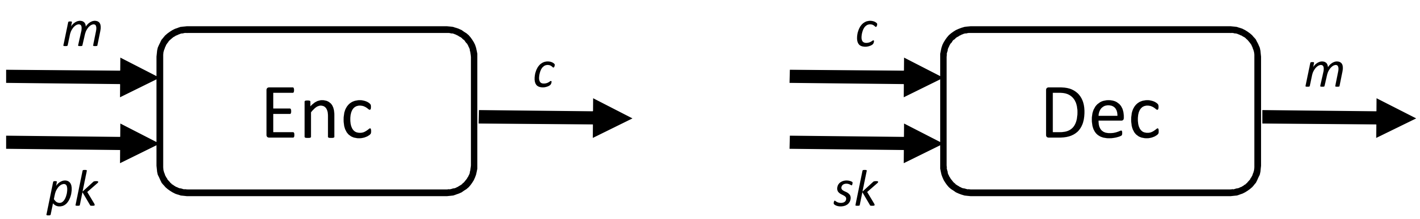
Practical Somewhat Homomorphic Encryption

David Wu
(joint work with Dan Boneh)

Homomorphic Encryption

Homomorphic encryption scheme: encryption scheme that allows computation on ciphertexts

Comprises of three functions:

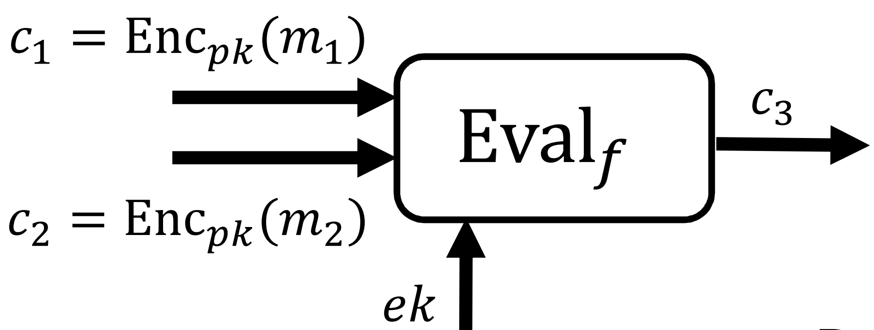


Must satisfy usual notion of semantic security

Homomorphic Encryption

Homomorphic encryption scheme: encryption scheme that allows computation on ciphertexts

Comprises of three functions:



$$\mathrm{Dec}_{sk}\left(\mathrm{Eval}_f(ek, c_1, c_2)\right) = f(m_1, m_2)$$

Fully Homomorphic Encryption (FHE)

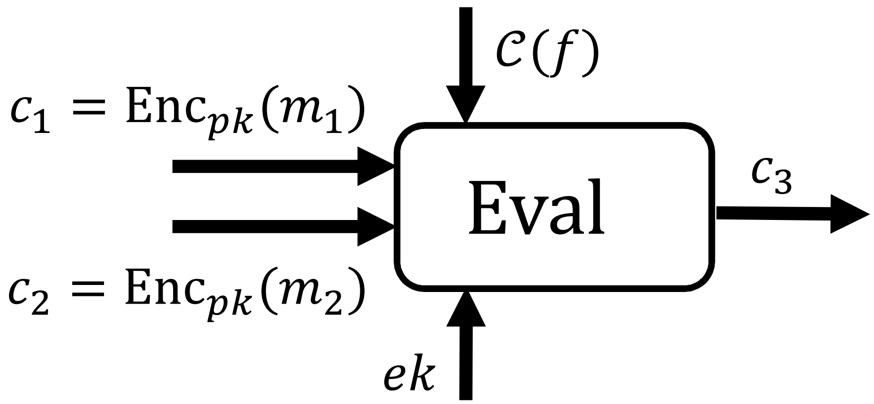
Many homomorphic encryption schemes:

- ElGamal: $f(m_0, m_1) = m_0 m_1$
- Paillier: $f(m_0, m_1) = m_0 + m_1$
- Goldwasser-Micali: $f(m_0, m_1) = m_0 \oplus m_1$

Fully homomorphic encryption: homomorphic with respect to **two** operations: addition and multiplication

- Can evaluate Boolean and arithmetic circuits
- [BGN05]: one multiplication, many additions
- [Gen09]: first FHE construction from lattices

Fully Homomorphic Encryption



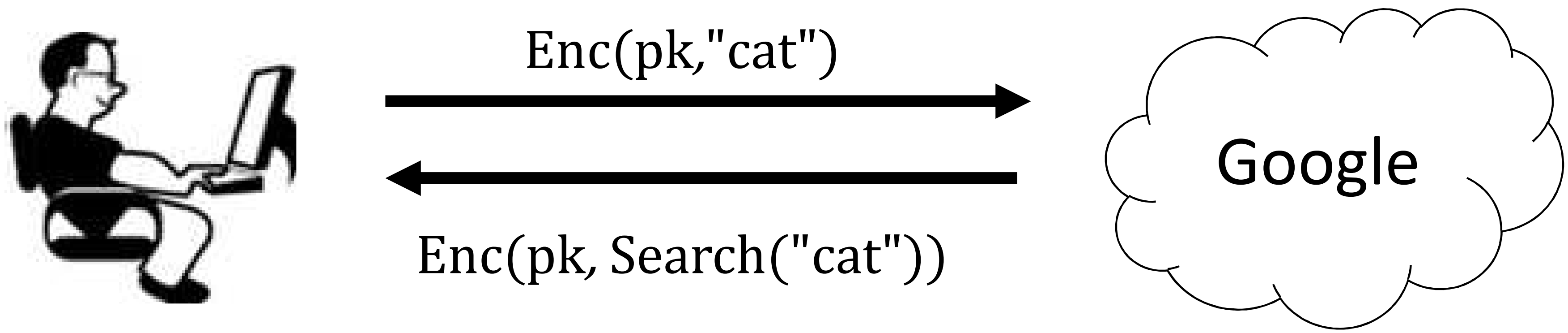
$\mathcal{C}(f)$: circuit for some function f

