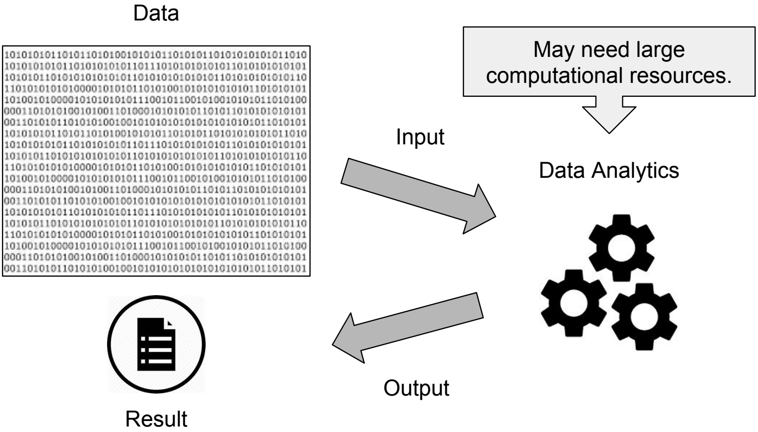
Securing Data Analytics on SGX With Randomization

Swarup Chandra, Vishal Karande, Zhiqiang Lin, Latifur Khan, Murat Kantarcioglu and Bhavani Thuraisingham

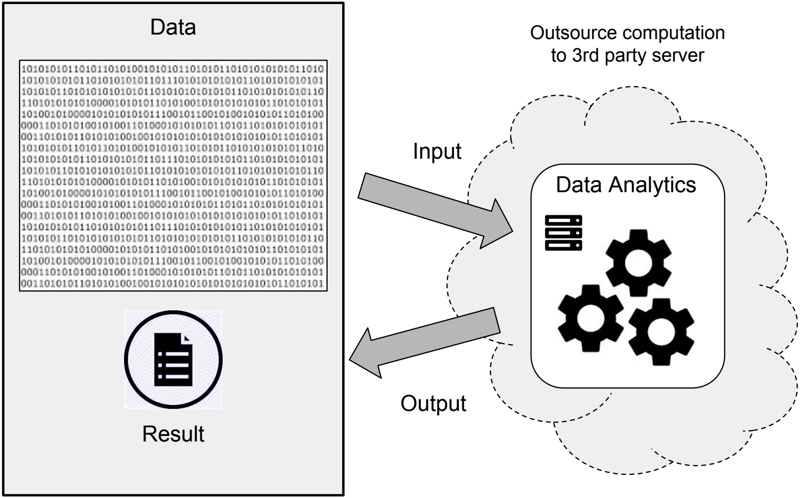
The University of Texas at Dallas

ESORICS 2017

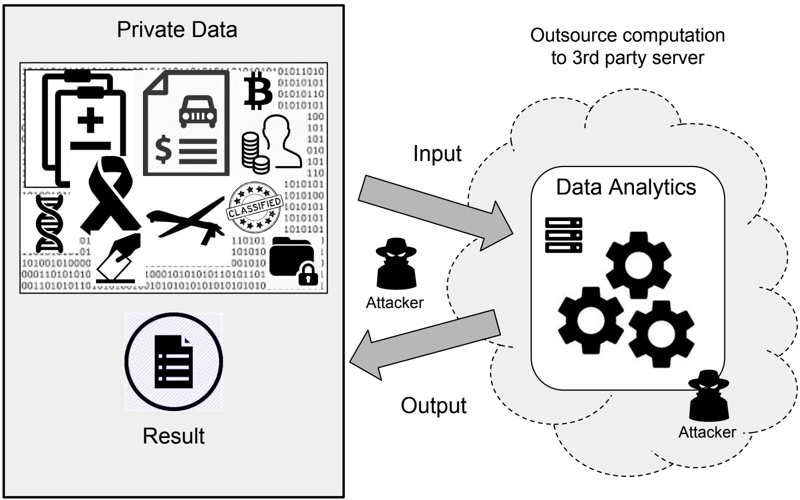
The Problem: Data Analytics in Untrusted Platform



