Decentralized Computing over Encrypted Data

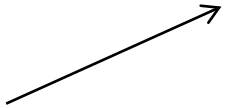
Chloé Hébant

Decentralization

Fully Homomorphic Encryption Gentry 2009

$x_1, \ldots, x_n$

$E_{hom}(x_1), \ldots, E_{hom}(x_n)$



$E_{hom}(f(x_1, \ldots, x_n))$

$f(x_1, \ldots, x_n)$

Fully Homomorphic Encryption

$x_1, \ldots, x_n$

$E_{hom}^{pk}(x_1), \ldots, E_{hom}^{pk}(x_n)$

$E_{hom}^{pk}(f(x_1, \ldots, x_n))$

Re-encryption

Distributed Controller

$E_{hom}^{pk_U}(f(x_1, \ldots, x_n))$

$f(x_1, \ldots, x_n)$

Decentralization

Decentralization = Distribution + No authority

⇒ Efficient decentralized key generation

Outline

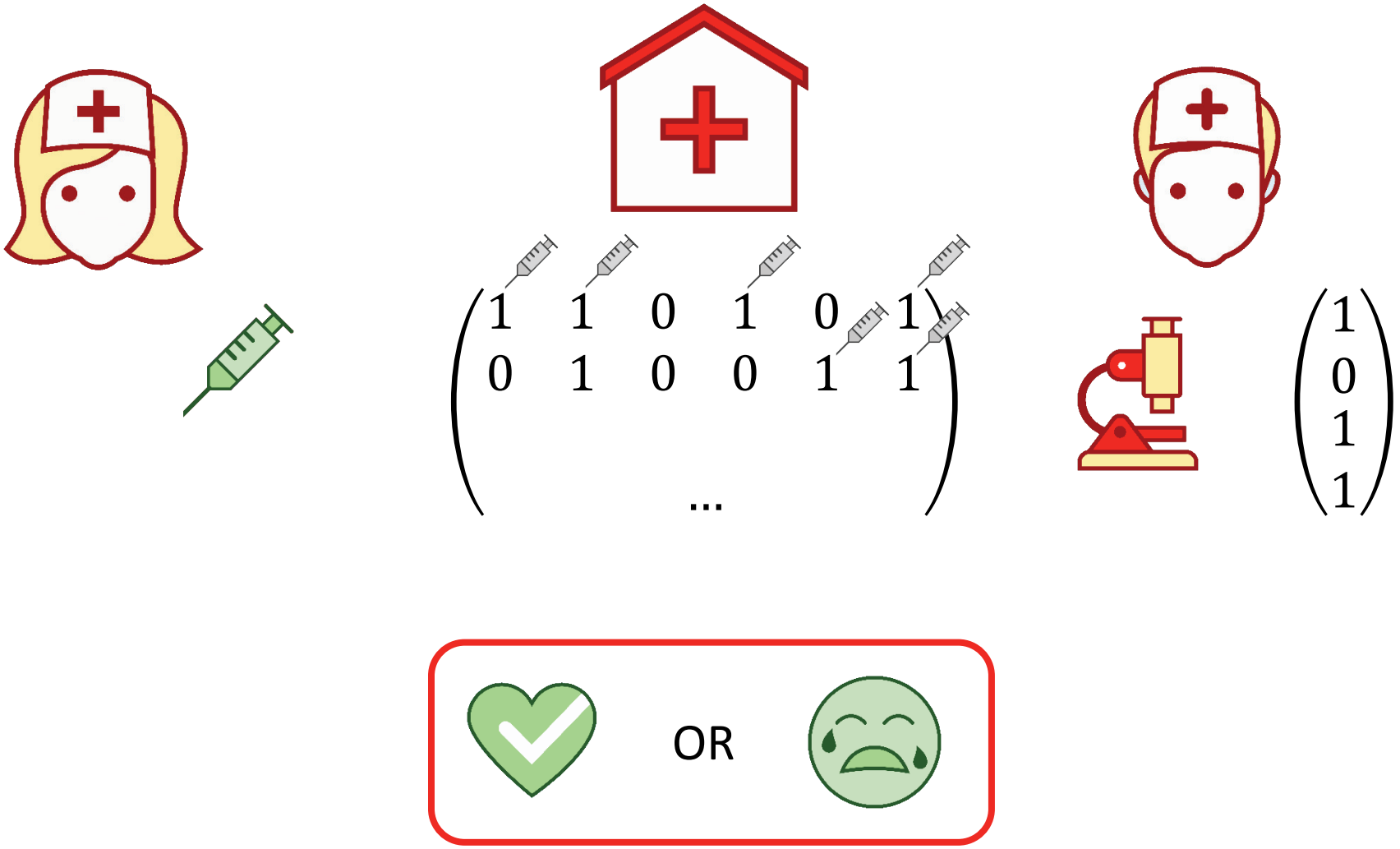
This talk :

