UT DALLAS

Securing Big Data in the Age of Artificial Intelligence

Murat Kantarcioglu

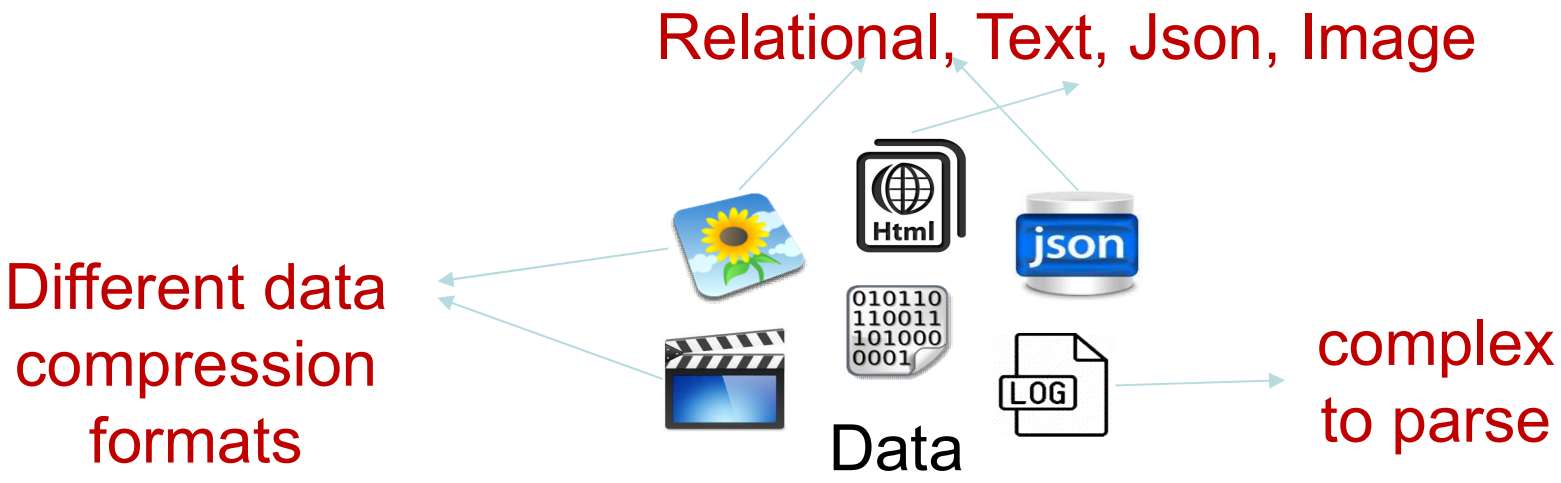
UT Dallas, Harvard University and DataSecTech

Big Data Revolution: Changing Landscape

- Amount of data that is generated is increasing.
- New storage and processing models
  - Apache Hadoop
  - Apache Spark
  - Others…
- Hybrid Cloud Architectures
- AI/ML is changing the landscape
- Need to secure the data
- Data privacy becomes more important due to compliance

Challenges for Big Data

- Volume (Google stores 10-15 exabytes, 1 exabyte=1 million terabytes)
- Variety: Unstructured, semi-structured
- Velocity (350 million new images uploaded to Facebook every day)
- Veracity (incomplete data)
- **No Single NoSQL/SQL database to rule them all**



