Duality

Fully Homomorphic Encryption for Machine Learning

Yuriy Polyakov

ypolyakov@dualitytech.com

Agenda

- Introduction to FHE
- Multiparty FHE
- FHE approaches for machine learning
- Multiparty FHE for ML
- Selected studies in PPML using FHE

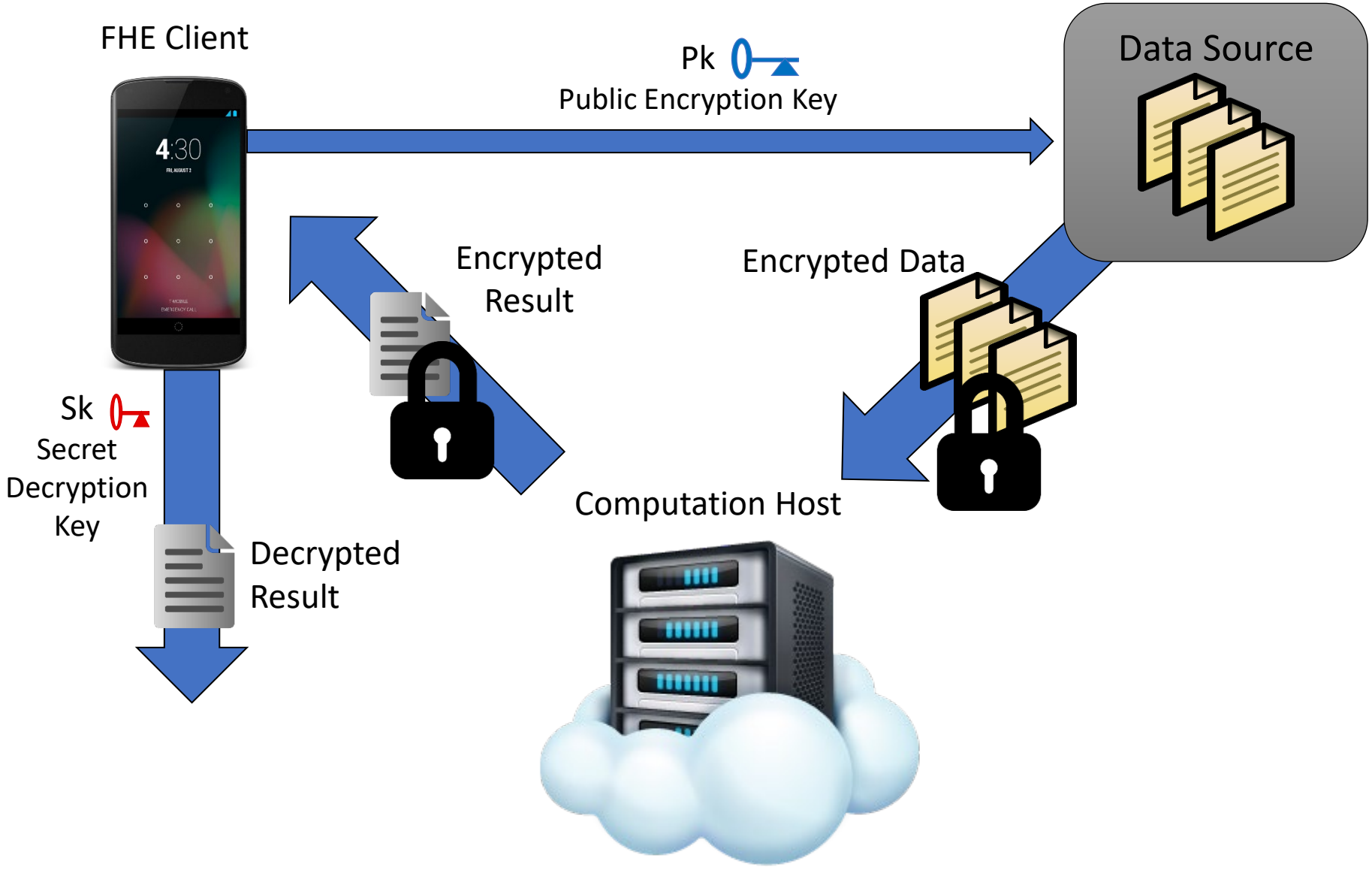
Introduction to FHE

WHAT IS HOMOMORPHIC ENCRYPTION?

- Encryption protocol with one extra operation: Evaluation
  - Allows for computation on encrypted data
  - Enables outsourcing of data storage/processing
- How is FHE related to symmetric and public key encryption?
  - FHE schemes provide efficient instantiations of post-quantum public-key and symmetric-key encryption schemes
  - Homomorphic encryption can be viewed as a generalization of public key encryption
- Key milestones in the history of homomorphic encryption
  - Rivest, Adleman, Dertouzos (1978) -- “On Data Banks and Privacy Homomorphisms”
  - Gentry (2009) -- “A Fully Homomorphic Encryption Scheme”
  - Multiple HE schemes developed after 2009

Yuriy Polyakov ypolyakov@dualitytech.com

EXAMPLE OF FHE WORKFLOW



Yuriy Polyakov ypolyakov@dualitytech.com

FHE vs OTHER SECURE COMPUTING APPROACHES

	FHE	MPC	Secure Enclaves/SGX
Performance	Compute-bound	Network-bound	Close to plaintext
Privacy	Encryption	Encryption / Non-collusion	Trusted Hardware
Non-interactive	✓	✗	✓
Cryptographic security	✓	✓	✗ (known attacks)

Hybrid approaches are also possible, e.g., MPC + FHE

Yuriy Polyakov ypolyakov@dualitytech.com

TYPICAL FHE OPERATIONS

- Encrypt bits and perform logical AND, OR, XOR operations on the ciphertexts.
  - 0 AND 1 → 0, 0 OR 1 → 1, 1 XOR 1 → 0
- Encrypt small integers and perform addition and multiplication, as long as the result does not exceed some fixed bound, for instance, if the bound is 10000
  - 123 + 456 → 579, 12 * 432 → 5184, 35 * 537 → overflow
- Encrypt 8-bit unsigned integers (between 0 and 255) and perform addition and multiplication modulo 256
  - 128 + 128 → 0, 2 * 129 → 2
- Encrypt fixed-point numbers and perform addition and multiplication with the result rounded to a fixed precision, for instance, two digits after the decimal point
  - 12.42 + 1.34 → 13.76, 2.23 + 5.19 → 11.57
- Different homomorphic encryption schemes support different plaintext types and different operations on them.