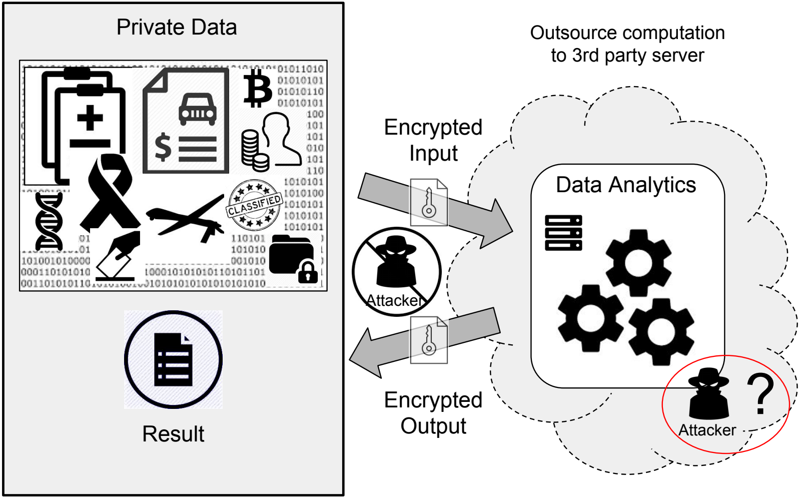
The Problem: Data Analytics in Untrusted Platform



The Problem: Data Analytics in Untrusted Platform



The Problem: Data Analytics in Untrusted Platform



Existing Efforts

Privacy Preserving Analytics

Modifications to data before sharing it to the third-party server [AP08].

Existing Efforts

Privacy Preserving Analytics

Modifications to data before sharing it to the third-party server [AP08].

Homomorphic Encryption

Using fully **homomorphic encryption** scheme (not practical) [LWN+15].

Existing Efforts

Privacy Preserving Analytics

Modifications to data before sharing it to the third-party server [AP08].

Homomorphic Encryption

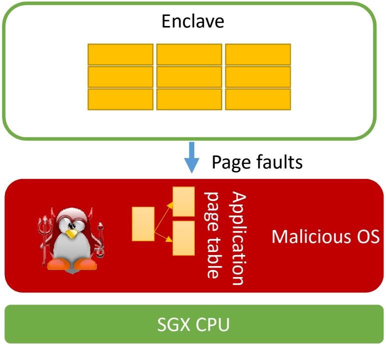
Using fully **homomorphic encryption** scheme (not practical) [LWN+15].

Leveraging Trusted Execution Environment (TEE)

- Recent TEEs: Intel SGX [MAB+13], ARM TrustZone.
- Perform analytics on private data within secure environment, isolated from adversary [SCF+15].

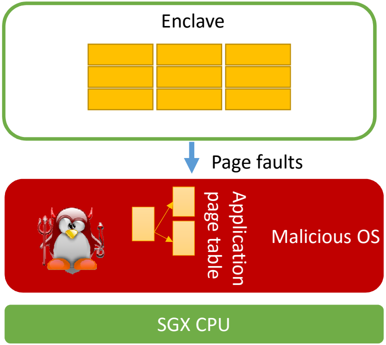
Issues w/ TEE Approaches

Information leakage via (controlled) side-channels harms data privacy



Issues w/ TEE Approaches

Information leakage via (controlled) side-channels harms data privacy



Attacker can use page-fault attack to observe execution flow and guess parameter values [XCP15].

State-of-the-Art

- **Balanced Execution** [SCNS16]: Execute each conditional branch.
- **Data oblivious execution** [OSF+16]: Manipulate execution sequence to be fully data independent.

State-of-the-Art

- **Balanced Execution** [SCNS16]: Execute each conditional branch.
- **Data oblivious execution** [OSF^{+}16]: Manipulate execution sequence to be fully data independent.

Execution overhead impractical for large analytics.

- When executing every path for each input, this creates bottleneck due to unnecessary path execution especially when **large parameters** are involved.

State-of-the-Art

- **Balanced Execution** [SCNS16]: Execute each conditional branch.
- **Data oblivious execution** [OSF+16]: Manipulate execution sequence to be fully data independent.

Execution overhead impractical for large analytics.

- When executing every path for each input, this creates bottleneck due to unnecessary path execution especially when **large parameters** are involved.

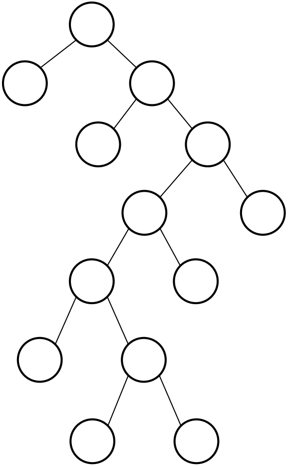
What are large parameters?

