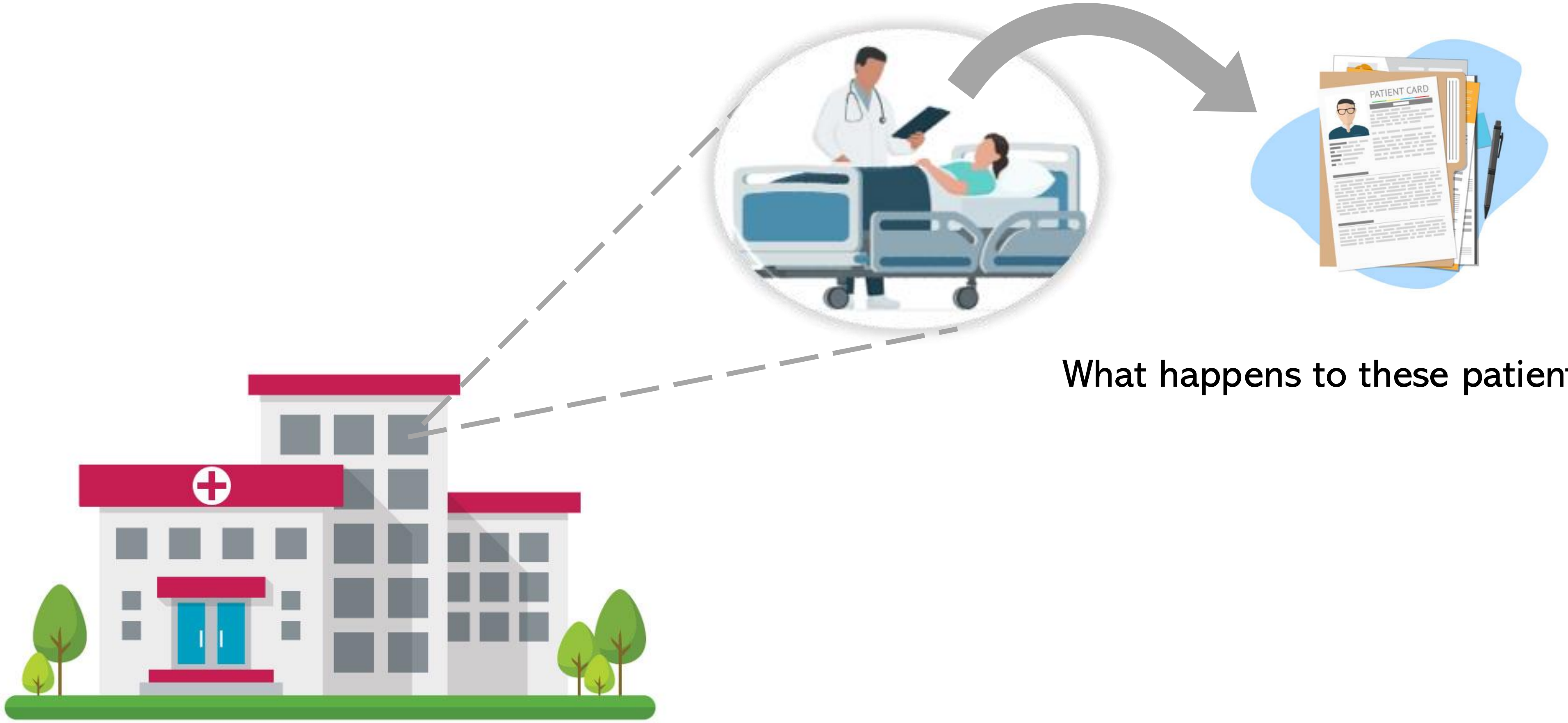
Private and Verifiable Delegation of Computation