Correctness: $\mathrm{Dec}_{sk}\left(\mathrm{Eval}_f(ek, c_1, c_2)\right) = f(m_1, m_2)$

Circuit Privacy: $\mathrm{Enc}_{pk}\big(\mathcal{C}(m_1, m_2)\big) \approx \mathrm{Eval}_f(ek, c_1, c_2)$

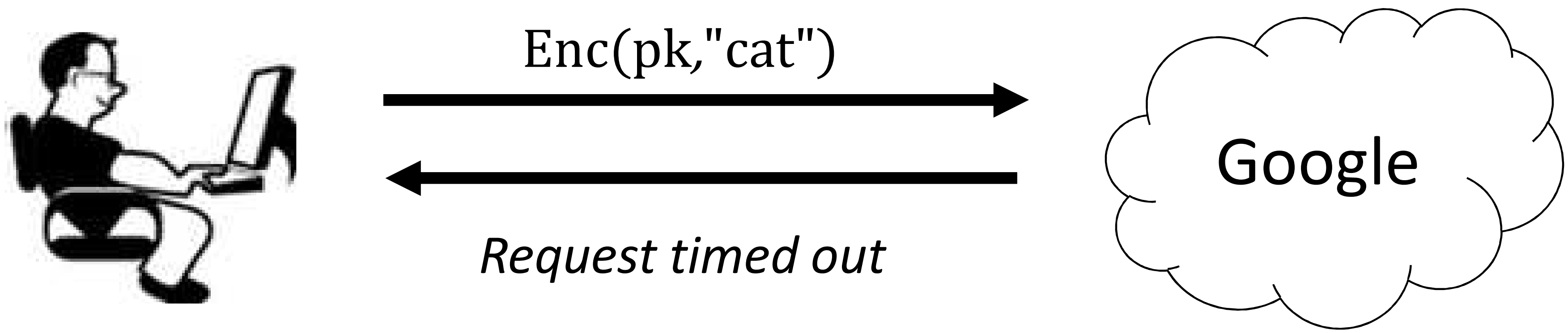
Compactness: Decryption circuit has size at most $\mathrm{poly}(\lambda)$

In Theory: Secure Computation using FHE



Represent “Search” function as a circuit and evaluate homomorphically

In Practice: Secure Computation using FHE



FHE schemes have tremendous overhead

Somewhat Homomorphic Encryption (SWHE)

FHE supports arbitrary number of operations

Compromise: Support a *limited* number of operations (e.g., evaluate circuits of a certain depth)

- Somewhat/leveled homomorphic encryption

Brakerski's SWHE [Bra12]

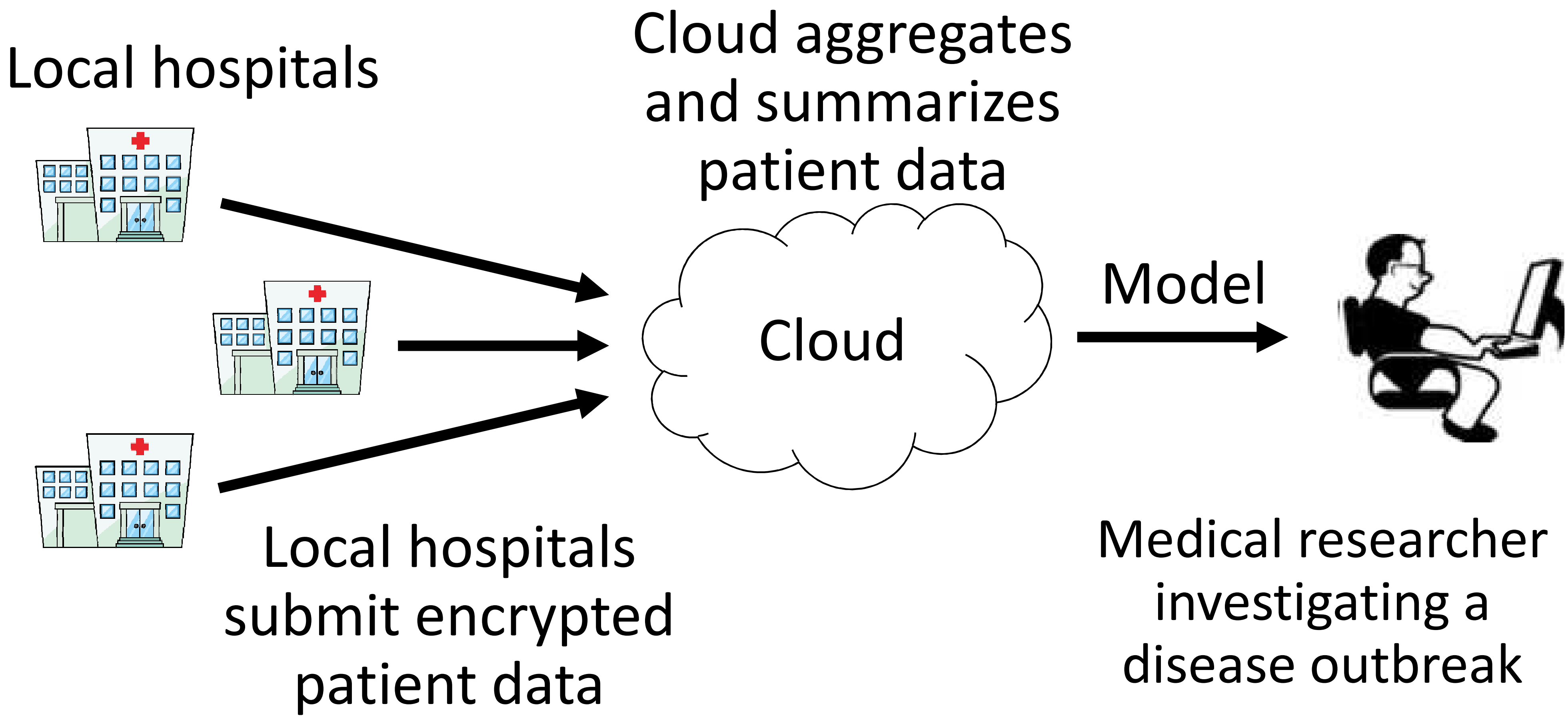
Operates over a polynomial ring: $R = \mathbb{Z}[x]/\Phi_m(x)$