- Decision Tree: **Large number of nodes**
- Naive Bayes: **Large domain size**
- K-Means: **Large domain size**

Limtations in State-of-the-Art

Example Decision Tree

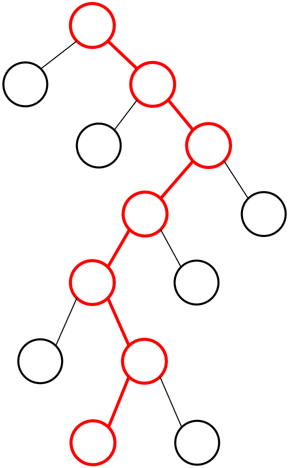
- Given a trained decision tree.



Limtations in State-of-the-Art

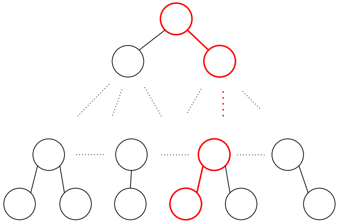
Example Decision Tree

- Given a trained decision tree.
- **Input**: Data instance X for evaluation.



Limtations in State-of-the-Art
Example Decision Tree

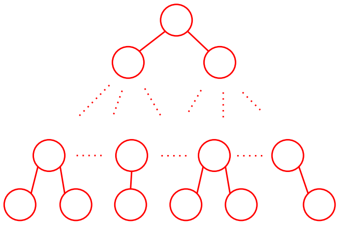
- Given a trained decision tree.
- **Input**: Data instance X for evaluation.
- **Defense**: Hide structure using dummy nodes.



Limtations in State-of-the-Art

Example Decision Tree

- Given a trained decision tree.
- **Input**: Data instance X for evaluation.
- **Defense**: Hide structure using dummy nodes.
- But, evaluation of X will explore all branches causing computational bottleneck.



Intel SGX

Intel Software Guard Extension (SGX)

Capabilities

- Offering secure enclave
- Confidentiality, Integrity

Intel SGX

Intel Software Guard Extension (SGX)

Capabilities

- Offering secure enclave
- Confidentiality, Integrity

- New extensions to Intel x86/64 architecture
- Applications keep secret data and code inside enclave
- Minimum attack surface

Intel SGX

Intel Software Guard Extension (SGX)

Capabilities

- Offering secure enclave
- Confidentiality, Integrity

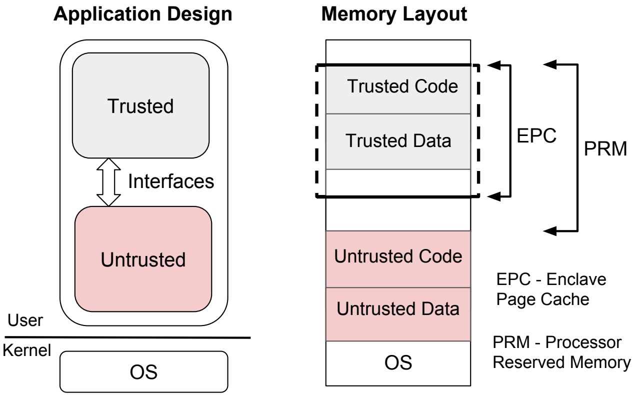
- New extensions to Intel x86/64 architecture
- Applications keep secret data and code inside enclave
- Minimum attack surface

Assumptions

- All privileged software (kernel, hypervisor) is malicious.

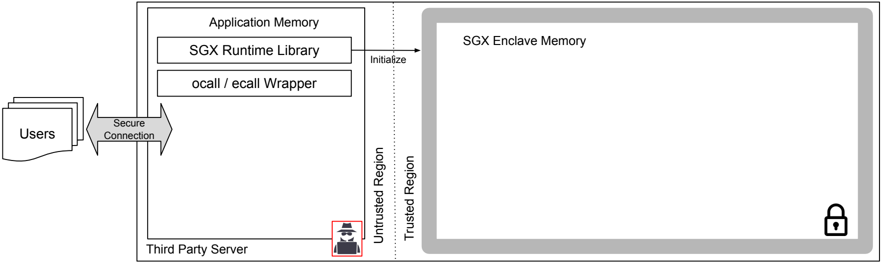
Intel SGX

Intel Software Guard Extension (SGX)