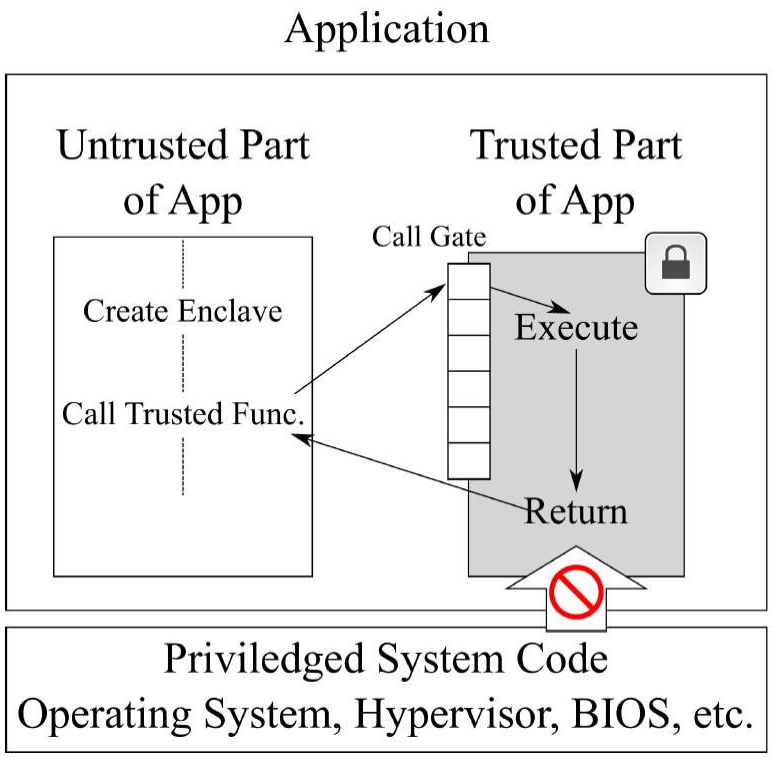
Problem

- Need to Protect Big Data Against Cyber Attacks
  - Big Data is critical for many organizations
  - New cyber attacks against big data storage systems
- Need to Comply with New Regulations
  - E.g., EU General Data Protection Directive
- **No simple and effective way to protect big data while complying with regulations across multiple databases**

Additional Problems Due to ML and Cloud

- Need to worry about the data security at rest and/or cloud outsourcing.
- Learned ML models could be vulnerable to attacks.
- Implications of deployed ML for privacy and security

Other Approach: Use Hardware Support for Efficient Oblivious Data Processing **



Intel Sgx Architecture

- ***Protect*** *the secrecy and integrity* of big data and the ML models using encryption and hardware support
- Enable **general language** for data processing while satisfying data obliviousness
- **Make it efficient enough for general use**

** ACM CCS 2017

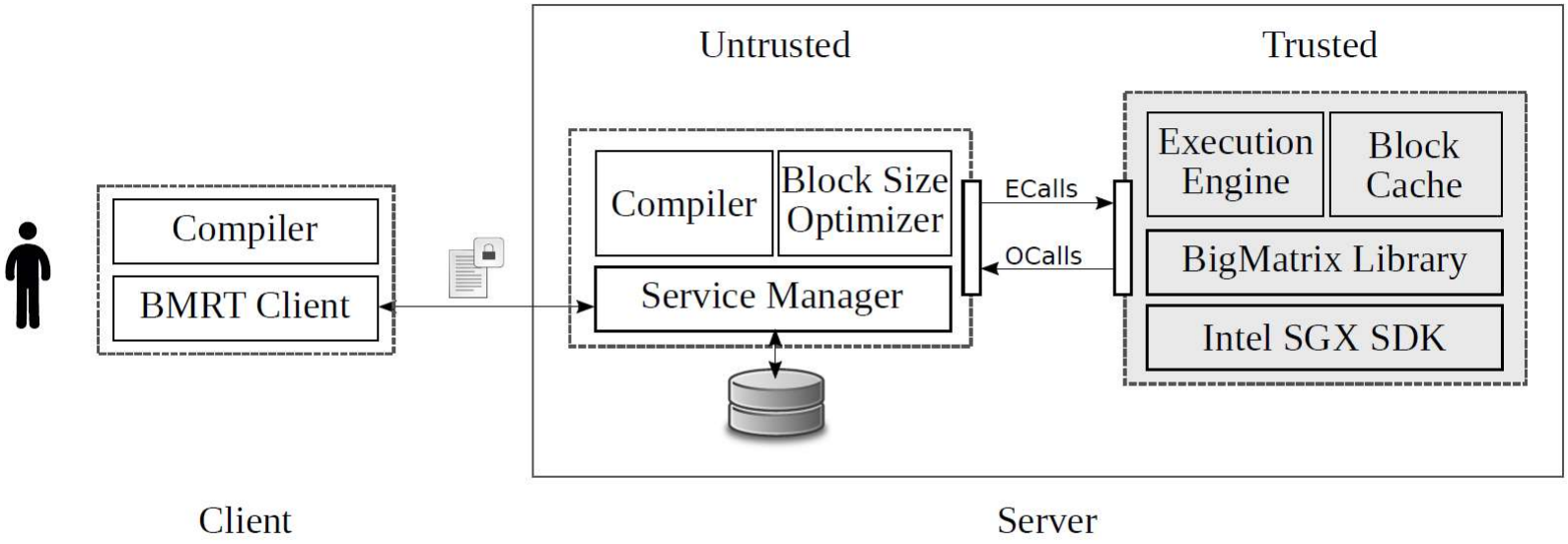
How to Support Data Obliviousness ??