Decentralized Re-encryption for a Quadratic Scheme

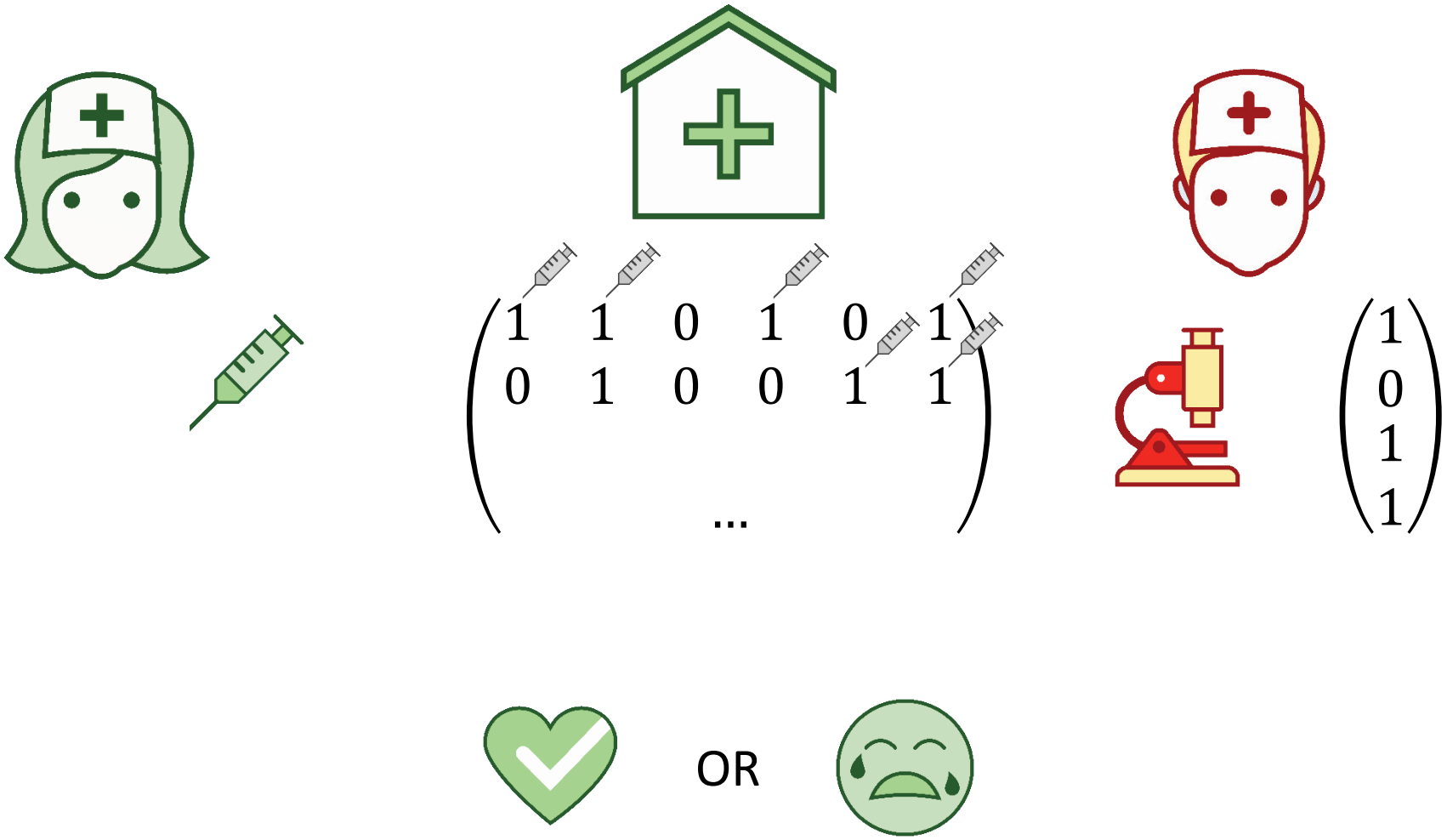
1. Example of application
2. Encryption scheme for quadratic multivariate polynomials
3. Decentralized scheme