Yuriy Polyakov ypolyakov@dualitytech.com

SOME EXAMPLES OF REAL-SCALE FHE APPLICATIONS

- Private information retrieval
  - https://eprint.iacr.org/2017/1142, IEEE S&P 2018
- Private set intersection
  - https://eprint.iacr.org/2017/299, ACM CCS 2017
  - https://eprint.iacr.org/2018/787, ACM CCS 2018
  - https://eprint.iacr.org/2021/1116, ACM CCS 2021
- Genome-wide association studies based on chi-square test and logistic regression training
  - https://eprint.iacr.org/2020/563, PNAS 2020
- Logistic regression training
  - https://eprint.iacr.org/2018/662, AAAI Conference on AI 2019
- Neural network inference (ResNet-20 to ResNet-110)
  - https://eprint.iacr.org/2021/1688, ICML 2022

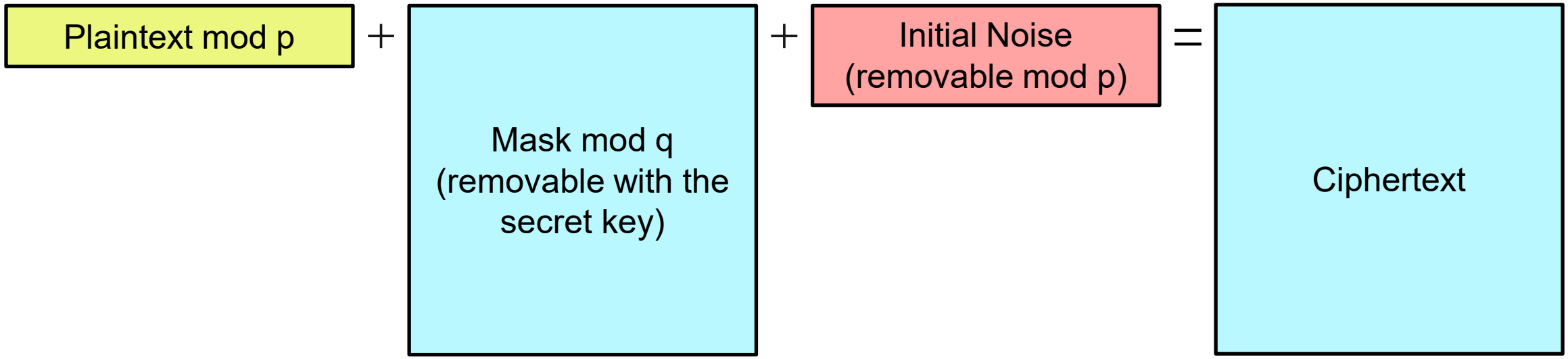
Yuriy Polyakov ypolyakov@dualitytech.com

MAIN CONCEPTS

- *Homomorphic*: a (secret) mapping from plaintext space to ciphertext space that preserves arithmetic operations.
- *Mathematical Hardness: (Ring) Learning with Errors Assumption*
  - Every image (ciphertext) of this mapping looks uniformly random in range (ciphertext space).
- *Security level*: the hardness of inverting this mapping without the secret key
  - Often estimated as a work factor.
    - Example: 128 bits → 2^{128} operations to break using best known lattice attack
- *Plaintext*: Elements and operations of a polynomial ring (mod $x^n + 1$, mod p).
  - Example: $3x^5 + x^4 + 2x^3 + ...$
  - For all practical purposes, you can think of it as a vector of (small) finite integers
- *Ciphertext*: elements and operations of a polynomial ring (mod $x^n + 1$, mod q).
  - Example: $7862x^5 + 5652x^4 + ...$
  - For all practical purposes, you can think of it as a vector of (larger) finite integers
- *Noise*: random integers with Gaussian distribution, which are “added” to the plaintext to achieve the desired security level based on Ring Learning With Errors

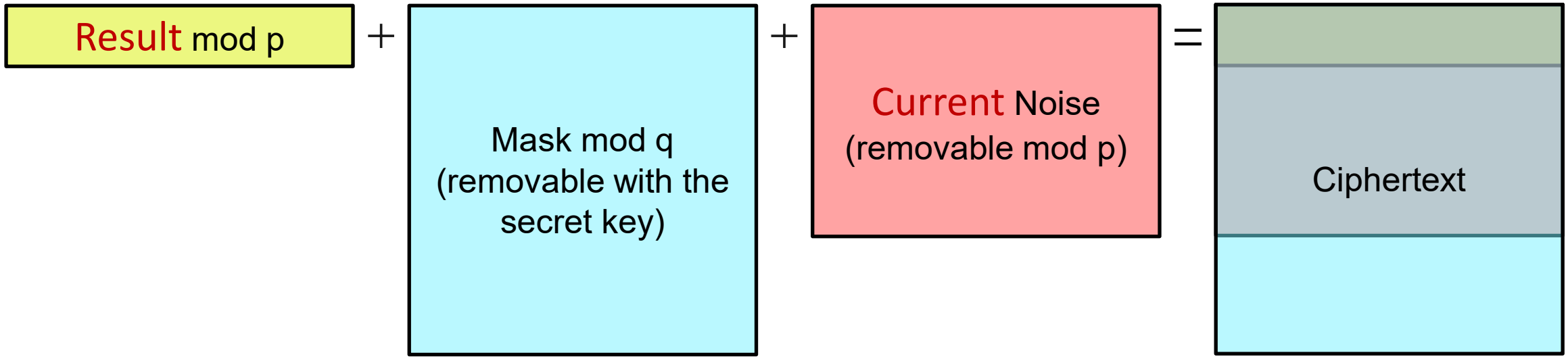
Yuriy Polyakov ypolyakov@dualitytech.com

FRESH ENCRYPTION



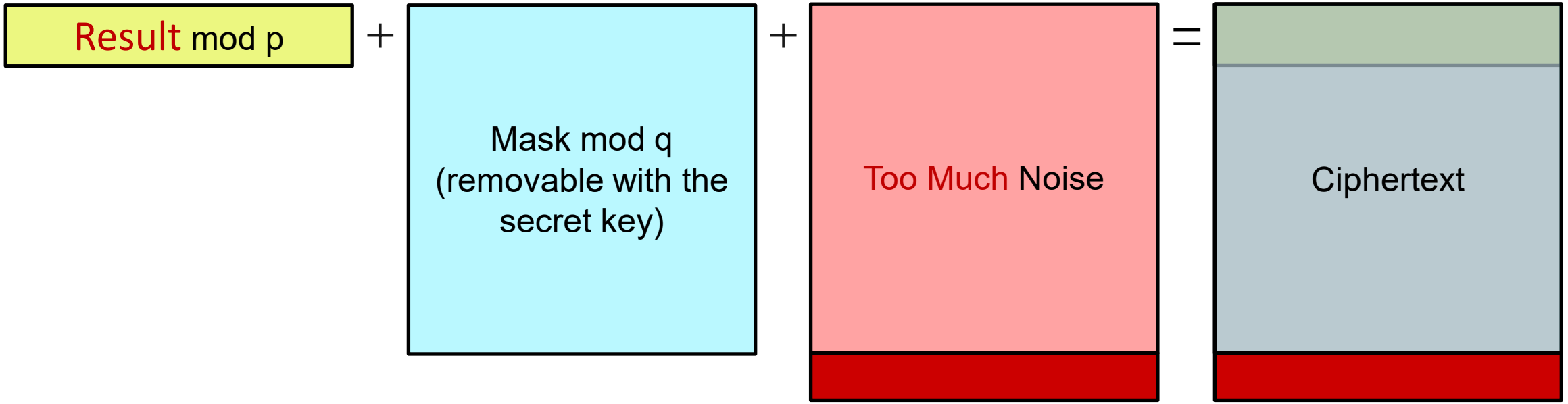
- Horizontal: each coefficient in a polynomial or in a vector.
- Vertical: size of coefficients.
- Initial noise is small in terms of coefficients' size.

AFTER SOME COMPUTATIONS



- Horizontal: each coefficient in a polynomial or in a vector.
- Vertical: size of coefficients.
- Initial noise is small in terms of coefficients' size.