Plaintext and ciphertext are vectors of ring elements

Homomorphic multiplication much more expensive than homomorphic addition

- Can evaluate *low* degree polynomials over encrypted data

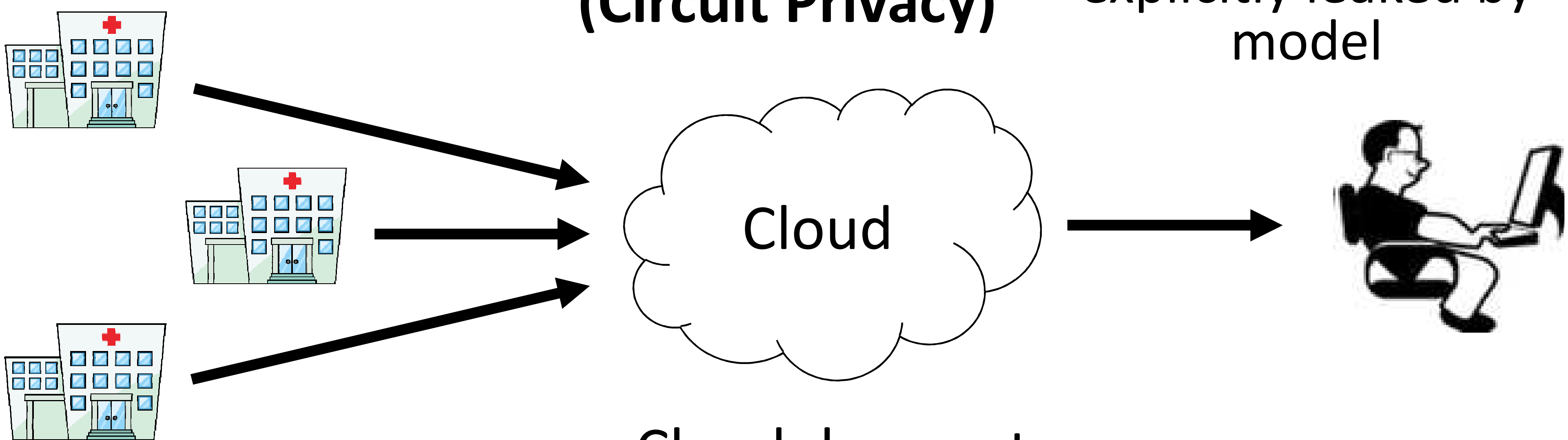
Application: Statistical Analysis



Security Model

Researcher does not learn individual patient data other than what is explicitly leaked by model

(Circuit Privacy)



Cloud does not learn patient data or model

(Semantic Security)

Application: Statistical Analysis

Given n vectors $x_1, \dots, x_n$ (e.g., patient profiles), define X to be the matrix with rows $x_1, \dots, x_n$

- Mean: $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$
- Covariance: $\Sigma_X = \frac{1}{n^2} (nX^T X - (n\mu)(n\mu\,)^T)$

Division difficult to support, so represent as rationals

Depth 0 circuit for mean, depth 1 for covariance

Application: Statistical Analysis

Can also perform linear regression on encrypted data

Given design matrix X and response vector y, evaluate normal equations:

$$\theta = (X^T X)^{-1} X^T y$$

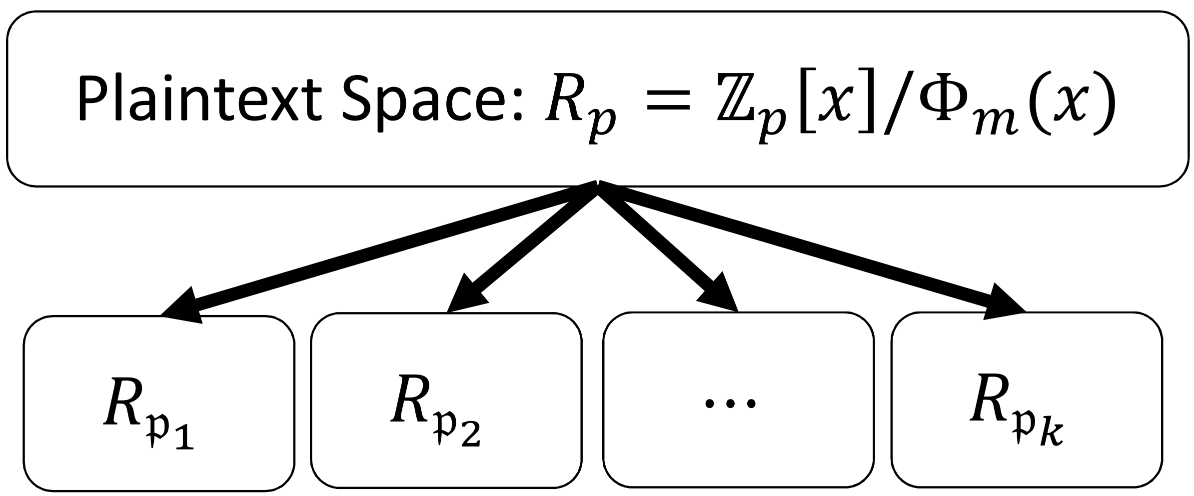
Invert over $\mathbb{Q}$ using Cramer's rule