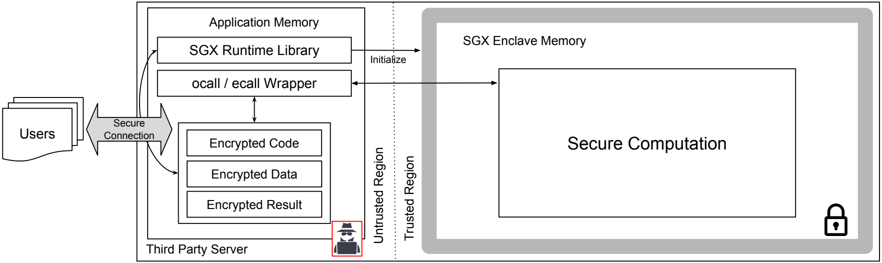
SGX-Rand: Secure Data Analytics

Architecture and the Threat Model



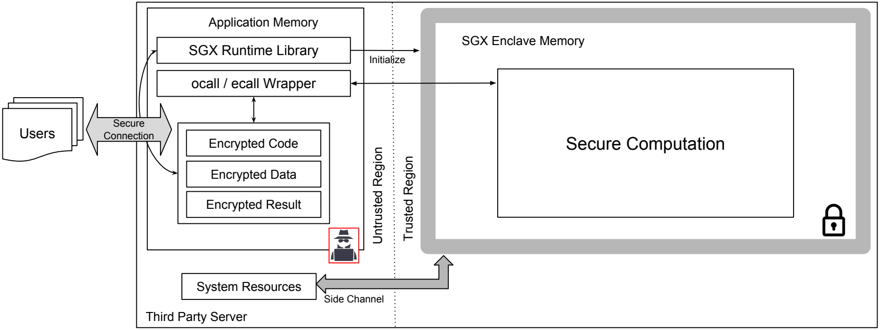
SGX-Rand: Secure Data Analytics

Architecture and the Threat Model



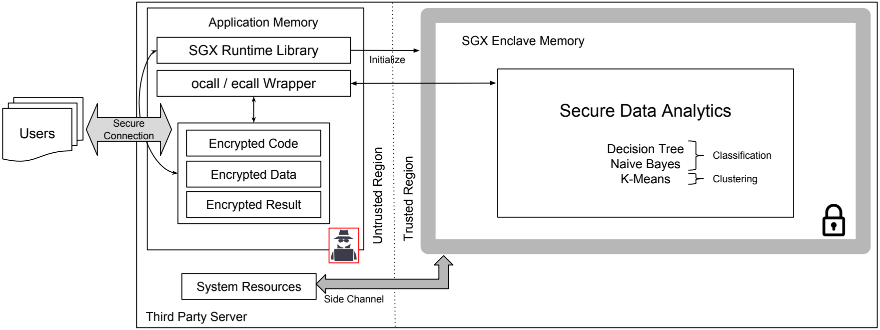
SGX-Rand: Secure Data Analytics

Architecture and the Threat Model



SGX-Rand: Secure Data Analytics

Architecture and the Threat Model



SGX-Rand: Secure Data Analytics

Objective: reduce leakage of private information during execution of analytic algorithms

What should we secure?

Model and data specific to each analytics

- **Trained parameters** such as posterior probability in Naive Bayes or internal node values in decision tree.
- **Learned structure** such as decision tree.

SGX-Rand: Secure Data Analytics

Objective: reduce leakage of private information during execution of analytic algorithms

What should we secure?

Model and data specific to each analytics

- **Trained parameters** such as posterior probability in Naive Bayes or internal node values in decision tree.
- **Learned structure** such as decision tree.

What need not be secure?

Public information in Analytics.

- Number of data instances.
- Total number of class labels.
- User-defined parameters such as number of clusters in K-Means clustering.

SGX-Rand

Key Idea

How to reduce computational cost?

- Create **dummy data instances**, equivalent to input data.
- **Randomly mix dummy data instances** with input test instances to form a contaminated test dataset.
- Evaluate with the **newly generated dataset**
- **Obliviously ignore** results from dummy data instances.

SGX-Rand
Key Idea

How to reduce computational cost?

- Create **dummy data instances**, equivalent to input data.
- **Randomly mix dummy data instances** with input test instances to form a contaminated test dataset.
- Evaluate with the **newly generated dataset**
- **Obliviously ignore** results from dummy data instances.

