New Systems and Algorithms for Preserving Big-data Privacy in Clouds

Abstract:
Driven by the rapidly increasing amount of data, many application vendors (e.g., Uber) store data on clouds. Meanwhile, application vendors and third parties often write self-defined queries (e.g., map/reduce) to process the data. This has caused severe privacy problems. One problem is ``computation leakage'': third-parties can easily acquire sensitive fields in data records (e.g., a credit card in an Uber transaction) using the self-defined queries. Existing techniques (e.g., differential privacy) add random noises to hide individual sensitive data, but due to the lack of precisely tracking how sensitive data flows to a query result, these techniques often add excessive noise and make query results useless. Another problem is ``cloud privacy'': cloud providers can compromise on external attacks and easily steal the data being queried.

This proposal takes a holistic methodology to tackle both the two problems with three objectives. First, to prevent ``computation leakage'', we will build Kakute, the first Data Flow Tracking (DFT) system for big-data queries. Kakute provides easy-to-use APIs for application vendors to tag sensitive fields in data records, and it automatically tracks and prevents these fields propagating to query results. An open challenge in existing DFT systems is that propagating tags in data-intensive queries is too slow. For instance, a notable DFT system incurred a 128X slowdown compared to native, insecure queries in our study. By leveraging subtle efficiency natures of big-data queries, we will create two fast tag propagation techniques called Reference Propagation and Tag Sharing. Our Kakute preliminary prototype presented in [ACSAC '17] shows that it incurs merely 32.3% overhead compared to native queries.

Second, we will develop Fine-grained Differential Privacy (FDP), a novel differential privacy technique and its new algorithms. Compared to DFT, differential privacy has complementary strength because it enables the aggregation of sensitive data while hiding individual privacy. However, existing differential techniques are too coarse-grained: they often add excessive noise to all fields of all data records and make results useless due to the lack of precisely tracking data flow. By leveraging Kakute, our new FDP technique will only add noise to sensitive fields, preserving strong differential privacy for sensitive data and good usability for most results.

Third, to tackle the ``cloud privacy'' problem, we will leverage the Intel SGX hardware to build the first just-in-time, privacy-preserving Java compiler for unmodified big-data frameworks (e.g., Spark). Existing SGX-based systems for big-data frameworks have two major challenges: they need to rewrite the Java big-data queries into SGX-compatible C++, or their trusted computing base is too large (e.g., the entire JVM). Our new compiler will run only the self-defined Java queries in SGX with a thin, verified just-in-time translator, and we will create fast SGX runtime techniques to achieve reasonable overhead compared to native queries.

The success of this proposal will effectively preserve big-data privacy in clouds, potentially benefiting every computer user, software vendor, organization, and government.

Long term impact:
The big-data and cloud computing trends bring fascinating opportunities to all parties, including data providers (e.g., application vendors such as Uber), cloud providers (e.g., Amazon), and computation providers (e.g., application vendors and their third-party contractors). Unfortunately, despite decades of effort, data leakage remains one of the most severe threats in clouds. In a data provider's perspective, both computation providers (e.g., the 2017 iCloud account leakage caused by third-parties) and cloud providers (e.g., the 2013 Yahoo Cloud compromise and the 2014 J.P. Morgan account leakage) have caused severe breaches on data privacy and huge financial loss.

Even in a private cloud (i.e., the data provider is the cloud provider), sensitive data such as credit cards, user identities, and healthcare records can easily be leaked in the self-defined queries written by computation providers (e.g., the 2017 iCloud leakage). Real-world breaches include directly acquiring particular user's identities from the query results and sending sensitive data outside the cloud through IO functions in the queries.

In the short term, we plan to accomplish both the Objective 1 and Objective 2 of this proposal, which can effectively prevent computation leakage in private clouds. Objective 1 proposes Kakute, the first Data Flow Tracking (DFT) system for big data queries. Although numerous DFT systems have been built to prevent accessing sensitive data in mobile phones and server programs, no DFT system exists for big-data frameworks. A key reason is that DFT's tag propagation incurs prohibitive overhead on data-intensive queries (e.g., we found a notable DFT system running with the WordCount query incurred a 128X slowdown compared to its native query).