Depth n for n dimensional data

Batch Computation [SV11]

Encrypt + process array of values at no extra cost

Main intuition: Chinese Remainder Theorem



Choose p such that R_p splits into smaller rings: $R_p \cong \bigotimes_{i=1}^{n} R_{\mathfrak{p}_i}$

Batch Computation

Encrypt + process array of values at no extra cost:

1	2	3	4

$+$

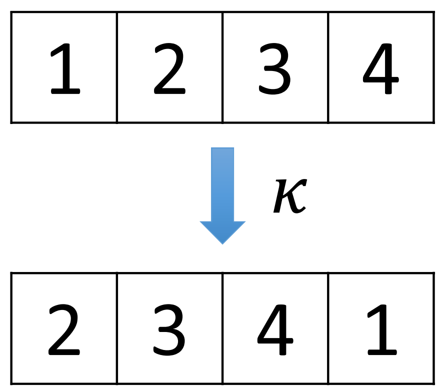
7	5	3	1

8	7	6	5

In practice: ≥ 5000 slots

Batch Computation

Can also permute slots (via Frobenius automorphisms) [BGV12, GHS12]



Batch Inner Products

Statistical analysis reduces to computing inner products:

$$\begin{pmatrix}1\\2\\3\\4\end{pmatrix}^T \cdot \begin{pmatrix}1\\0\\0\\1\end{pmatrix} = 1\cdot 1 + 2\cdot 0 + 3\cdot 0 + 4\cdot 1 = 5$$

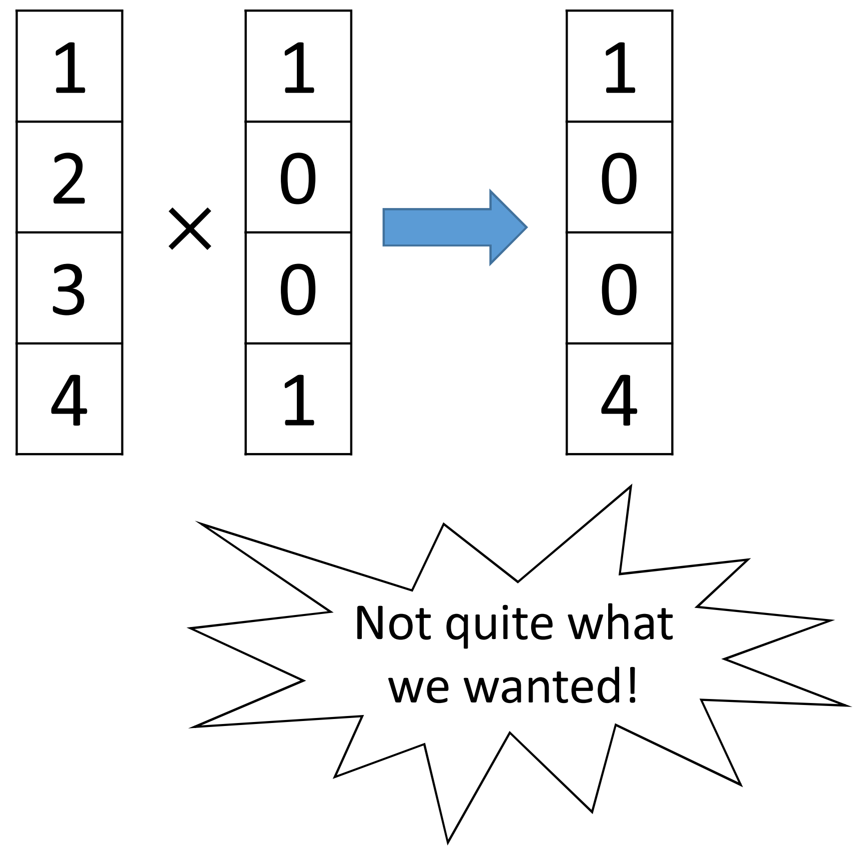
Naïve method: Encrypt each component separately.

Requires 4 multiplications!

Batch Inner Products

Batch inner product: encrypt multiple components in each ciphertext.

Requires 1 multiplication!



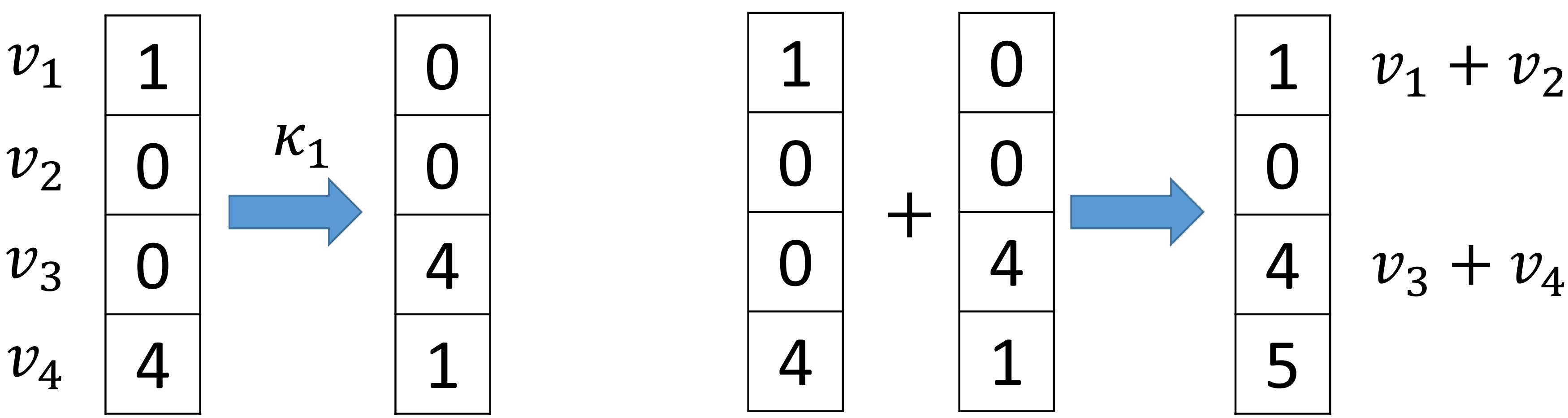
Batch Inner Products

Result of batch multiplication: $\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline 0 \\ \hline 4 \\ \hline \end{array}$

