Weakly Randomized Encryption

And the Strength of Weak Randomization

David Pouliot, Scott Griffy, **Charles V. Wright**
Portland State University

This work to appear in DSN 2019

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under Contract No. N66001-15-C-4070. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or SSC Pacific.

“Executive” Summary

Weakly Randomized Encryption

- A safer upgrade to deterministic encryption
- Secure against most common “snapshot” attacks
- Easy to deploy
- ACID properties*
- Low overhead

Research Questions

1. What security can we achieve if **easy deployability** is a **hard constraint**?

2. Are there PPE-like constructions that provide **any meaningful security** against inference???

RELATED WORK

Property-Preserving Encryption (PPE)

- Deterministic and Efficiently Searchable Encryption [BBO07,ABO07]
- CryptDB [PRZB11]
- Microsoft SQL Server "Always Encrypted"

Parallel Invention

- [LP18] Lacharité and Paterson. Frequency Smoothing Encryption: Preventing snapshot attacks on deterministically encrypted data.
  - https://eprint.iacr.org/2017/1068
  - Most similar to our *Proportional Salt Allocation*

Inference Attacks

1. Offline inference (the “snapshot” model)
   – IKK12, NKW15
   – CGPR15, GSBNR17

2. Online inference
   – KKNO16, LMP18
   – GLMP18, GLMP19

3. Inference from database/OS artifacts
   – GRS17

Defense Against Inference Attacks

1. Offline inference:
   – IKK12, NKW15
   – CGPR15, GSBNR17

Focus of this work
- Defend against the **most common attacks** (i.e. snapshots / SQL injection)
- Maximize backwards compatibility
- What security & performance can we get?

2. Online inference
   – KKNO16, LMP18
   – GLMP18, GLMP19

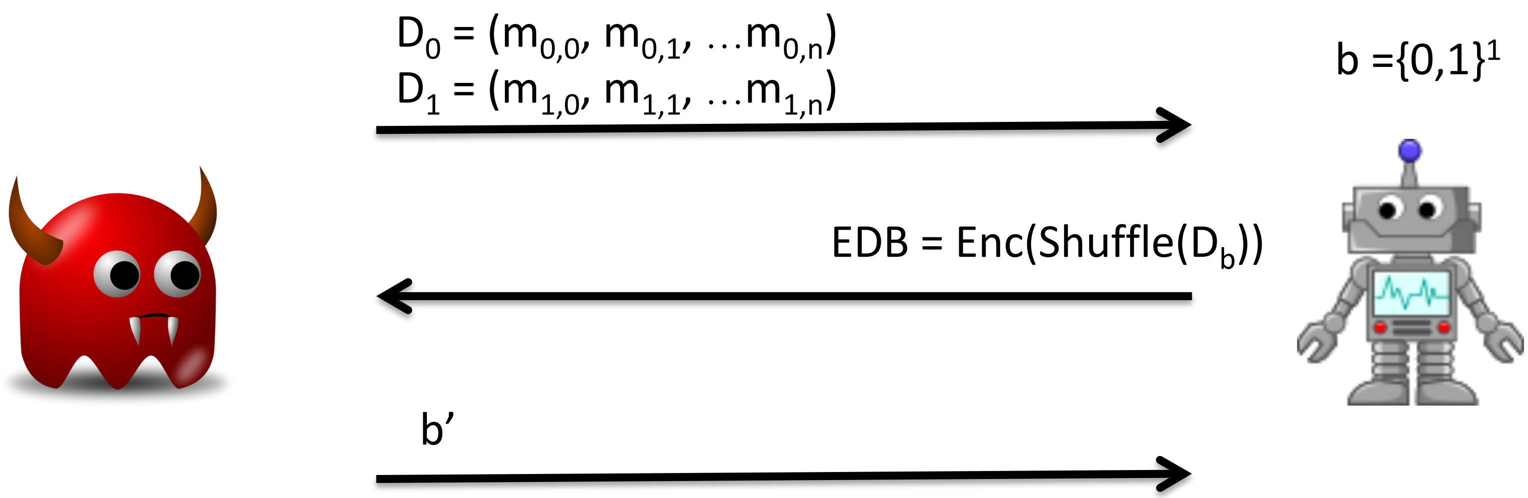
Harder problem / Future work
- Attacks apply to stronger constructions too

3. Inference from database/OS artifacts
   – GRS17

Mostly engineering??
- Not worth trying to fix this if you can't also defend #1

SECURITY GOALS

Security Game



Adversary wins iff b’ == b

Statistical Distance and Security

Definition 3 (Statistical Distance). *The* statistical distance Δ *between two random variables* X, Y *over a common domain* ω *is defined as:*

$$\Delta(X,Y) = \frac{1}{2} \sum_{\alpha \in \omega} \left| Pr(X = \alpha) - Pr(Y = \alpha) \right|$$

Definition 4 (Distinguishing Two Distributions). *Let* P_0 *and* P_1 *be probability distributions on a finite set R. Then for every adversary* $\mathcal{A}$*, we have the distinguishing advantage of* $\mathcal{A}$ *between* P_0 *and* P_1*,*

$$Pr[\mathsf{Dist}_{\mathcal{A}}(P_0, P_1)] \leq \Delta(P_0, P_1)$$

CONSTRUCTIONS

Efficiently Searchable Encryption [BBO07, ABO07]