NOISE OVERFLOW (RESULTS IN DECRYPTION FALURE)

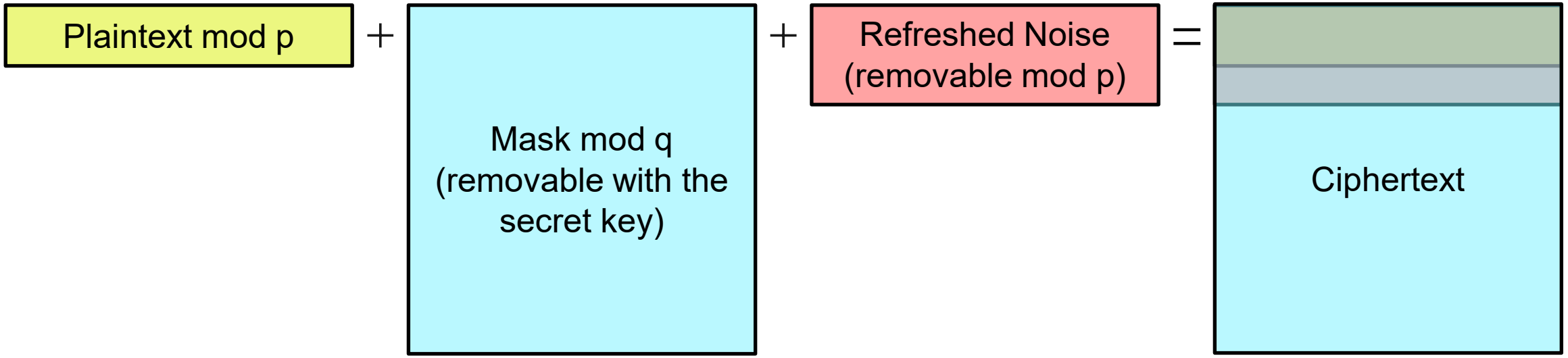


- Horizontal: each coefficient in a polynomial or in a vector.
- Vertical: size of coefficients.
- Initial noise is small in terms of coefficients' size.

Yuriy Polyakov ypolyakov@dualitytech.com

BOOTSTRAPPING (NOISE REFRESHING PROCEDURE)

Evaluates the decryption circuit homomorphically and resets the noise.



- Horizontal: each coefficient in a polynomial or in a vector.
- Vertical: size of coefficients.
- Initial noise is small in terms of coefficients' size.

TYPES OF HOMOMORPHIC ENCRYPTION

- Partially homomorphic encryption (weakest notion)
  - supports only one type of operation, e.g. addition or multiplication.
- Somewhat homomorphic encryption schemes
  - can evaluate two types of gates/operations, but only for a subset of circuits.
- **Leveled fully homomorphic encryption**
  - supports more than one operation but only computations of a predetermined size (typically multiplicative depth); supports much deeper circuits than somewhat homomorphic encryption
- **Fully homomorphic encryption**
  - supports arbitrary computation on encrypted data; it is the strongest notion of homomorphic encryption.

Yuriy Polyakov ypolyakov@dualitytech.com

CLASSES OF HOMOMORPHIC SCHEMES

1. Modular (Exact) Integer Arithmetic: BGV / BFV
   - Plaintext data represented as **vectors modulo a plaintext modulus "*t*"** (or their vectors)
   - Computations expressed as **vectors arithmetic mod *t***

2. Functional (Programmable) Bootstrapping: DM (FHEW) / CGGI (TFHE)
   - Plaintext represented as **integers/Boolean values**
   - Supports evaluation of arbitrary functions using Look-Up Tables (LUTs)
   - Computation for each integer is evaluated separately

3. Approximate Number Arithmetic: CKKS
   - Plaintext data represented as **vectors of real numbers** (or complex numbers)
   - Compute model similar to **floating-point arithmetic** but dealing with fixed-point numbers

MODULAR (EXACT) INTEGER ARITHMETIC APPROACH

- Features:
  - Efficient SIMD computations over vectors of integers (using batching, also called CRT packing)
  - Fast high-precision integer arithmetic
  - Fast private information retrieval/private set intersection/secure database query
  - Leveled design (often used without bootstrapping)
- Main schemes:
  - Brakerski-Vaikuntanathan (BV) [BV11] - foundation for other schemes
  - Brakerski-Gentry-Vaikuntanathan (BGV) [BGV12]
  - Brakerski/Fan-Vercauteren (BFV) [Bra12, FV12]

FUNCTIONAL (PROGRAMMABLE) BOOTSTRAPPING APPROACH

- Features:
  - Fast number comparison
  - Initially proposed for Boolean circuits
  - Supports arbitrary functions by evaluating LUTs
  - Fast bootstrapping (noise refreshing procedure)
  - Does not support batching/CRT packing
- Related schemes:
  - Gentry-Sahai-Waters (GSW) [GSW13] – used in bootstrapping
  - Fastest Homomorphic Encryption in the West (DM/FHEW) [DM15]
  - Fast Fully Homomorphic Encryption over the Torus (CGGI/TFHE) [CGGI16,CGGI17]
  - Efficient FHEW Bootstrapping with Small Evaluation Keys [LMCKDEY23]

APPROXIMATE NUMBER ARITHMETIC APPROACH

- Features:
  - Efficient SIMD computations over vectors of real numbers (using batching)
  - Fast polynomial approximation
  - Relatively fast multiplicative inverse and Discrete Fourier Transform
  - Deep approximate computations, such as logistic regression learning
  - Leveled design, but also used with approximate bootstrapping in many ML applications
  - Best amortized bootstrapping time

- Selected schemes:
  - Cheon-Kim-Kim-Song (CKKS) [CKKS17]

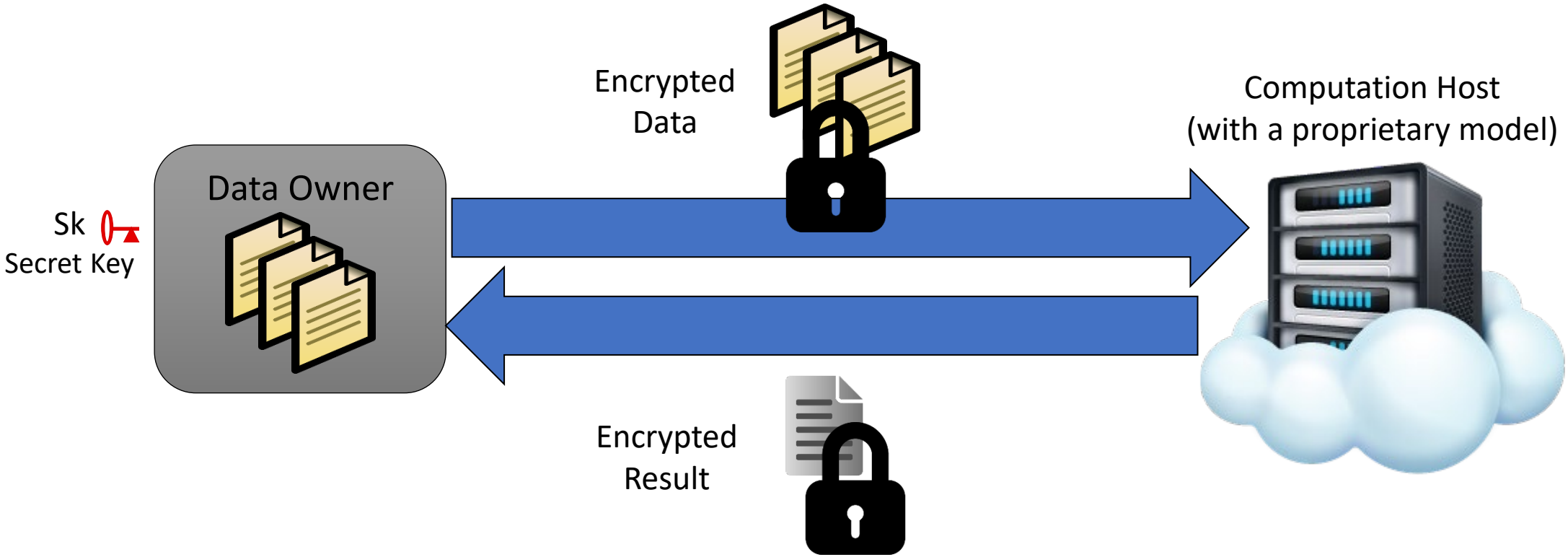
SELECTING SECURITY PARAMETERS

The ciphertext dimension (degree of polynomial) should be chosen according to the security tables published at HomomorphicEncryption.org