Aarushi Goel

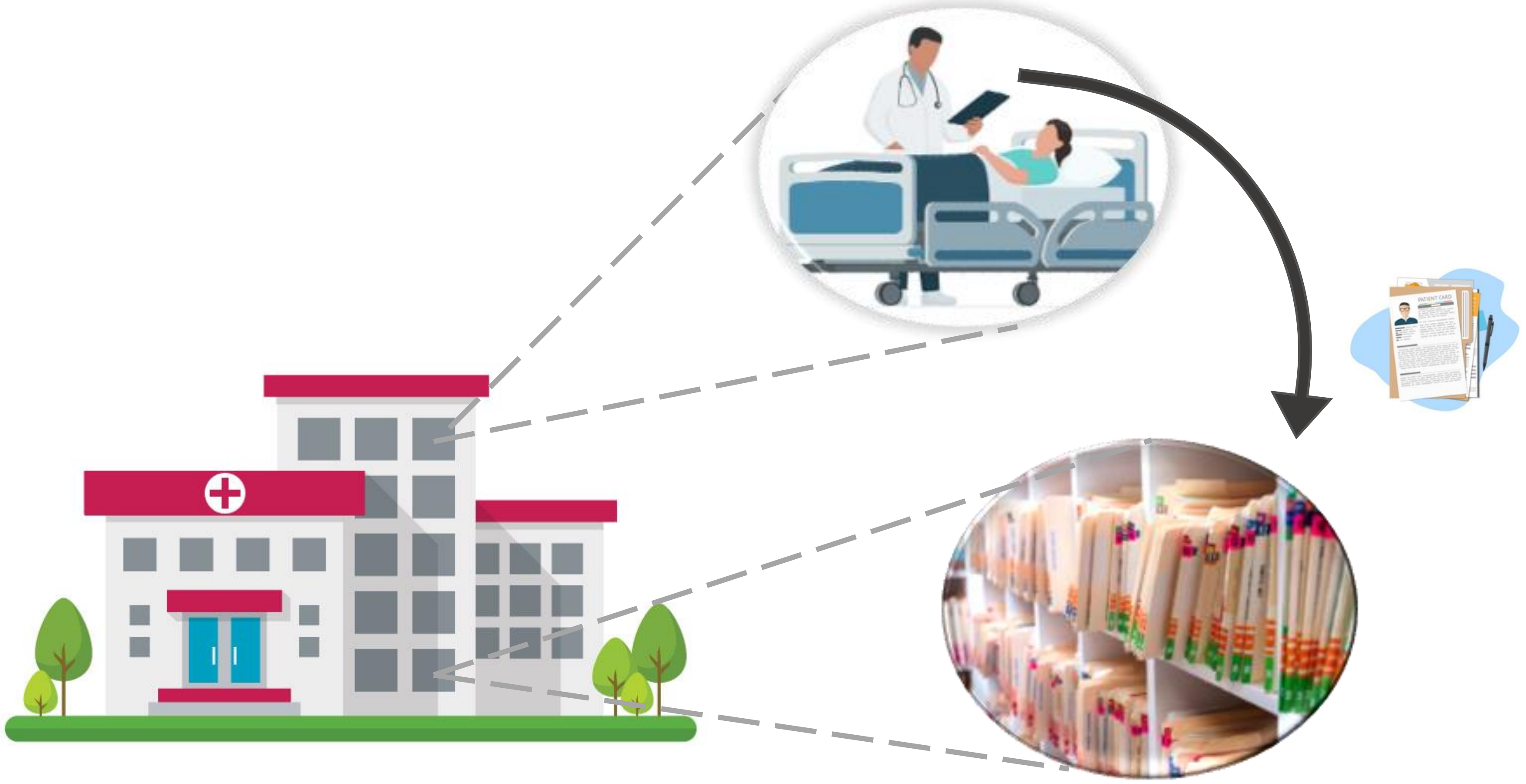
PURDUE UNIVERSITY



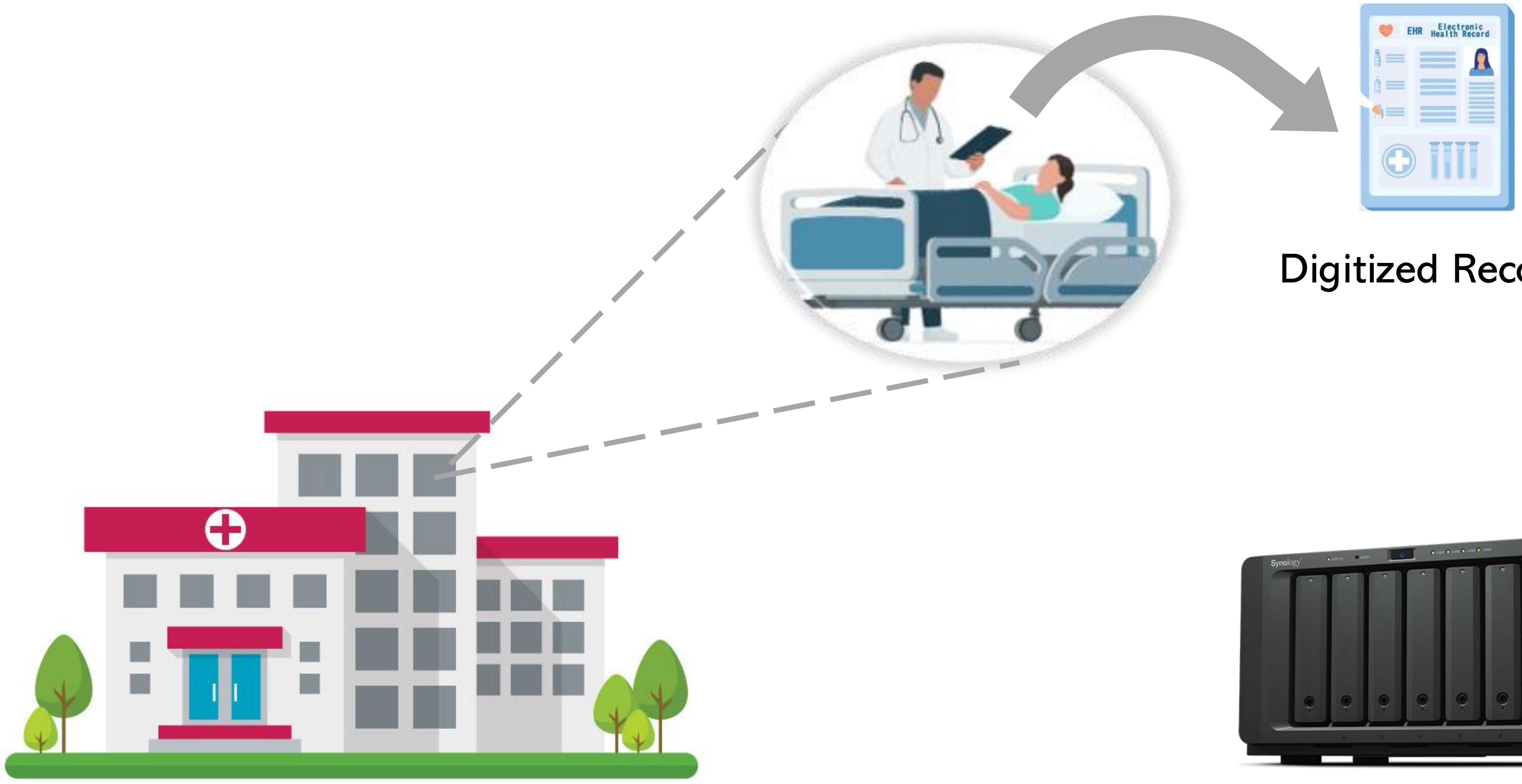
What happens to these patient records?

Hospital

Until Late 1980s



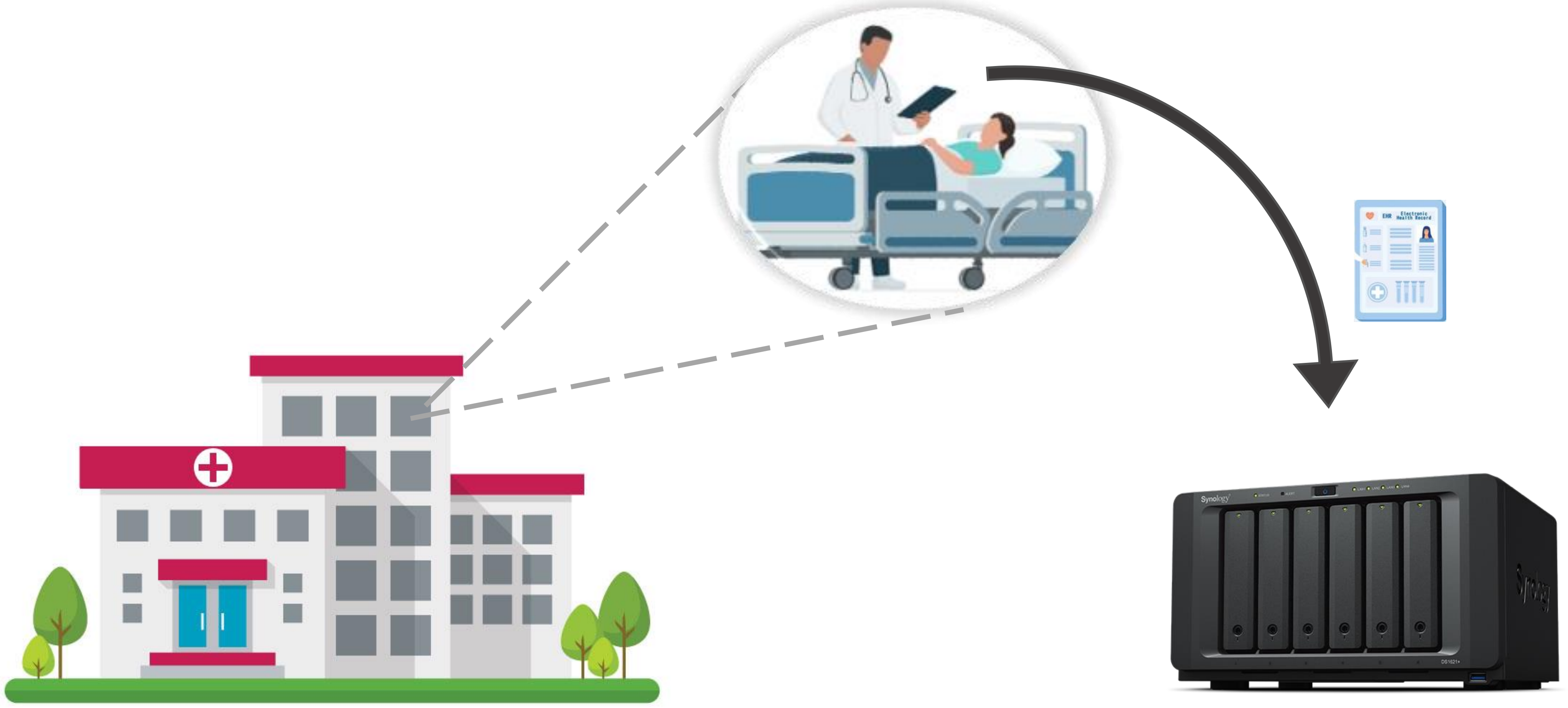
Late 1980s – Late 2000s



Digitized Records

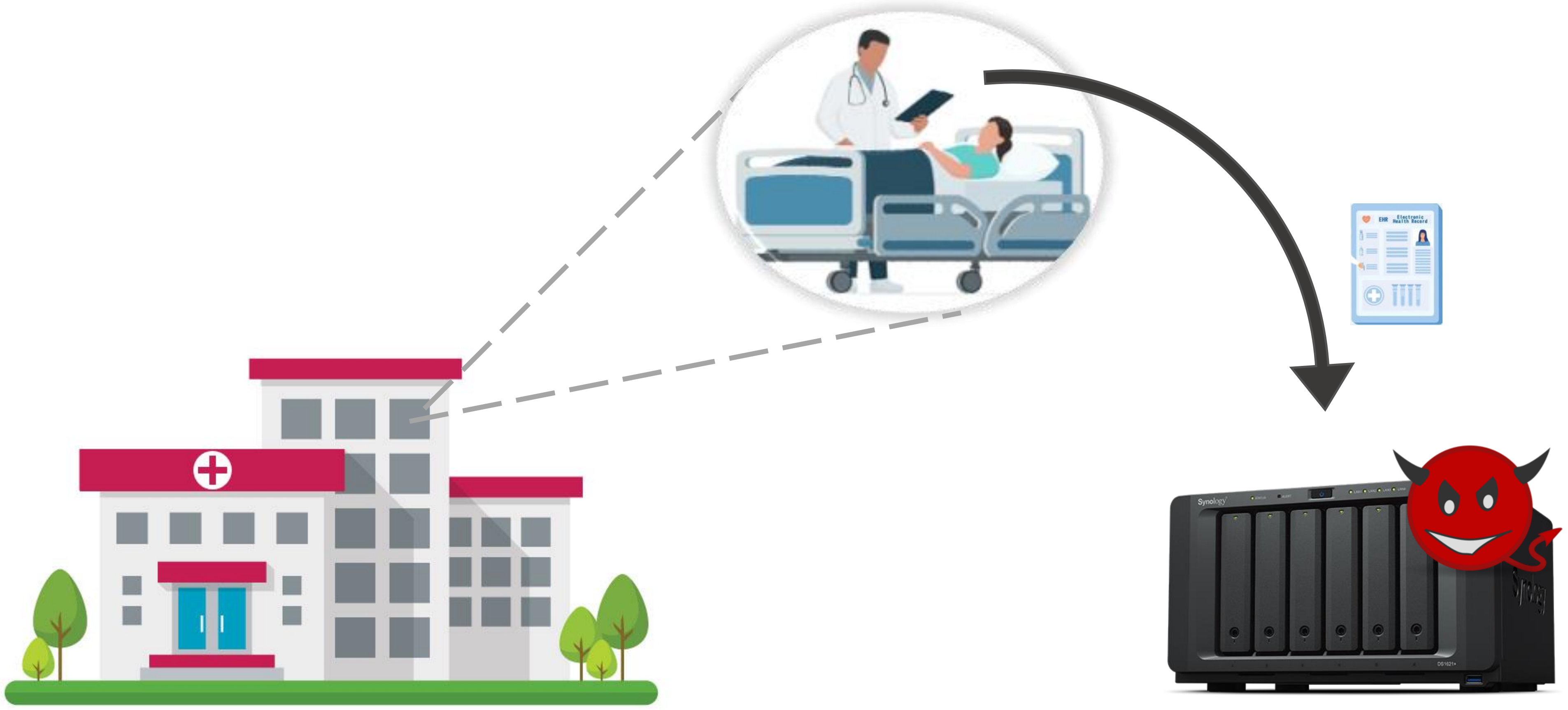
Network attached storage/
Storage area network