- Idea, remove **If statements using vectorization**

```
sum = 0, count = 0
for i = 0 to Person.length:
    if Person.age >= 50:
        count++
        sum += P.income
print sum / count
```

```
S = where(Person, "Person['age'] >= 50")
print (S .* Person['income'] ) / sum(S)
```

SGX- BigMatrix Architecture



SGX BigMatrix

Compiler

- Compiles our python like language into **basic commands**
- Data obliviousness using **data oblivious building blocks** and **operation vectorizations**

Input

```
x = load('path/to/X_Matrix')
y = load('path/to/Y_Matrix')
xt = transpose(x)
theta = inverse(xt * x) * xt * y
publish(theta)
```

Compiler-Output

Output

```
x = load(X_Matrix_ID)
y = load(Y_Matrix_ID)
xt = transpose(x)
t1 = multiply(xt, x)
unset(x)
t2 = inverse(t1)
unset(t1)
t3 = multiply(t2, xt)
unset(xt)
unset(t2)
theta = multiply(t3, y)
unset(y)
unset(t3)
publish(theta)
```

Support for Basic Data Science

- E.g., SQL, Matrix Operations etc.

Input

```
I = sql('SELECT *
FROM person p
JOIN person_income pi (1)
ON p.id = pi.id
WHERE p.age > 50
AND pi.income > 100000')
```

Other Important Features

- **Automatic Sensitivity Analysis** for flagging sensitive information disclosure
  - I.e., using sensitive output for allocating a new array
- **Cost based and secure optimization** for optimizing blocking
  - Sgx do not support efficient data buffering

Experimental Evaluation

- Performed linear regression on two popular datasets

Data Set	Rows	BigMatrix Encrypted
USCensus1990	2,458,285	3m 5s 460ms
OnlineNewsPopularity	39,644	2s 250ms

Table: Time results of linear regression on real datasets

- Performed Page Rank on three popular datasets

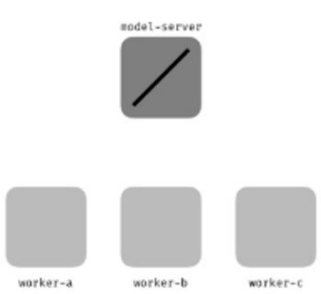
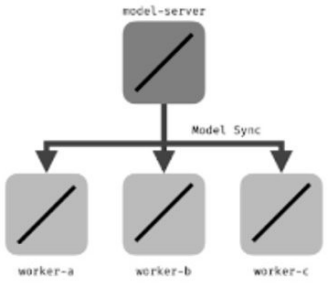
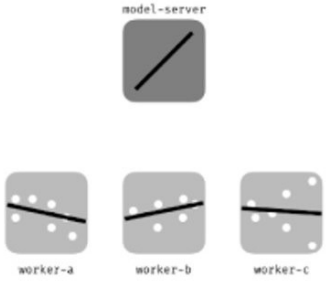
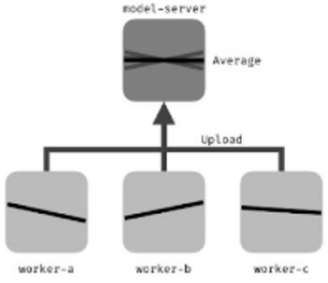
Data Set	Nodes	BigMatrix Encrypted
Wiki-Vote	7,115	97s 560ms
Astro-Physics	18,772	6m 41s 200ms
Enron Email	36,692	23m 19s 700ms

Comparison with ObliVM

Matrix Dimension	ObliVM	BigMatrix SGX Enc.	BigMatrix SGX Unenc.
100	28s 660ms	10ms	10ms
250	7m 0s 90ms	93ms	88ms
500	53m 48s 910ms	706.66ms	675.66ms
750	2h 59m 40s 990ms	2s 310ms	2s 260ms
1,000	6h 34m 17s 900ms	10s 450ms	10s 330ms