distribution	n	security level	logq	uSVP	dec	dual
(-1, 1)	1024	128	27	131.6	160.2	138.7
		192	19	193.0	259.5	207.7
		256	14	265.6	406.4	293.8
	2048	128	54	129.7	144.4	134.2
		192	37	197.5	233.0	207.8
		256	29	259.1	321.7	273.5
	4096	128	109	128.1	134.9	129.9
		192	75	194.7	212.2	198.5
		256	58	260.4	292.6	270.1
	8192	128	218	128.5	131.5	129.2
		192	152	192.2	200.4	194.6
		256	118	256.7	273.0	260.6

Multiparty FHE

SINGLE-KEY FHE WORKFLOW

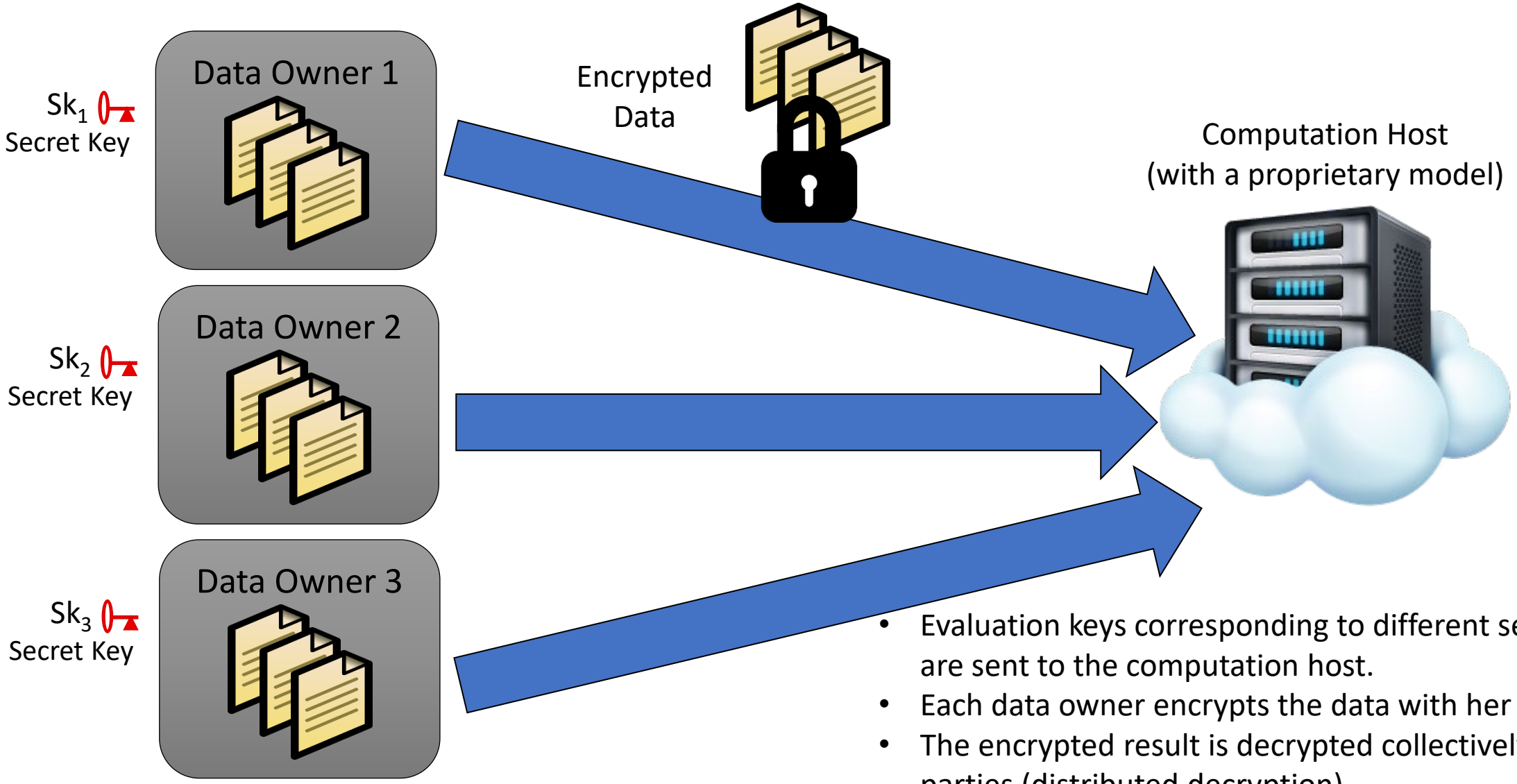


How can this model be extended to multiple data owners that do not want to share a secret key or data?

What if the model needs to be encrypted by model provider and sent to the computation host? What key should the model provider use for encryption?

Yuriy Polyakov ypolyakov@dualitytech.com

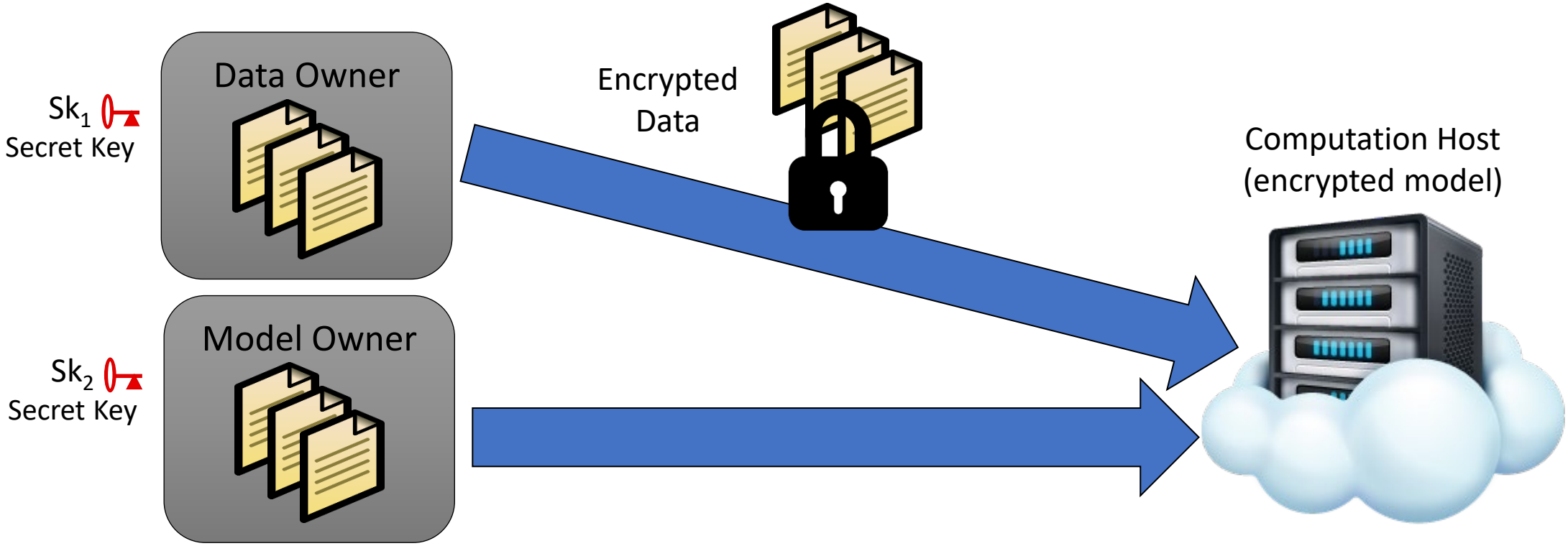
SOLUTION 1: MULTIKEY FHE (MULTIPLE DATA OWNERS)



- Evaluation keys corresponding to different secret keys are sent to the computation host.
- Each data owner encrypts the data with her own key.
- The encrypted result is decrypted collectively by all parties (distributed decryption).