Late 1980s – Late 2000s

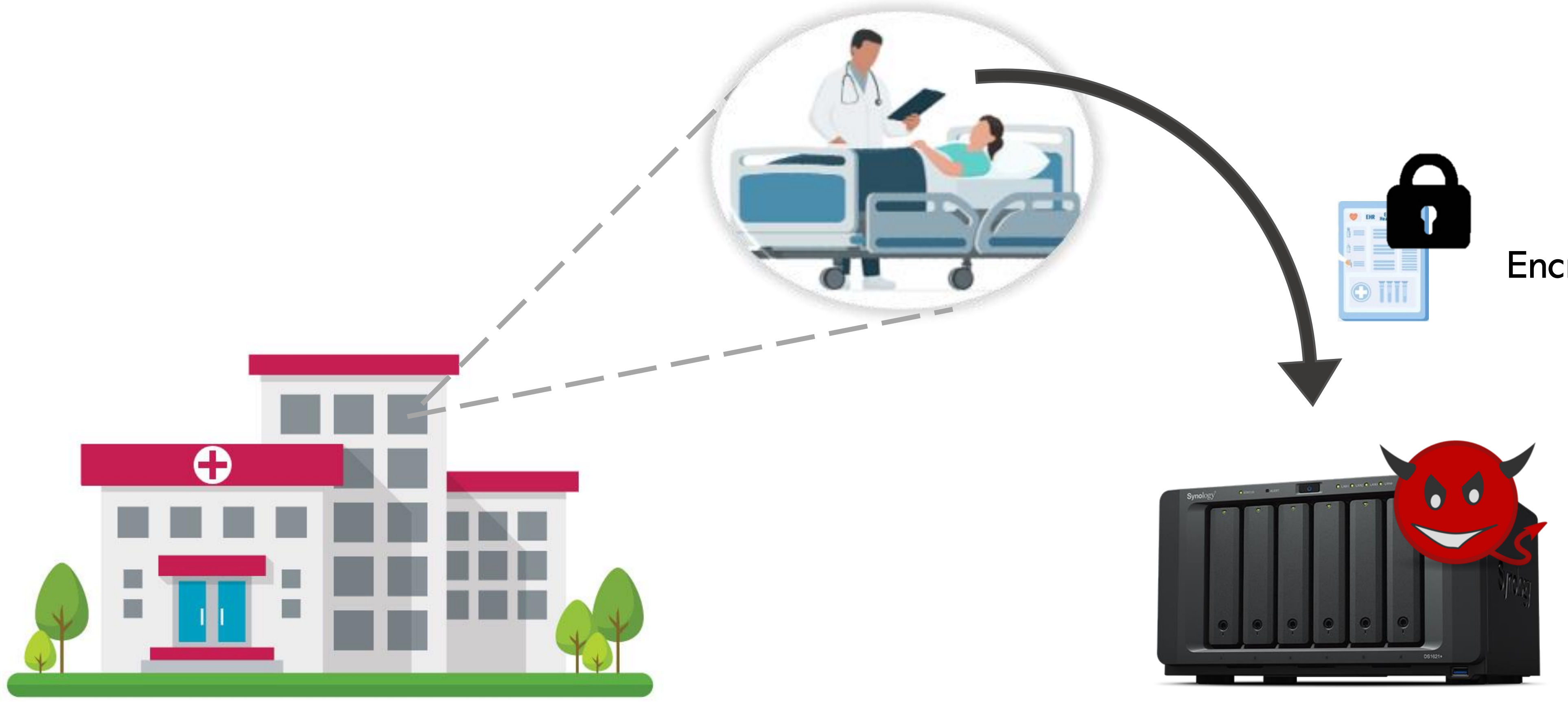


Network attached storage/
Storage area network

Late 1980s – Late 2000s



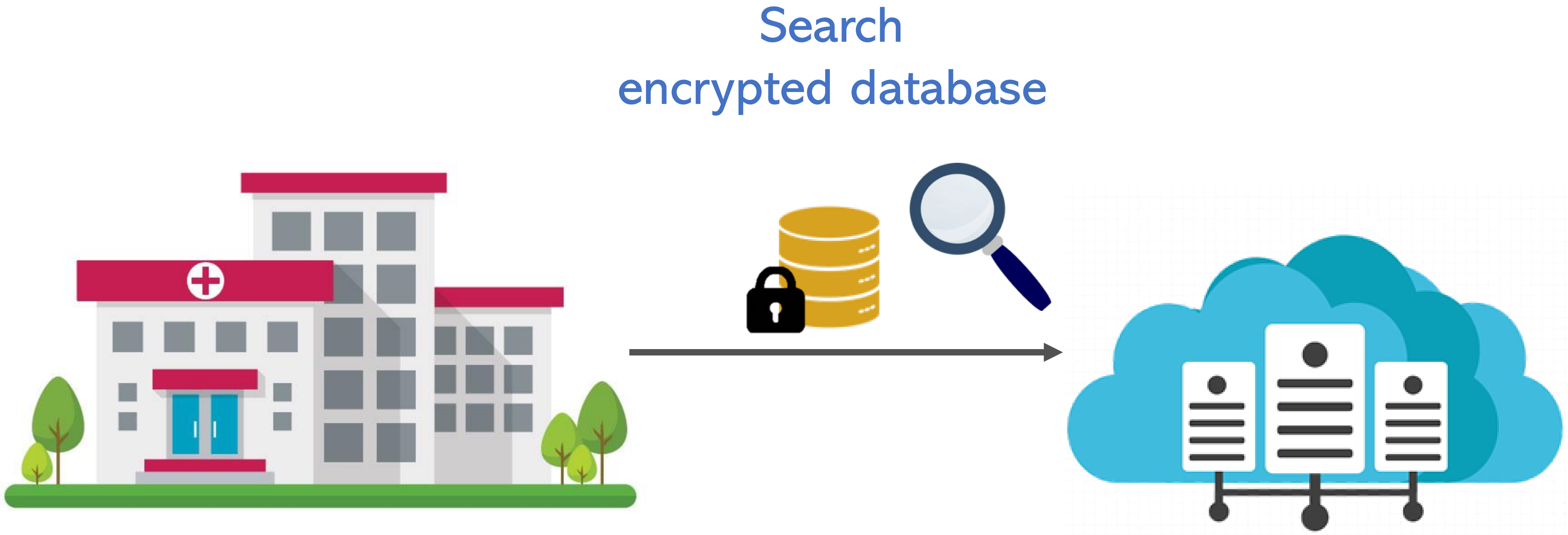
Late 1980s – Late 2000s



Since Late 2000s

Cloud Computing Platforms

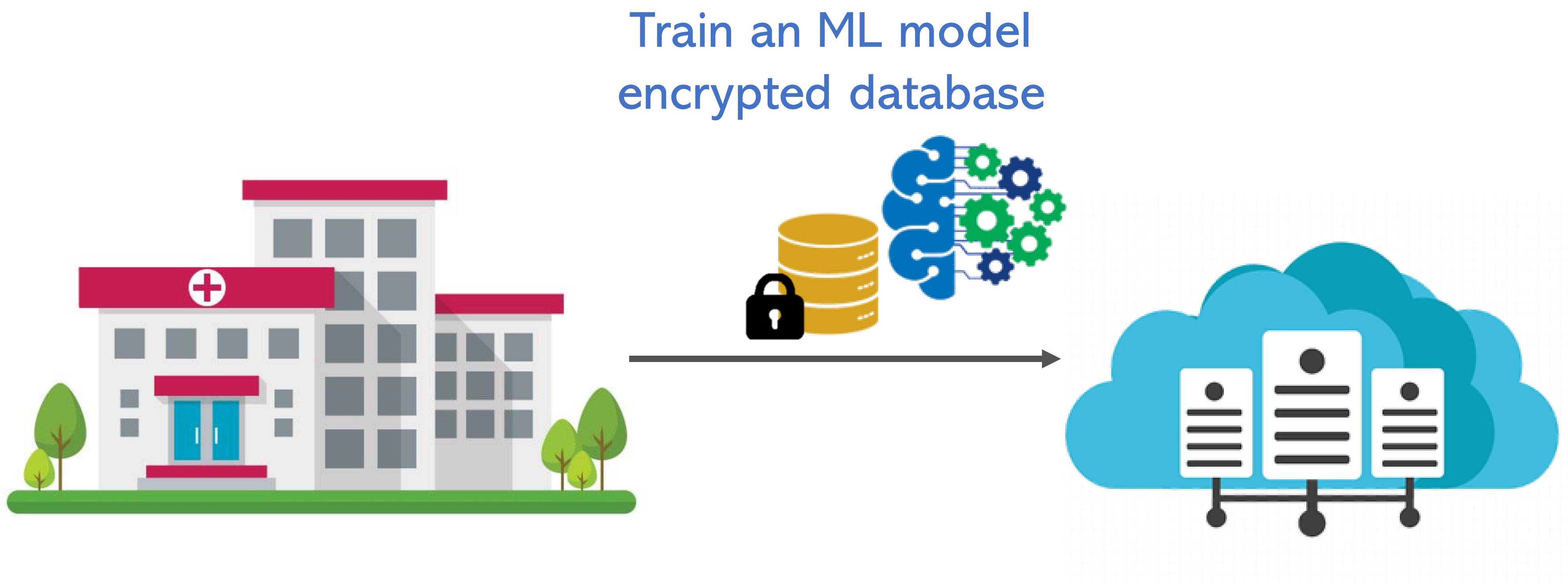
Since Late 2000s



Since Late 2000s

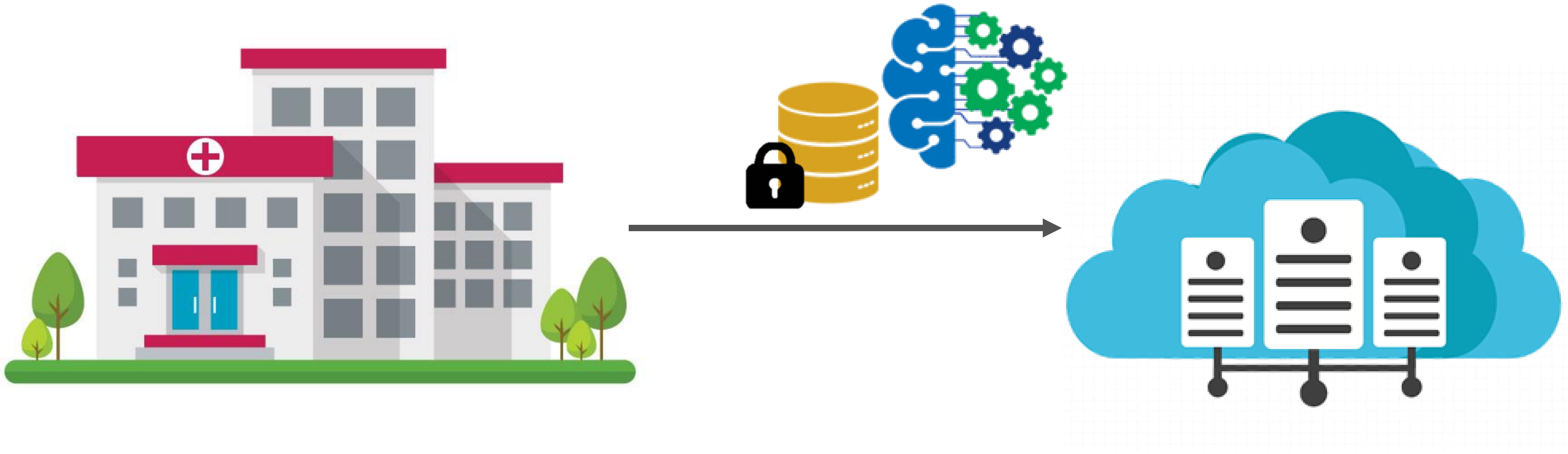
Statistical analysis on encrypted database