Table: Two-party matrix multiplication time in ObliVM vs BigMatrix

Federated Learning: Privacy vs Robustness*

Step 1	Step 2	Step 3	Step 4
			
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

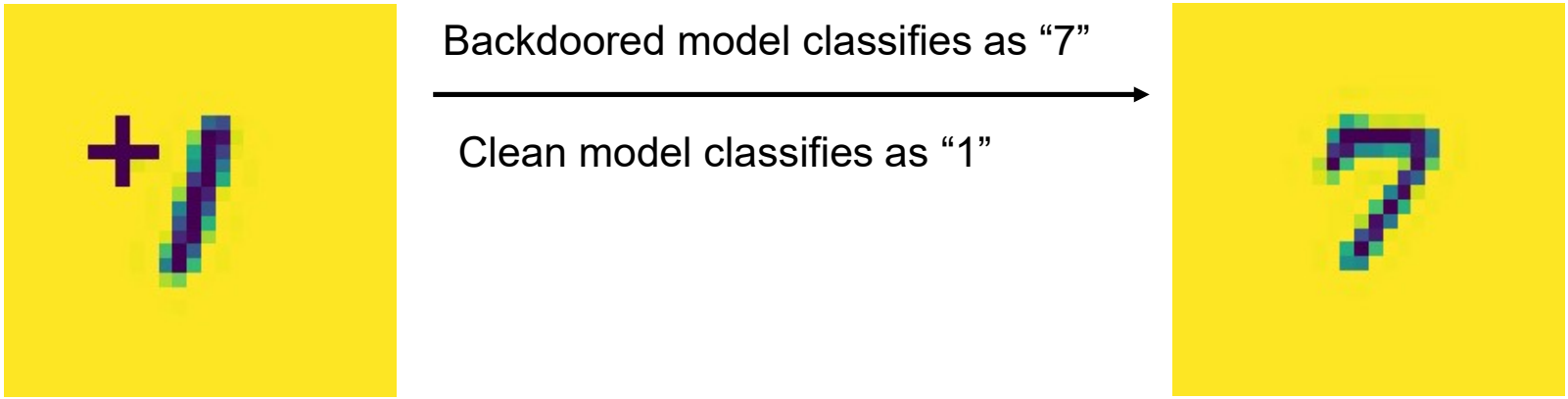
https://en.wikipedia.org/wiki/Federated_learning

* AAAI 2021

Backdoor Attacks

- Backdoor: a targeted misclassification functionality

- Can be introduced via data poisoning in centralized setting
  - Gain access to training data
  - Add pixel-pattern to some samples of a class, re-label them to a target class



Backdoor Attacks in FL context

- Backdoor attacks can be carried via model poisoning [1-2]
  - Corrupt agents train their models on poisoned data
  - Send the malicious update to server for aggregation

- Aggregation function should be robust
  - A single adversary can arbitrarily skew FedAvg

$$w_{t+1} = w_t + \eta \frac{\sum_{k \in S_t} n_k \cdot \Delta_t^k}{\sum_{k \in S_t} n_k}$$

w_t: weights at round t
S_t: selected agents at round t
Δ_t^k: update of k'th agent at round t
n_k: dataset size of k'th agent
η: server's learning rate

Overview

- A defense against backdoor attacks in federated learning (FL) context
- Main idea: adjust learning rate of aggregation server, per round and per dimension, based on updates' sign
  - No structural changes
  - Can be used with any aggregation function
- Evaluation in both iid, and non-iid settings
  - Comparison with a few recent defenses

Our Defense: Robust Learning Rate (RLR)

- Let w_{adv}, w_{hon} be two distinct points on parameter space
  - w_{adv} : minimizes loss on backdoor, and main tasks
  - w_{hon}: minimizes loss on main task

- For some dimensions, honest and corrupt agents will try to move the model to different directions

- Sign information of updates can be treated as votes for directions.

