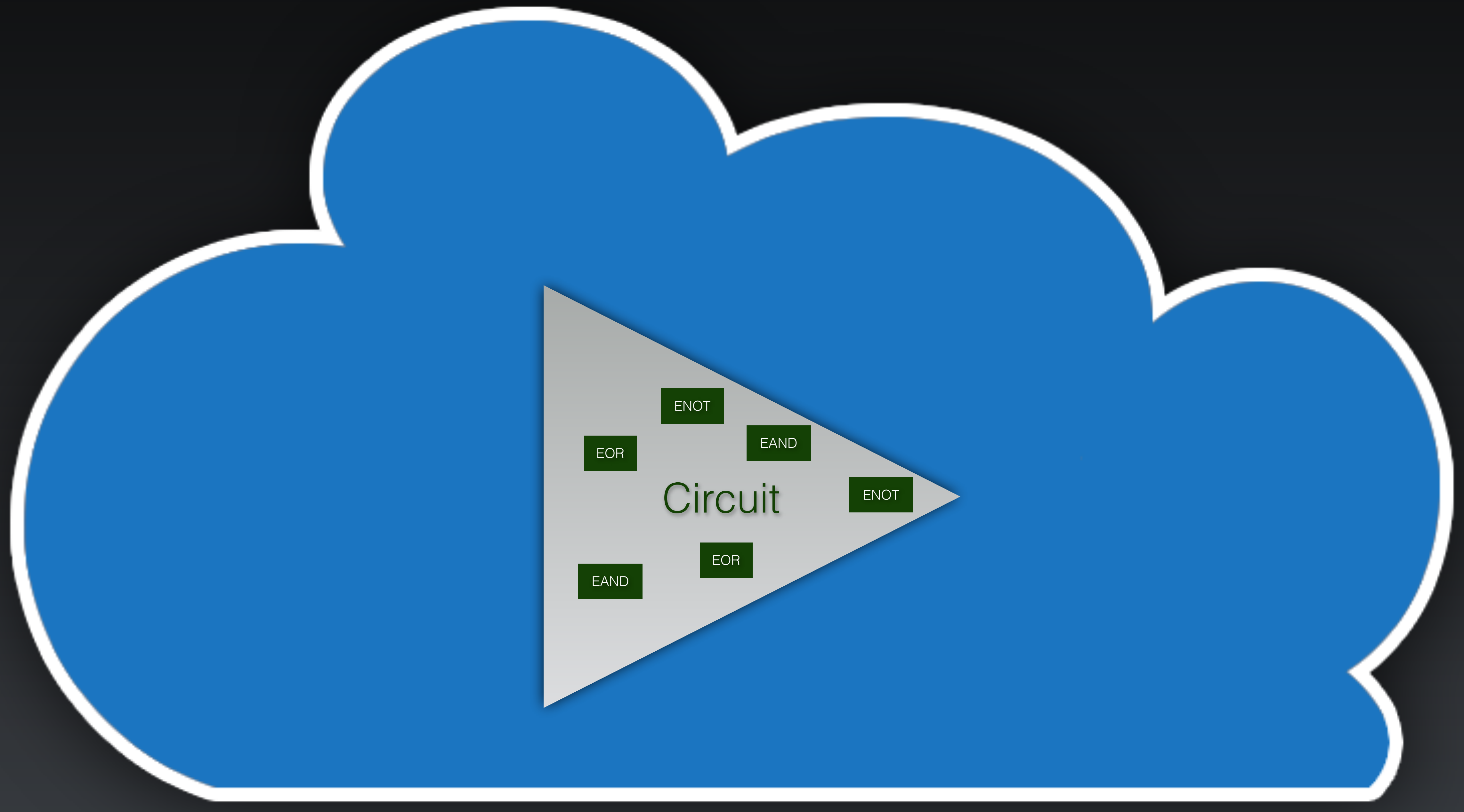
[Gentry - STOC ’09]

With a "virtual" decryption: one reduces the noise

Fully Homomorphic Encryption: any computation!