Attacker's perspective

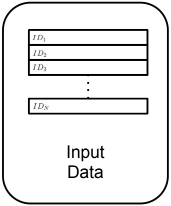
- Obtain traces from **side-channel** during evaluation.
- **Cannot distinguish traces** corresponding to **real vs. dummy** data instances.

SGX-Rand

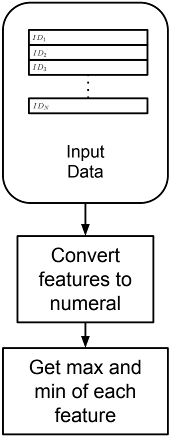
Two Key Primitives

1. **Dummy data generation**
2. **Data oblivious methods**
   - shuffling dummy data with input data to create contaminated dataset.
   - ignoring results of dummy data.

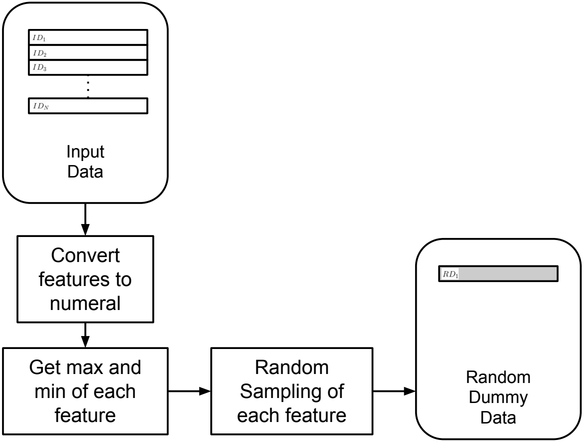
SGX-Rand: **Dummy Data Generation** Primitive



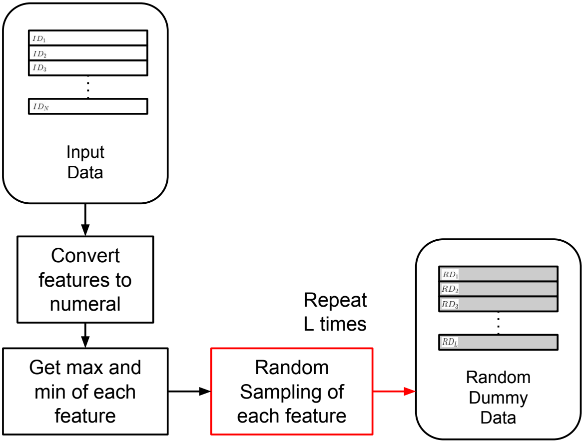
SGX-Rand: **Dummy Data Generation** Primitive



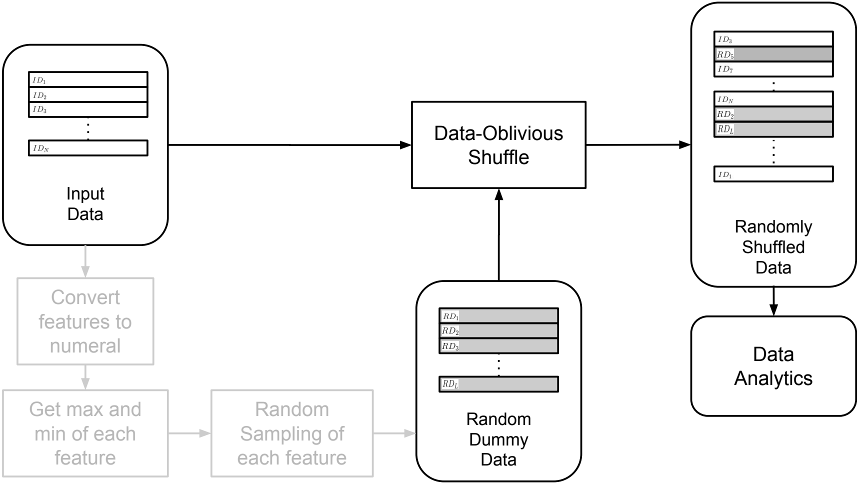
SGX-Rand: **Dummy Data Generation** Primitive



SGX-Rand: **Dummy Data Generation** Primitive



SGX-Rand: **Dummy Data Generation** Primitive



SGX-Rand: **Data Oblivious Comparison** Primitives

- Example function foo.
- Call function bar on different variables depending on condition.

```
int foo(int x, int y)
{
    ...
    if (x > y) {
          bar(x);
    } else {
          bar(y);
    }
    ...
}
```

