Squirrel: A Scalable Secure Computation Framework for Training GBDT

Wen-jie Lu, Zhicong Huang, Qizhi Zhang, Yuchen Wang, Cheng Hong

Agenda

- Building Gradient Boosting Decision Tree (GBDT)
- Secure Two-party Computation (2PC) Primitives
- Pipeline of Squirrel Framework
- Optimizations
- Results

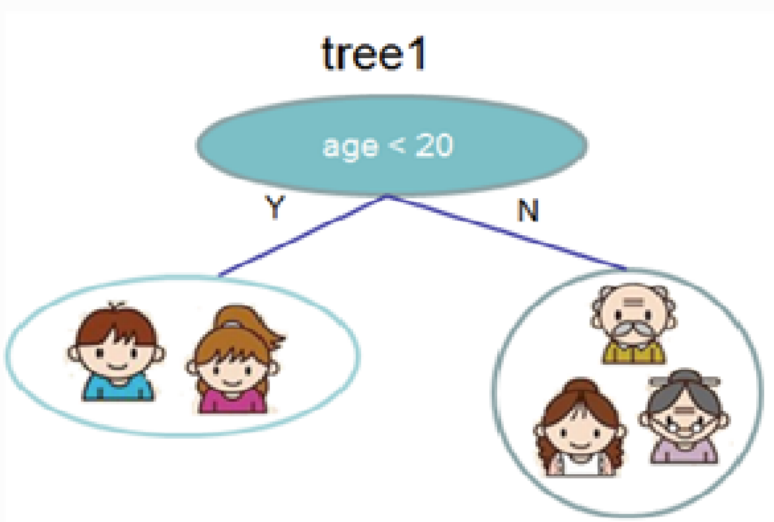
Backgrounds: Decision Tree

- Build branching model F(X) closes to label Y
  - (binary classification for simplicity)

ID	X_1 (Age)	X_2 (Use computer daily)	Y (Like computer game)
	6	Y	1
	12	N	-1
	30	Y	1
	65	N	-1
	70	N	-1

X (Attributes)

Y (Label)



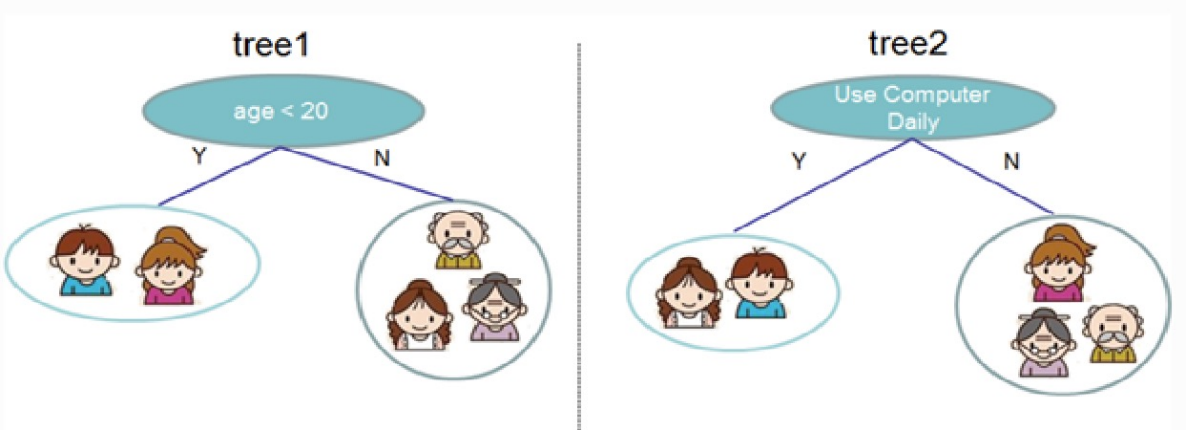
F(X)=
- 1 (X_1<20)
- -1 (X_1≥20)

E.g. a decision tree with depth = 1

cite: https://xgboost.readthedocs.io/en/stable/tutorials/model.html

Background: Gradient Boosting Decision Tree (GBDT)

- Train many decision trees
- Uses the “boosting” technique
  - N^{th} tree tries is built according the gradient info from previous N-1 trees
  - Final prediction = Sum of all the predictions on each tree.



Model: would someone like computer game?

cite: https://xgboost.readthedocs.io/en/stable/tutorials/model.html

GBDT is Good

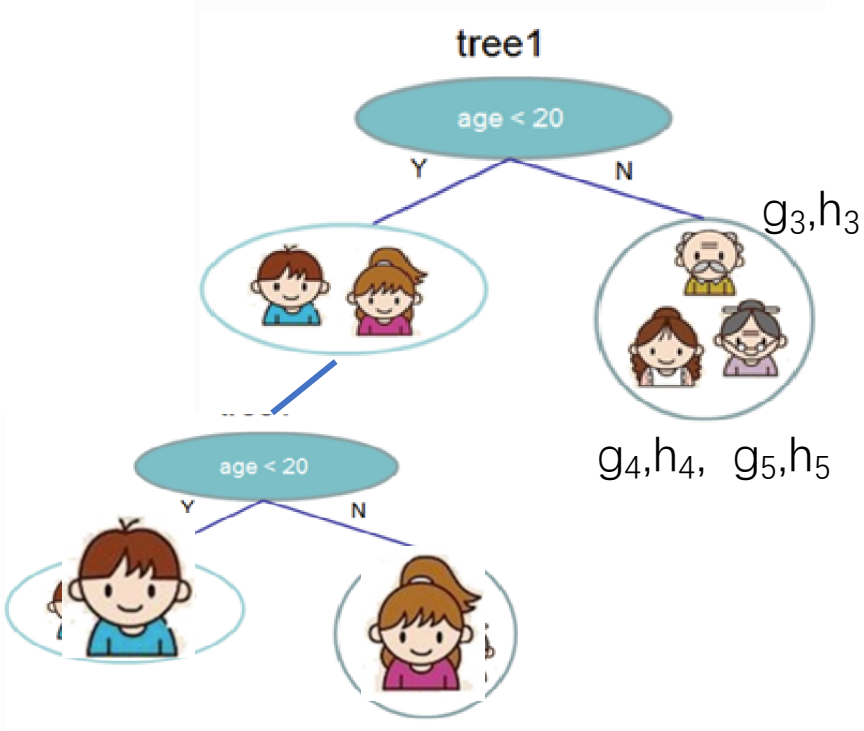
- High accuracy
  - Used by many winners in Kaggle
- Interpretability
  - Customers often ask for explanations
    E.g: Why is my account blocked ?