Group Testing

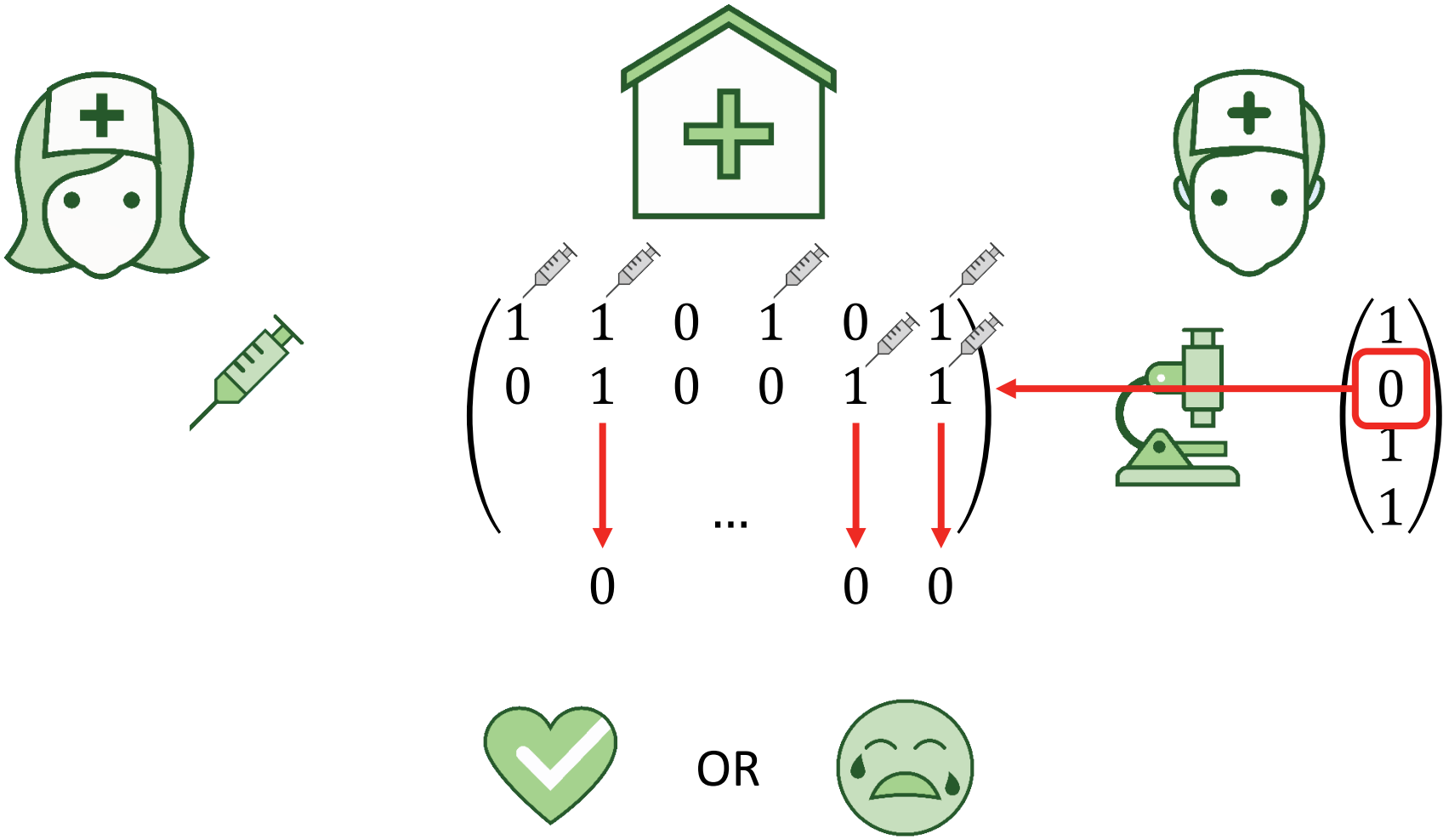
Motivation: Group Testing



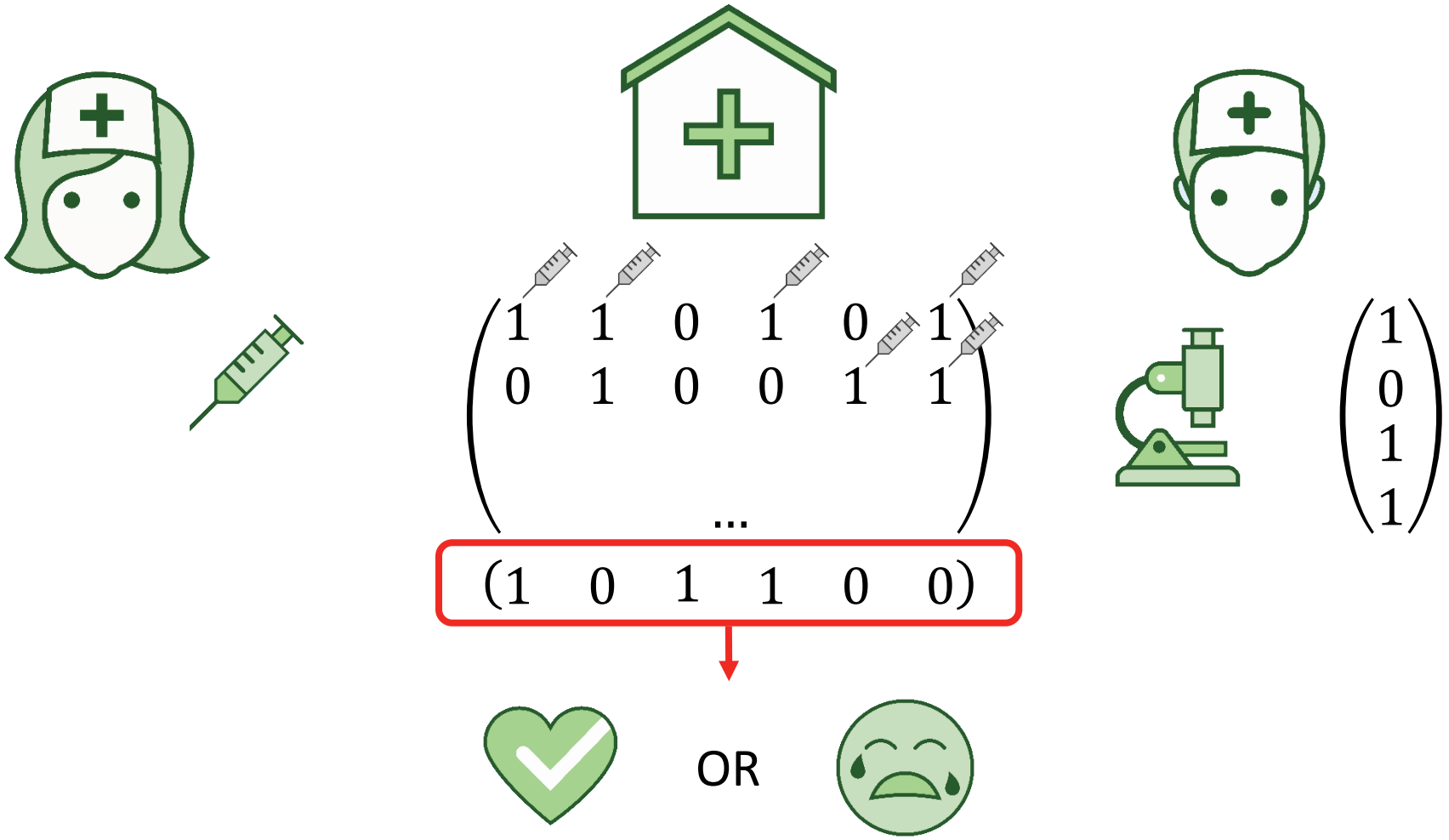
Motivation: Group Testing



Motivation: Group Testing



Motivation: Group Testing



Motivation: Group Testing

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ & & \cdots & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \qquad \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix}$$