Modern Applications



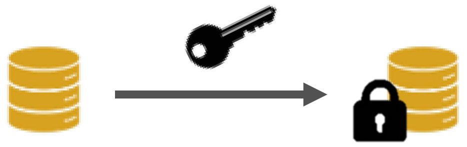
Since Late 2000s

How does one compute on encrypted data?



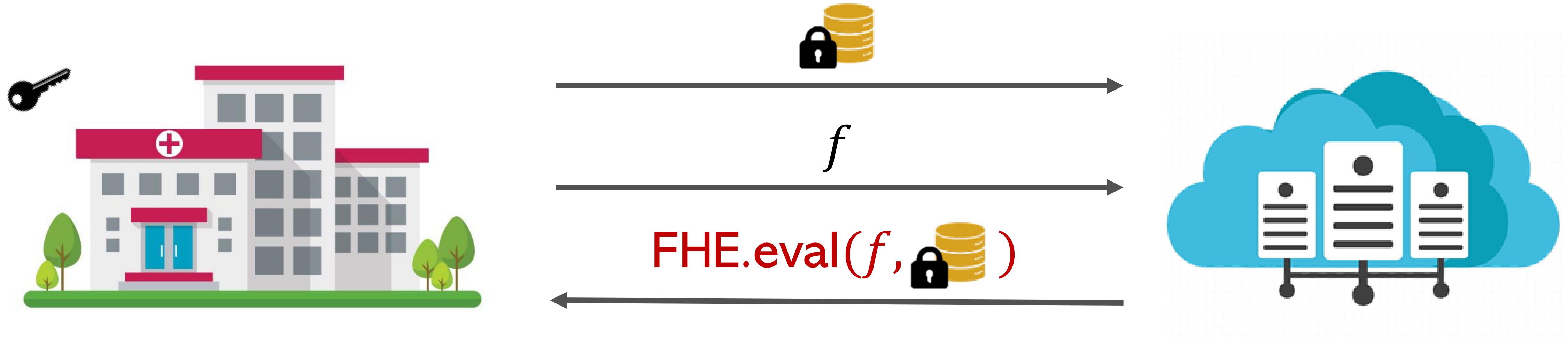
Fully Homomorphic Encryption

[Gentry09]



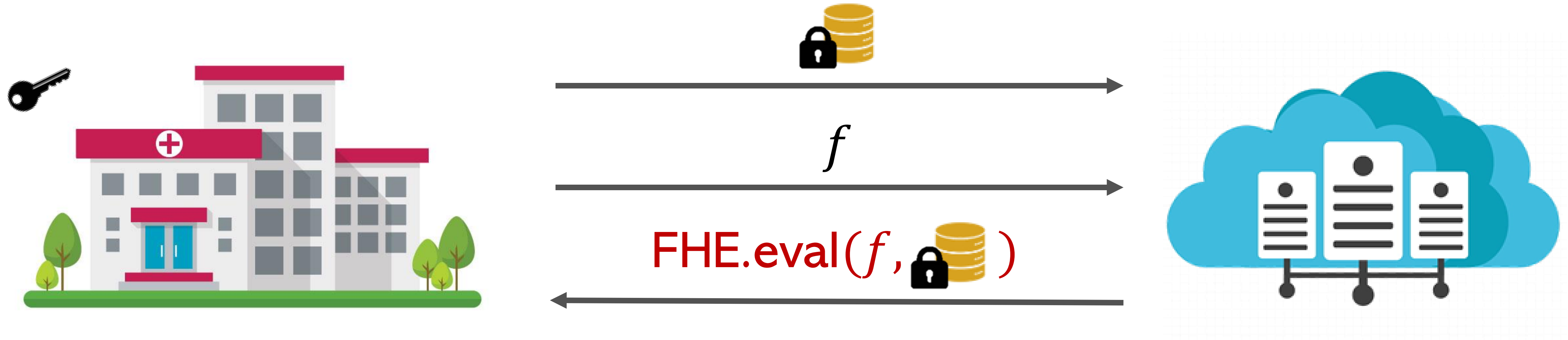
Fully Homomorphic Encryption

[Gentry09]



Fully Homomorphic Encryption

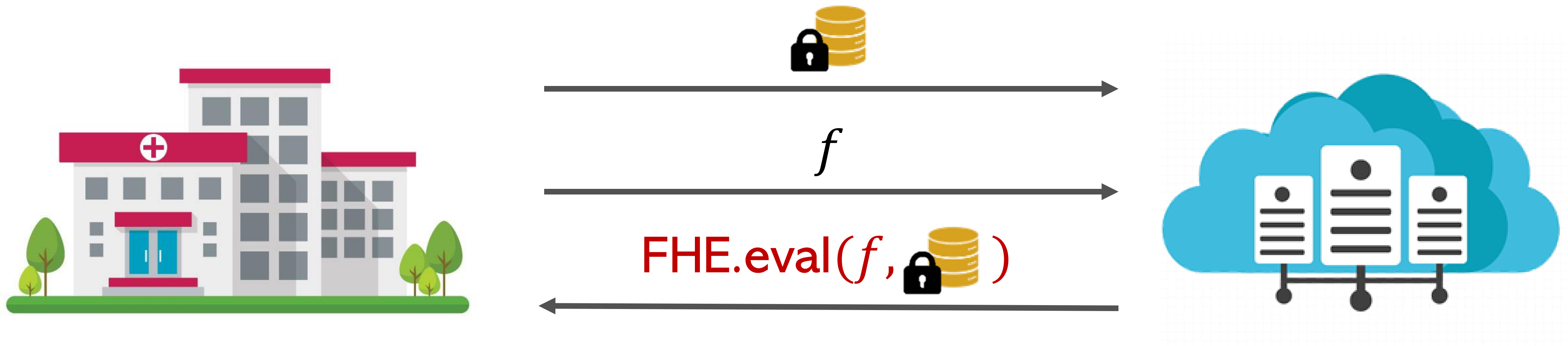
[Gentry09]