Plain Table

Row ID	Animal
1	Dog
2	Horse
3	Cat
4	Cat
5	Dog
6	Horse
7	Dog
8	Dog
9	Cat

Efficiently Searchable Encryption [BBO07, ABO07]

Plain Table

Row ID	Animal
1	Dog
2	Horse
3	Cat
4	Cat
5	Dog
6	Horse
7	Dog
8	Dog
9	Cat

Encrypted Table

Row ID	Tag	Cipher
1	F(Dog)	E(Dog)
2	F(Horse)	E(Horse)
3	F(Cat)	E(Cat)
4	F(Cat)	E(Cat)
5	F(Dog)	E(Dog)
6	F(Horse)	E(Horse)
7	F(Dog)	E(Dog)
8	F(Dog)	E(Dog)
9	F(Cat)	E(Cat)

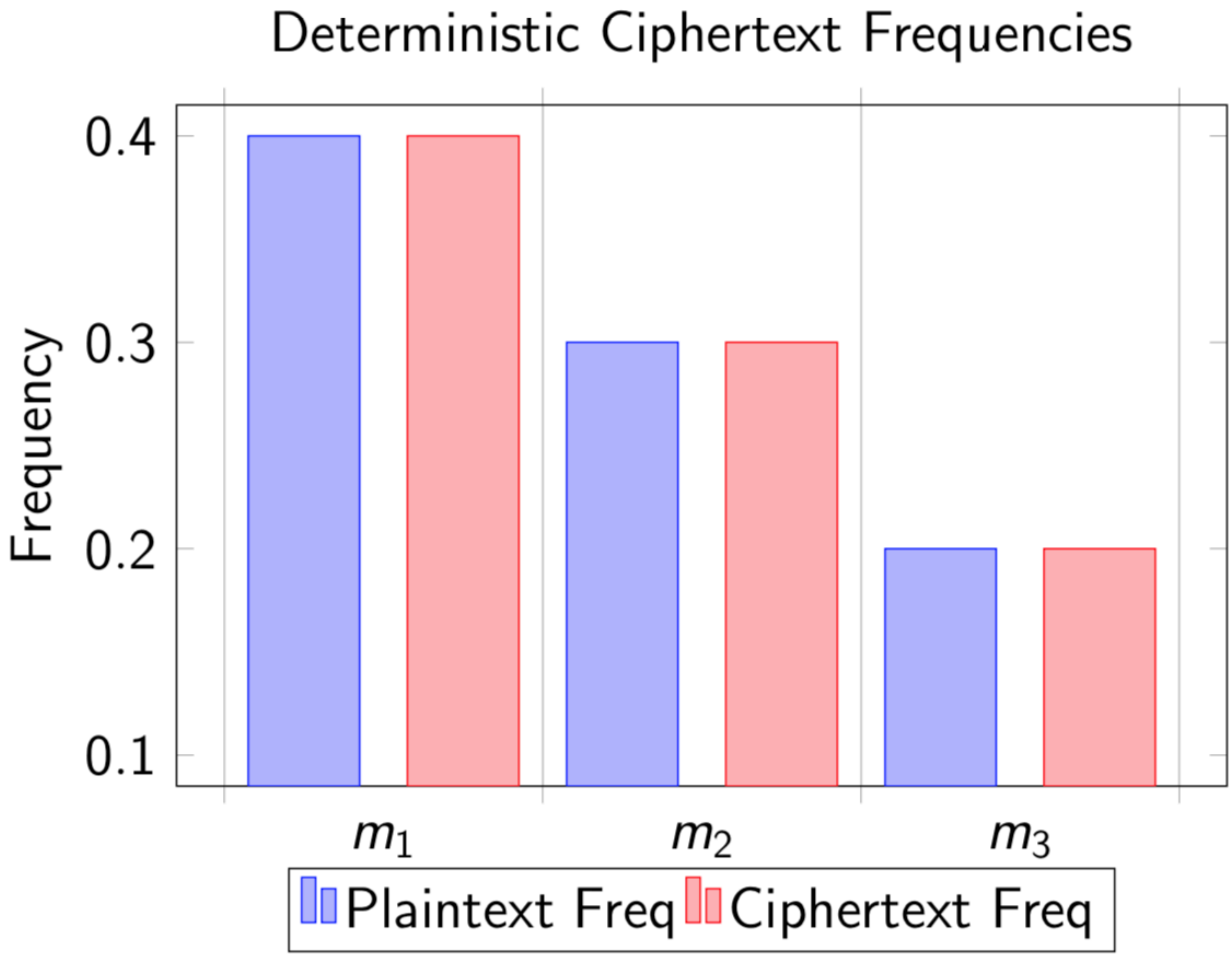
Efficiently Searchable Encryption [BBO07, ABO07]

Plain Table

Row ID	Animal
1	Dog
2	Horse
3	Cat
4	Cat
5	Dog
6	Horse
7	Dog
8	Dog
9	Cat

Encrypted Table

Row ID	Tag	Cipher
1	eb3f	653c
2	137a	bb21
3	6f20	e0f3
4	6f20	9201
5	eb3f	bbcf
6	137a	d830
7	eb3f	c971
8	eb3f	ee26
9	6f20	7a0b