SGX-Rand: **Data Oblivious Comparison** Primitives

- Example function `foo`
- Call function bar on different variables depending on condition
- Instead, we **define a temporary variable (t) to capture condition**
- Use it to pass appropriate value to bar, making the function **data oblivious**

```
int foo(int x, int y)
{
    ...
    int t;
    if (x > y) {
            t=1;
    } else {
            t=0;
    }
    bar(t*x + (1-t)*y);
    ...
}
```

SGX-Rand: Secure Data Analytics

How to Use SGX-Rand

- Train model using confidential training data.
- Provide input to third-party server
  - Trained model
  - Unknown data set for evaluation.
- Obtain output from third-party server.
- Discard dummy output.

SGX-Rand: Secure Data Analytics

How to Use SGX-Rand

- Train model using confidential training data.
- Provide input to third-party server
  - Trained model
  - Unknown data set for evaluation.
- Obtain output from third-party server.
- Discard dummy output.

Securing popular analytics during evaluation

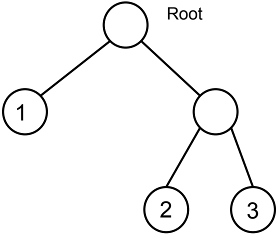
- Classification
  - **Decision Tree**
  - Naive Bayes
- Clustering
  - K-Means

Case-Study: Securing Decision Tree

Private Information	Public Information
Tree Structure Node values Data distribution	Number of input data instances. Number of class labels.

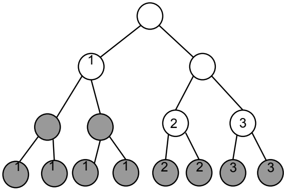
Case Study: Securing Decision Tree

- Train a decision tree offline.



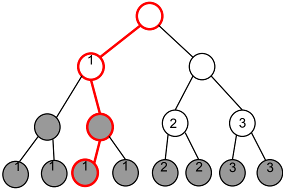
Case Study: Securing Decision Tree

- Train a decision tree offline.
- Balance the tree with **dummy nodes** such that leaf has same label as real-tree parent leaf. (extra step to secure tree structure).
- Load it into SGX enclave.



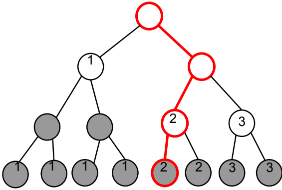
Case Study: Securing Decision Tree

- Train a decision tree offline.
- Balance the tree with **dummy nodes** such that leaf has same label as real-tree parent leaf. (extra step to secure tree structure).
- Load it into SGX enclave.
- Within enclave, tree evaluated on contaminated dataset.
- Obliviously ignore results from dummy data instances.



Case Study: Securing Decision Tree

- Train a decision tree offline.
- Balance the tree with **dummy nodes** such that leaf has same label as real-tree parent leaf. (extra step to secure tree structure).
- Load it into SGX enclave.
- Within enclave, tree evaluated on contaminated dataset.
- Obliviously ignore results from dummy data instances.
- Attacker cannot differentiate between 2 traces.



Empirical Evaluation

Objectives

- At minimum privacy, how much reduction in execution overhead can randomization achieve?
- How much is the trade-off between computation time and privacy-guarantee?

Empirical Evaluation

Objectives

- At minimum privacy, how much reduction in execution overhead can randomization achieve?
- How much is the trade-off between computation time and privacy-guarantee?

Dataset

Dataset	Size	Feature	Classes
Arrhythmia (A)	452	280	13
Defaulter (D)	30,000	24	2
ForestCover (F)	50,000	55	7
Synthetic (S)	50,000	71	7

Experiment Setup

Baseline

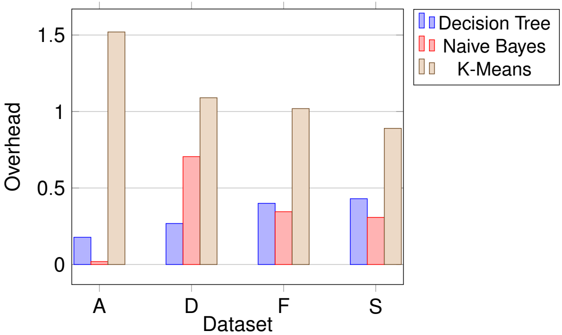
- **SGX**: Analytics within SGX enclave, *without* any defense.
- **SGX+Obliv**: Analytics within SGX enclave, *with* fully data-oblivious defense.
- **SGX+Rand**: Analytics within SGX enclave, *with* our randomization defense.