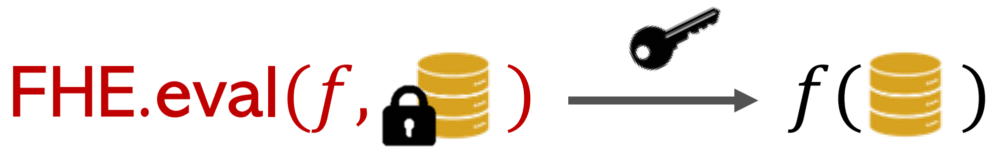


FHE.eval(f,)

Fully Homomorphic Encryption

[Gentry09]





FHE in Industry

Optalysys

Duality

inpher

intel®

ZAMA

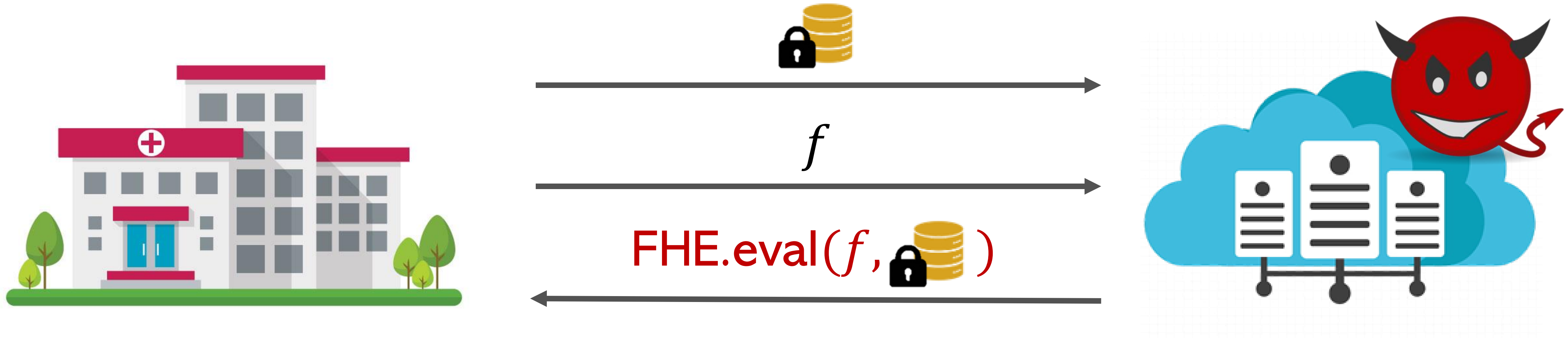
galois

IBM

CORNAMI®
Intelligent Computing

N NIOBIUM

Can we trust the cloud to compute FHE.eval(f,) honestly?



Can we trust the cloud to compute FHE.eval$(f,$ 🔒 $)$ honestly?

FHE enables private delegation of computation

Main Question:

Can we enable private and verifiable delegation of computation?

Main Question:

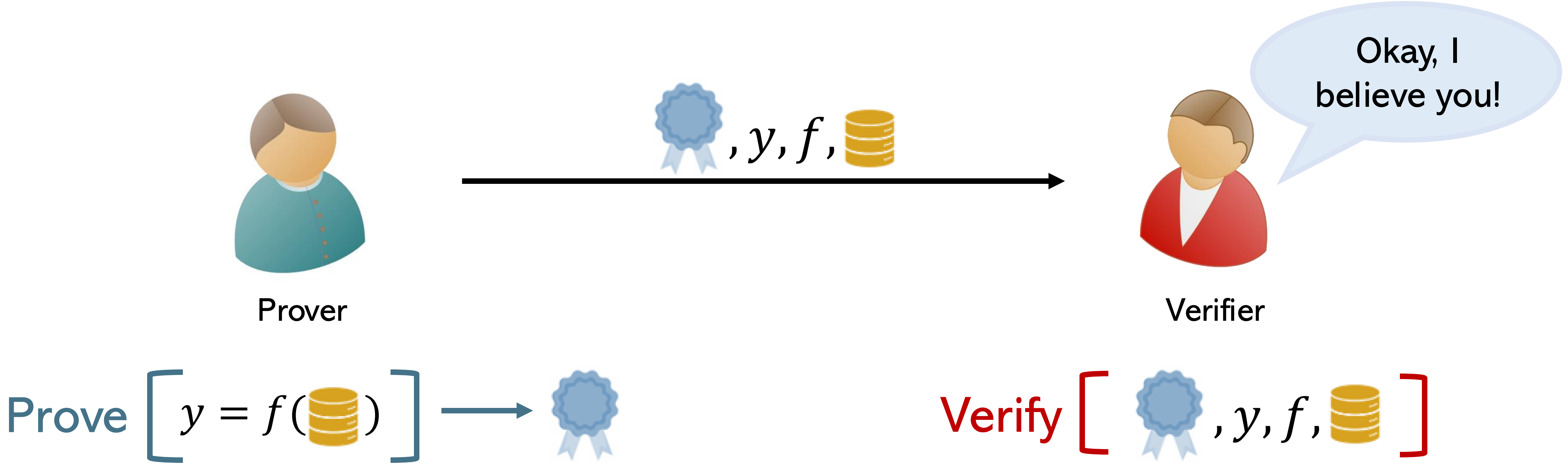
Can we enable private and verifiable delegation of computation?

FHE

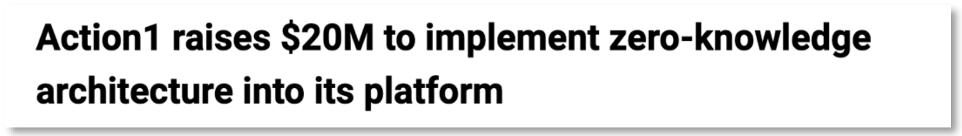
??

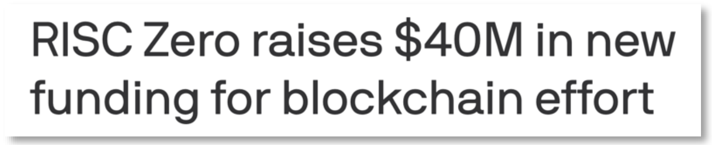
Verifiable Computation

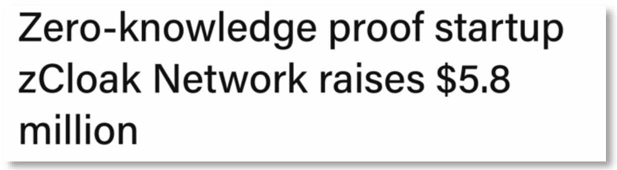
Succinct Proofs/SNARKs [Kil92, Mic94]

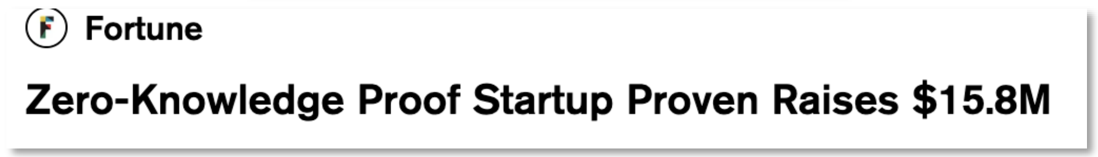


SNARKs in Industry

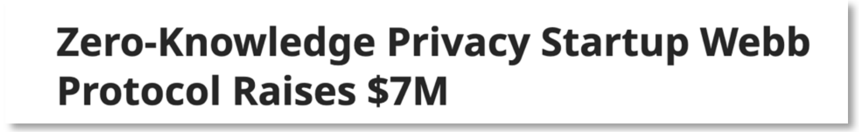

Action1 raises $20M to implement zero-knowledge architecture into its platform

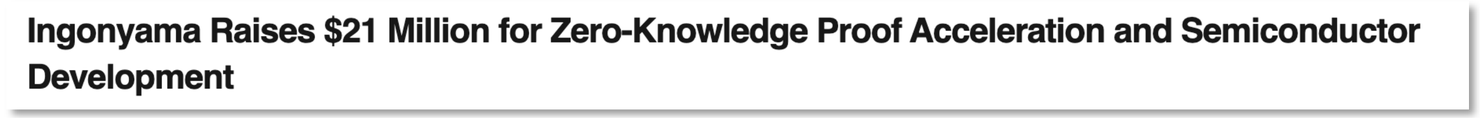

RISC Zero raises $40M in new funding for blockchain effort


Zero-knowledge proof startup zCloak Network raises $5.8 million


Fortune

Zero-Knowledge Proof Startup Proven Raises $15.8M


Zero-Knowledge Privacy Startup Webb Protocol Raises $7M


Ingonyama Raises $21 Million for Zero-Knowledge Proof Acceleration and Semiconductor Development