Desired result: $\boxed{5}$

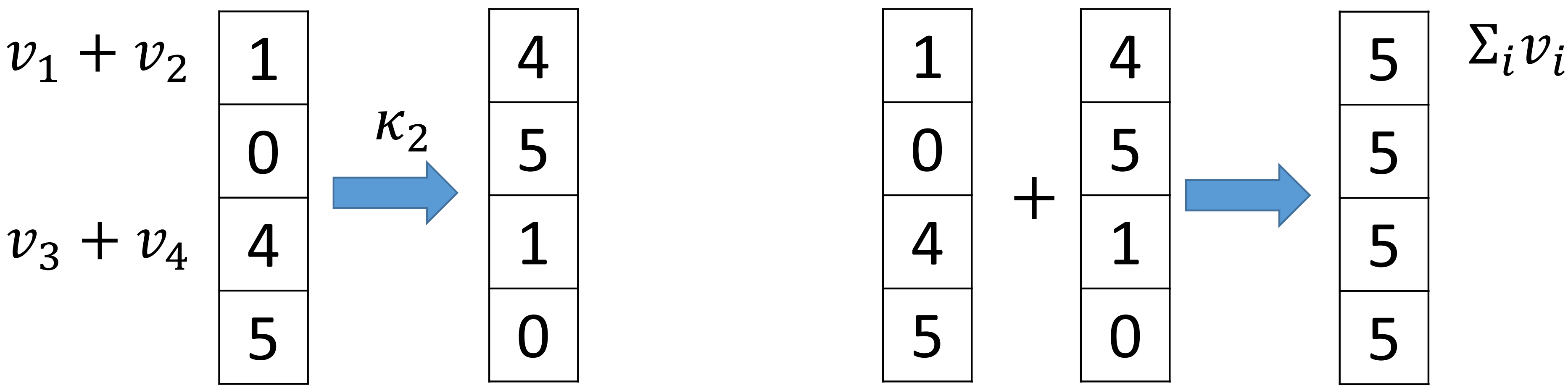
Batch Inner Products

Use automorphisms to sum up components:

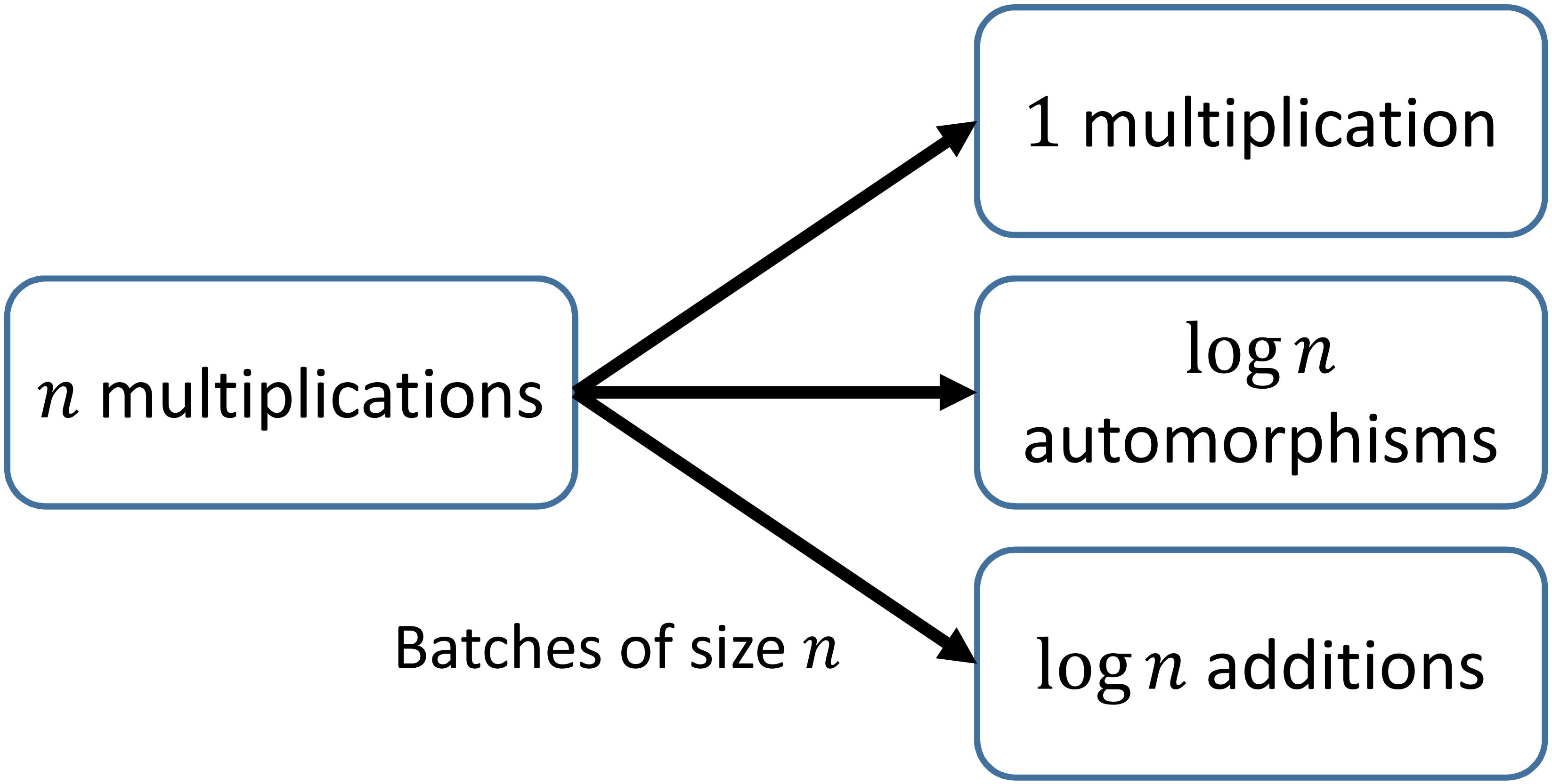


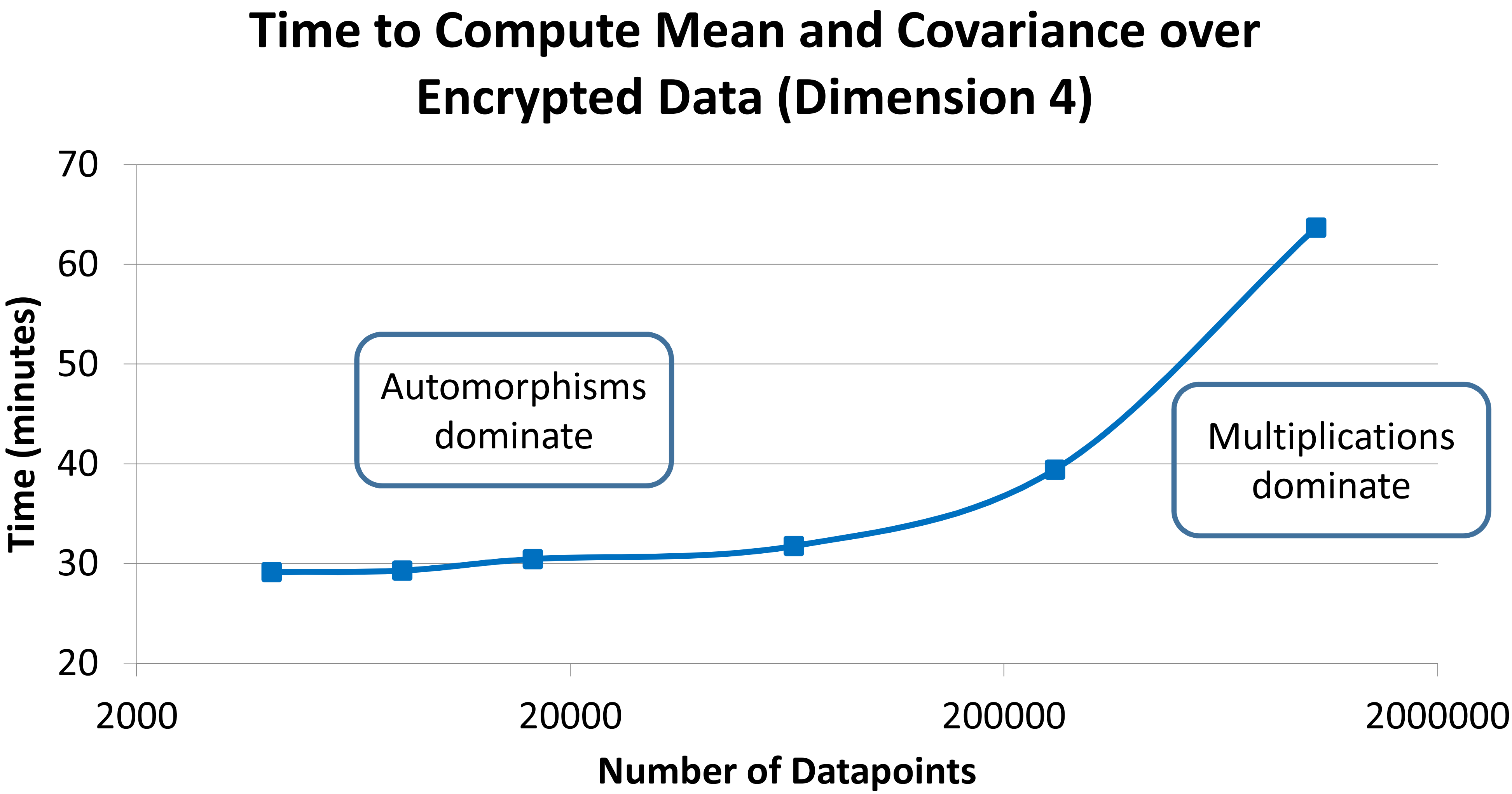