2 Private GBDT via Secure Computation

GBDT Building

- Compute the 1st and 2nd gradients (g_i, h_i) for each sample on each tree
- Iterate all the candidate split point:
  - Find the optimal split point that maximize "gain"

$$\text{Gain} = \frac{\left(\sum_{\text{Left}} g_i\right)^2}{r+\sum_{\text{Left}} h_i} + \frac{\left(\sum_{\text{Right}} g_i\right)^2}{r+\sum_{\text{Right}} h_i} - \frac{\left(\sum g_i\right)^2}{r+\sum h_i}$$



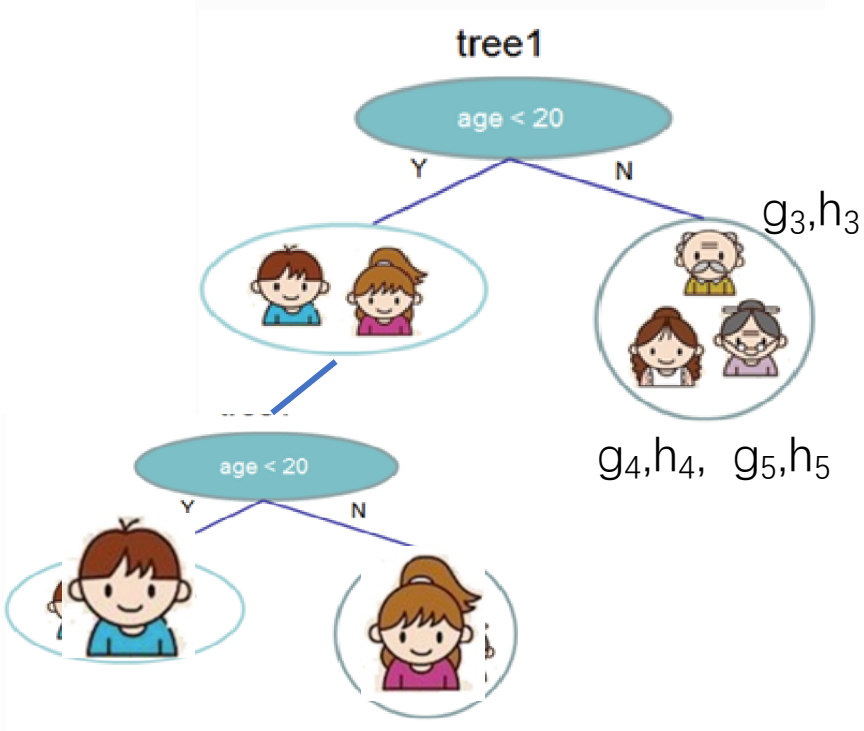
cite: https://xgboost.readthedocs.io/en/stable/tutorials/model.html

GBDT Building

- Compute the 1st and 2nd gradients (g_i, h_i) for each sample on each tree
- Iterate all the candidate split point:
  - Find the optimal split point that maximize "gain"

$$\text{Gain} = \frac{\left(\sum_{\text{Left}} g_i\right)^2}{r+\sum_{\text{Left}} h_i} + \frac{\left(\sum_{\text{Right}} g_i\right)^2}{r+\sum_{\text{Right}} h_i} - \frac{\left(\sum g_i\right)^2}{r+\sum h_i}$$

- Samples are sepearated according to the optimal split



cite: https://xgboost.readthedocs.io/en/stable/tutorials/model.html

Goal: Collaboration on Vertically-Split Dataset

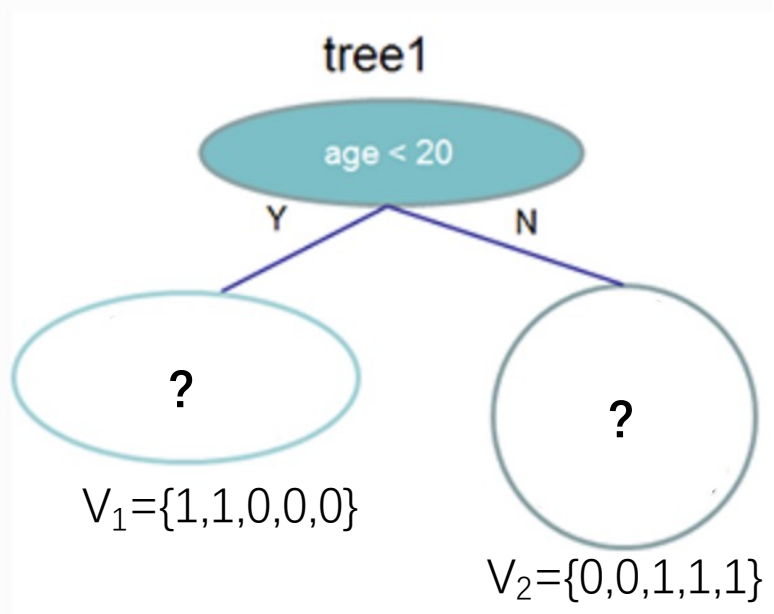
ID	Yearly expense	Yearly income	Auto insurance claim filed per year	Hospital visits per year	Label
Alice	$50,000	$150,000	1	3	y0
Bob	$35,000	$90,000	2	1	y1
Cathy	$45,000	$70,000	3	3	y2
David	$72,000	$300,000	2	4	y3

Aligned ID (eg. via PSI) | Bank A knows this | Insurance company B knows this | Distributed in A/B

- Suppose someone wants to apply a loan from a bank A
  - A could make better decisions under B's help

Privacy Requirements

- Semi-honest two party computation
  - Nothing beyond the final model should be revealed
- Part of final model: split point "Age <20”
  - Reveal to the attribute holder only
- NOT part of final model: sample distribution (SD)
  - In practice, a secret indicator vector V_i is used to keep sample distribution private



The party without the “age” column does not know V_1 or V_2

Related Work

- Federated learning based
  - Secureboost and VFboost
  - Leaks sample distribution ☹
  - Computes gain in clear ☹
- Secure Two-party Computation based
  - [FZT21] and [YSX20]
  - Private indicator vector 😊
  - Computes gain in MPC 😊
  - Heavy computation & communication ☹

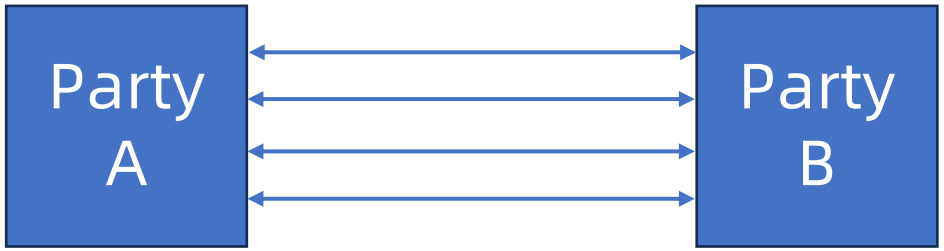
[1] Cheng K, Fan T, Jin Y, et al. Secureboost: A lossless federated learning framework[J]. IEEE Intelligent Systems, 2021
[2] Fu F, Shao Y, Yu L, et al. Vf2boost: Very fast vertical federated gradient boosting for cross-enterprise learning, SIGMOD 2021
[3] Fang W, Zhao D, Tan J, et al. Large-scale secure XGB for vertical federated learning, CIKM 2021
[4] Wu Y, Cai S, Xiao X, et al. Privacy Preserving Vertical Federated Learning for Tree-based Models, VLDB 2020

Challenges

- Privacy-preserving GBDT training using 2PC is promising
  - Why not federated learning ? => Want “minimal” information leak
- Efficient 2PC GBDT on millions of samples
  - Faster computation
  - Lighter communication, e.g., 200 Mbps

03 2PC Tools

Secure Two-party Computation



- Intermediate information are all secretly shared or encrypted
- Decrypt (reveal) the final result only