Zcash

AZTEC

NEXUS

Algorand FOUNDATION

Aleo

ETHERIUM

ZKPASS

TRAIL OF BITS

PROVEN

RISC ZERO

STARKWARE

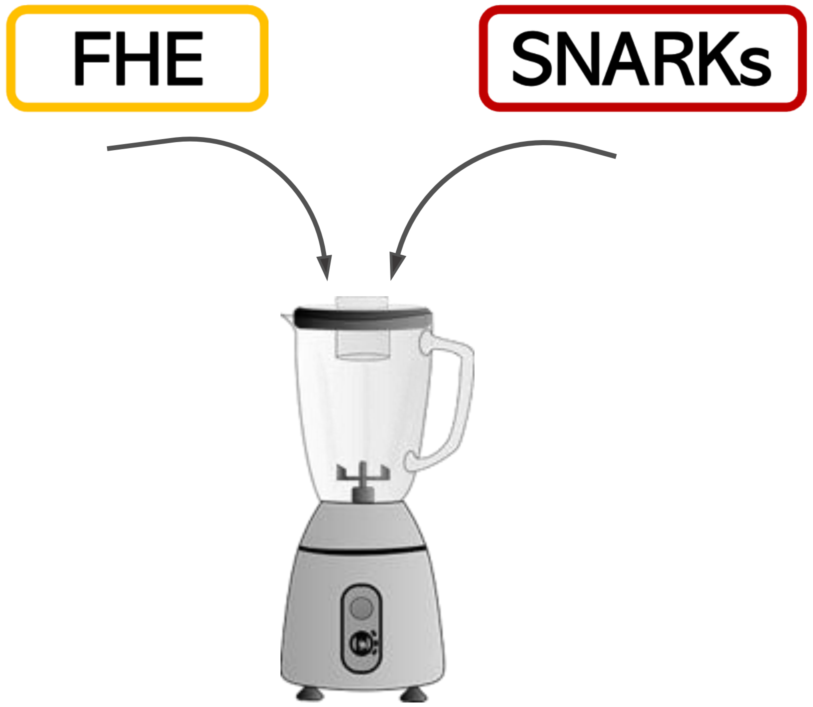
webb

Can we enable private and verifiable delegation of computation?

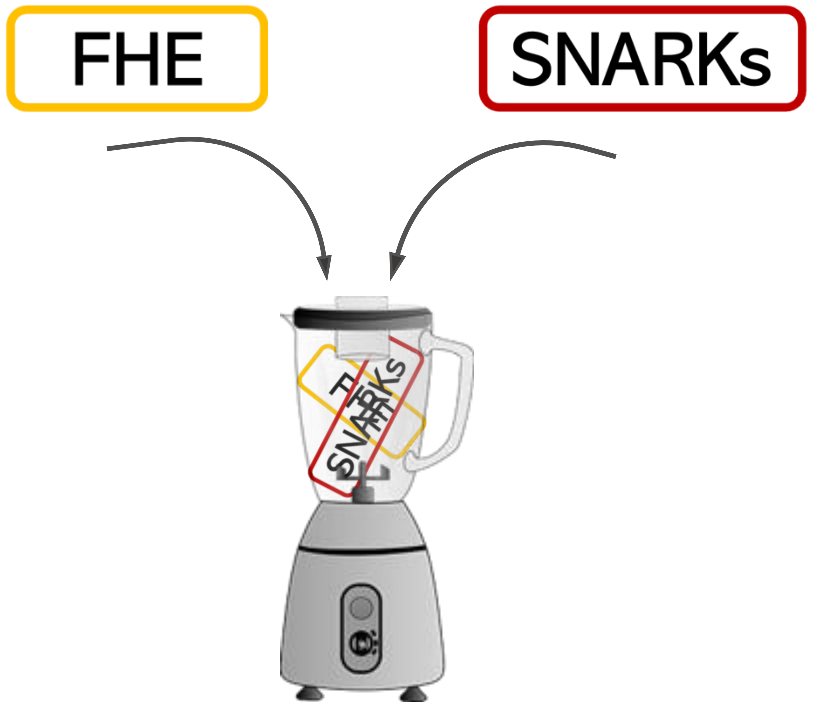
FHE

SNARKs

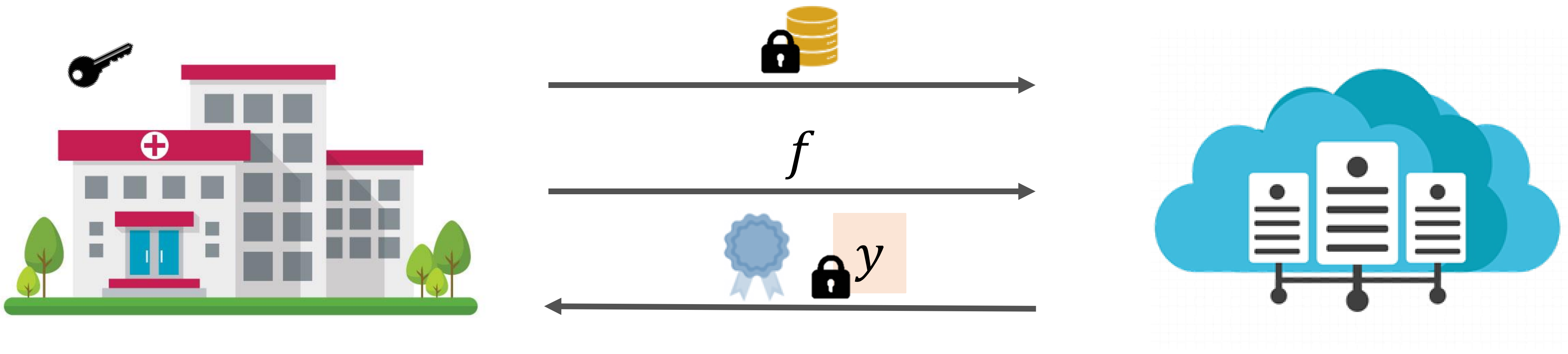
Can we enable private and verifiable delegation of computation?



Can we enable private and verifiable delegation of computation?



FHE + SNARKs: Strawman Approach



Verify [🏅 , 🔒y , f , 🔒🛢]

🔒y $= \text{FHE.eval}(f,$ 🔒🛢 $)$

🔒y $\xrightarrow{\text{🔑}}$ $y = f($🛢$)$

Prove [🔒y $= \text{FHE.eval}(f,$ 🔒🛢 $)$] → 🏅

These Technologies Have Overheads!

$$\mathrm{Time}\left[\mathrm{Prove}\left[y = f(🛢)\right]\right] \gg a \times \mathrm{Time}\left[f(🛢)\right]$$

$$\mathrm{Time}\left[\mathrm{FHE.eval}(f, 🛢🔒)\right] \gg b \times \mathrm{Time}\left[f(🛢)\right]$$

FHE + SNARKs: Strawman Approach

Time [Prove [$y =$ FHE.eval$(f,$ 🔒🛢 $)$]] $\gg$ ab $\times$ Time [$f($🛢$)$]

Prohibitively slow!

Our Result

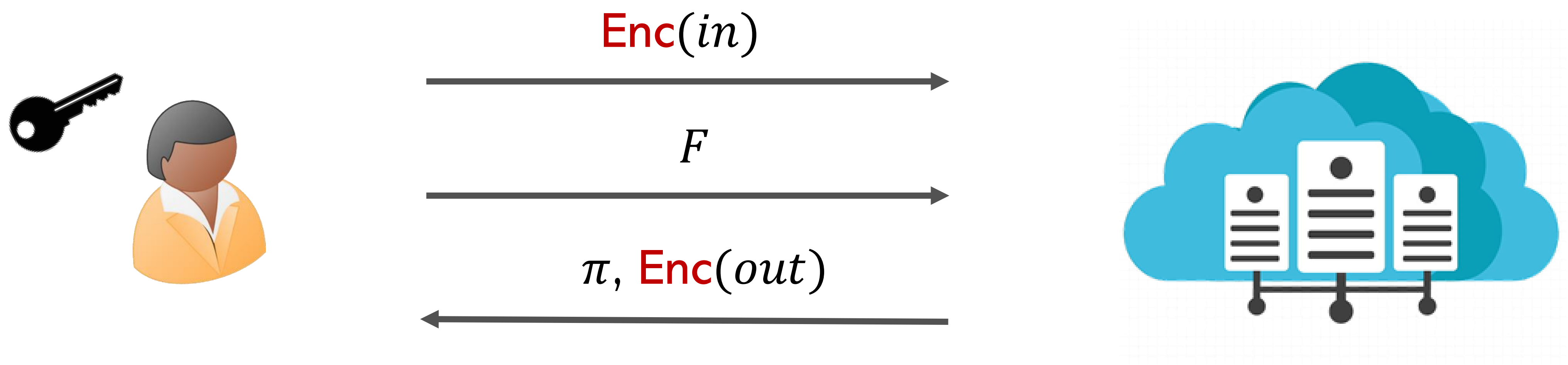
[GGW | CRYPTO24]

An efficient way to combine FHE + SNARKs for private and verifiable delegation of computation to a single untrusted server.