Yuriy Polyakov ypolyakov@dualitytech.com

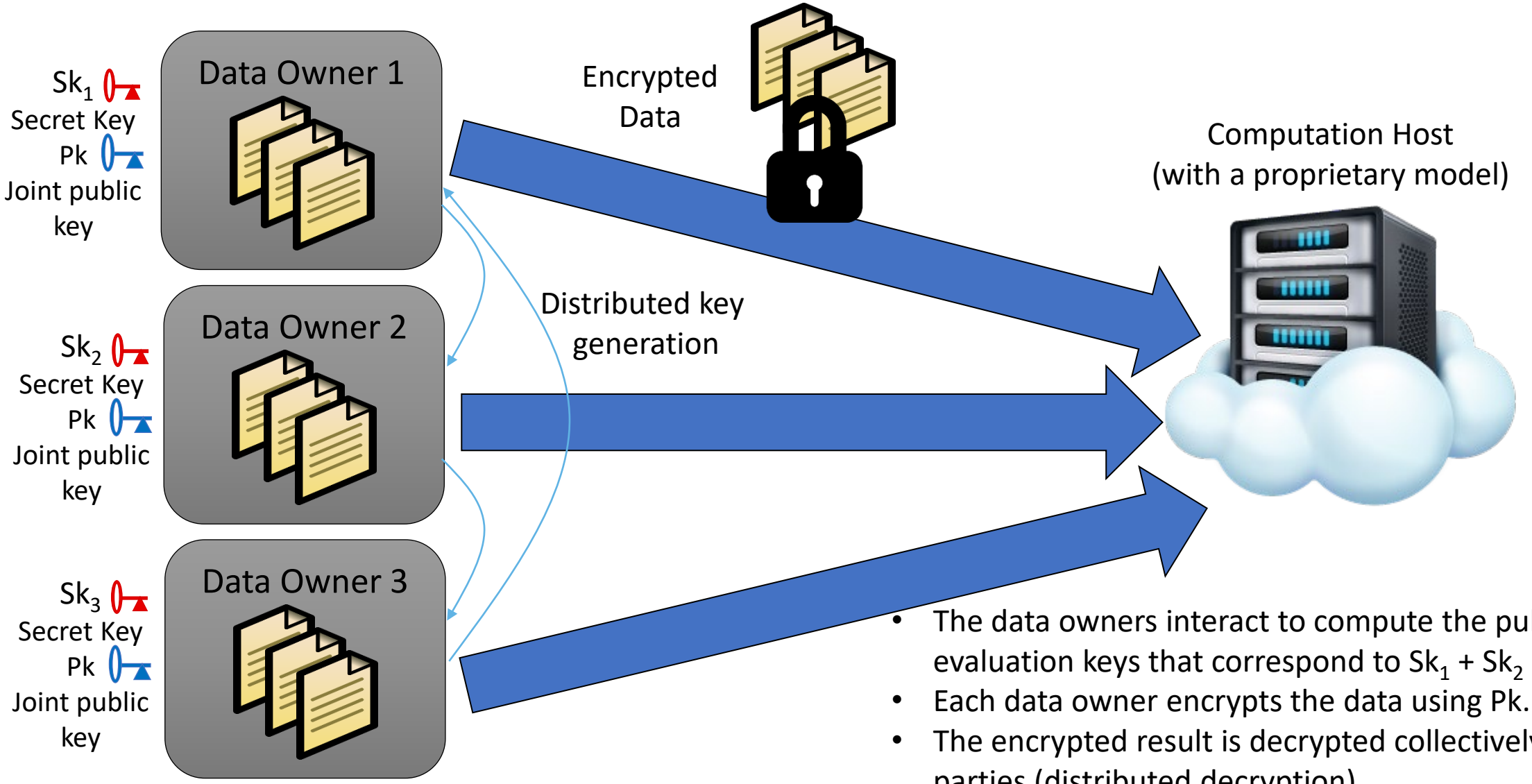
SOLUTION 1: MULTIKEY FHE (ENCRYPTED MODEL)



- Evaluation keys corresponding to different secret keys are sent to the computation host.
- Data and model owners encrypt the data with their own keys.
- The encrypted result is decrypted collectively by all parties (distributed decryption).

Yuriy Polyakov ypolyakov@dualitytech.com

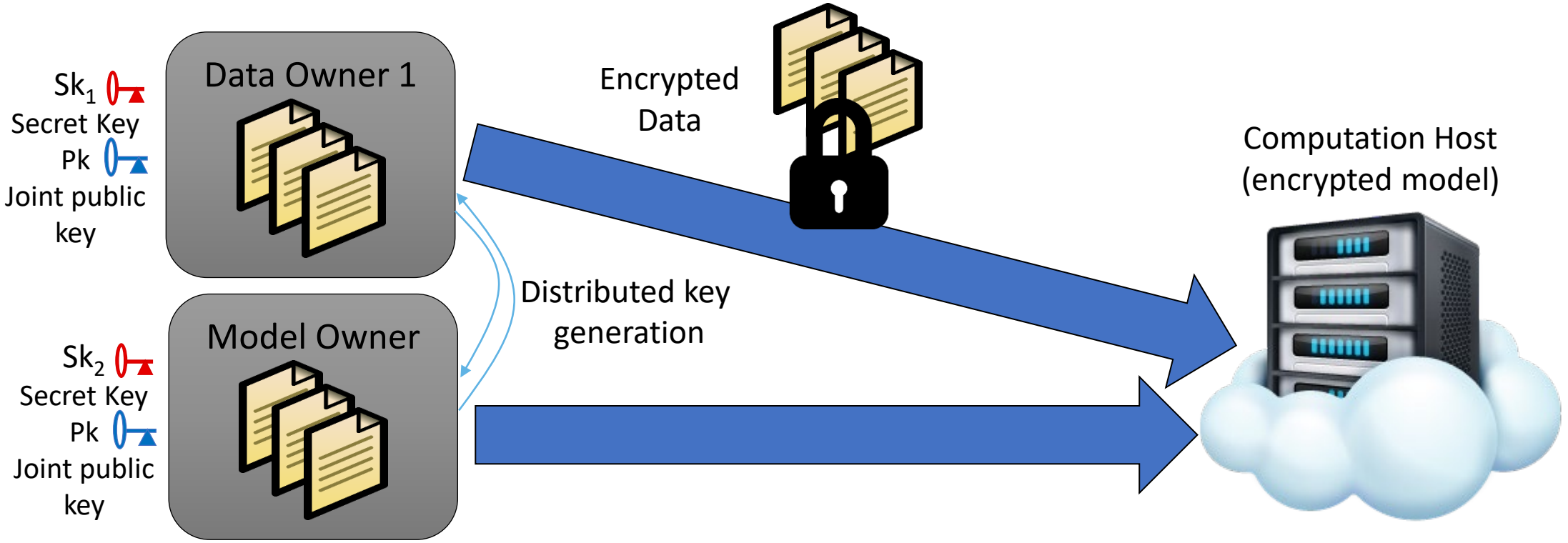
SOLUTION 2: THRESHOLD FHE (MULTIPLE DATA OWNERS)



- The data owners interact to compute the public and evaluation keys that correspond to $Sk_1 + Sk_2 + Sk_3$.
- Each data owner encrypts the data using Pk.
- The encrypted result is decrypted collectively by all parties (distributed decryption).