SS & OT & HE

- Arithmetic Secret Sharing (SS)
  - Info. is randomly splitted $x_0 + x_1 = x \% p$ e.g., p=2^64
  - Uses Oblivious Transfer (OT) as building block
  - Protocols for many (nonlinear) functions eg., Div, ArgMax and Sqrt
- Linear Homomorphic Encryption (HE)
  - Enc(x; pk) + Enc(y; pk) => C if Dec(C; sk) can give then sum x + y
  - Eg., Paillier or BFV HE scheme
  - However, inefficient for evaluating nonlinear functions

Mixed Primitives Evaluation

- Use secret sharing for non-linear functions e.g., private division
- Switch to LHE for a lower communication
  - A2H: Enc(x0; pk0) ➔ Enc(x0; pk0) + x1 = Enc(x; pk0)
  - H2A: Dec(C; sk0) = m – r ➔ Enc(m; pk0) + r

04 Squirrel

Data form: Plaintext & Share & HE

ID	Age
	8
	12
	34
	67
	70

Plaintext
Hold privately

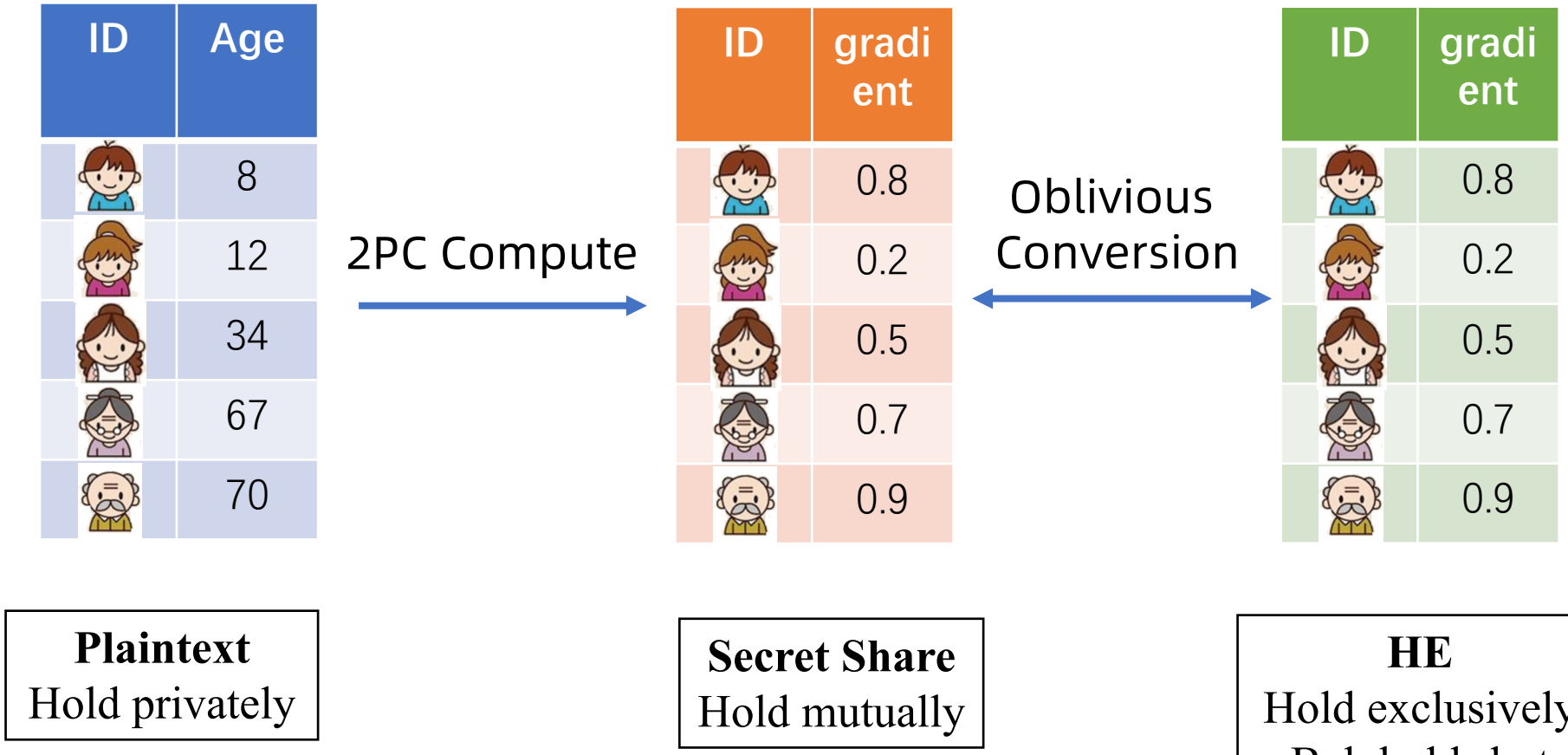
ID	gradient
	0.8
	0.2
	0.5
	0.7
	0.9

Secret Share
Hold mutually

ID	gradient
	0.8
	0.2
	0.5
	0.7
	0.9

HE
Hold exclusively
Bob holds but encrypted under Alice's key

Data form: Plaintext & Share & HE



Pipeline of Squirrel GBDT Building

- One-shot Step 1/3: Private Set Intersection

ID	Age
	8
	12
	34
	67
	70

Held by Alice

ID	Height
	5ft
	5.2ft
	6ft
	5.4ft
	6.2ft

Held by Bob

Pipeline of Squirrel GBDT Building

- One-shot Step 2/3: Data Encoding (ie. Binning)

ID	Age
	8
	12
	34
	67
	70

ID	Age ≤20	20<Age ≤60	Age >60
	1	0	0
	1	0	0
	0	1	0
	0	0	1
	0	0	1

Pipeline of Squirrel GBDT Building

- One-shot Step 2/3: Local private of (sparse) binary matrix

ID	Age ≤20	20<Age ≤60	Age >60
	1	0	0
	1	0	0
	0	1	0
	0	0	1
	0	0	1

Held by Alice

ID	Height ≤ 4ft	4ft<Height ≤6ft	Height> 6ft
	1	0	0
	0	1	0
	0	0	1
	0	1	0
	0	0	1

Held by Bob

Pipeline of Squirrel GBDT Building

- One-shot Step 3/3: Initialize the 1st prediction vector

ID	pred
	0.5
	0.5
	0.5
	0.5
	0.5

Pipeline of Squirrel GBDT Building

- Iterative Step 1/5: 2PC Compute 1st and 2nd gradient

ID	pred
	0.5
	0.5
	0.5
	0.5
	0.5

2PC Compute →

ID	1st grad	2nd grad
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

For the i^{th} sample in the t^{th} tree:

$$g_i = P_i - y_i$$
$$h_i = P_i(1 - P_i)$$

// P_i denote the prediction on i^{th} sample

$$P_i = \text{sigmoid}\left(\sum_{j=1}^{t-1} j\text{Tree\_output}(i)\right)$$

Pipeline of Squirrel GBDT Building

- Iterative Step 2/5: Gradient aggregation using HE

ID	1st grad	2nd grad
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

Secret Share

ID	g	h
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

HE

Pipeline of Squirrel GBDT Building

- Iterative Step 2/5: Gradient aggregation using HE

ID	1st grad	2nd grad
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

ID	g	h
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

HE

ID	Age ≤20	20<Age ≤60	Age >60
	1	0	0
	1	0	0
	0	1	0
	0	0	1
	0	0	1

Bob's Local Plaintext

	Age ≤20	20<Age ≤60	Age >60
G	0.3	0.7	1
H	0.8	0.2	0.8

Aggregated gradient table (Encrypted under Alice's key)

Pipeline of Squirrel GBDT Building

- Iterative Step 2/5: Gradient aggregation usi

ID	1st grad	2nd grad
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

ID	g	h
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

ID	Age ≤20	20<Ag ≤60	
	1	0	
	1	0	0
	0	1	0
	0	0	1
	0	0	1

G	0.3	0.7	1
H	0.8	0.2	0.8

HE

Bob's Local Plaintext

Why convert to HE ?