Our Defense: Robust Learning Rate (RLR) -2

- A hyperparameter called learning threshold, θ, at server-side
- For a dimension i, if sum of signs is less than θ, negate learning rate for dimension i
  - To maximize loss on that dimension

$$\eta_{\theta,i} = \begin{cases} \eta & \left|\sum_{k\in S_t} \operatorname{sgn}(\Delta_{t,i}^k)\right| \geq \theta \\ -\eta & \text{otherwise.} \end{cases}$$

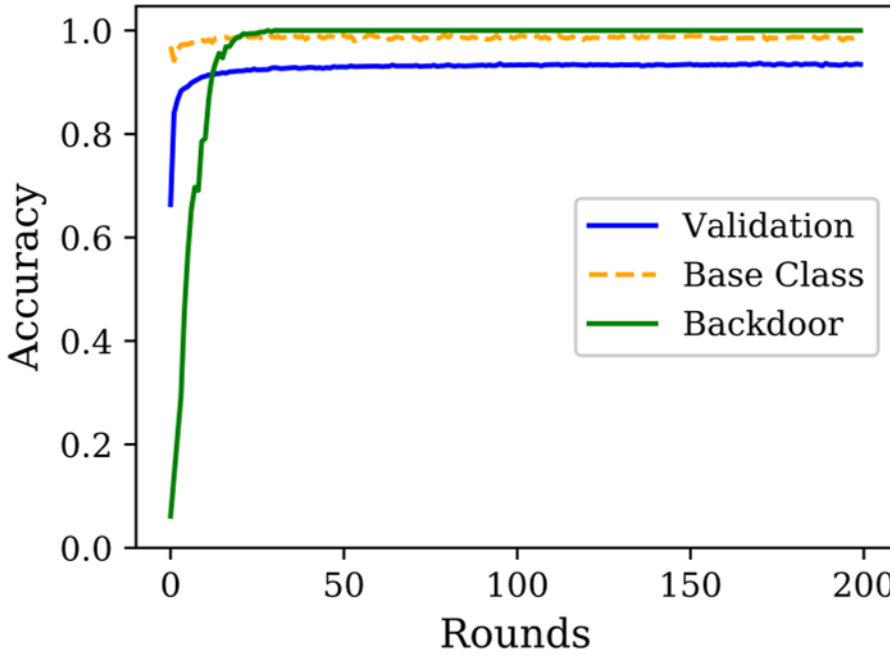
$$w_{t+1} = w_t + \eta_\theta \odot \frac{\sum_{k\in S_t} n_k \cdot \Delta_t^k}{\sum_{k\in S_t} n_k}$$

Experiments

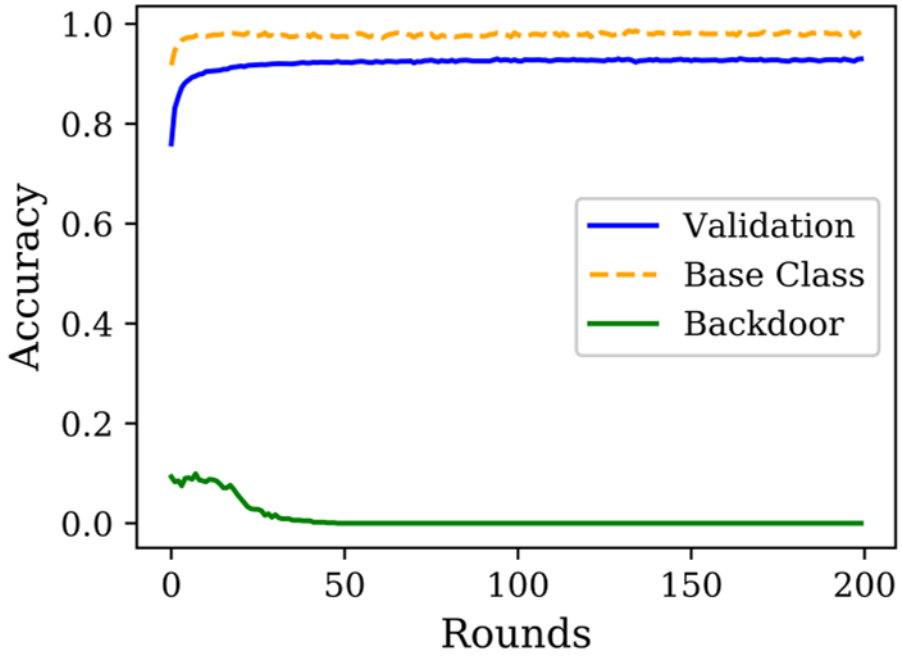
- Simulate FL where 10% of agents are corrupt
  - Corrupt agents poison their local data via trojans
  - IID setting with 10 agents on Fashion-MNIST [4]
  - NIID setting with ~3k agents on FEMNIST [5]
    - ~ 30 agents per round