Measurement

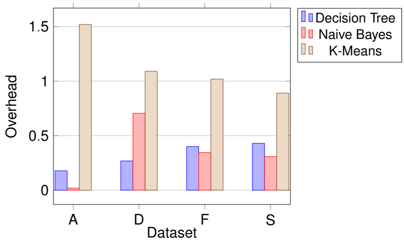
- Execution time $e = \frac{\mathit{time}(SGX+x)}{\mathit{time}(SGX)}$, where $x \in \{\mathrm{Obliv}, \mathrm{Rand}\}$
- Classification accuracy

Results

- With minimum privacy, $L = N$.
- Overhead $o = \frac{e(SGX+Rand)}{e(SGX+Obliv)}$
- Lower o is better, with $o < 1$ is gain in computation time.



Results



Good

Decision Tree and **Naive Bayes** have significantly large gain in execution time.

Not Good

K-Means for SGX+Rand has larger overhead than SGX+Obliv since every cluster is accessed even for dummy data instances.

Security Evaluation

Objective

- Is there noise in access traces?
- Are traces of input and dummy data instances indistinguishable?

Security Evaluation

Objective

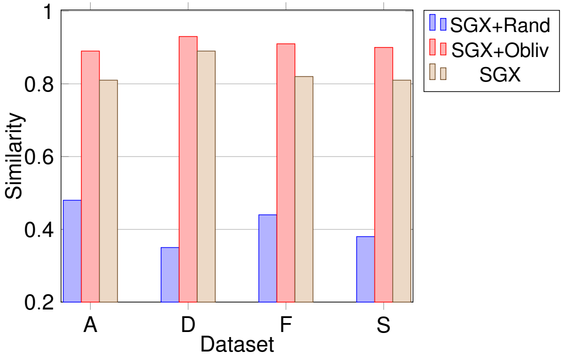
- Is there noise in access traces?
- Are traces of input and dummy data instances indistinguishable?

Methodology

- Using Intel PIN to obtain traces during execution.
- Measure **Levenshtein distance** between memory access sequence.
  - Greater distance is dissimilar traces, i.e., less similarity.
  - More similarity is similar traces.

Security Evaluation

Measuring similarity of execution traces obtained from two **independent datasets**



- Smaller value indicates greater randomization.

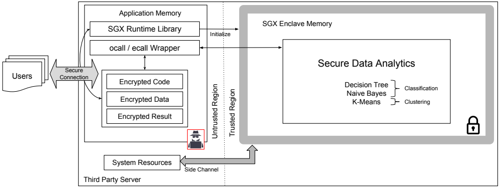
Conclusion

- **Randomization** is effective in reducing execution time overhead in data analytics compared to **fully data oblivious** solution.
- We quantitatively measure privacy and provide a choice in trade-off between privacy and efficiency.
- Our evaluation demonstrates both advantages and disadvantages in employing randomization.

Open-Source

Project code available at `https://github.com/utds3lab/secure-analytics-sgx`

Thank You



Thanks also to the AFOSR, NSA, and NSF for their sponsorship of this research.

References I

- Charu C Aggarwal and S Yu Philip, A general survey of privacy-preserving data mining models and algorithms, Privacy-preserving data mining, Springer, 2008, pp. 11–52.
- Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi, Oblivm: A programming framework for secure computation, Security and Privacy (SP), 2015 IEEE Symposium on, IEEE, 2015, pp. 359–376.
- Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar, Innovative instructions and software model for isolated execution, Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP) (Tel-Aviv, Israel), 2013, pp. 1–8.
- Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa, Oblivious multi-party machine learning on trusted processors., USENIX Security Symposium, 2016, pp. 619–636.
- Felix Schuster, Manuel Costa, CÃ©dric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich, Vc3: Trustworthy data analytics in the cloud using sgx, Proceedings of the 2015 IEEE Symposium on Security and Privacy (Washington, DC, USA), SP '15, IEEE Computer Society, 2015, pp. 38–54.
- Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena, Preventing page faults from telling your secrets, Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, ACM, 2016, pp. 317–328.
- Yuanzhong Xu, Weidong Cui, and Marcus Peinado, Controlled-channel attacks: Deterministic side channels for untrusted operating systems, Security and Privacy (SP), 2015 IEEE Symposium on, IEEE, 2015, pp. 640–656.