- HE enables local compute instead of expensive oblivious sorting for the vertical setting

Pipeline of Squirrel GBDT Building

- Iterative Step 2/5: Gradient aggregation using HE

ID	1st grad	2nd grad
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

ID	g	h
	0.2	0.1
	0.1	0.7
	0.7	0.2
	0.4	0.5
	0.6	0.3

ID	Age ≤20	20<Age ≤60	Age >60
	1	0	0
	1	0	0
	0	1	0
	0	0	1
	0	0	1

	Age ≤20	20<Age ≤60	Age >60
G	0.3	0.7	1
H	0.8	0.2	0.8

	Age ≤20	20<Age ≤60	Age >60
G	0.3	0.7	1
H	0.8	0.2	0.8

HE

Bob's Local Plaintext

Pipeline of Squirrel GBDT Building

- Iterative Step 3/5: Find the best split with maximum gains
- Find max (ie ArgMax)

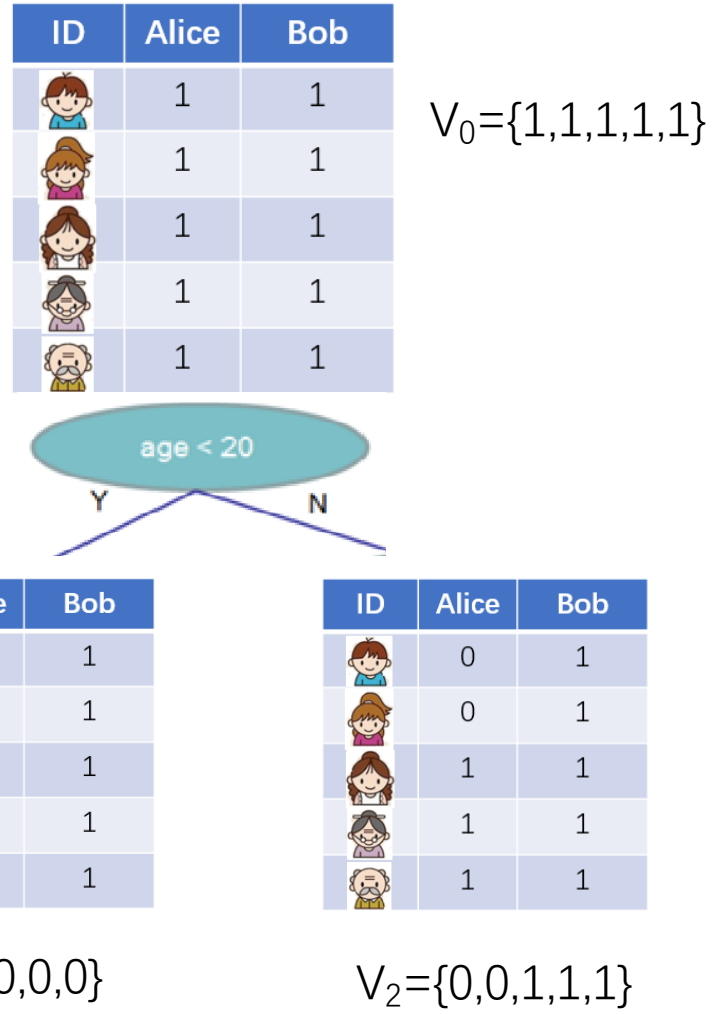
$$\frac{\left(\sum_{Left} g_i\right)^2}{r+\sum_{Left} h_i} + \frac{\left(\sum_{Right} g_i\right)^2}{r+\sum_{Right} h_i} - \frac{\left(\sum g_i\right)^2}{r+\sum h_i}$$

- Use general secret-share-based MPC protocols
  - Secure division\ secure argmax ...
  - Instantiate the OT with Ferret-OTs [6] for lower communication
- The max index is revealed to the corresponding holder

[6] Yang K, Weng C, Lan X, et al. Ferret: Fast extension for correlated OT with small communication, CCS 2020

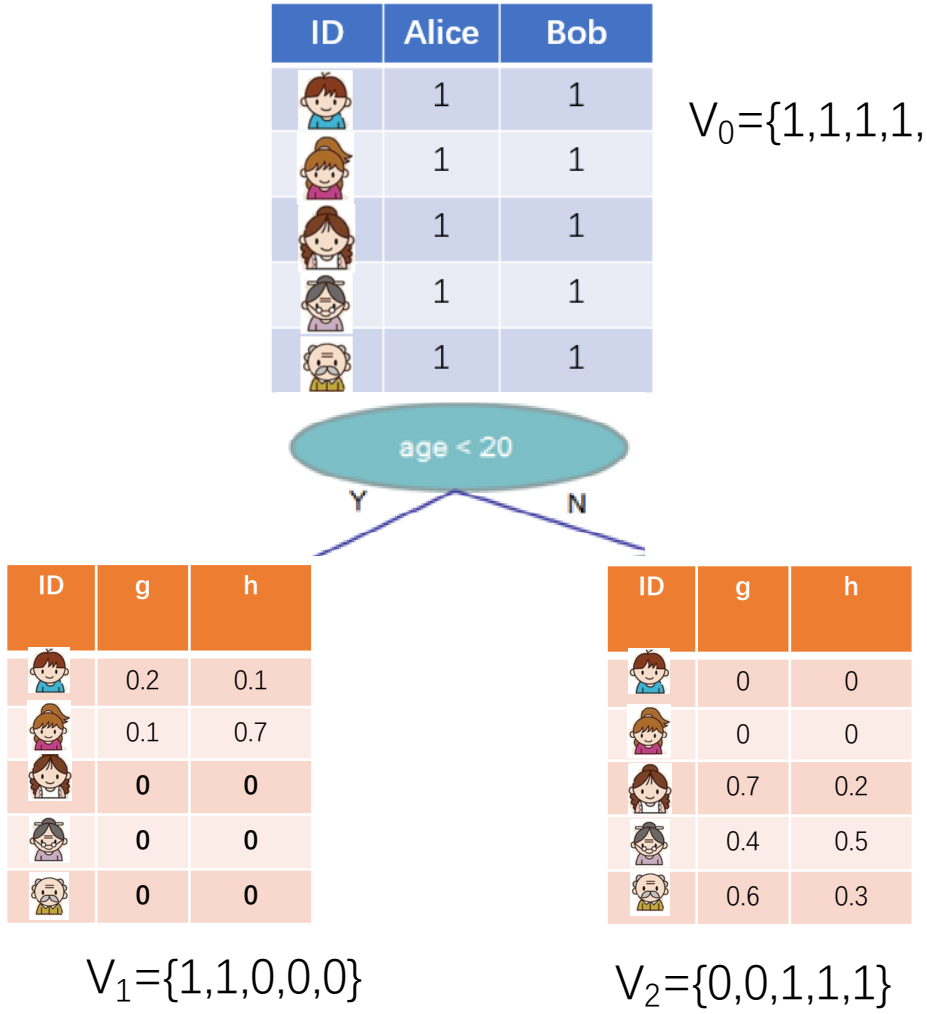
Pipeline of Squirrel GBDT Building

- Iterative Step 4/5: Split the node obliviously
  - Starting at V_0={1,1,1,1,1} sample indicator
  - Update V_i accoring to the split
- Update the gradient tables using OT