Deterministic Ciphertext Frequencies
Frequency
0.4
0.3
0.2
0.1
m_1
m_2
m_3
Plaintext Freq
Ciphertext Freq

Randomizing Deterministic Encryption

- Too random → Not useful ☹

- Too predictable → Not secure ☹

- Just enough randomness → ☺

To Encrypt

1. Choose random, **low entropy** salt s
2. Tag $t = F_{k1}(s \mid\mid m)$
3. (Randomized) ciphertext $c = E_{k2}(m)$

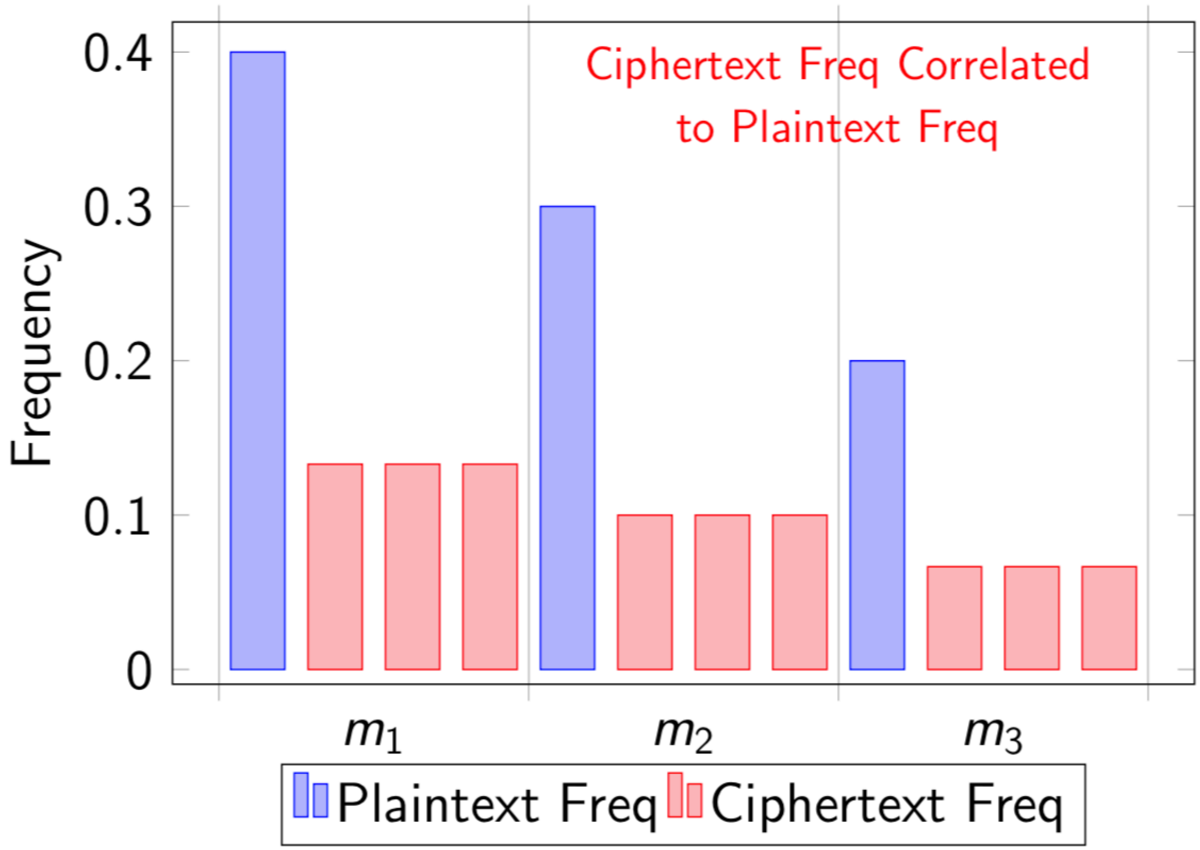
To Search

1. Generate all possible tags for msg m
   – For each salt s_i:
     Let $t_i = F_{k1}(s_i \mid\mid m)$