Batch Inner Products

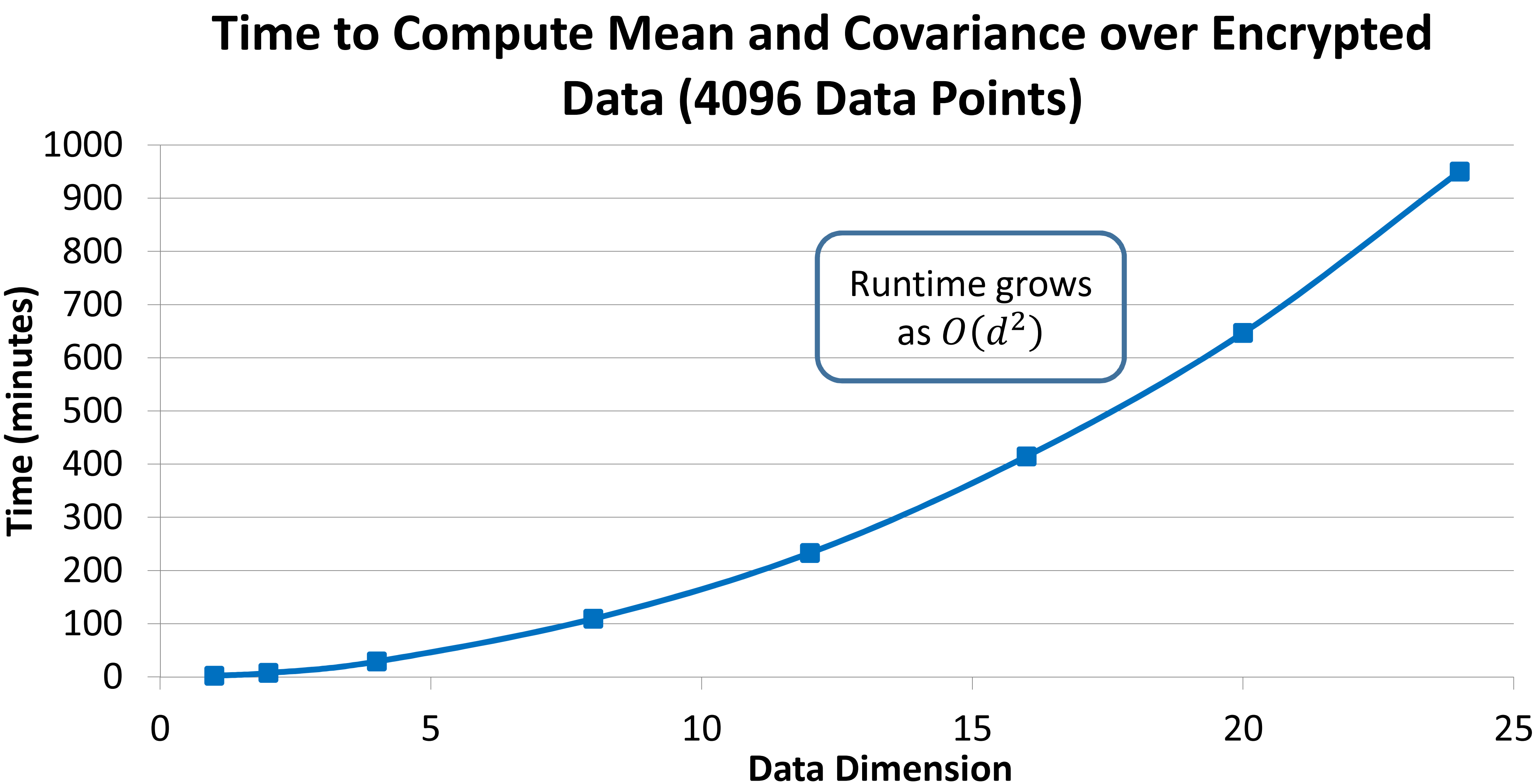
Use automorphisms to sum up components:

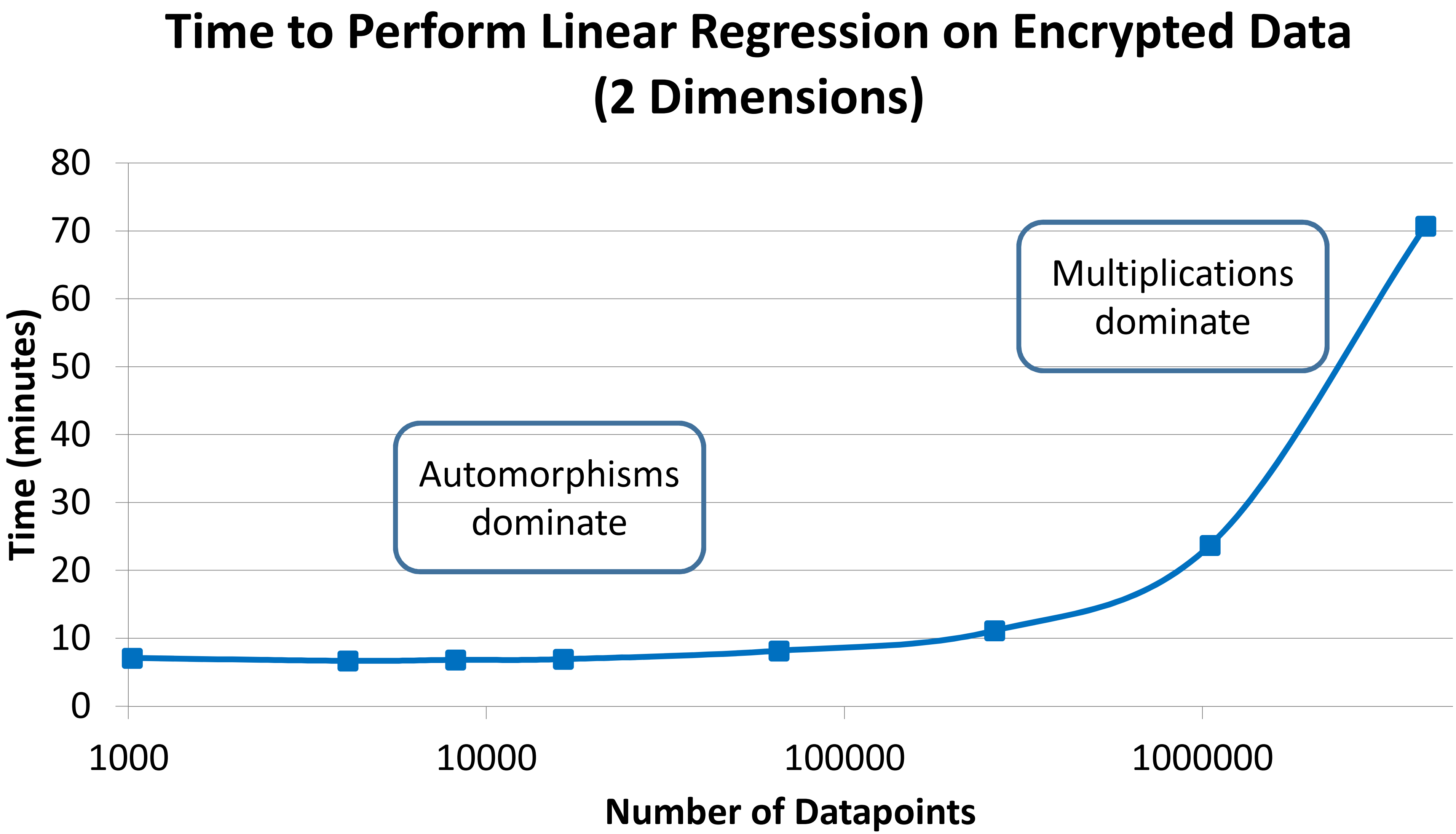


Batch Inner Products

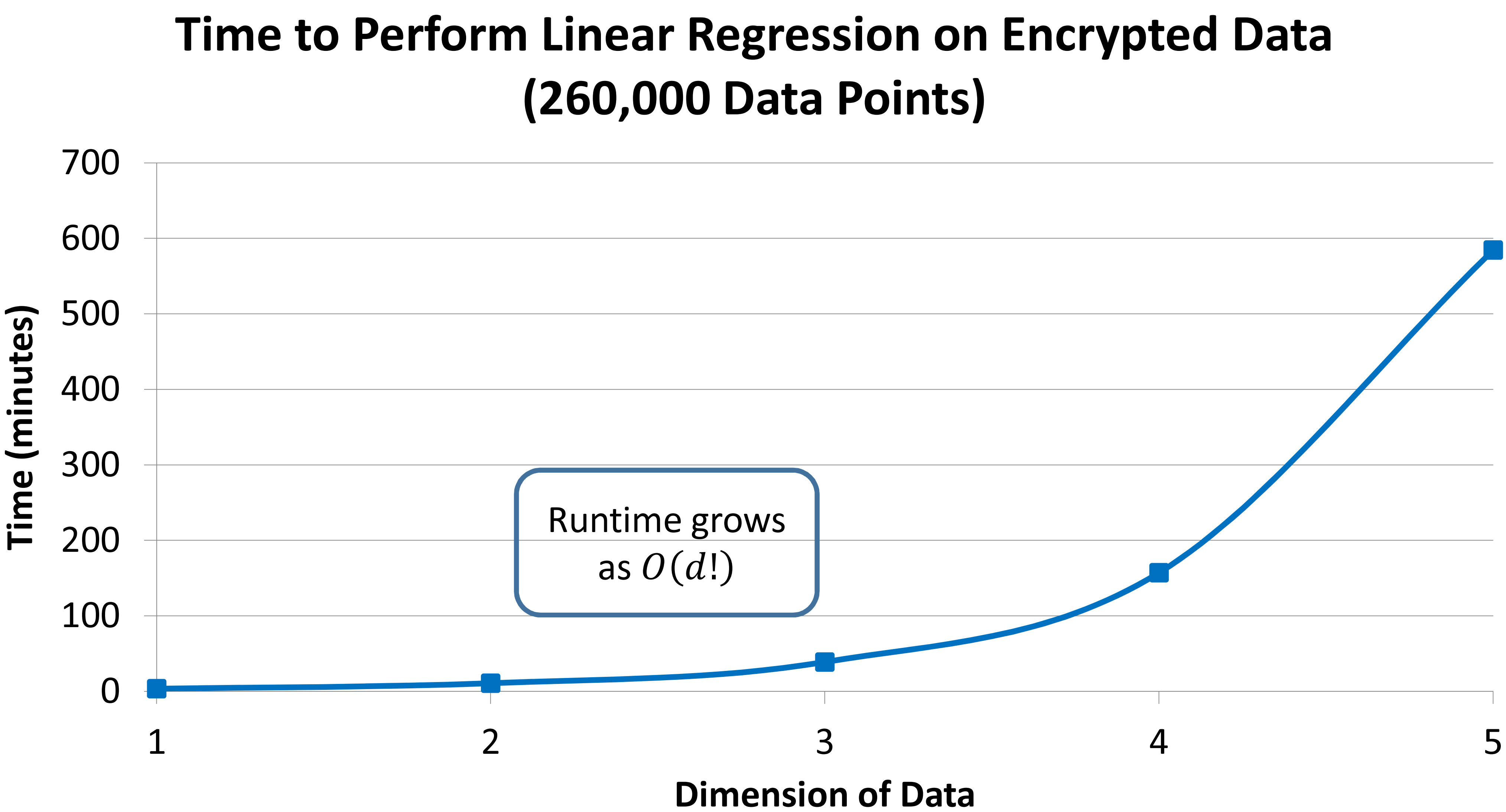



Time to Compute Mean and Covariance over Encrypted Data (Dimension 4)
Time (minutes)
70
60
50
40
30
20
Automorphisms dominate
Multiplications dominate
2000
20000
200000
2000000
Number of Datapoints


Time to Compute Mean and Covariance over Encrypted Data (4096 Data Points)
Runtime grows as $O(d^2)$
Time (minutes)
0
100
200
300
400
500
600
700
800
900
1000
0
5
10
15
20
25
Data Dimension



Time to Perform Linear Regression on Encrypted Data
(2 Dimensions)
Time (minutes)
0
10
20
30
40
50
60
70
80
1000
10000
100000
1000000
Number of Datapoints
Automorphisms dominate
Multiplications dominate



Time to Perform Linear Regression on Encrypted Data
(260,000 Data Points)
700
600
500
400
300
200
100
0
Time (minutes)
Runtime grows
as $O(d!)$
1
2
3
4
5
Dimension of Data

Conclusions

SWHE allows computation of circuits of low-depth

Batching enables scaling to nontrivial datasets

Can perform statistical analysis on encrypted data with “reasonable” overhead

Open Source FHE Implementation:

`https://github.com/dwu4/fhe-si`