Yuriy Polyakov ypolyakov@dualitytech.com

SOLUTION 2: THRESHOLD FHE (ENCRYPTED MODEL)



- The data and model owner interact to compute the public and evaluation keys that correspond to $Sk_1 + Sk_2$.
- Data owner encrypts the data and model owner encrypts the model using Pk.
- The encrypted result is decrypted collectively by both parties (distributed decryption).

Yuriy Polyakov ypolyakov@dualitytech.com

COMPARISON OF MULTIKEY AND THRESHOLD HE

Parameter	Multikey HE	Threshold HE
Key generation	Non-interactive (asynchronous)	Interactive (synchronous)
Number of parties	Supports a variable number of parties, bounding only the number of parties involved in a specific computation	The number of parties is fixed
Decryption	Interactive (all parties compute partial decryptions and merge them)	Interactive (all parties compute partial decryptions and merge them)
Computation runtime	Grows quadratically (asymptotically; slightly better in practice) with the number of parties [CDKS19]	Roughly the same as in single-key HE
Evaluation and ciphertext size	Linear in the number of parties [CDKS19]	Roughly the same as in single-key HE

[CDKS19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference. CCS'19.

Yuriy Polyakov ypolyakov@dualitytech.com

RECENT RESEARCH RESULTS CLOSING THE GAP BETWEEN MULTIKEY AND THRESHOLD HE

- Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. A Unified Framework of Homomorphic Encryption for Multiple Parties with Non-Interactive Setup, https://eprint.iacr.org/2021/1412.
- Taechan Kim, Hyesun Kawk, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically Faster Multi-Key Homomorphic Encryption from Homomorphic Gadget Decomposition, https://eprint.iacr.org/2022/347

FHE Approaches for ML

THREE MOST COMMON WAYS TO DO ENCRYPTED ML USING FHE

Here we focus on (supervised) learning using the models based on logistic regression, decision trees, and neural networks, i.e., relatively deep computations requiring bootstrapping.

1. Approximate approach based on CKKS
   - Fastest for most ML applications, especially with larger problem sizes.
   - Polynomial approximations should be used with care.
2. Hybrid approximate/LUT approach based on CKKS and DM (FHEW) /CGGI (TFHE)
   - Significantly slower than approach 1 for most ML applications.
   - Evaluates “tricky” non-linear functions, such as comparison, using functional bootstrapping in DM/CGGI.
3. LUT approach based on functional bootstrapping in DM/CGGI
   - Slowest option (typically by orders of magnitude compared to option 1) and does not scale well with the problem size.
   - Easiest to use in most cases.

Yuriy Polyakov ypolyakov@dualitytech.com

APPROXIMATE APPROACH BASED ON CKKS

- Complicated functions are approximated using polynomials
  - Typically Chebyshev interpolations or similar minimax methods are used
  - Evaluates functions over vectors of real numbers, e.g., a function is evaluated for 32K real numbers at once
  - When building the polynomial interpolation, the input range has to be specified
- Approximate CKKS bootstrapping is used to support deep computations, such as logistic regression training
  - Approximate bootstrapping significantly increases the approximation error after first bootstrapping, but further bootstrapping operations typically have a small effect for many ML applications (the ML computations are often approximate in nature)
  - Multiprecison CKKS (META-BTS) bootstrapping was recently proposed in https://eprint.iacr.org/2022/1167 (CCS'22)
- Amortized cost of CKKS bootstrapping gets as low as 1ms per real number [BMT+21]
- Inference is fast for many models, e.g., logreg inference for hundreds of samples and dozens of features takes less than 1 second

HYBRID APPROXIMATE/LUT APPROACH BASED ON CKKS+DM/CGGI

- CKKS is used for all polynomial/matrix arithmetic computations
- For “tricky” non-linear functions, LUT evaluation using functional bootstrapping with DM/CGGI is used
  - Useful for comparisons
  - Can be used for piecewise polynomial evaluation, addressing the input range issue in polynomial approximations
- Requires scheme switching from CKKS to/from DM/CGGI
- LUT evaluation is often the main bottleneck