Pipeline of Squirrel GBDT Building

- Iterative Step 5/5: Update Predication Vector using OT
- Each leaf node is attached with shared gradient vectors
  - Some entries might be 0s (but still kept private)

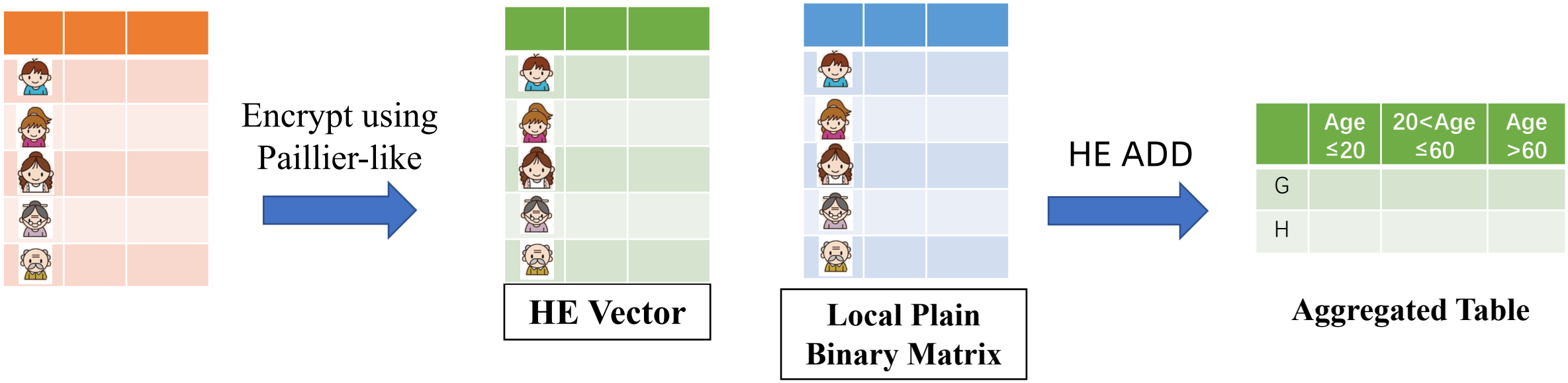


05 Optimizations

The iterative steps are similar to [FZT21] and [YSX20] except with more optimizations

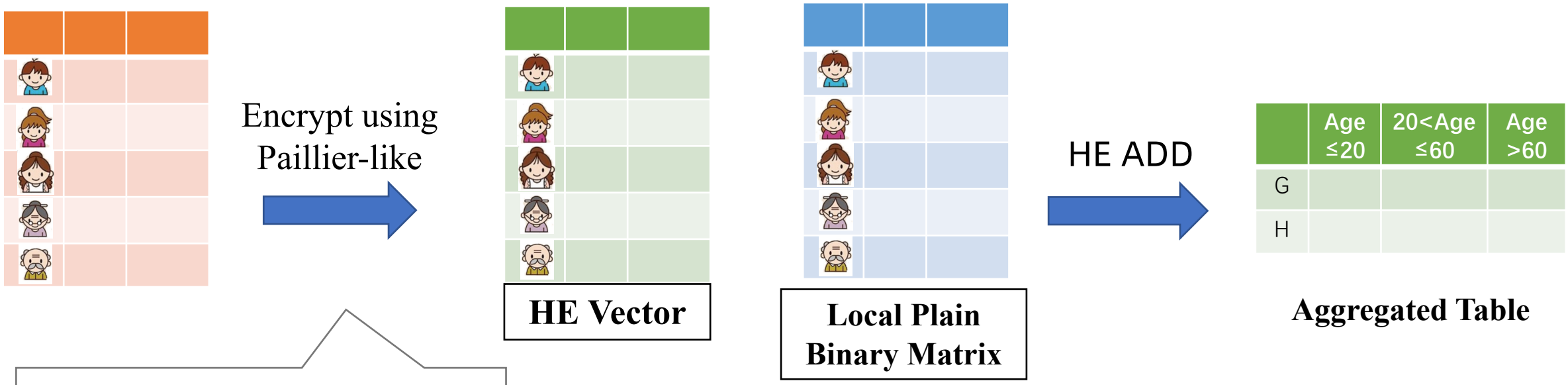
Opt. ½ Fast Gradient Aggregation Using Ring-LWE

[FZT21] and [YSX20] use Paillier-like HE scheme for gradient aggregation



Opt. ½ Fast Gradient Aggregation Using Ring-LWE

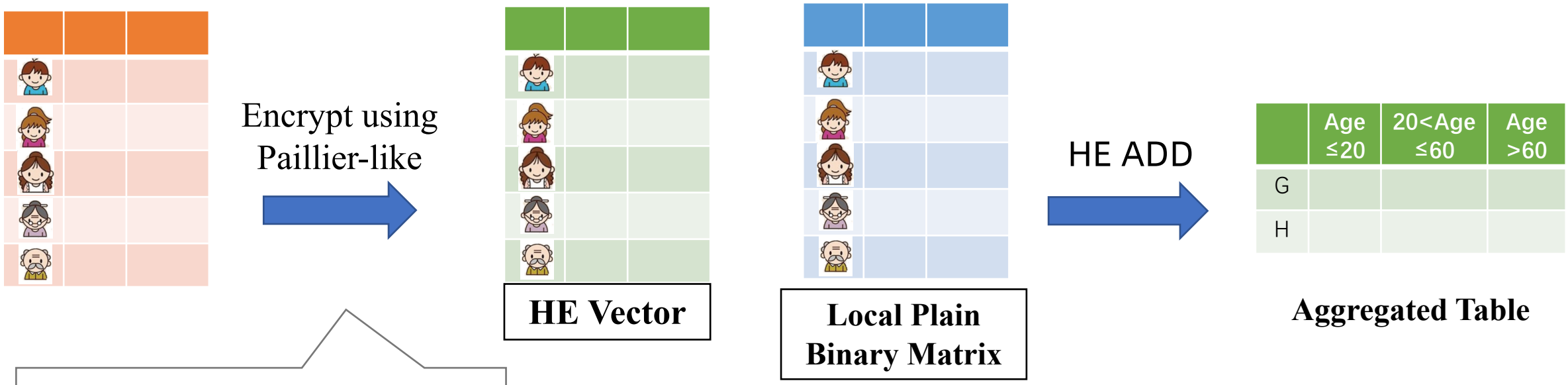
[FZT21] and [YSX20] use Paillier-like HE scheme for gradient aggregation



- CPU intensive:
  - 1~2ms per encryption
- Bandwidth intensive:
  - 256 -- 512B per sample