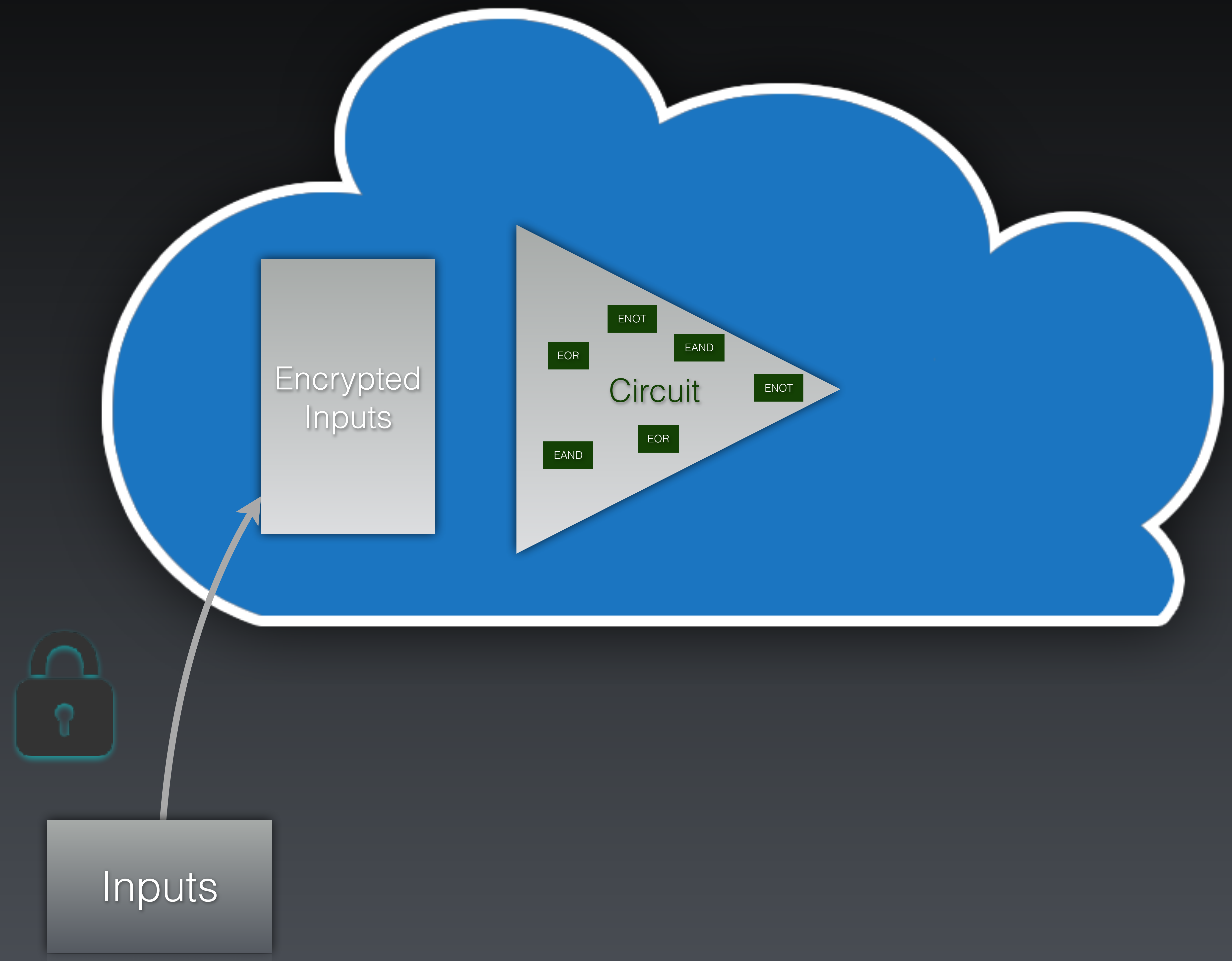


Outsourced Computations

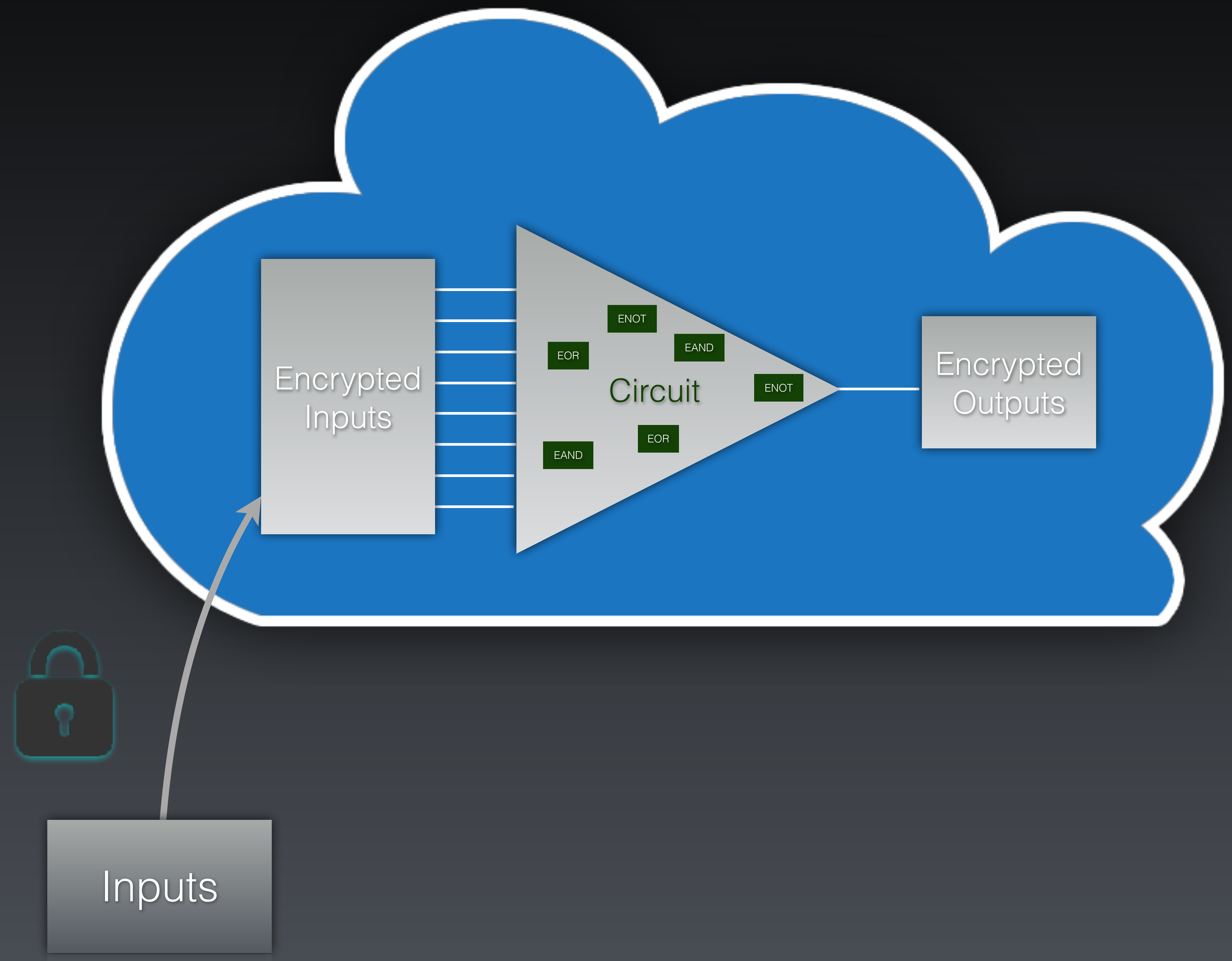


Inputs

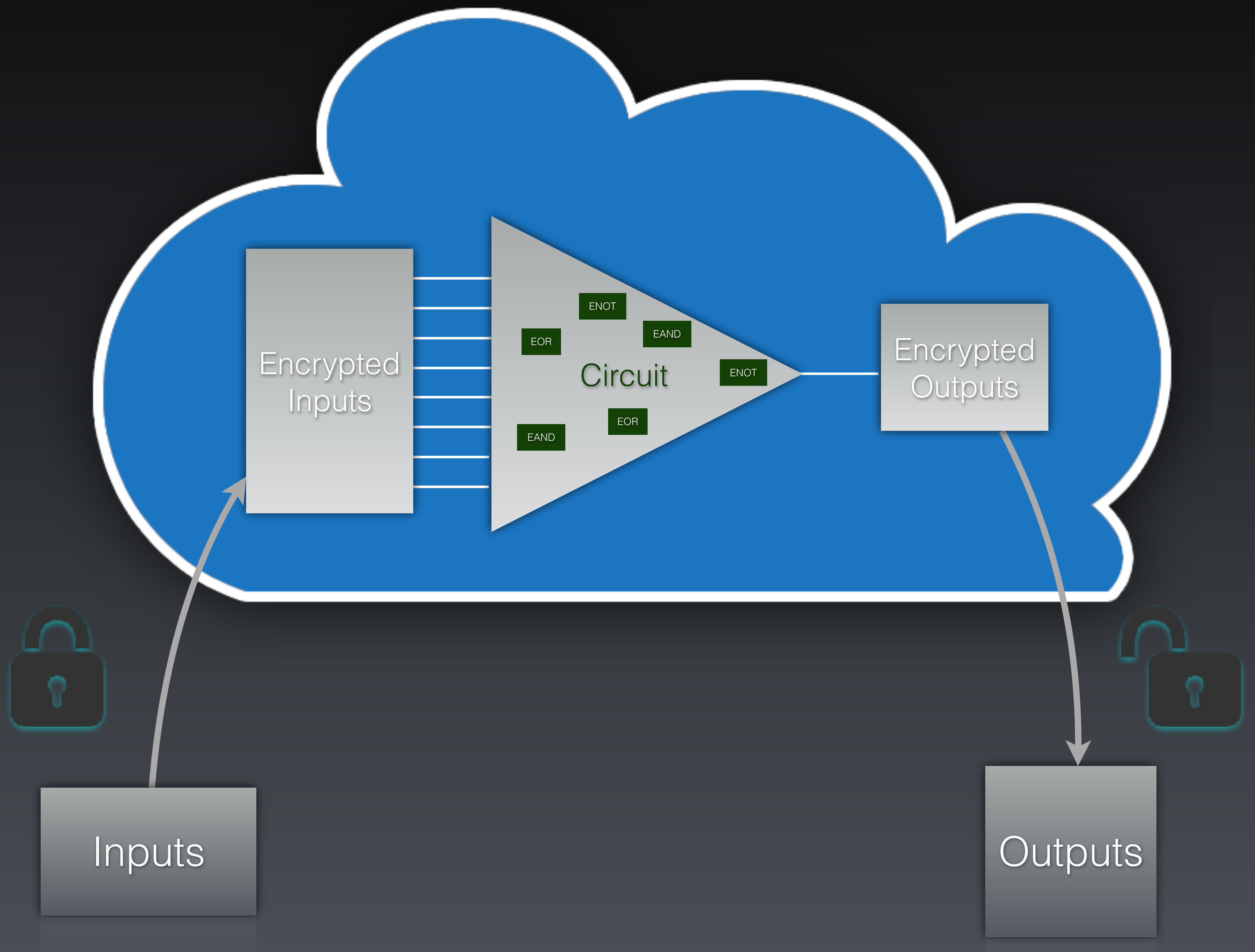