Yuriy Polyakov ypolyakov@dualitytech.com

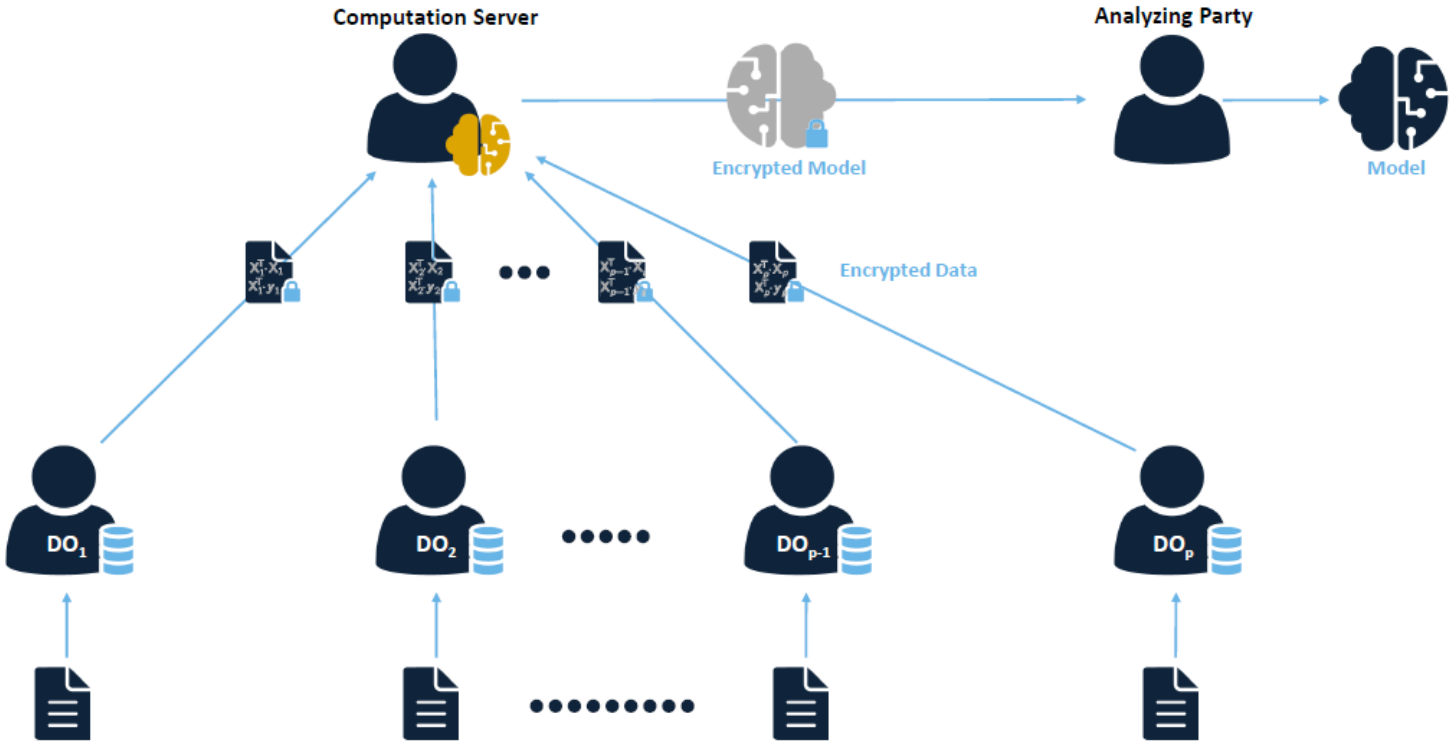
LUT APPROACH BASED ON FUNCTIONAL BOOTSTRAPPING IN DM/CGGI

- Any deep learning algorithm can be evaluated using LUTs for non-linear functions
- The main drawback is performance
  - Functional bootstrapping does not support the evaluation of a LUT over a vector of integers in a SIMD manner
  - LUT evaluation even for a small plaintext modulus (few bits of precision) requires 100ms or so
  - To evaluate an arbitrary function for a larger plaintext space, many small LUT evaluations are needed
  - Typically orders of magnitude slower than the approximate approach based on CKKS
  - Even inference takes substantial time, e.g., logreg inference for hundreds of samples and dozens of features takes more than 10 minutes

Yuriy Polyakov ypolyakov@dualitytech.com

Multiparty FHE for ML

COLLABORATIVE TRAINING WORKFLOW: MULTIPARTY FHE

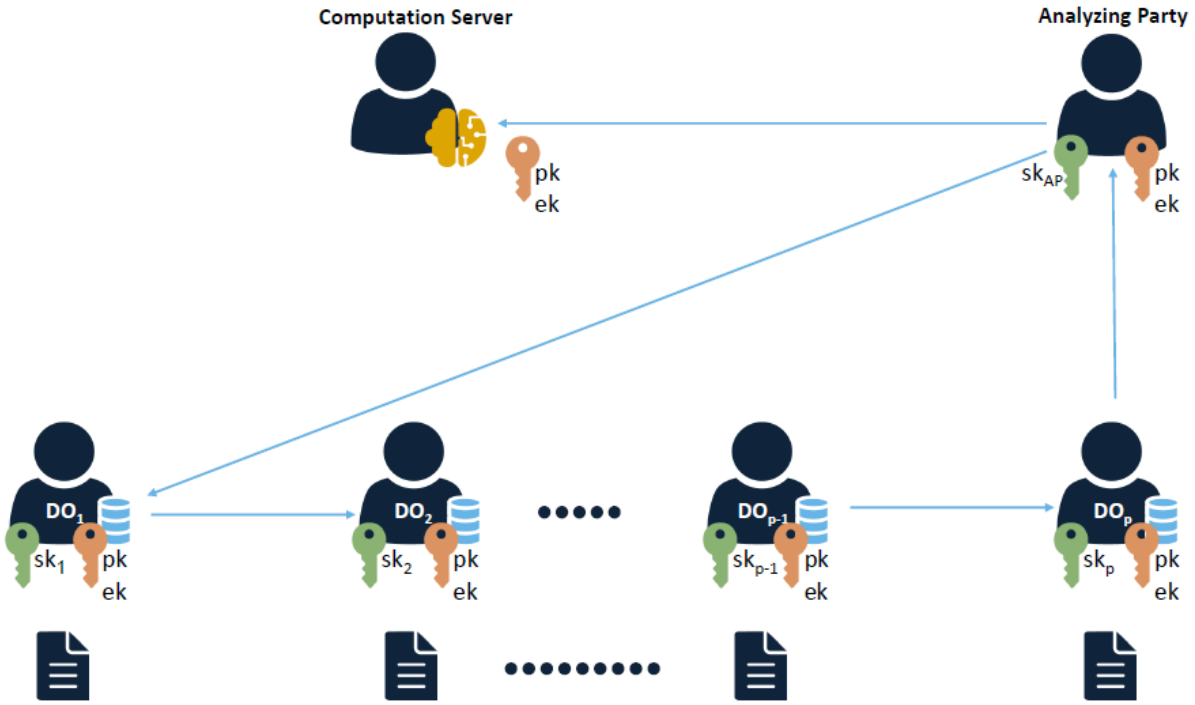


- Compute the model that corresponds to the whole dataset
- Protect the data of each party using a privacy enhancing technology (Fully Homomorphic Encryption)

Yuriy Polyakov ypolyakov@dualitytech.com

MULTIPARTY FHE: DISTRIBUTED KEY GENERATION

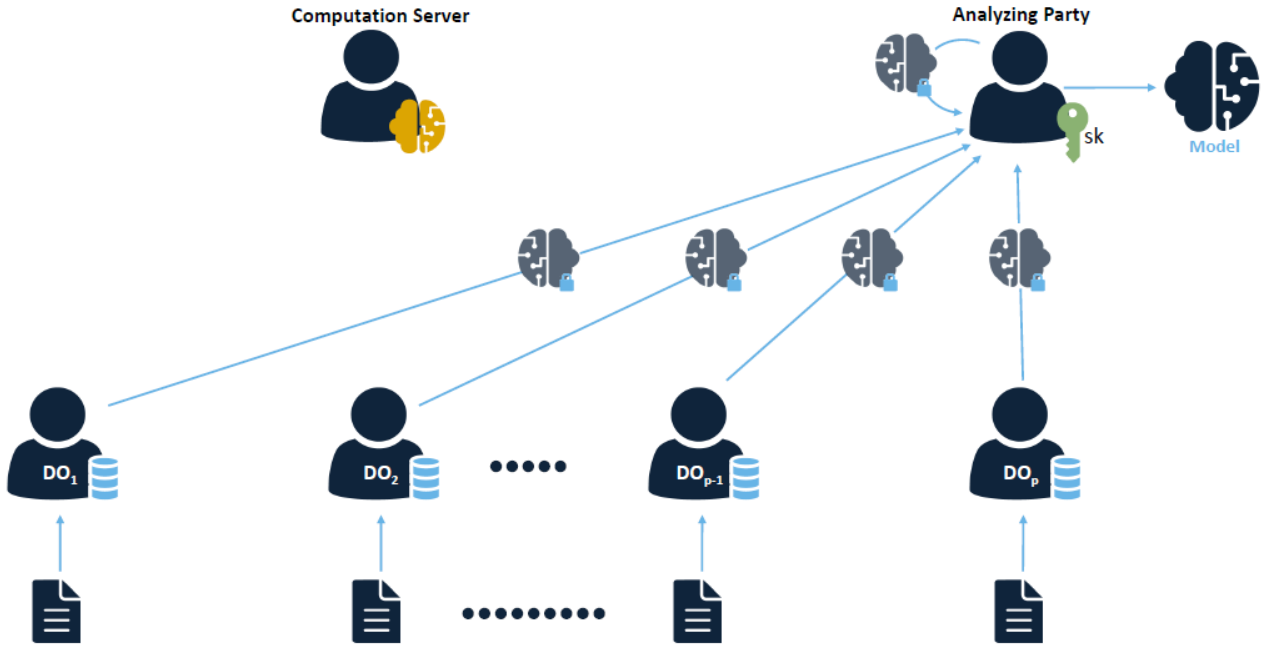
- Each party generates a secret share
- Public keys are computed for a joint secret key, which is a sum of secret shares
- The public keys are computed homomorphically