Opt. ½ Fast Gradient Aggregation Using Ring-LWE

[FZT21] and [YSX20] use Paillier-like HE scheme for gradient aggregation



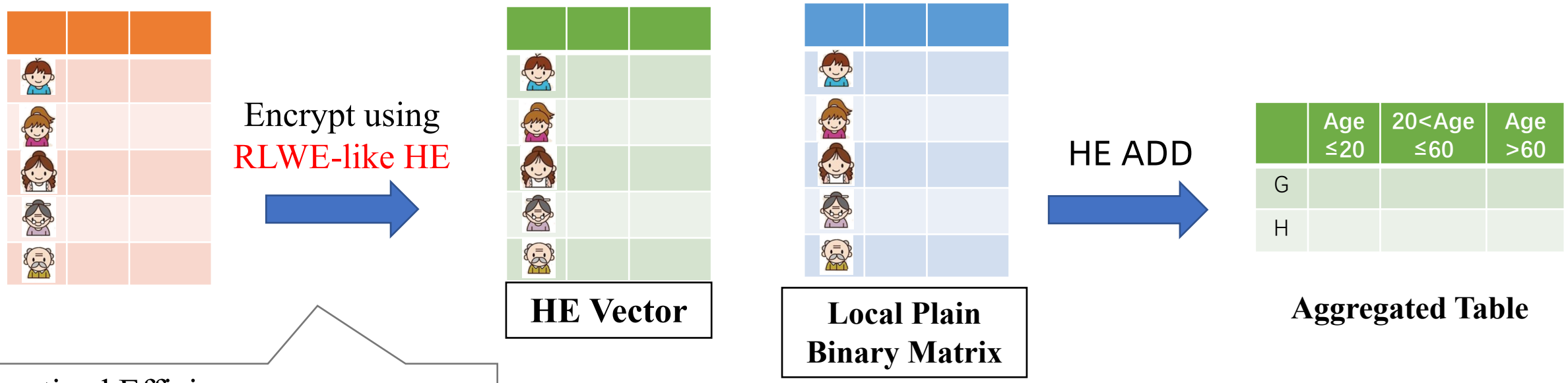
- CPU intensive:
  - 1~2ms per encryption
- Bandwidth intensive:
  - 256 -- 512B per sample

- 1M samples
  - 30min for encryption
  - 250 MB for transition

- Repeat for half nodes in the tree

Opt. ½ Fast Gradient Aggregation Using Ring-LWE

We suggest to encode the gradients in RLWE coefficients



Amortized Efficiency

- 2~3ms for encryption 4k samples
  - 3 order higher throughput
- 55KB cipher for 4k samples
  - 95 - 98% less comm

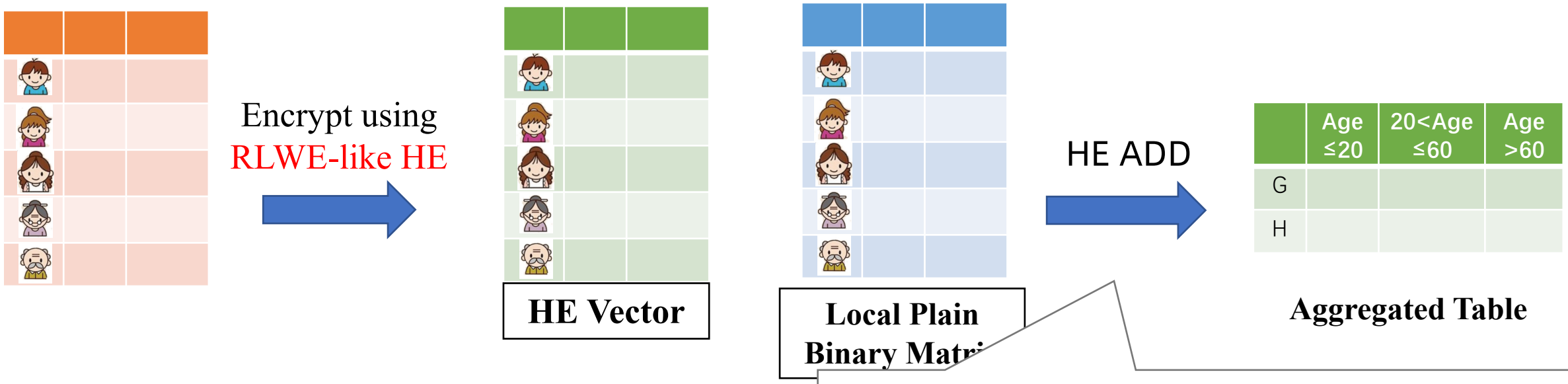
$$\text{RLWE}(m(X))$$

$$m(X) = m_0X^0 + m_1X^1 + m_2X^2 + \cdots + m_{4095}X^{4095}$$

$$m_i \in [-2^{63}, 2^{63})$$

Opt. ½ Fast Gradient Aggregation Using Ring-LWE

We suggest to encode the gradients in RLWE coefficients



$$\text{RLWE}(m(X)) \rightarrow \text{LWE}(m_i)$$

- RLWE-to-LWE is trivial on X^N + 1 ring
  - No crypto op, simple coefficient re-ordering
- HE-ADD over LWE can be easily accelerated using AVX
- 61k ADDs per sec, per CPU core using AVX2

Opt. ½ Fast Gradient Aggregation Using Ring-LWE

We suggest to encode the gradients in RLWE coefficients

Encrypt using
RLWE-like HE

HE Vector

Local Plain Binary Matri

	Age ≤20	20<Age ≤60	Age >60
G			
H			

	Age ≤20	20<Age ≤60	Age >60
G			
H			

$$\text{RLWE}(m(X)) \rightarrow \text{LWE}(m_i)$$

- Each aggregated entry is now an LWE cipher
- Compress hundreds of LWE ciphers into one RLWE for communication efficiency

See paper

Opt. 2/2 A New Sigmoid Protocol

- Binary classification need Sigmoid for the 1st gradient
  - Sigmoid(x) = 1/(1 + exp(-x))
- [FZT21] approximates $0.5 + \frac{0.5x}{1+|x|} \approx \mathrm{sigmoid}(x)$
- [YSX20] computes $2^{x \cdot \log 2(e)}$ for Exp(x) and
- 2PC for large numbers of Div and Exp are still costly
  - Garbled circuits: High communication ☹
  - Numerically Approximation: Too many rounds ☹