Outsourced Computations



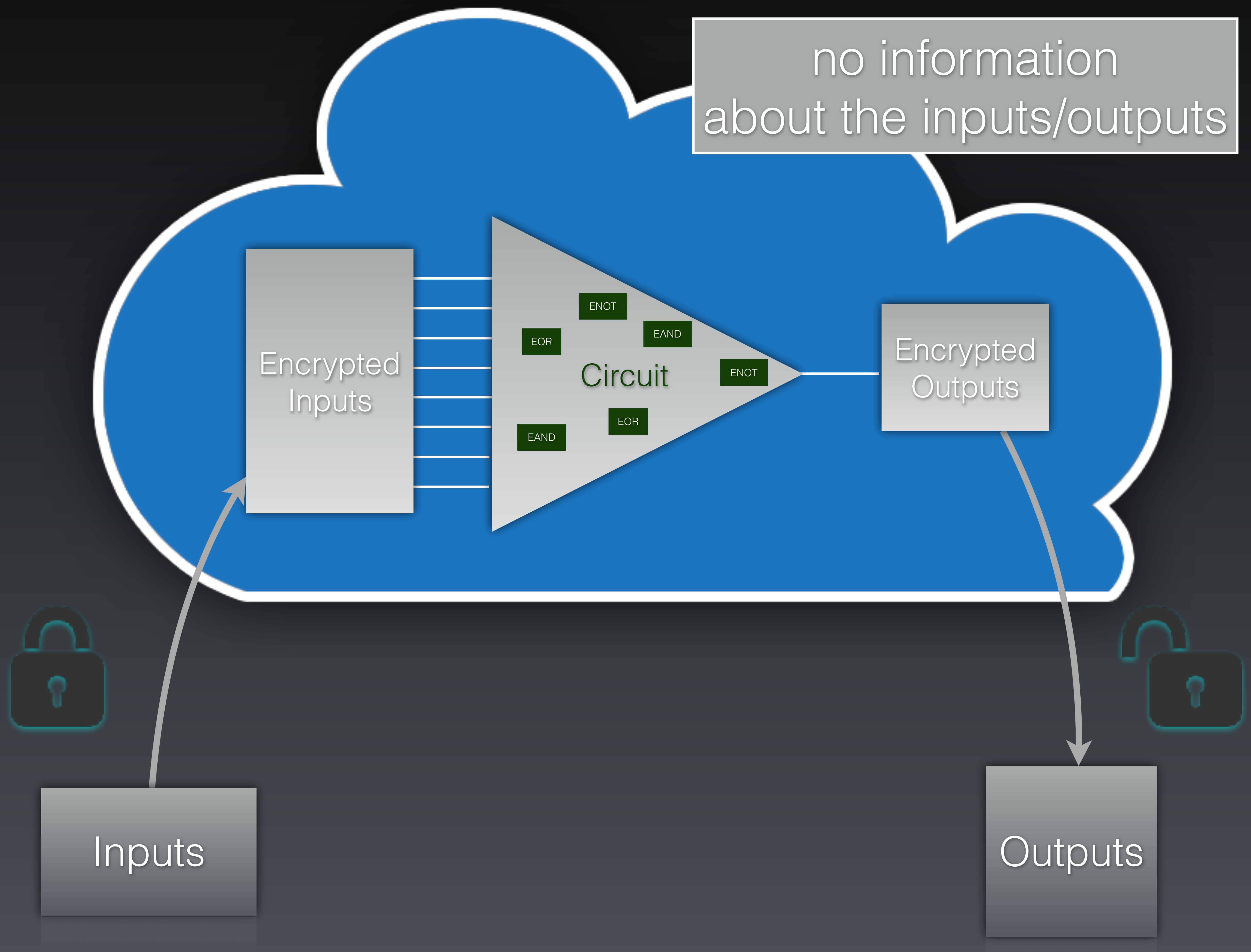
Outsourced Computations



Outsourced Computations



Outsourced Computations



FHE allows

- Any computation on private inputs
- Private « googling »

SNARGs: Succinct Proofs of correct computation

Any computation

But no possible sharing!