Yuriy Polyakov ypolyakov@dualitytech.com

MULTIPARTY FHE: DISTRIBUTED DECRYPTION

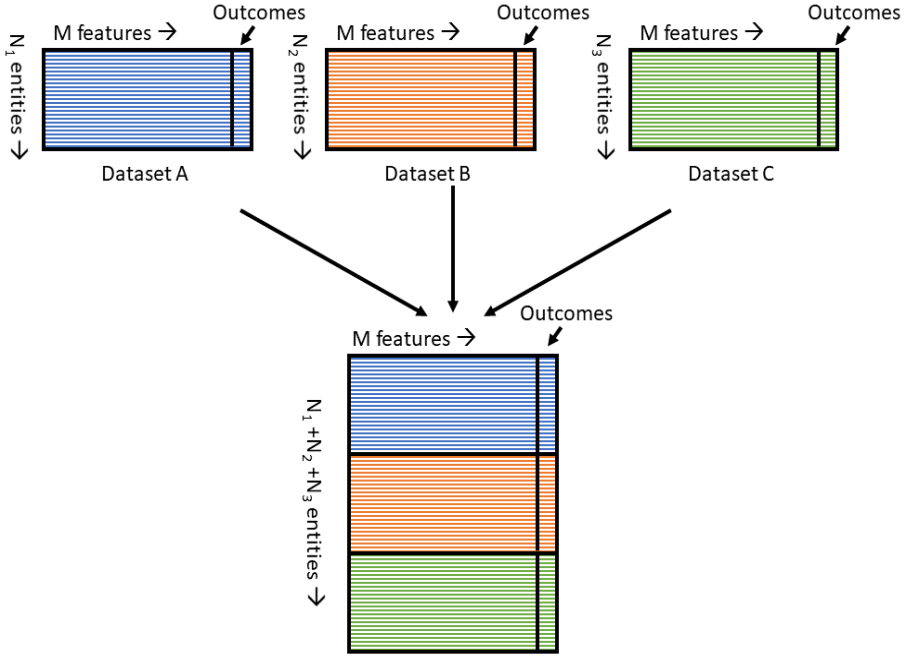
- Each party computes a partial decryption using their own secret share
- The partial decryptions are combined to obtain the decrypted model



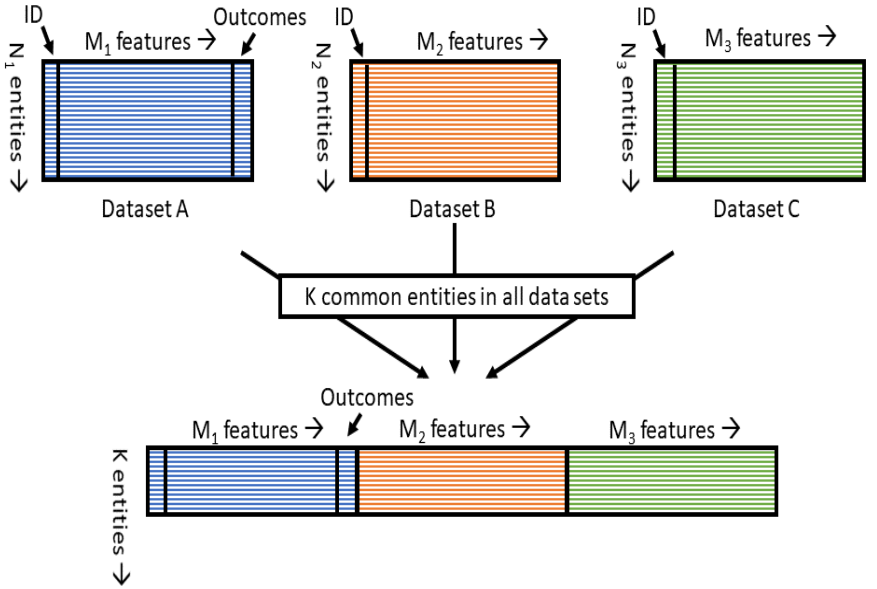
Yuriy Polyakov ypolyakov@dualitytech.com

LINKING ENCRYPTED DATA FROM MULTIPLE DATA OWNERS

Stacking/union



Private Join and Compute



Yuriy Polyakov ypolyakov@dualitytech.com

Selected studies in PPML using FHE

RECENT DARPA PROGRAMS

- Cooperative Secure Learning (CSL) [August 2020 – January 2022]
  - Develop methods to protect data, models, and model outputs among a community of entities desiring to securely share their information to better inform ML model development
  - Enable multiple parties to cooperate for the purpose of improving each other's ML models while assuring that each entity's individual, pre-existing datasets and models will remain private
- Data PRotection in Virtual Environments (DPRIVE) [January 2021 – present]
  - Develop a hardware accelerator for FHE computations that will dramatically reduce the compute runtime overhead compared to software-based FHE approaches
  - Motivating applications are logistic regression training, CNN inference, and CNN training

SELECTED PAPERS FOR APPROXIMATE METHOD BASED ON CKKS

- Logistic regression training
  - https://eprint.iacr.org/2018/662, AAAI Conference on AI 2019
  - 422108 samples over 200 features in 17 hours on a single machine
- Genome-wide association studies based on chi-square test and logistic regression training
  - https://eprint.iacr.org/2020/563, PNAS 2020
  - 500000 SNPs and 100000 individuals in 5.6 hours on a single machine
- ResNet-20 deep neural network evaluation
  - https://eprint.iacr.org/2021/783; CIFAR-10 in 3 hours on a single server with 64 threads
  - https://eprint.iacr.org/2021/1688; CIFAR-10 in 40 minutes on a single server with 1 thread; Resnet-110 for the same setup took about 3.7 hrs

SELECTED PAPERS FOR HYBRID METHOD

- Semi-parallel logistic regression training (GWAS)
  - https://eprint.iacr.org/2019/101, BMC Medical Genomics 2020
  - 10643 SNPs, 245 patients, and 3 covariates: 4 min. to 3 hrs on a single machine
  - The best approaches based on the CKKS method took few minutes for the problem sizes that required few hours for the hybrid approach
- Decision tree evaluation and K-means clustering
  - https://eprint.iacr.org/2020/1606, IEEE S&P 2021
  - Decision tree evaluation: for 60 internal nodes, 57 features, and 2 classification labels the runtime was about 7 seconds
  - K-means clustering: for 4096 data points and 8 clusters, the runtime was 52 minutes

Yuriy Polyakov ypolyakov@dualitytech.com

References

[BGV14] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):1–36, 2014.

[Bra12] Z. Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In CRYPTO 2012. Pages 868 – 886.

[BV11] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Annual cryptology conference, pages 505–524. Springer, 2011.

[CGGI16]: I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Asiacrypt 2016 (Best Paper), pages 3-33.

[CGGI17]: I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. ASIACRYPT (1) 2017: 377-408.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic Encryption for Arithmetic of Approximate Numbers. In ASIACRYPT 2017. Pages 409–437.

[DM15]: L. Ducas and D. Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. EUROCRYPT 2015.

[FV12] J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive. Report 2012/144, 2012. http://eprint.iacr.org/2012/144.

[GSW13]: C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. CRYPTO 2013.

[LMKCDEY13]: Y. Lee, A. Kim, D. Micciancio, R. Choi, Deryabin M., J. Eom, D. Yoo. Efficient FHEW Bootstrapping with Small Evaluation Keys, and Applications to Threshold Homomorphic Encryption, EUROCRYPT 2023.

Yuriy Polyakov ypolyakov@dualitytech.com

Duality

Thank You! Have questions?

Ask them at https://openfhe.discourse.group/

Yuriy Polyakov

ypolyakov@dualitytech.com