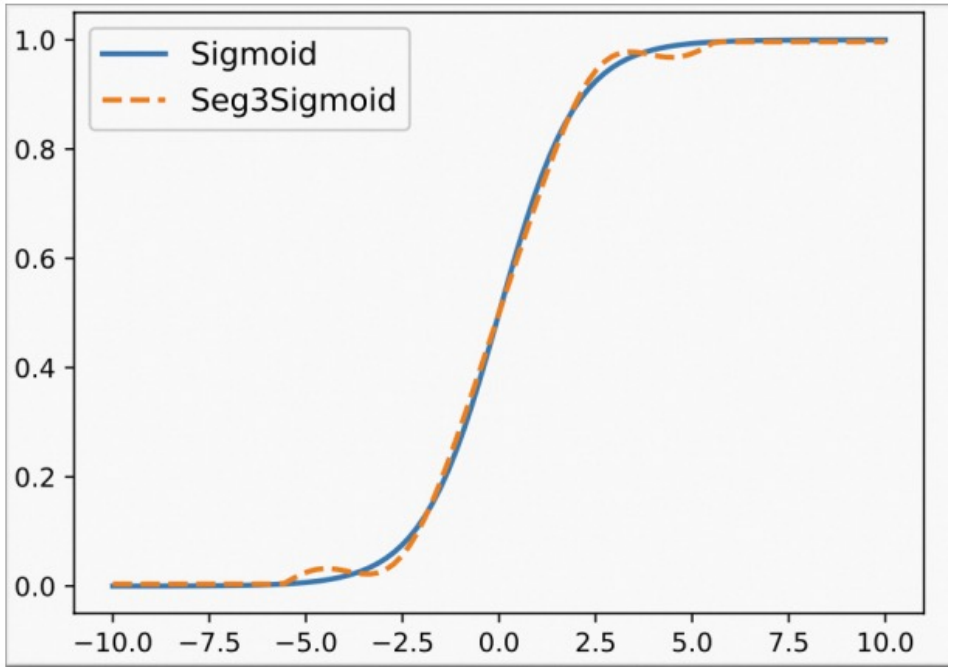
Opt. 2/2 A New Sigmoid Protocol

- 3-segments Approximation

$$\text{Seg3Sigmoid}(x) = \begin{cases} \epsilon & \text{if } x < -5.6 \\ F(x) & \text{if } x \in [-5.6, 5.6] \\ 1 - \epsilon & \text{if } x > 5.6 \end{cases}$$

  - A small number of branching
- Middle segment Fourier series:

$$F(x) = \omega_0 + \sum_{j=1}^{9} \omega_j \sin\left(\frac{2\pi \cdot j \cdot x}{2^{L+1}}\right) \text{ for } |x| \leq 2^L$$



Opt. 2/2 A New Sigmoid Protocol

- Insight: the Fourier can be computed very efficiently (single round!) on secret share

$$
\begin{aligned}
\sin(2\pi x/2^{\delta}) &= \sin(2\pi(\langle x\rangle_0 + \langle x\rangle_1 + \varepsilon 2^{\ell})/2^{\delta}) \\
&= \sin\left(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0) + 2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)\right) \\
&= \sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)) \\
&\quad + \cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)).
\end{aligned}
$$

Definition of Arith share

Opt. 2/2 A New Sigmoid Protocol

- Insight: the Fourier can be computed very efficiently (single round!) on secret share

$$
\begin{aligned}
\sin(2\pi x/2^{\delta}) &= \sin(2\pi(\langle x\rangle_0 + \langle x\rangle_1 + \varepsilon 2^{\ell})/2^{\delta}) \\
&= \sin\left(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0) + 2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)\right) \\
&= \sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)) \\
&\quad + \cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)).
\end{aligned}
$$

Definition of Arith share

Periodic property

Opt. 2/2 A New Sigmoid Protocol

- Insight: the Fourier can be computed very efficiently (single round!) on secret share

$$
\begin{aligned}
\sin(2\pi x/2^{\delta}) &= \sin(2\pi(\langle x\rangle_0 + \langle x\rangle_1 + \varepsilon 2^{\ell})/2^{\delta}) \\
&= \sin\left(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0) + 2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)\right) \\
&= \sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)) \\
&\quad + \cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)).
\end{aligned}
$$

Definition of Arith share

Periodic property

Double-angle of sin()

Opt. 2/2 A New Sigmoid Protocol

- Insight: the Fourier can be computed very efficiently (single round!) on secret share

$$
\begin{aligned}
\sin(2\pi x/2^{\delta}) &= \sin(2\pi(\langle x\rangle_0 + \langle x\rangle_1 + \varepsilon 2^{\ell})/2^{\delta}) \\
&= \sin\left(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0) + 2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)\right) \\
&= \sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)) \\
&\quad + \cos(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_0))\sin(2\pi \mathrm{Fr}_{\delta}(\langle x\rangle_1)).
\end{aligned}
$$

Locally computable by A
Locally computable by B

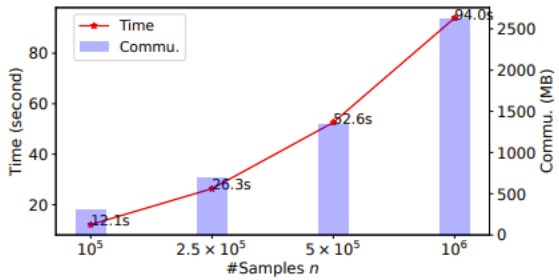
$$
F(x) = \omega_0 + \sum_{j=1}^{9} \omega_j \sin\left(\frac{2\pi \cdot j \cdot x}{2^{L+1}}\right) \text{ for } |x| \le 2^{L}
$$

2*9 MULs
is enough

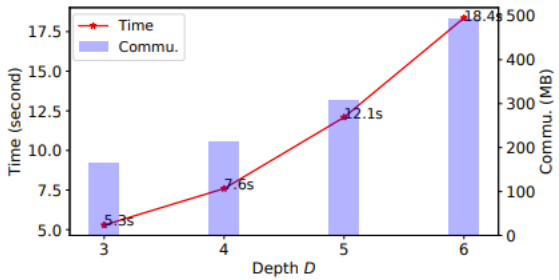
06 Experiments

Experiment results

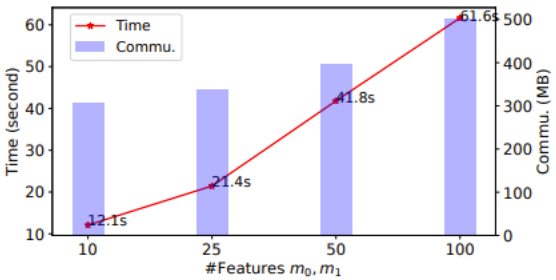
- 170+ seconds; 2.5GB per tree; (exclude PSI)
  - WAN (200mbps/20ms ping), 1 million samples * 20 features, tree depth = 5 (ie., 31 nodes)



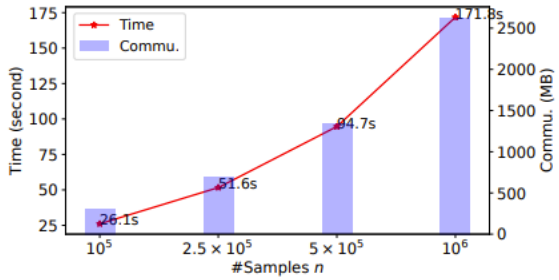
(a) Time (LAN) vs. Sample size n



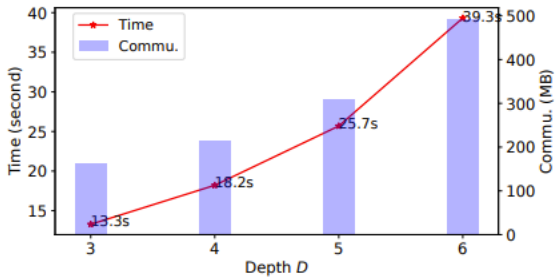
(b) Time (LAN) vs. Depth D



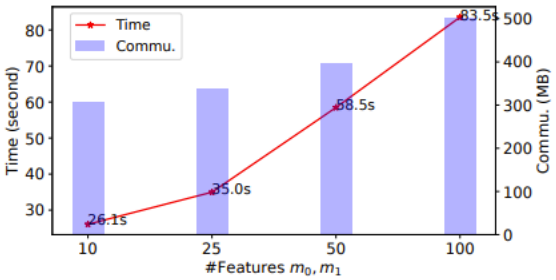
(c) Time (LAN) vs. Feature size m_0, m_1