Server's RunTime $\approx (a + b) \times$ Time $\left[\, f(🛢) \,\right]$

Only makes black-box use of FHE.

FHE + SNARKs: Strawman Approach



$\mathsf{Verify}\left[\pi, \mathsf{Enc}(out), \mathsf{Enc}(in), \mathsf{FHE.eval}\,(F,.)\right]$

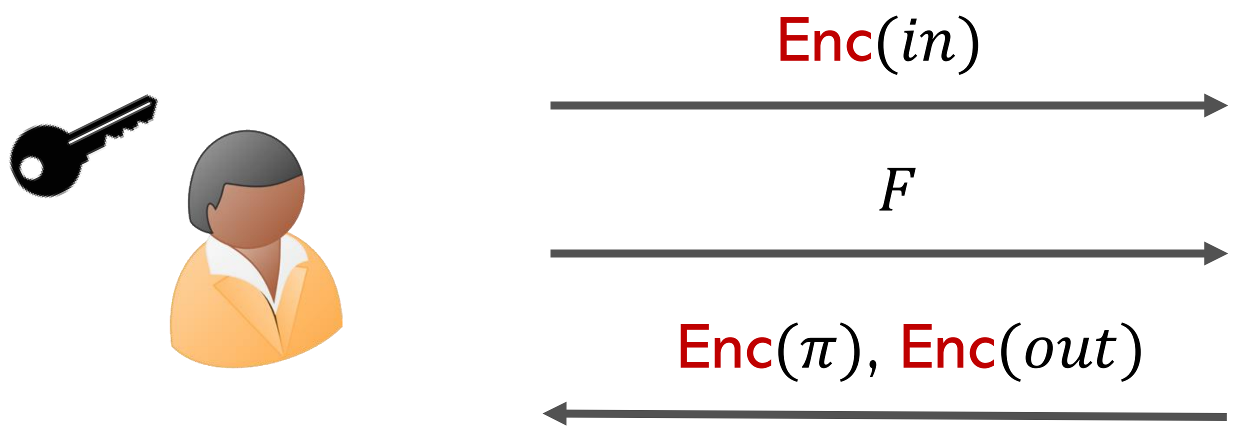
$\mathsf{Enc}(out) = \mathsf{FHE.eval}\,(F, \mathsf{Enc}(in))$

$\mathsf{Dec}\left[\mathsf{Enc}(out)\right] \longrightarrow out$

$\mathsf{Prove}\left[\mathsf{Enc}(out) = \mathsf{FHE.eval}\,(F, \mathsf{Enc}(in))\right] \longrightarrow \pi$

FHE + SNARKs: Another Idea?

Compute an encrypted proof



$\text{Dec}\left[\text{Enc}(\pi)\right] \longrightarrow \pi$

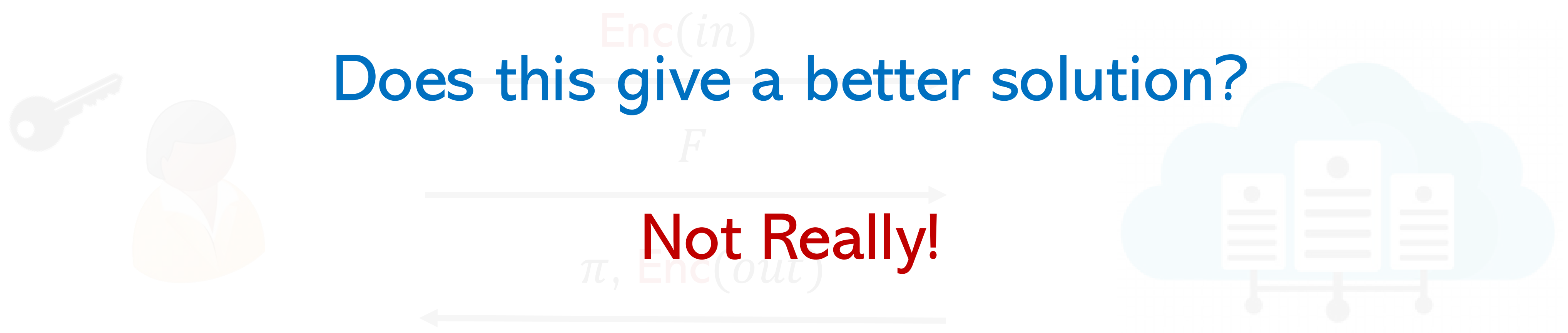
$\text{Dec}\left[\text{Enc}(out)\right] \longrightarrow out$

$\text{Verify}\left[\pi, F, in, out\right]$

$\text{Enc}(out) = \text{FHE.eval}\ (F, \text{Enc}(in))$

$\text{Enc}(\pi) = \text{FHE.eval}\ (\text{Prove}, F, \text{Enc}(in), \text{Enc}(out))$

FHE + SNARKs: Another Idea?



Does this give a better solution?

Not Really!

The overheads from FHE and SNARKs will still get multiplied.

$\text{Dec}\left[\text{Enc}(\pi)\right] \longrightarrow \pi$

$\text{Enc}(out) = \text{FHE.eval}\left(F, \text{Enc}(in)\right)$

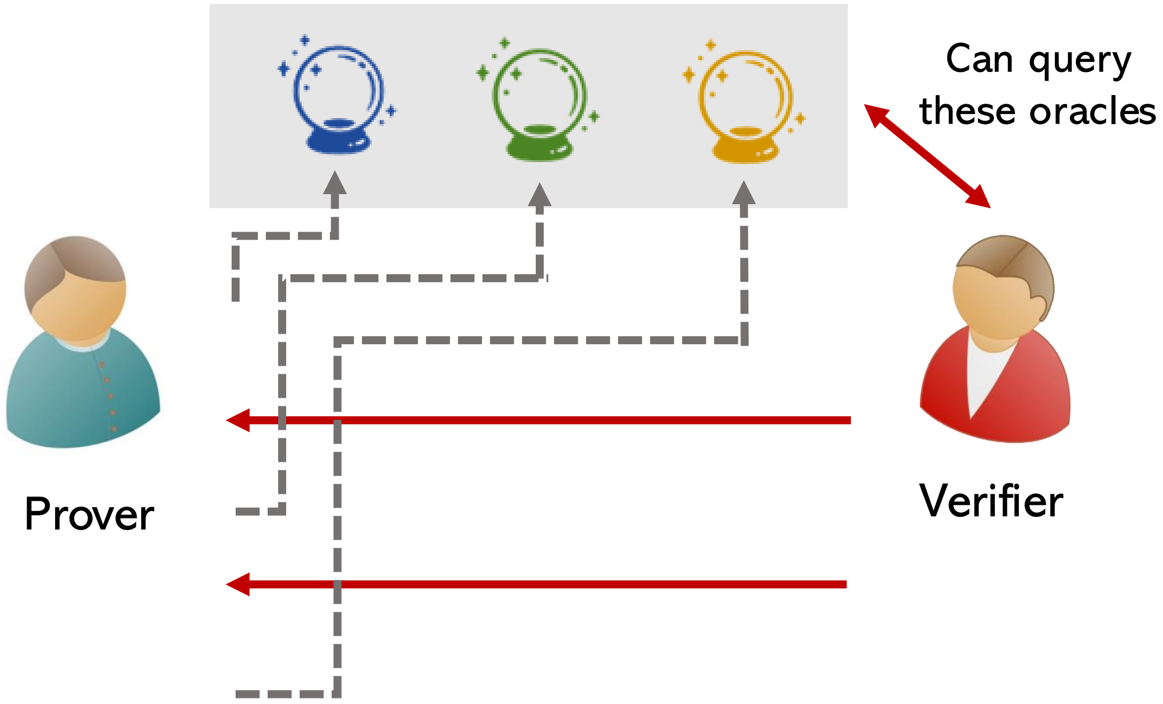
$\text{Dec}\left[\text{Enc}(out)\right] \longrightarrow out$

$\text{Enc}(\pi) = \text{FHE.eval}\left(\text{Prove}, F, \text{Enc}(in), \text{Enc}(out)\right)$