$$\bar{F}_j = \bigvee_i (x_{ij} \wedge \bar{y}_i)$$

OR

Motivation: Group Testing

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ & & \cdots & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \qquad \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix}$$

$$\bar{F}_j = \sum_i (x_{ij} \cdot (1 - y_i))$$

OR

2-DNF on Encrypted Data

$$x_1, \dots, x_n \in \{0,1\}$$

2-DNF:

$$\bigvee_{i=1}^{m} (\ell_{i,1} \wedge \ell_{i,2}) \qquad \ell_{i,1} \wedge \ell_{i,2} \in \{x_1, \dots, x_n\} \cup \{\overline{x_1}, \dots, \overline{x_n}\}$$

Multivariate polynomial degree 2:

$$\sum_{i=1}^{m} (y_{i,1} \cdot y_{i,2}) \qquad \begin{cases} y_{i,j} = \ell_{i,j} & \text{if } \ell_{i,j} \in \{x_1, \dots, x_n\} \\ y_{i,j} = 1 - \ell_{i,j} & \text{if } \ell_{i,j} \in \{\overline{x_1}, \dots, \overline{x_n}\} \end{cases}$$

Encryption Scheme

The Encryption Scheme

- BGN 2005
- Freeman 2010
- Our Scheme
  - Multi-user setting
  - Efficient distributed decryption
  - Efficient distributed re-encryption
  - Decentralized key generation

Notations

$\mathbb{G}_s = < g_s >$

$$a \in \mathbb{Z}_p, \qquad [a]_s = g_s^a$$

$$\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{Z}_p^n, \qquad [\boldsymbol{x}]_s = (g_s^{x_1}, \dots, g_s^{x_n})$$

$e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

$$[a]_1 \bullet [b]_2 = [a \otimes b]_T$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11} \cdot \boldsymbol{B} & a_{12} \cdot \boldsymbol{B} \\ a_{21} \cdot \boldsymbol{B} & a_{22} \cdot \boldsymbol{B} \end{pmatrix}$$

The Encryption Scheme

Keygen

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

The Encryption Scheme

Keygen

Projection

$$\boldsymbol{P}_s = \left(\boldsymbol{B}_s^{-1}\right)\left(\boldsymbol{U}_2\right)\left(\boldsymbol{B}_s\right) \in \mathrm{GL}_2(\mathbb{Z}_p)$$

$$\mathrm{sk}_s = \boldsymbol{P}_s$$

$$\boldsymbol{p}_s \in \ker(\boldsymbol{P}_s) = \{\boldsymbol{x}: \boldsymbol{x} \cdot \boldsymbol{P}_s = (0 \quad 0)\}$$

$$\mathrm{pk}_s = [\boldsymbol{p}_s]_s \Rightarrow [\boldsymbol{p}_s]_s \cdot \boldsymbol{P}_s = [(0 \quad 0)]_s$$

The Encryption Scheme

- Keygen:

$$\mathrm{sk}_S = \boldsymbol{P_s} = \left(\boldsymbol{B_s^{-1}}\right)\left(\boldsymbol{U_2}\right)\left(\boldsymbol{B_s}\right) \qquad \mathrm{sk}_T = (\mathrm{sk}_1, \mathrm{sk}_2)$$

$$\mathrm{pk}_S = [\boldsymbol{p}_s]_s \Rightarrow [\boldsymbol{p}_s]_s \cdot \boldsymbol{P_s} = [\boldsymbol{0}]_s \qquad \mathrm{pk}_T = (\mathrm{pk}_1, \mathrm{pk}_2)$$

- Encrypt:
  - $C_s = \left([\boldsymbol{c}_{s,1}]_s, [\boldsymbol{c}_{s,2}]_s\right) = (m \cdot [a_s]_s + \underbrace{r \cdot [\boldsymbol{p}_s]_s}_{\in \ker(\boldsymbol{P_s})}, [a_s]_s)$ $\quad r \in_\$ \mathbb{Z}_p$
  - $C_T = \left([\boldsymbol{c}_{T,1}]_T, [\boldsymbol{c}_{T,2}]_T\right) = (m \cdot [a_1]_1 \bullet [a_2]_2 + \underbrace{[\boldsymbol{p}_1]_1 \bullet [\boldsymbol{r}_2]_2 + [\boldsymbol{r}_1]_1 \bullet [\boldsymbol{p}_2]_2}_{\in \ker(\boldsymbol{P_1} \otimes \boldsymbol{P_2})},$
    $[a_1]_1 \bullet [a_2]_2)$ $\quad [r_1]_1 \in_\$ \mathbb{G}_1^2, [r_2]_2 \in_\$ \mathbb{G}_2^2$