(d) Time (WAN) vs. Sample size n



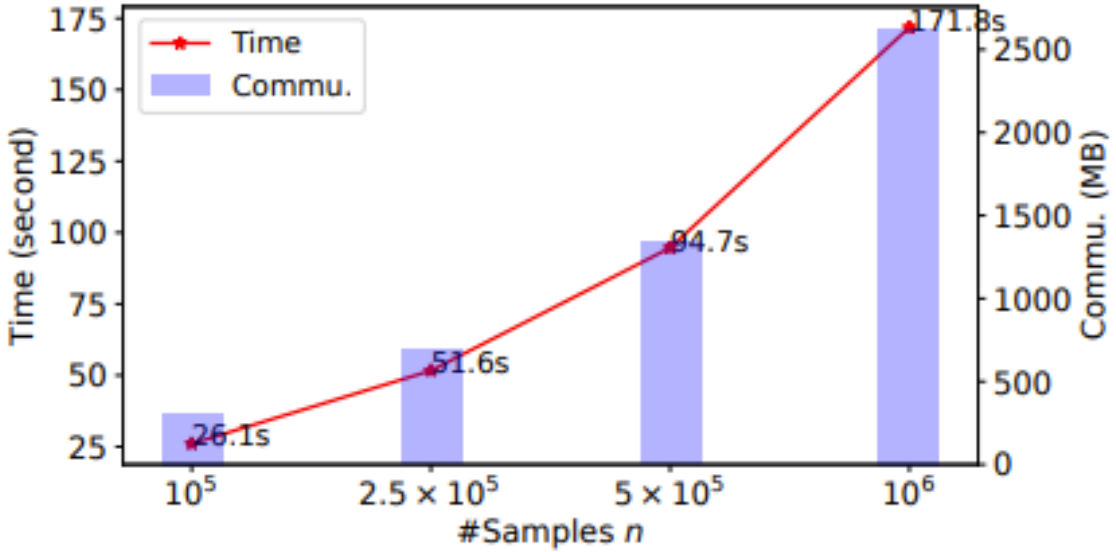
(e) Time (WAN) vs. Depth D



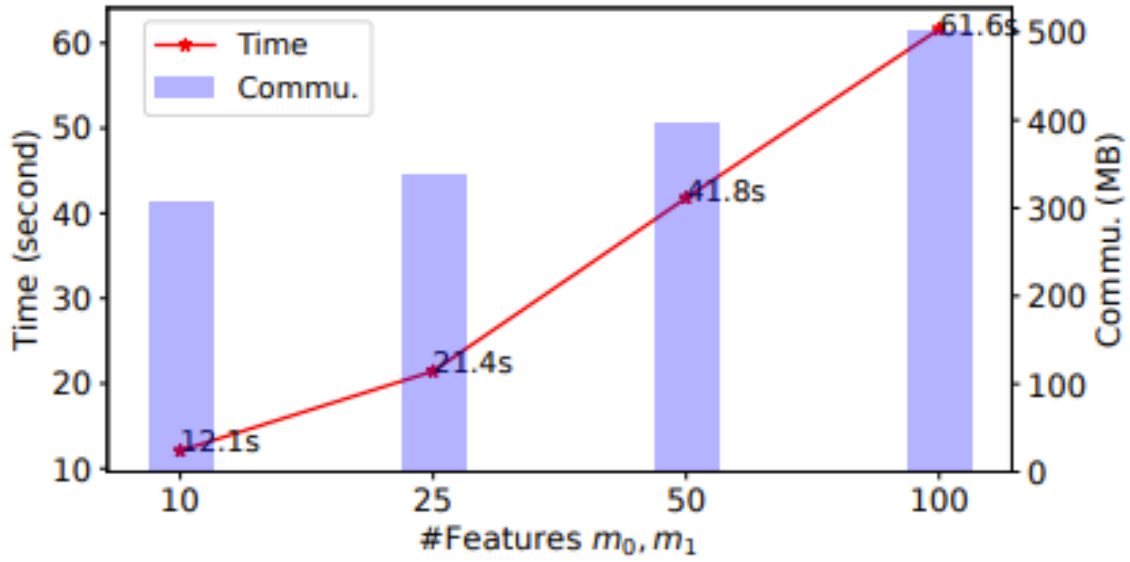
(f) Time (WAN) vs. Feature size m_0, m_1

Experiment results

- 170+ seconds; 2.5GB per tree;
  - WAN (200mbps/20ms ping), 1 million samples * 20 features, tree depth = 5 (ie., 31 nodes)



(d) Time (WAN) vs. Sample size n



(c) Time (LAN) vs. Feature size m_0, m_1

Time scales linearly
Level-wise Full Tree

Comm. scales gently

Experiment results

- 28-40x speedup vs Pivot [YSX20]
- 3-4x speedup vs HEP-XGB [FZT21]
  - Even they use Trusted hardware to generate multiplication triples

Approach	Parameters	Settings	Time
Ours	$n = 5 \times 10^4, D = 4$	LAN	6.0s
[58]†	$m_0 = 8, m_1 = 7, B = 8$	6 threads	168s (28×)
Ours	$\mathbf{n = 2 \times 10^5}, D = 4$	LAN	11.1s
[58]	$m_0 = 8, m_1 = 7, B = 8$	6 threads	448s (40×)
Ours	$n = 1.4 \times 10^5, D = 5$	LAN	11.4s
[22]‡	$m_0 = 7, m_1 = 16, B = 10$	32 threads	47.6s (4×)
Ours	$n = 1.4 \times 10^5, D = 5$	**100 Mbps**	40.0s
[22]	$m_0 = 7, m_1 = 16, B = 10$	32 threads	151s (3.7×)

†`Pivot` [58] did not report the pre-processing time.
‡`HEP-XGB` [22] have used TEE to accelerate their computation.

[3] Fang W, Zhao D, Tan J, et al. Large-scale secure XGB for vertical federated learning, CIKM 2021
[4] Wu Y, Cai S, Xiao X, et al. Privacy Preserving Vertical Federated Learning for Tree-based Models, VLDB 2020

Conclusion

- RLWE -> LWE for fast gradient aggregation
- LWE -> RLWE for less communication for aggregated table
- Smoother and cheaper Sigmoid
  - Compatible small comm. with Functional Secret sharing-based sigmoid
  - Some follow ups opt. are given in our e-print version

Amit Agarwal et al. *Communication Efficient Secure Logistic Regression*

— THANKS —

QA

- Why only two-party, can use more ?
  - Should work for ABY3; but we only meet needs of two-party
- BGV or BFV ?
  - BFV, because we want 2^k plaintext modulus
- Can we keep the split point private to both parties ?
  - Unfornately no;
- Level-wise or leaf-wise ? Why ?
  - Level-wise; leaf-wise might introudce too many communication rounds
- Why such low bandwidth ?
  - For our case study, a high speed physical cable is very expensive.