2. Encrypt query
   – SELECT …
     FROM enc_table
     WHERE tag in (t_1, t_2, …, t_n);

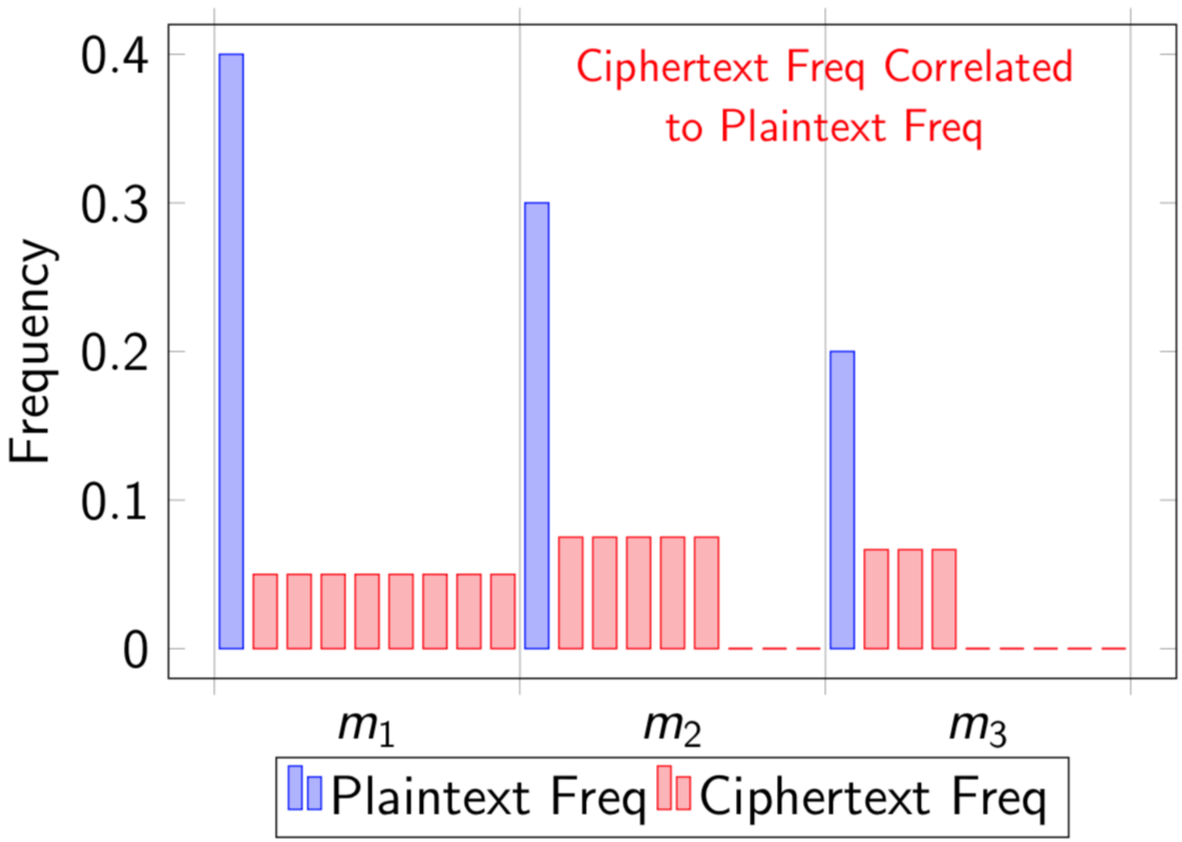
Strawman Construction: Fixed Salts

- Choose salt uniformly from [1..N]
  - e.g. N = 3



Proportional Salt Allocation

- Allocate salts in proportion to frequency



Frequencies are closer to Uniform

Some *aliasing* effects

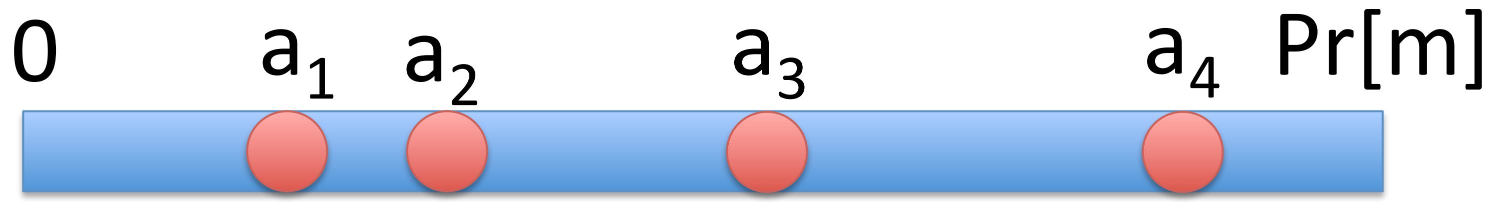
Poisson Salt Allocation

Question:
How to allocate message *m*'s probability mass to the ciphertexts?

Poisson Salt Allocation

Idea:

Sample points from a Poisson process w rate param λ

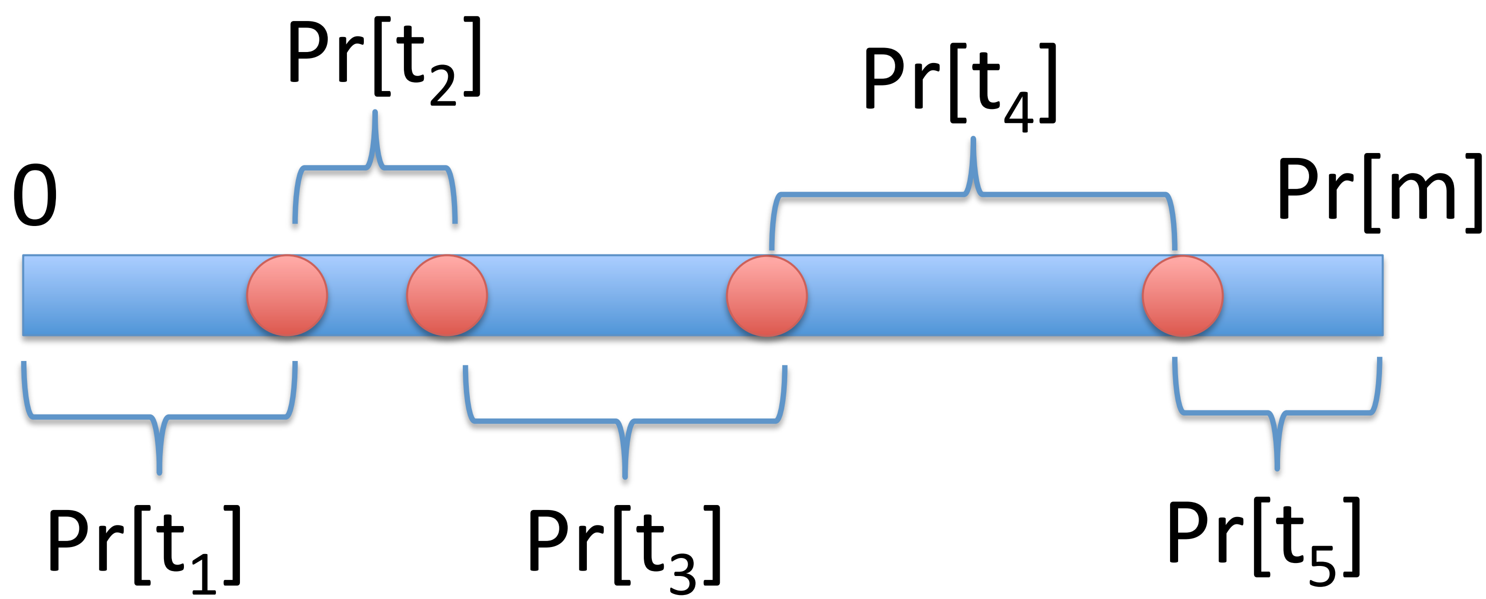


Poisson Salt Allocation

Idea:

Sample points from a Poisson process w rate param λ

Distances between points (“inter-arrivals”) give tag frequencies



Poisson Security

- Ciphertext freqs are identically distributed!
  - $\Pr[t_j] \sim \text{Exponential}(\lambda)$ for all j

Poisson Security

- Ciphertext freqs are identically distributed!
  - $Pr[t_j] \sim Exponential(\lambda)$ for all j

- Identical distribution → No statistical distance

Poisson Security

- Ciphertext freqs are identically distributed!
  - $Pr[t_j] \sim Exponential(\lambda)$ for all j

- Identical distribution → No statistical distance

- No statistical distance → No guessing advantage

Poisson Security

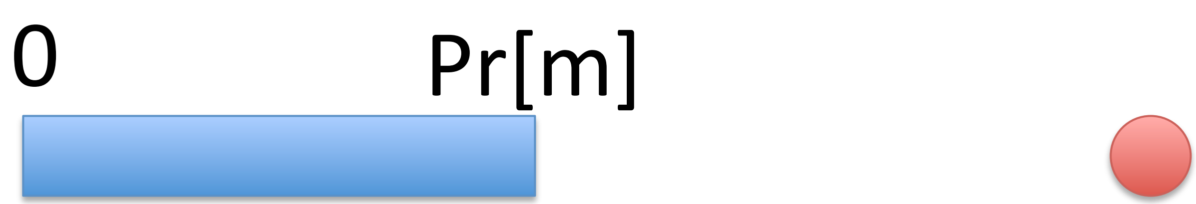
- Ciphertext freqs are identically distributed!
  - Pr[t_j] ~ Exponential(λ) for all j

> **Whoops… Not quite true..**
>
> **They are *almost identically* distributed. :-**

- Identical distributi … e
- No statistical distance → No guessing advantage

Something Fishy About Poisson

Problem:
What if there are no arrivals in the interval [0, Pr[m]] ???

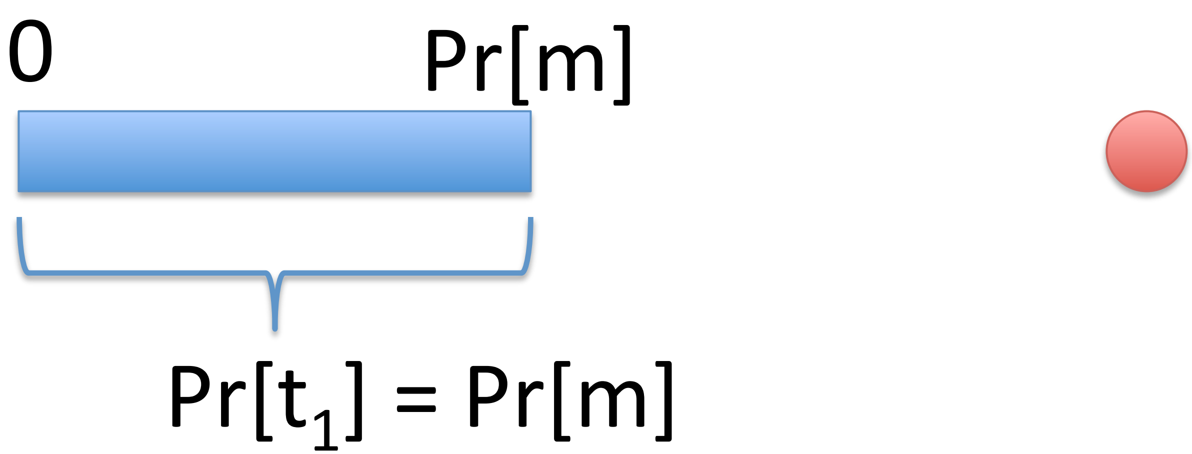


Something Fishy About Poisson

Problem:

What if there are no arrivals in the interval [0, Pr[m]] ???