Our key insight to address the DFT efficiency challenge is that most fields of a record often have the same tags. Leveraging this insight, we present two new techniques, Reference Propagation and Tag Sharing. To achieve a robust DFT architecture for distributed big-data frameworks (e.g., Spark), Kakute completely captures the frameworks' inter-computer data flows (i.e., shuffles). We have implemented a Kakute prototype and integrated it with Spark. Kakute carries built-in checkers for four security and reliability problems: sensitive data leakage, data provenance, programming bugs, and performance bugs. Kakute not only incurs a moderate performance overhead of 32.3% compared to native queries, but it also effectively detects 13 real-world security and performance bugs. These promising preliminary results have been presented in [ACSAC '17] and [TPDS '17].

In the security community, DFT and differential privacy are complementary: DFT enforces mandatory access control on sensitive data, but it may cause some critical query results to be missing; differential privacy allows the aggregation results of sensitive fields while hiding individual privacy, but due to the lack of precisely tracking data flow, it may suffer from excessively added noise and inaccurate (useless) query results. Therefore, the Objective 2 of this proposal takes the first step to integrate DFT and differential privacy and to develop a novel Fine-grained Differential Privacy (FDP) technique, reaching the best of the both worlds.

In the intermediate term, we will tackle the ``cloud privacy'' problem in a public cloud (e.g., Amazon) by accomplishing Objective 3. Recently, Intel SGX has attracted high attention on protecting the privacy of data being queried, because it enforces hardware protection for the confidentiality of data and code even if the cloud compromises. Meanwhile, SGX is good fit for big-data queries because these queries are data-intensive in userspace and they hardly invoke system calls. Existing SGX-based systems for big-data have two major challenges: they need to rewrite the Java big-data queries into SGX-compatible C++, or their trusted computing base is too large. To tackle these two challenges, Objective 3 will build the first just-in-time, privacy-preserving Java compiler for unmodified big-data frameworks. Our new compiler will run only the self-defined Java queries in SGX with a thin, verified just-in-time translator, and the rest of the JVM is outside SGX without affecting the privacy of the data being queried. Therefore, our compiler will be the first work to support fast, unmodified Java big-data queries with minimal trusted computing base.

In the long term, by integrating the outcomes of all the three objectives in this proposal and extensively applying them to real-world big-data frameworks, we will help data providers enforce comprehensive privacy against both computation providers and cloud providers. This will benefits almost all individuals, software vendors, organizations, and governments. For instance, many HK finance entrepreneurs demand strong privacy for their data deployed in clouds. Moreover, we envision that the outcomes of this proposal will broadly promote other new security techniques, including strengthening the integrity and availability of real-world software.

Objectives:

1. [To build the first Data Flow Tracking (DFT) system for private clouds]
We will create Kakute, a fast DFT system that can track and prevent sensitive data leakage in self-defined big-data queries. We will make Kakute support diverse big-data queries on large, popular datasets, and we aim will design Kakute to incur reasonable performance overhead compared to the native, insecure queries.

2. [To develop a novel Fine-grained Differential Privacy (FDP) technique for private clouds]
We will leverage Kakute to develop FDP and its new algorithm, which will only add noise to sensitive data fields, preserving strong differential privacy for sensitive data and good accuracy for most query results. We will extensively study FDP's accuracy improvements on both sensitive and insensitive data compared to existing differential privacy techniques.

3. [To create the first compiler for big-data privacy in public clouds]
Our compiler will support unmodified big-data frameworks by creating a thin translator to automatically convert Java bytecode to SGX-compatible code. We will verify the translator with state-of-the-art verification techniques so that our compiler will have a minimum trusted computing base (i.e., SGX only). We will evaluate whether our compiler can protect the privacy of data being queried against real, privileged attacks. We will evaluate the compiler's performance overhead.