- Three metrics measured at each round
  - Validation accuracy
  - Backdoor accuracy
    - e.g., whether trojaned “1”s are classified as “7’
  - Base class accuracy
    - e.g., whether clean “1’s are classified as “1"

Learning Curves - IID

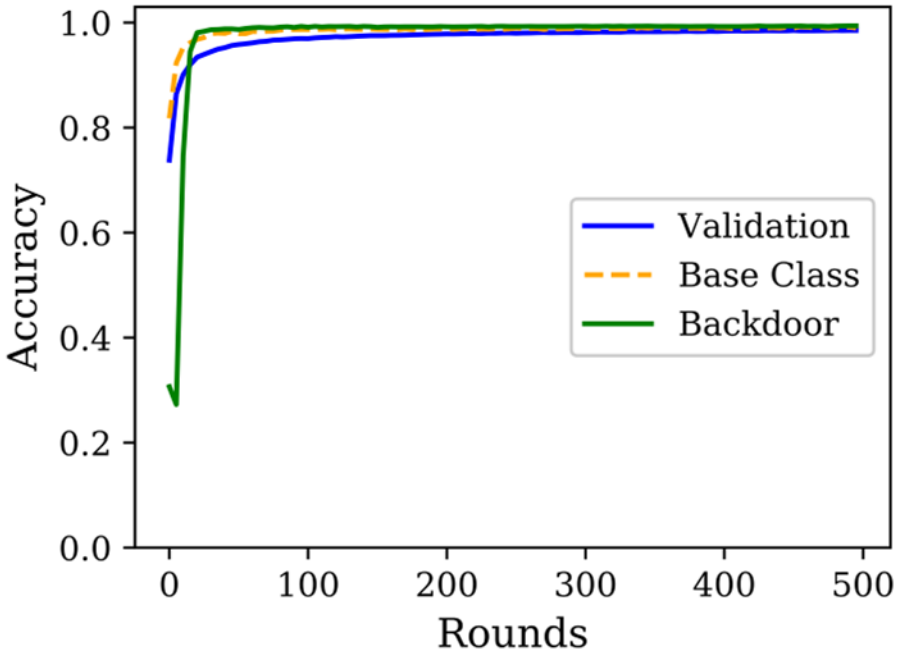


FedAvg

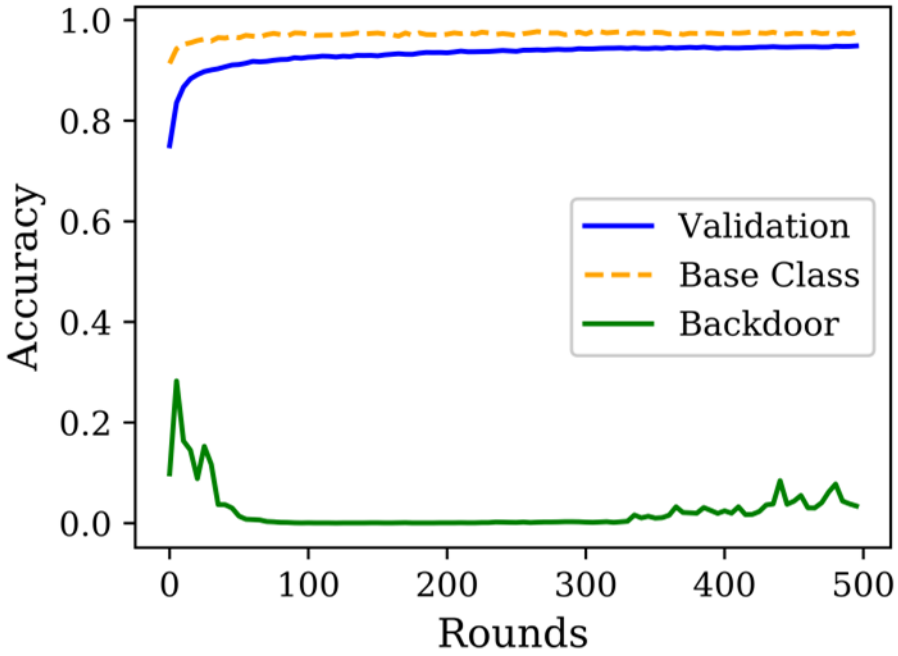


FedAvg with RLR

Learning Curves - NIID



FedAvg



FedAvg with RLR

Comparison with Other Defenses - IID

Aggregation	M	σ	Backdoor (%)	Validation (%)	Base (%)
FedAvg-*No Attack*	0	0	1	**93.5**	98.5
FedAvg	0	0	100	93.4	98.5
FedAvg	4	1e-3	100	93.2	**99.1**
FoolsGold	0	0	100	93.1	98.9
FoolsGold	4	1e-3	100	93.3	98.5
Comed	0	0	100	92.8	99.0
Comed	4	1e-3	99.5	92.8	98.4
Sign	0	0	100	92.9	98.7
Sign	4	1e-3	99.7	93.1	98.6
FedAvg with RLR	0	0	**0**	92.9	98.3
FedAvg with RLR	4	1e-3	0.5	92.2	97.4

DP can be applied to limit contribution of each agent

- For fairness/privacy purposes [6]
- Can deter label-flipping backdoors [7]

M: L_2-norm threshold on updates

σ: std.dev of Gaussian noise

FoolsGold [8]
Comed [9]
Sign [3]

Comparison with Other Defenses - NIID

Aggregation	M	σ	Backdoor (%)	Validation (%)	Base (%)
FedAvg*-*No Attack*	0	0	21.1	**98.6**	**99.1**
FedAvg	0	0	99.3	98.5	99.0
FedAvg	0.5	1e-3	99.2	98.0	98.7
FoolsGold	0	0	98.5	98.9	99.5
FoolsGold	0.5	1e-3	99.1	97.9	98.6
Comed	0	0	82.3	96.3	98.4
Comed	0.5	1e-3	95.2	95.5	98.1
Sign	0	0	99.8	97.6	98.7
Sign	0.5	1e-3	99.7	97.8	98.5
FedAvg with RLR	0	0	3.4	94.8	97.6
FedAvg with RLR	0.5	1e-3	**0.4**	93.2	97.7

Conclusion: FL Poisoning Attacks

- A simple defense that requires no changes to FL
- Agnostic to the aggregation function
- Outperforms some of the recent defenses
- Full version contains the following.
  - Distributed backdoor attacks [10]
  - Combining RLR with other aggregation functions
  - Extended set of experiments
    - More trojan patterns, higher corruption rates
    - M, σ values etc.

Learned ML models and Privacy Implications