No choice but to give all of *m*'s probability mass to a single t*ag*



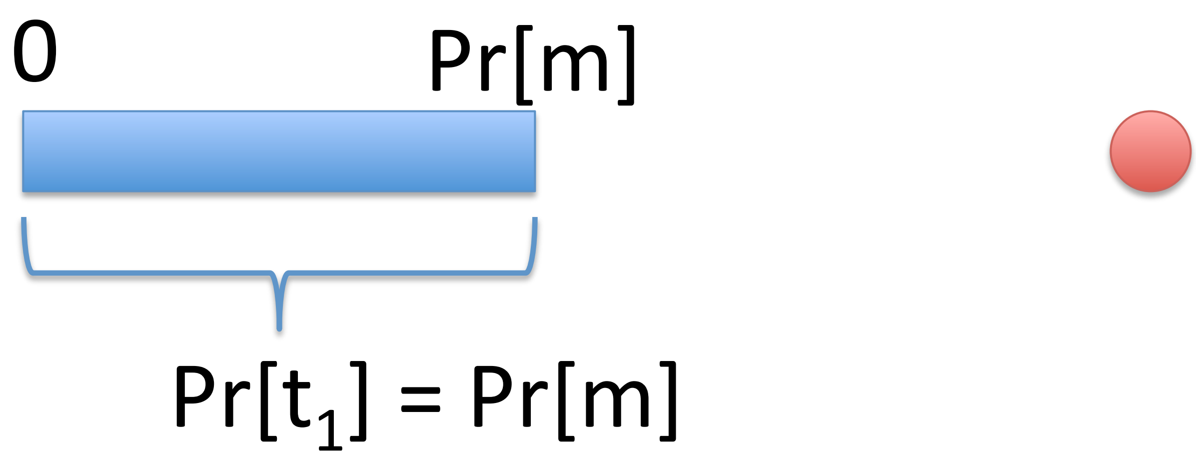
Something Fishy About Poisson

Problem:

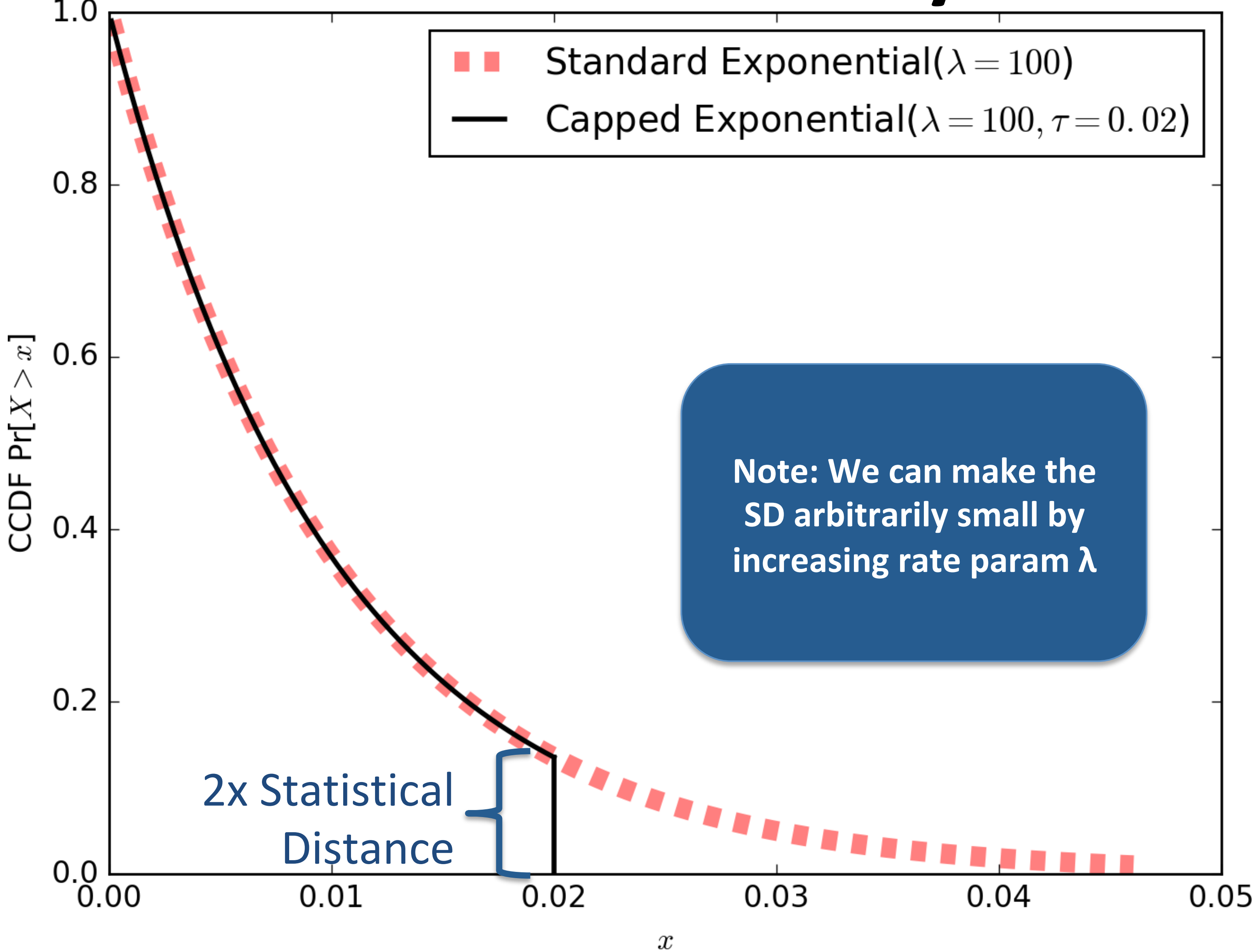
What if there are no arrivals in the interval [0, Pr[m]] ???

