CERIAS

The Center for Education and Research in Information Assurance and Security

# An Authenticated Blocked-Based Graph Datastore

Servio Palacios, Ananth Grama, Byoungyoung Lee, Bharat Bhargava
Computer Science and CERIAS, Purdue University

## 1. Motivation

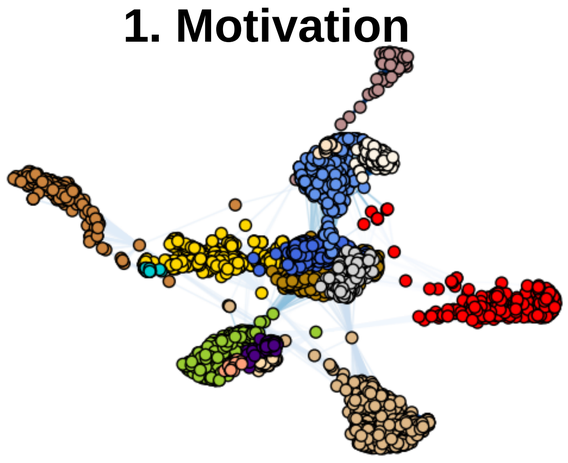


**Figure 1**. Authenticated-Blocked graph example. Python PoC.

- ❑ Many modern big-data applications work on top of graph-structured datasets. For instance, graph databases such as Neo4j, TitanDB, OrientDB, and others handle large-scale graphs that model biological interactions and social networks.
- ❑ Accessing network-structured data from/to secondary storage must ensure **access locality for graphs**.
- ❑ In multi-tenant environments, protecting sensitive information from unauthorized access and leakage is imperative. Unfortunately, although existing solutions for the graph processing model include secure communication and authentication (for independent graphs and authenticated entities); **there is no solution for multiple authenticated entities (data owners) sharing a diversity of data graphs**.

## 2. Threat Model

- ❑ We assume a probabilistic polynomial time adversary. **Leakage-free** is when the user cannot infer any information from the subgraph $\mathbf{G_i}$ that belongs to **G** but no $\mathbf{G_i}$.
- ❑ **Inference attack**: A user $\mathbf{U_i}$ with access to one or more subgraphs $\mathbf{G_i} \subset \mathbf{G}$, attempts to infer sensitive information from the signature and $\mathbf{G_i}$ (e.g., node or edge inference).
- ❑ **Data tampering attack**: An attacker tampers the structural order, the content, the relation between two nodes (edges), or a set of blocks $B \in \boldsymbol{\beta}$.
- ❑ Data-processing services involve **mutually distrustful parties**.

## 3. Our Contributions

- ❑ The definition of the practical and formal security model of authentication for the blocked graph.
- ❑ **Three new algorithms for**
  - block construction with high locality
  - disk layout for authenticated and blocked graphs
  - unique oblivious data structures and programming API for graph access utilizing trusted hardware **[4]**
- ❑ An open source implementation and evaluation of the proposed techniques (AB-Graph) **[2]**.

## 4. Authentication Graph and Layout Ordering (Proposed Solution)

Locality-Aware Block Formation → Authenticated-Blocked Formation → Authenticated Sets Layout Ordering

**Figure 2**. Authenticated Data Sets.

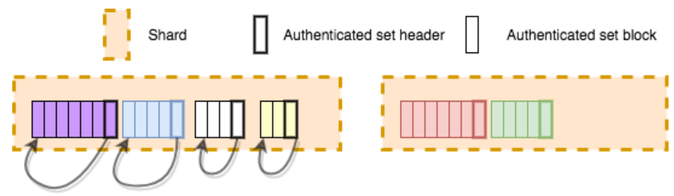



**Figure 3**. Disk Layout for **Fig. 2**.

## 5. Implementation

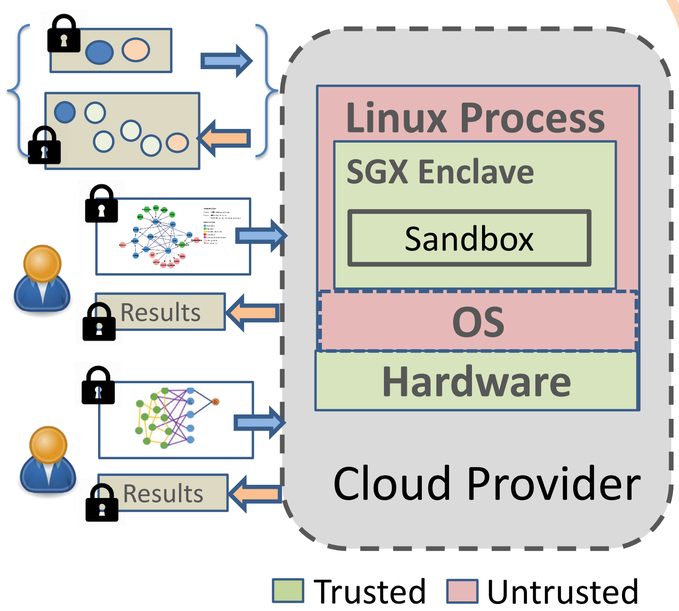
- ❑ Python and **NetworkX** based framework for studying blocked-graphs and networks. Blocked-based traversal algorithms.
- ❑ We use **METIS** partitioning framework to compare against our implementation (experimental results).
- ❑ JAVA API with Java Native Interface to call native applications (trusted hardware **[4]**)

```
// blocked graph simulation with authorization graph
blocked_graph = BlockedGraph()
blocked_graph.generate_authentication_model(1000, 100,
100, 4, GraphType.PREFERENTIAL_ATTACHMENT)
blocked_graph.draw_blocked_graph_manager()
blocked_graph.plot_values(3)
blocked_graph.blocked_single_source_shortest_path()

// independent graph generation including lattices
// and preferential attachment
graph = BlockedGraph()
G1 = graph.generate_preferential_attachment(200, 1)
G2 = graph.generate_lattice(Lattices.HEXAGONAL, 3, 3)
G3 = graph.generate_lattice(Lattices.HYPERCUBE, 3)
G4 = graph.generate_lattice(Lattices.TRIANGULAR, 3, 4)
G5 = graph.generate_lattice(Lattices.MESH, 8, 8)

graph.set_blocked_graph(G1)
partitions = graph.partition(G1)
graph.draw_blocked_graph(G1, partitions)
```

**Listing 1**. AB-Graph Python Driver.

## 6. Applications

1. **Secure Traffic Analytics** and Routing Algorithms for **ITS** Applications **[3]**.
2. **Authenticated and Private Network Sharing for enhanced Graph Analytics**. Our project provides novel approaches for Analyzing Biological Networks.
3. Authenticated and Private **Smart Farming**.
4. **Data Leakage** Detection on Social Networks.
5. Authenticated Peer-to-Peer Distributed Live Media Streaming with WebRTC **[1].**




## 7. References

1. Palacios, S., Santos, V., Barsallo, E., & Bhargava, B. (2018). **OwnStream**: A Peer-to-Peer Distributed Live Media Streaming with WebRTC. *Multimedia Tools and Applications* (Submitted).
2. Palacios, S. (2018). AB-Graph. Retrieved from https://github.com/maverick-zhn/block-graph
3. Palacios, S., & Bhargava, B. "Secure Traffic Analytics and Routing Algorithms for ITS Applications", IEEE CLOUD 2018 (Submitted).
4. Costan, V., & Devadas, S. (2016). Intel SGX Explained. *Cryptology ePrint Archive, Report 2016/086*.

PURDUE UNIVERSITY Discovery Park

CERIAS