Functional Encryption

[Boneh-Sahai-Waters - TCC ‘11]

Functional Encryption

[Boneh-Sahai-Waters - TCC '11]

The authority generates functional decryption keys dk_f according to functions f

Functional Encryption

[Boneh-Sahai-Waters - TCC '11]

The authority generates functional decryption keys dk_f according to functions f

Functional Encryption

[Boneh-Sahai-Waters - TCC '11]

The authority generates functional decryption keys dk_f according to functions f

- From C = **Encrypt**(x), **Decrypt**(dk_f, C) outputs $f(x)$

Functional Encryption

[Boneh-Sahai-Waters - TCC ‘11]

The authority generates functional decryption keys dk_f according to functions f

- From C = **Encrypt**(x), **Decrypt**(dk_f, C) outputs $f(x)$
- This allows **controlled sharing of data**

Result in clear
for a Specific Function
for Specific Users

Functional Encryption is Powerful

Functional Encryption allows **access control**, from $C = \textbf{Encrypt}(x \mid\mid U)$

- with $f_{\mathsf{id}}(x \mid\mid U) = (\text{if id} = U, \text{ then } x, \text{ else } \perp)$: **identity-based** encryption
- with $f_{\mathsf{id}}(x \mid\mid U) = (\text{if id} \in U, \text{ then } x, \text{ else } \perp)$: **broadcast** encryption

but this is still all-or-nothing

Functional Encryption allows **computations**:

- any function f: in theory, with *iO* (Indistinguishable Obfuscation)
- concrete functions:
  - inner product, from $C = \textbf{Encrypt}(\vec{x}), f_{\vec{y}}(\vec{x}) = \vec{x} \cdot \vec{y}$
  - quadratic functions, from $C = \textbf{Encrypt}(\vec{x}, \vec{y}), f_Q(\vec{x}, \vec{y}) = \vec{x}^{\mathsf{T}} \cdot Q \cdot \vec{y}$

FE: Inner Product

[Abdalla-Bourse-De Caro-P. - PKC '15]

Time series data: $\overrightarrow{x_t}$

A few distinct linear statistic parameters $\overrightarrow{a_i}$ to get $\overrightarrow{a_i} \cdot \overrightarrow{x_t}$

- Each time period, $\overrightarrow{x_t}$ is encrypted
- For each parameter $\overrightarrow{a_i}$, the decryption key dk_i is generated

Can be done from any linearly homomorphic encryption:

- Master Secret Key: $sk = \vec{s}$, Functional Decryption Key: $dk_{\vec{y}} = \vec{s} \cdot \vec{y}$
- Encryption of $\vec{x}$: $c_0 = r$, $\vec{c} = \vec{x} + r \cdot \vec{s}$, for random r
- Decryption: $\vec{c} \cdot \vec{y} = \vec{x} \cdot \vec{y} + r \cdot \vec{s} \cdot \vec{y} = \vec{x} \cdot \vec{y} + r \cdot dk_{\vec{y}}$
- One-time pad: insecure… but can be made secure with ElGamal, Regev, etc (based on Discrete Logarithm, Lattices, etc)

Multi-Client Functional Encryption

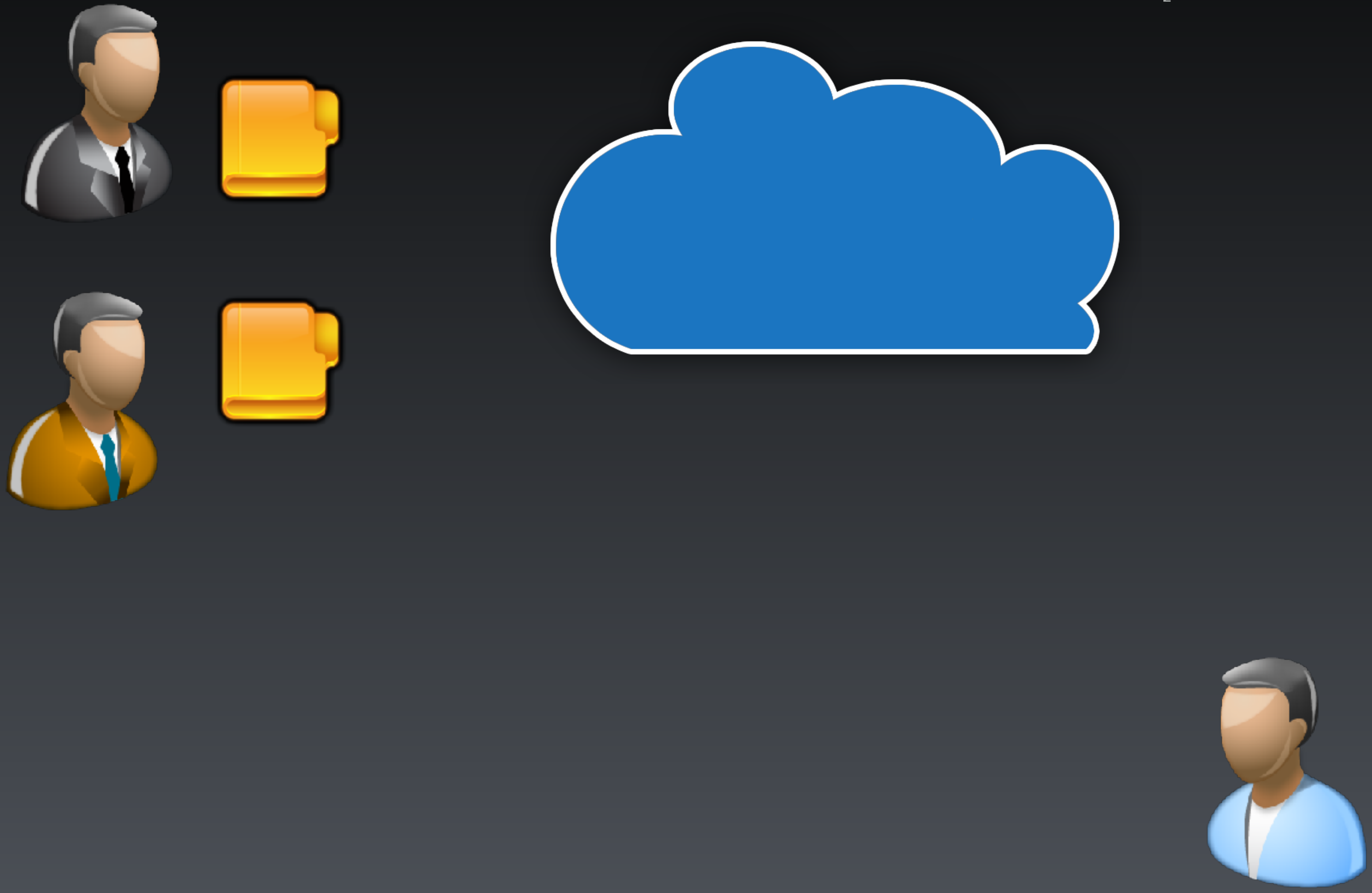
- one key limits to one function on any vector 🙂
- a unique sender only can encrypt all the inputs 😞
  - Multi-Client Functional Encryption (MCFE) 🙂
  - Client C_j generates $\mathsf{E}(t, j, x_{t,j})$ for the time period t