- Decrypt:
  - $C_s \cdot \boldsymbol{P_s} = (m \cdot [a_s]_s \cdot \boldsymbol{P_s} + [\boldsymbol{0}]_s, [a_s]_s \cdot \boldsymbol{P_s})$
  - $C_T \cdot (\boldsymbol{P_1} \otimes \boldsymbol{P_2}) = (m \cdot [a_1]_1 \bullet [a_2]_2 \cdot (\boldsymbol{P_1} \otimes \boldsymbol{P_2}) + [\boldsymbol{0}]_T, [a_1]_1 \bullet [a_2]_2 \cdot (\boldsymbol{P_1} \otimes \boldsymbol{P_2}))$

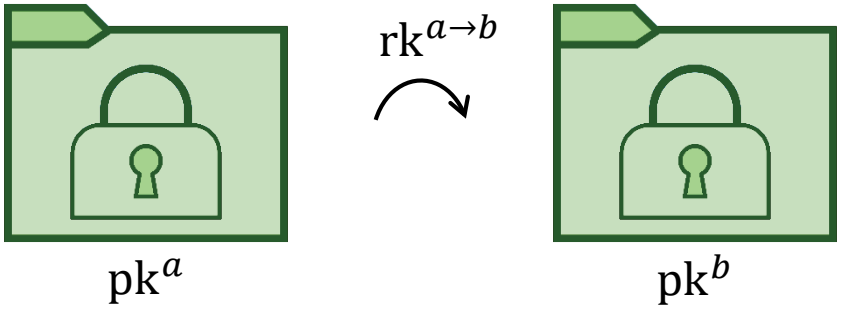
The Homomorphic Properties

- Add: *Many times*
  - $[\boldsymbol{c}_s]_s + [\boldsymbol{c'}_s]_s = (m + m') \cdot [\boldsymbol{a}_s]_s + (r + r') \cdot [\boldsymbol{p}_s]_s$
  - $[\boldsymbol{c}_T]_T + [\boldsymbol{c}_T]_T = (m + m') \cdot [\boldsymbol{a}_1]_1 \bullet [\boldsymbol{a}_2]_2 + [\boldsymbol{p}_1]_1 \bullet [\boldsymbol{r}_2 + \boldsymbol{r'}_2]_2 +$
    $[\boldsymbol{r}_1 + \boldsymbol{r'}_1]_1 \bullet [\boldsymbol{p}_2]_2$
- Multiply: *Once*
  - $[\boldsymbol{c}_1]_1 \bullet [\boldsymbol{c}_2]_2 = (m_1 \cdot m_2) \cdot [\boldsymbol{a}_1]_1 \bullet [\boldsymbol{a}_2]_2 + [\boldsymbol{p}_1]_1 \bullet [\boldsymbol{r'}]_2 + [\boldsymbol{r}]_1 \bullet [\boldsymbol{p}_2]_2$

    with $[\boldsymbol{r}]_1 = m_1 r_2 \boldsymbol{a}_1$

    $[\boldsymbol{r'}]_2 = m_2 r_1 \boldsymbol{a}_2 + r_1 r_2 \boldsymbol{p}_2$

Re-Encryption



$\boldsymbol{P = B^{-1}U_2B}$ sk^{a} sk^{b} $\boldsymbol{P' = B'^{-1}U_2B'}$

$\mathrm{rk}^{a\to b}$

$\boldsymbol{R = B^{-1}B'}$

Problem

Problem

- Distributed decryption and re-encryption ?
  - Yes, with distributed keys
- Decentralized key generation ?

$$\boldsymbol{P}_s = \left(\boldsymbol{B}_s^{-1} \right) \left(\boldsymbol{U}_2 \right) \left(\boldsymbol{B}_s \right)$$