- Your Facebook likes can expose your personality
- Your profile picture can reveal your satisfaction with life/ personal traits ??
- Who you follow on Twitter can detect account anomalies

Your Are What You

→ High IQ

→ Dissatisfied With Life

→ Shy & Reserved

→ Few Friends

Attacking models to improve privacy and fairness *

- Cost function $c(x, x')$
- Target classifier f(x)
- Ethical/legal transformation set F_x
- Find the low cost transformation that achieves the goal:

$$\arg\min_{x'} \quad c(x, x')$$
$$subject\ \ to\ x' \in F_x, f(x') = t$$

GameSec 2020

Example: Attacking Image Classifiers

- Prevent image classifier from predicting private info.
  - E.g., Sexual Orientation.

- Make sure the noise added ϵ satisfies certain domain constraints for some suitable norm:

$$\arg\min_{\epsilon} \quad f(x+\epsilon) = t$$

$$subject \;\; to \; \|\epsilon\| \leq \delta, \epsilon \in \mathcal{X}_m$$

Domain constraint Example

Example: Prevent Gender Prediction

- CelebA: Celebrity Picture Data Set
- Attacked Model: VGGFACE+ VGC16
- Accuracy on Clean Data: 94.44%
- Attack:
  - Random Pictures: 297 Female, 266 Male
  - Using the glasses as the constraints, and just change those pixels.
  - Changes normalized to [-1,1] range for those pixels
  - 100% of the pictures can be attacked successfully.

Change Images Using Glasses

Conclusion

- Protect data by pushing data protection closer to data sources
- Need to consider compliance and data privacy
- Securing ML vs Attacking ML for Increasing Privacy

Questions?

- **This work is supported by the following grants:**
  - Air Force Office of Scientific Research Grant FA9550-12-1-0082, National Institutes of Health Grants 1R01LM009989 and 1R01HG006844, National Science Foundation (NSF) Grants Career-CNS-0845803, CNS-0964350, CNS-1016343, CNS-1111529, CNS-1228198, Army Research Office Grant W911NF-12-1-0558, NSF SBIR Phase 1 and Phase 2 Grants.