No choice but to give all of *m*'s probability mass to a single tag

Not really a true Exponential. Can the Adv now distinguish?



Poisson: Security

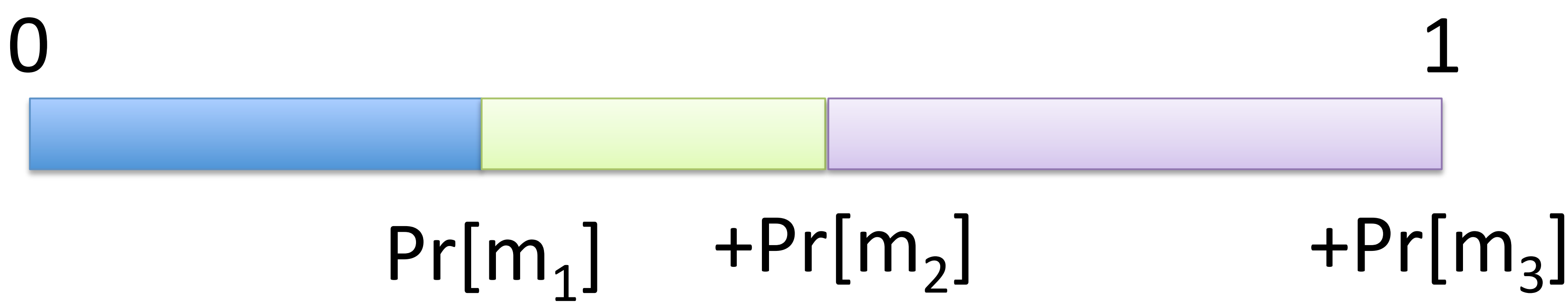


Poisson: One More Problem

- Lacharite-Paterson attack: What if Adv looks at more than one ciphertext?
  - Goal: Find a set of search tags $t_1, t_2, \ldots, t_n$ s.t.
    - $Pr[m] = \sum_j Pr[t_j]$
    - These records are *probably* (???) the encryptions of m
  - Difficulty: Bin packing problem :-\
    - On the bright side:
      - Might be a hard (NP) instance
      - Solution might (tend to) select the wrong records

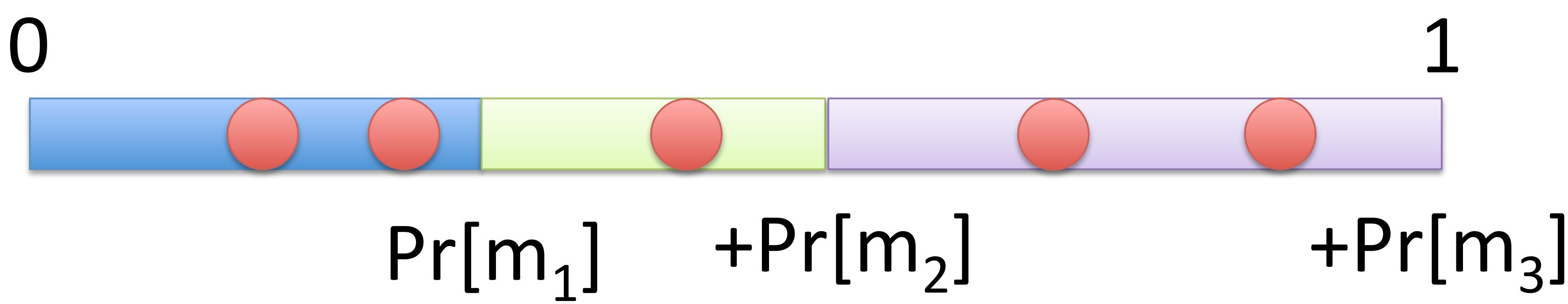
Bucketized Poisson

Lay out plaintext freqs on the number line [0..1]