$\text{Verify}\left[\pi, F, in, out\right]$

State-of-the-art SNARKs

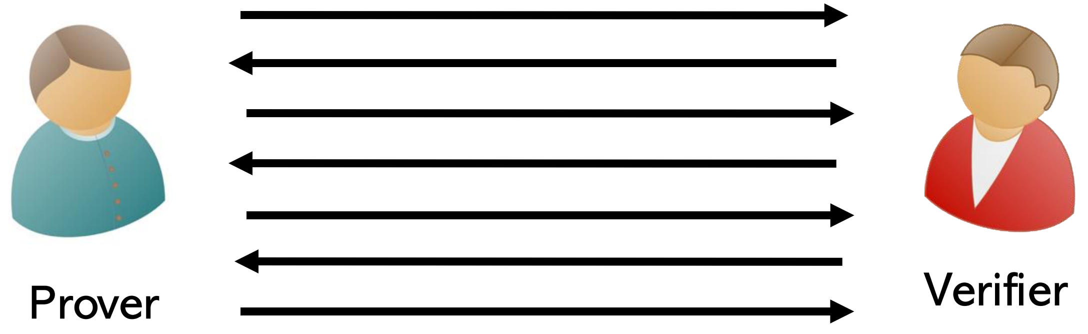
Interactive Oracle Proofs → Interactive Proofs → Non-Interactive Proofs



These are information-theoretic primitives

State-of-the-art SNARKs

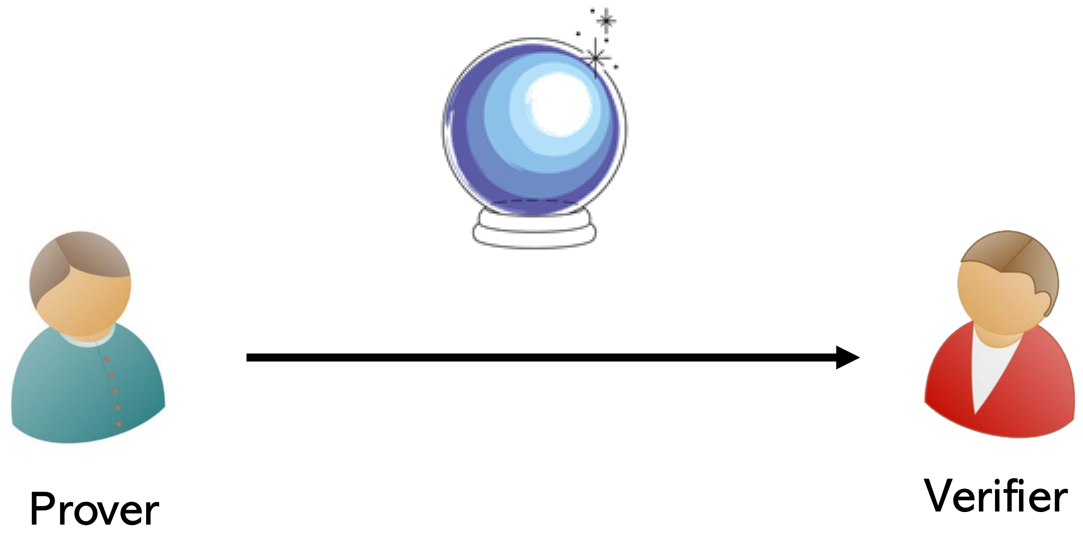
Interactive Oracle Proofs → Interactive Proofs → Non-Interactive Proofs



The transformation to interactive proofs requires the use of cryptography

State-of-the-art SNARKs

Interactive Oracle Proofs → Interactive Proofs → Non-Interactive Proofs



Non-interactive in the random oracle model

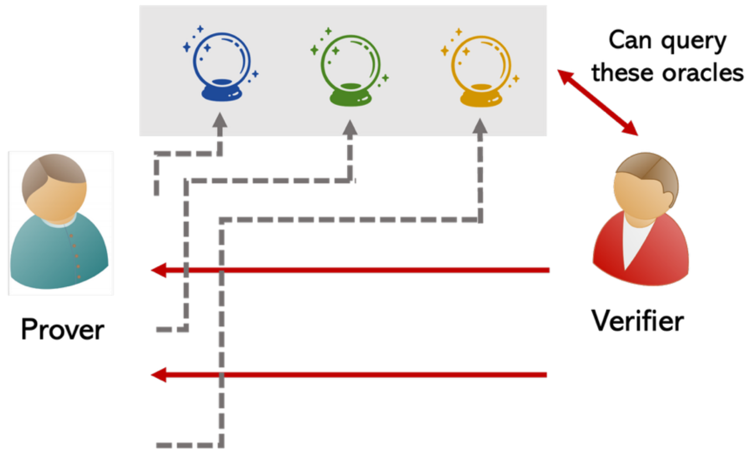
Our Idea

A new compiler for compiling encrypted IOPs

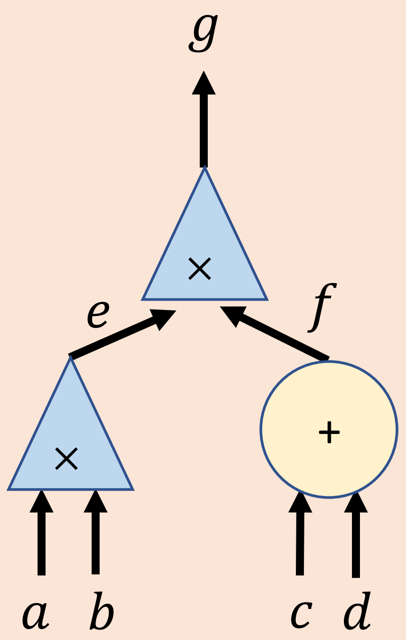
Interactive Oracle Proofs $\rightarrow$ Interactive Proofs $\rightarrow$ Non-Interactive Proofs

Encrypted interactive oracle proof

Interactive Oracle Proofs



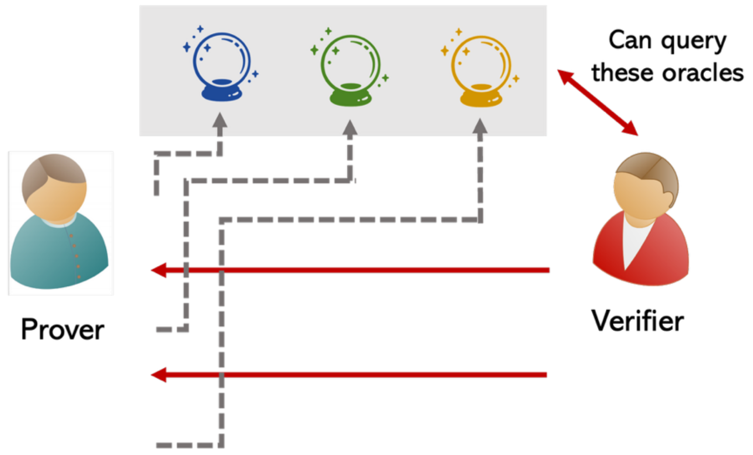
Represent the function to be proven as a circuit



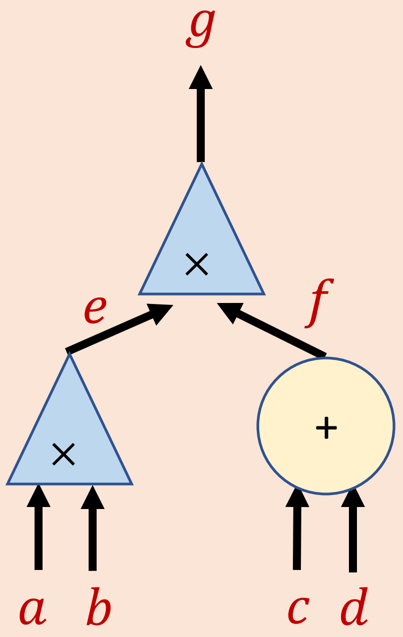
Each oracle can be viewed as a vector of values. The verifier can query linear functions over these values

The computation of each oracle is a function of a, b, c, d, e, f, g

Encrypted Interactive Oracle Proofs



Represent the function F as a circuit. The server uses FHE.eval to compute encryptions of all wire values



Use FHE.eval on encrypted values a, b, c, d, e, f, g to compute encrypted oracles

Each oracle is now a vector of encrypted values.

What does the verifier query in the encrypted oracle?

$Enc(\alpha)$	$Enc(\beta)$	$Enc(\gamma)$	$Enc(\delta)$	$Enc(\varepsilon)$	$Enc(\eta)$	$Enc(\vartheta)$

We observe: The verifier needs to query an encryption of a linear function of the $\alpha, \beta, \gamma, \delta, \ldots$ values.

We design: A new cryptographic compiler for this that remains black-box in FHE.

Thanks!

✉ aarushi@purdue.edu

🌐 https://aarushigoel.github.io/