CS 6431: Security and Privacy Technologies
Homework #3

Due: April 11, 2018

Instructors: Tom Ristenpart and Vitaly Shmatikov

Problem 1

Suppose we wish to design countermeasures for the attacks discussed in the Yarom-Faulkner and Meltdown papers.

1. Blinding is an oft-mentioned mechanism for protecting cryptographic algorithms from side-channels. Does using blinding of exponentiation with a fixed random multiplier (instead of randomized for each exponentiation) suffice to prevent the Flush+Reload attack against RSA? Explain why or why not.

2. Explain the simplest configuration-based mechanism (meaning, one need not change software or hardware) by which one can prevent the Flush+Reload attack. Explain the Prime+Probe protocol for inferring cache usage. Does the countermeasure for Flush+Reload prevent a Prime+Probe attack?

3. The Meltdown paper lists a number of potential countermeasures. Explain the approach actually taken by some major OS.

Problem 2

How would you implement a program to detect, with a minimum of false positives, that a website is using tracking cookies? canvas fingerprinting? battery API fingerprinting?

Problem 3a

Prove that the Laplace mechanism satisfies differential privacy.

Problem 3b

What is the exponential mechanism? Show that the Laplace mechanism is a special case of the exponential mechanism.

Problem 4

D is the dataset containing annual salaries of all Cornell employees. $bsdcount(D)$ returns the number of entries in D that are greater than $\$1,000,000$; $max(D)$ returns the maximum salary in the dataset.

Let San be the standard Laplacian mechanism for ϵ-differential privacy. Given any function f, San generates random ξ from the Laplacian distribution with variance that depends on the sensitivity of function f and the privacy parameter ϵ, and returns $f(D) + \xi$.

Problem 4a

What is the sensitivity of $bsdcount$ and max? State all assumptions you needed to calculate the answers.

Problem 4b

For the "same level of privacy," which function requires "more noise" to be added? Given a function, how does the "noise distribution change" in order to achieve "higher level of privacy"? Your answers should make precise all terms in quotes.

Problem 4c

Let $\epsilon = 0.001$, let $p = 0.01$ be your prior probability that Big Red Bear makes $\$10,000$ a year, and let p' be the probability after learning the differentially private values of $bsdcount$ and max. What is the maximum value of p'?

Problem 5

Order-preserving encryption (OPE) encrypts plaintexts with a shared secret key so that for any key K and plaintexts $m_1 < m_2$ it is the case that $\mathcal{E}_K(m_1) < \mathcal{E}_K(m_2)$. The first formal treatment of OPE is due to Boldyreva, Chenette, Lee, and O'Neill (BCLO).[1]) They give a scheme that has been deployed in industry to help secure

[1] `http://www.cc.gatech.edu/~aboldyre/papers/bclo.pdf`

data outsourced to cloud services.

An implementation of the BCLO scheme in Python is available[2]. Your goal in this assignment is to assess the security of the BCLO scheme empirically.

- Find a public dataset of names, birthdates, or other data that would be a reasonable stand-in for information that would be sensitive. The data should also be order-relevant, i.e., be plausibly encrypted using OPE to allow servers to perform sorting, range queries, etc. The dataset should not be "dense", i.e., a reasonable fraction of the possible plaintext values should be absent. (OPE is trivally insecure when used on dense data.)

- Implement a simulation using the BCLO cipher. Choose domain and range sizes appropriate for the data, and following any recommendations you find (see the original BCLO paper as well as the follow-up by Boldyreva, Lee, and O'Neill (BLO)[3]). Your simulation should be able to encrypt all the data under a fresh key. The resulting ciphertexts will be used to perform attacks.

- Implement an attack based on the plaintext leakage. You might look at the follow-up BLO paper (see proof of Theorem 4.2 in Appendix F) as well as our recent paper on this topic[4]). Your attack should at least recover non-empty prefixes of the plaintext (equivalently, determine a subset of the domain within which a plaintext falls). Should your attack recovers a variable number of bits depending on the target plaintext, provide a graph for which the x-axis is the plaintexts (from smallest to largest) and the y-axis is the number of bits of the plaintext recovered by your attack. In either case, give summary statistics, e.g., the (average) number of bits recovered, the error rate (an error being an incorrect plaintext prefix guess). Discuss the implications for the type of data in your dataset.

[2] `https://pypi.python.org/pypi/pyope/`
[3] `http://www.cc.gatech.edu/~aboldyre/papers/operev.pdf`
[4] `http://eprint.iacr.org/2016/895.pdf`