    $\Rightarrow$ only one ciphertext for each index j and each time period t

    $\Rightarrow$ all the individual ciphertexts globally encrypt $\vec{x}_t$
- still a unique authority for the functional key generation 😞
  - Decentralized Multi-Client Functional Encryption (DMCFE) 🙂
  - With Independent and Distrustful Clients

Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt ’18]

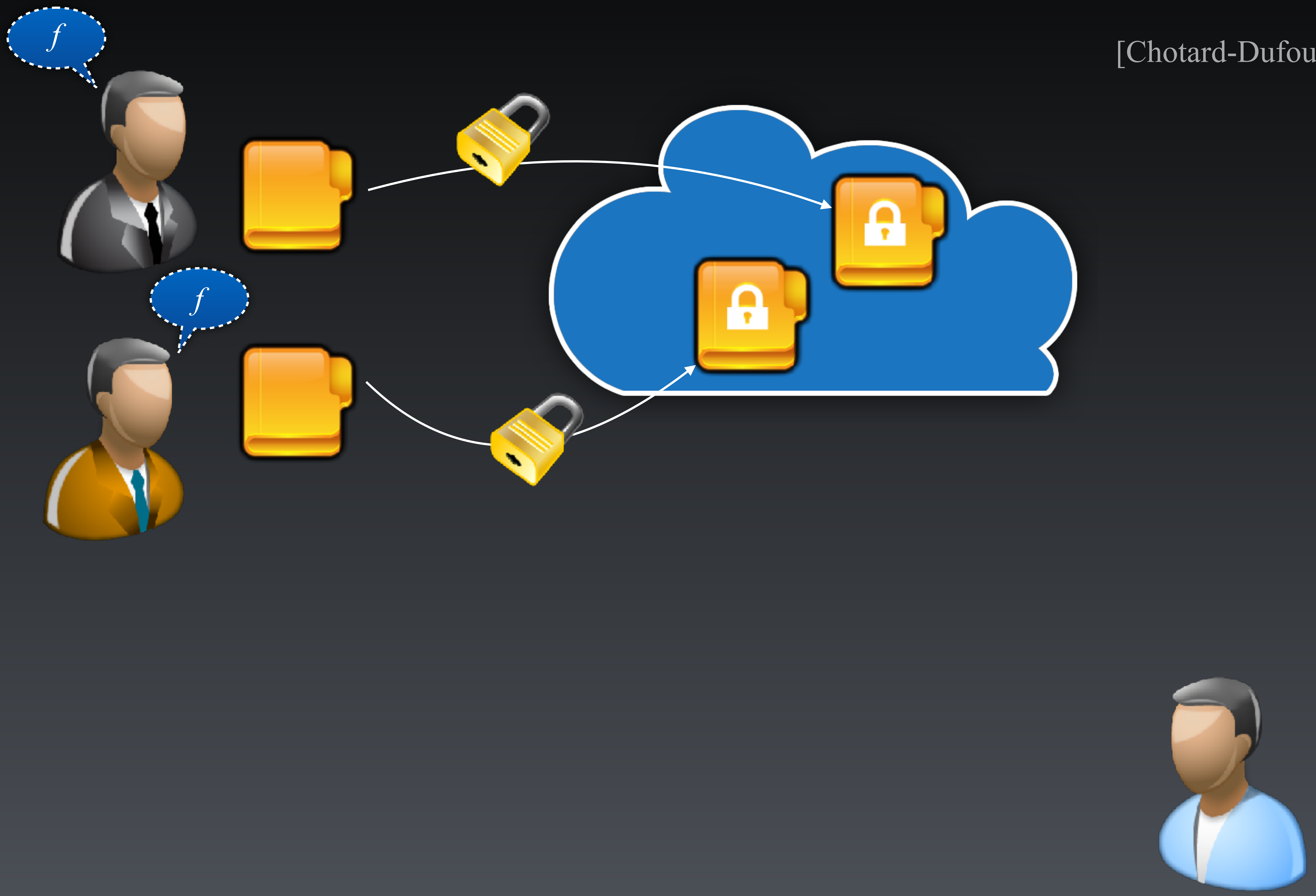


Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt '18]

Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt '18]

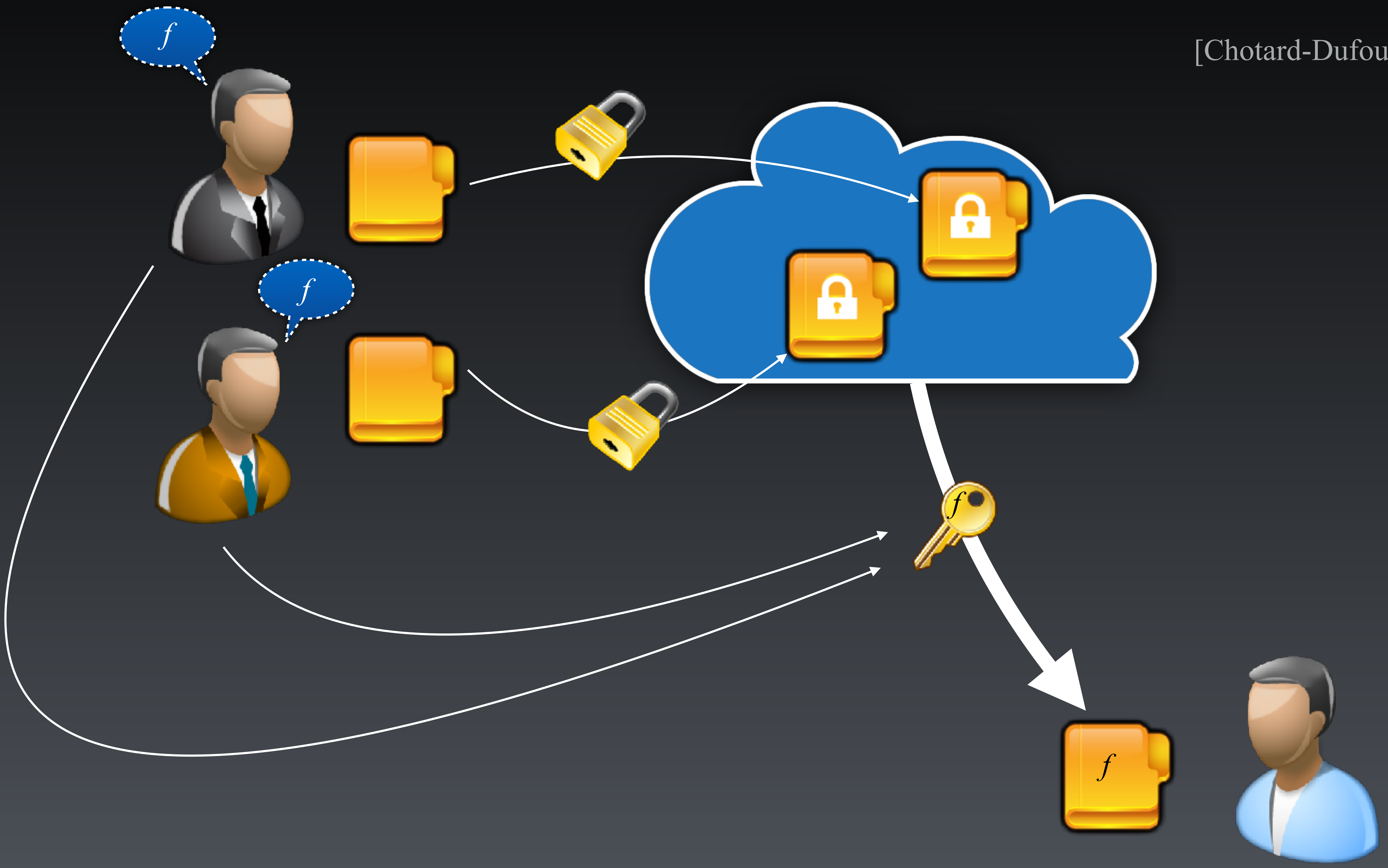


Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt ’18]

Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt ’18]



Decentralized MCFE

[Chotard-Dufour Sans-Gay-Phan-P. - Asiacrypt '18]

- **KeyGen**(i) → secret key sk_i and encryption key ek_i for client i
- **Encrypt**($ek_i, \lambda, \boldsymbol{x}_i$) → c_i = **E**($ek_i, \lambda, \boldsymbol{x}_i$) for the label λ (or time period t)
- **DKeyGen**($(sk_i)_i, f$) → dk_f
- **Decrypt**(dk_f, λ, **C**) → $f(\boldsymbol{x})$ if **C** = (c_i = **E**($ek_i, \lambda, \boldsymbol{x}_i$))$_i$