  - No …

Simplification

- Size of the keys:

$$\boldsymbol{P}_s = \begin{pmatrix} 1 & 0 \\ x & 0 \end{pmatrix} \qquad \mathrm{sk}_s = x$$

$$[\boldsymbol{p}_s]_s = [-x \quad 1]_s \qquad \mathrm{pk}_s = [-x]_s$$

- Size of the ciphertexts:

$$[\boldsymbol{a}_s]_s = [1 \quad 0]_s$$

$$\begin{array}{l} C_s \in \mathbb{G}_s^2 \times \mathbb{G}_s^2 \Rightarrow \\ C_T \in \mathbb{G}_T^4 \times \mathbb{G}_T^4 \Rightarrow \end{array} \boxed{\begin{array}{c} C_s \in \mathbb{G}_s^2 \\ C_T \in \mathbb{G}_T^4 \end{array}}$$

The Optimized Encryption Scheme

- Keygen:

$\mathrm{sk}_s = x$ $\qquad$ $\mathrm{sk}_T = (\mathrm{sk}_1, \mathrm{sk}_2)$

$\mathrm{pk}_s = [-x]_s$ $\qquad$ $\mathrm{pk}_T = (\mathrm{pk}_1, \mathrm{pk}_2)$

- Encrypt:
  - $C_s = (g_s^m \cdot \mathrm{pk}_s^r, g_s^r)$ $\qquad r \in_\$ \mathbb{Z}_p$
  - $C_T = \begin{cases} c_{T,1} = e(g_1, g_2)^m \cdot e(g_1, \mathrm{pk}_2)^{r_{11}} \cdot e(\mathrm{pk}_1, g_2)^{r_{21}} \\ c_{T,2} = e(g_1, g_2)^{r_{11}} \cdot e(\mathrm{pk}_1, g_2)^{r_{22}} \\ c_{T,3} = e(g_1, \mathrm{pk}_2)^{r_{12}} \cdot e(g_1, g_2)^{r_{21}} \\ c_{T,4} = e(g_1, g_2)^{r_{12}+r_{22}} \end{cases}$ $\qquad r_{11}, r_{12}, r_{21}, r_{22} \in_\$ \mathbb{Z}_p^4$

- Decrypt:
  - $c_{s,1} \cdot c_{s,2}^{\mathrm{sk}_s}$
  - $c_{T,1} \cdot c_{T,2}^{\mathrm{sk}_2} \cdot c_{T,3}^{\mathrm{sk}_1} \cdot c_{T,4}^{\mathrm{sk}_1 \cdot \mathrm{sk}_2}$

Decentralization

Decentralization:
1) Decentratized Key Generation

Shamir Secret Sharing 1979

- k points $(x_1, y_1), \ldots, (x_k, y_k)$ with distinct abscissa
- Theorem (Lagrange interpolation):

$$\exists!\, P(X) \text{ s.t. } \deg(P) = k - 1 \text{ and } P(x_i) = y_i$$

- Shamir Secret Sharing:
  - $sk = x = P(0),\ pk = g^x$
  - $\mathrm{sk}_i = P(i)$ for $i = 1 \ \ldots n$
  - For any subset S of k indices:

$$x = \sum_{j \in S} \lambda_{S,j} sk_j$$

$$y = \prod_{j \in S} v_j^{\lambda_{S,j}} \text{ for } v_j = g^{sk_j}$$

Decentralization:
2) Distributed Re-Encryption

Distributed Re-encryption

- $c_s = (c_{s,1}, c_{s,2})$ under $pk_s \quad \rightarrow \quad C_s = (C_{s,1}, C_{s,2})$ under PK_s
- Shamir Secret Sharing: $\boxed{sk_s = \sum_i \lambda_i \cdot sk_{s,i}}$
- Player i computes:

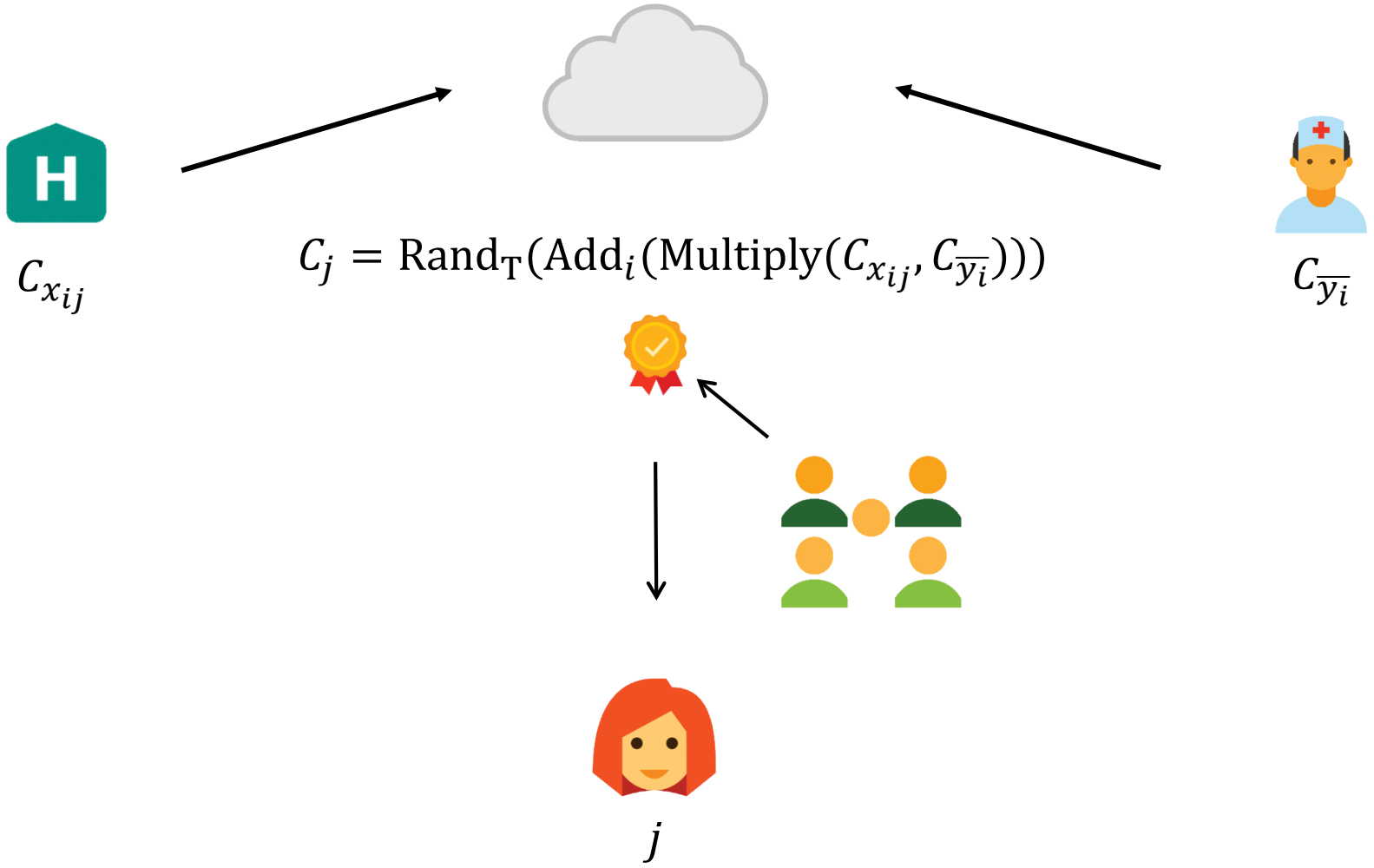
$$r_i' \in_R \mathbb{Z}_p, \alpha_i = c_{s,2}^{sk_{s,i}} \cdot PK_s^{r_i'}, \beta_i = g_s^{r_i'}$$

- Anybody can compute:

$$C_s = (c_{s,1} \times \prod_i \alpha_i^{\lambda_i}, \prod_i \beta_i^{\lambda_i})$$

$$= (g_s^m \cdot PK_s^{r'}, g_s^{r'}) \qquad r' = \sum_i \lambda_i \cdot r_i'$$

Solution: Group Testing



Conclusion

Conclusion

- Efficient scheme to evaluate quadratic multivariate polynomials
  - Distributed decryption
  - Distributed re-encryption
  - Decentralized key generation

- Open problem:

Decentralized FHE

Thank you

ia.cr/2018/1019

Joined work with David Pointcheval and Duong-Hieu Phan