- **Encrypt/Decrypt** are non-interactive algorithms
- **KeyGen/DKeyGen** might be interactive protocols between the clients
  - but should be **one-round** protocols only

DMCFE: Concrete Case

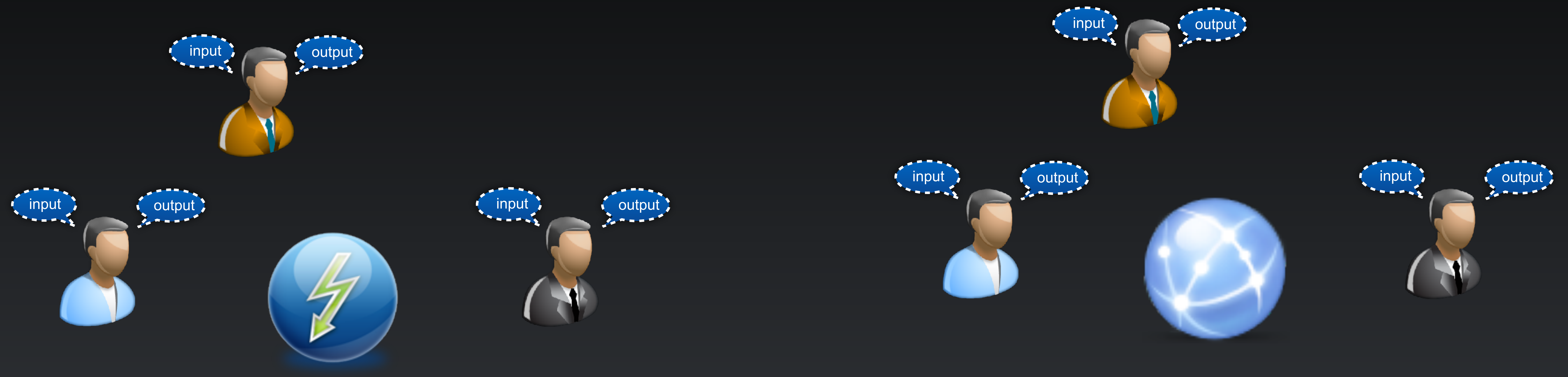
Jan 2020	Theft	Fire	Water	Auto	Falls

Feb 2020	Theft	Fire	Water	Auto	Falls
Co. 1	7	1	8	3	1
Co. 2	7	1	2	3	2
Co. 3	3	2	10	1	4
Total	17	4	20	7	7

- Insurance companies: list of damages
- Each individual line is quite sensitive: cannot be shared
  - encrypted by each company every month
- Monthly totals are valuable for everybody
  - functional key for each sub-total: generated together once for all
  - can be applied every month, on fresh ciphertexts, without interactions

Multi-Party Computation

[Yao - 1982]



Cloud = possible interactions between the parties

- Private computation with a Trusted Third Party
- MPC = without any TTP
  - only interactions between the players with their secrets
  - no additional information leaks

2-PC and Machine Learning

[Ryffel-Tholoniat-P.-Bach - arXiv '20]

Two-Party Computation = Particular case of MPC

- data owner vs. model owner
- can be applied to federated learning

Main ingredients:

- secret sharing
- comparisons: activation function

Multiple iterations until the secret is reconstructed:

- multiple layers in the network
- multiple data sources for training

FHE/FE and Machine Learning

Fully Homomorphic Encryption: any function

- one can apply a private model on private data for a client
- one can help a client to refine a model with private data

Functional Encryption: only quadratic functions

- quadratic activation function (instead of classical ReLU)
- one hidden layer only: the output is in clear

Experiments on the MNIST Data Set [Ryffel-Dufour Sans-Gay-Bach-P. - NeurIPS ’19]

What is Data Privacy?

FE/DMCFE: no leakage excepted the decrypted result

FHE: no leakage excepted the input/output for user

MPC/2-PC: no leakage excepted the output result

- What is the result?
  - the model (training phase), the inference (decision phase)
- The model contains information about the training set
  - the model owner will learn information about the training set
- Inference leaks information about the model
  - the data owner will learn information about the model
  - and then about the training set

Differential Privacy

To reduce information about the training set: noise addition

- Differential privacy
  - the output is indistinguishable whether any user A is in the set or not
  - the model does not leak individual data from the training set
- Cryptography: the protocol does not leak more than the output

- The training phase does not leak
  - any individual data from the training set to the model owner
- Inferences do not leak
  - any individual data from the training set to the client
  - the user's input to the model owner

Conclusion

- Functional Encryption / DMCFE
  - can handle any statistics on data series
  - without interactions
  - with strong control on the authorized computations
- Fully Homomorphic Encryption
  - allows outsourced computations
  - without interactions (one-round query-answer)
  - but still several milliseconds per gate on the server-side
- Two-Party Computation / MPC
  - very versatile and quite efficient
  - but highly interactive
- But one has to take care about the **information revealed by the result**