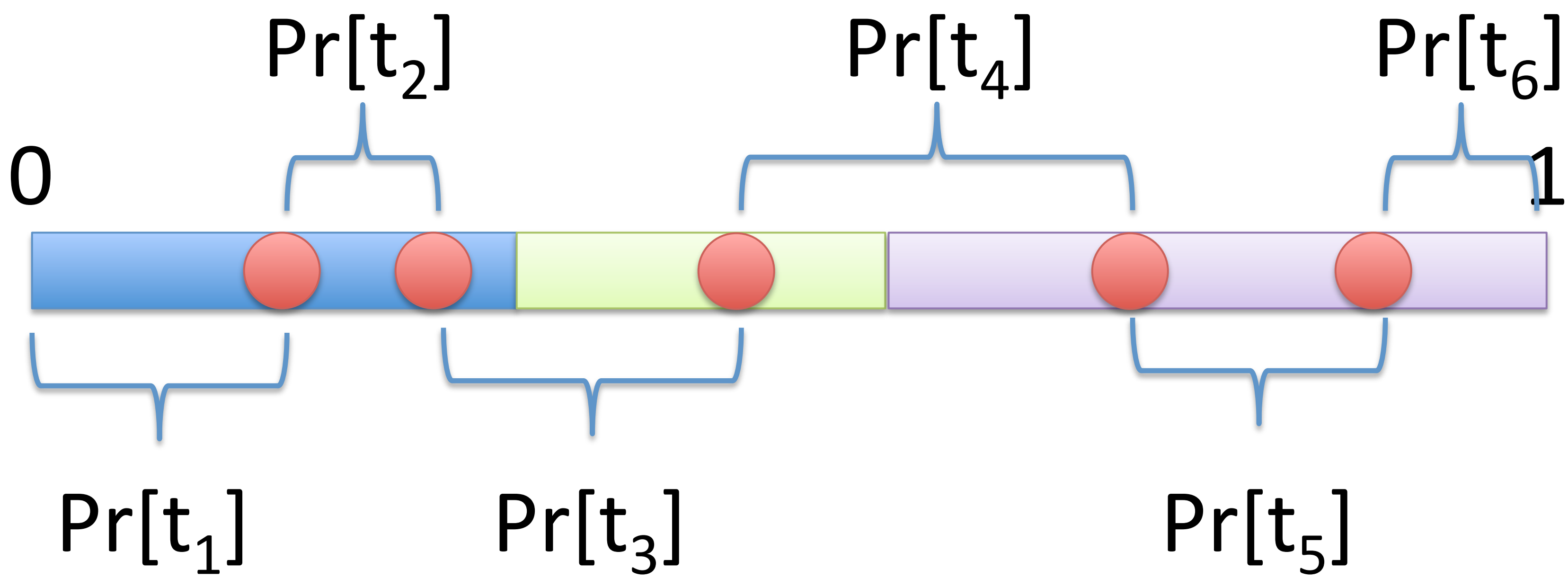
Bucketized Poisson

Lay out plaintext freqs on the number line [0..1]
Sample from the Poisson process



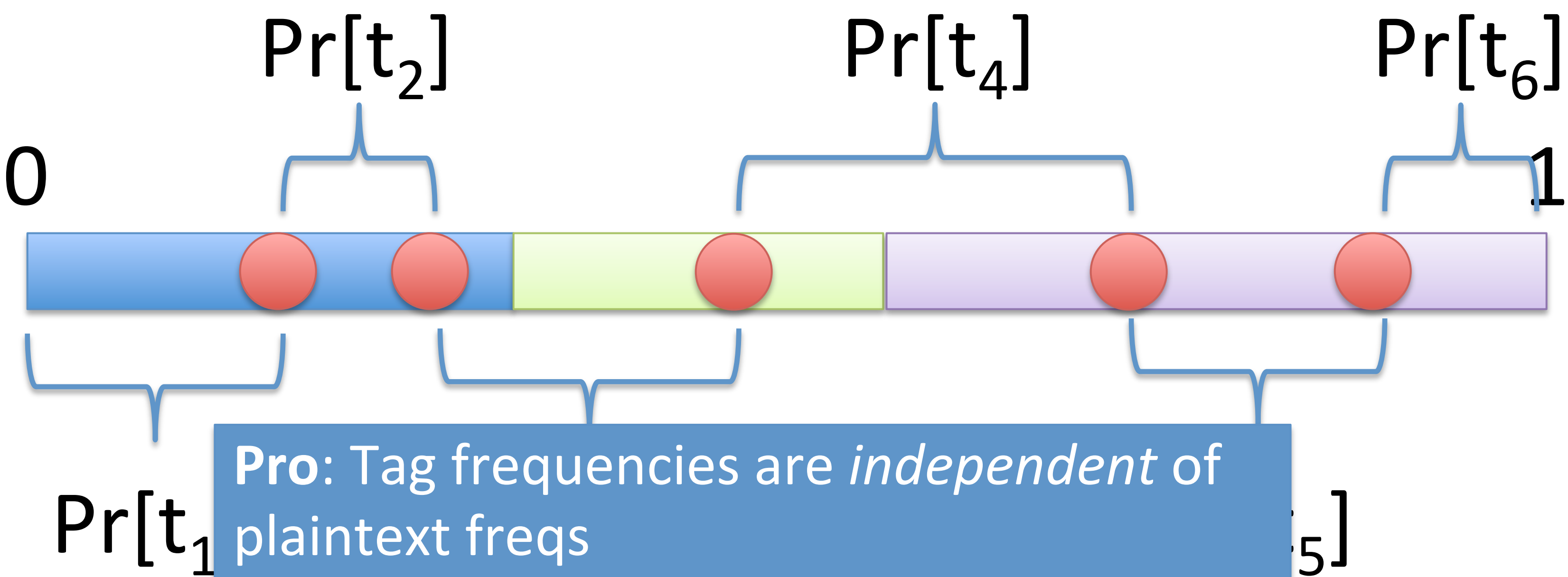
Bucketized Poisson

Lay out plaintext freqs on the number line [0..1]
Sample from the Poisson process
Use inter-arrivals to fix a set of search tags for **all** plaintexts to share



Bucketized Poisson

Lay out plaintext freqs on the number line [0..1]
Sample from the Poisson process
Use inter-arrivals to fix a set of search tags for **all** plaintexts to share



Pro: Tag frequencies are *independent* of plaintext freqs

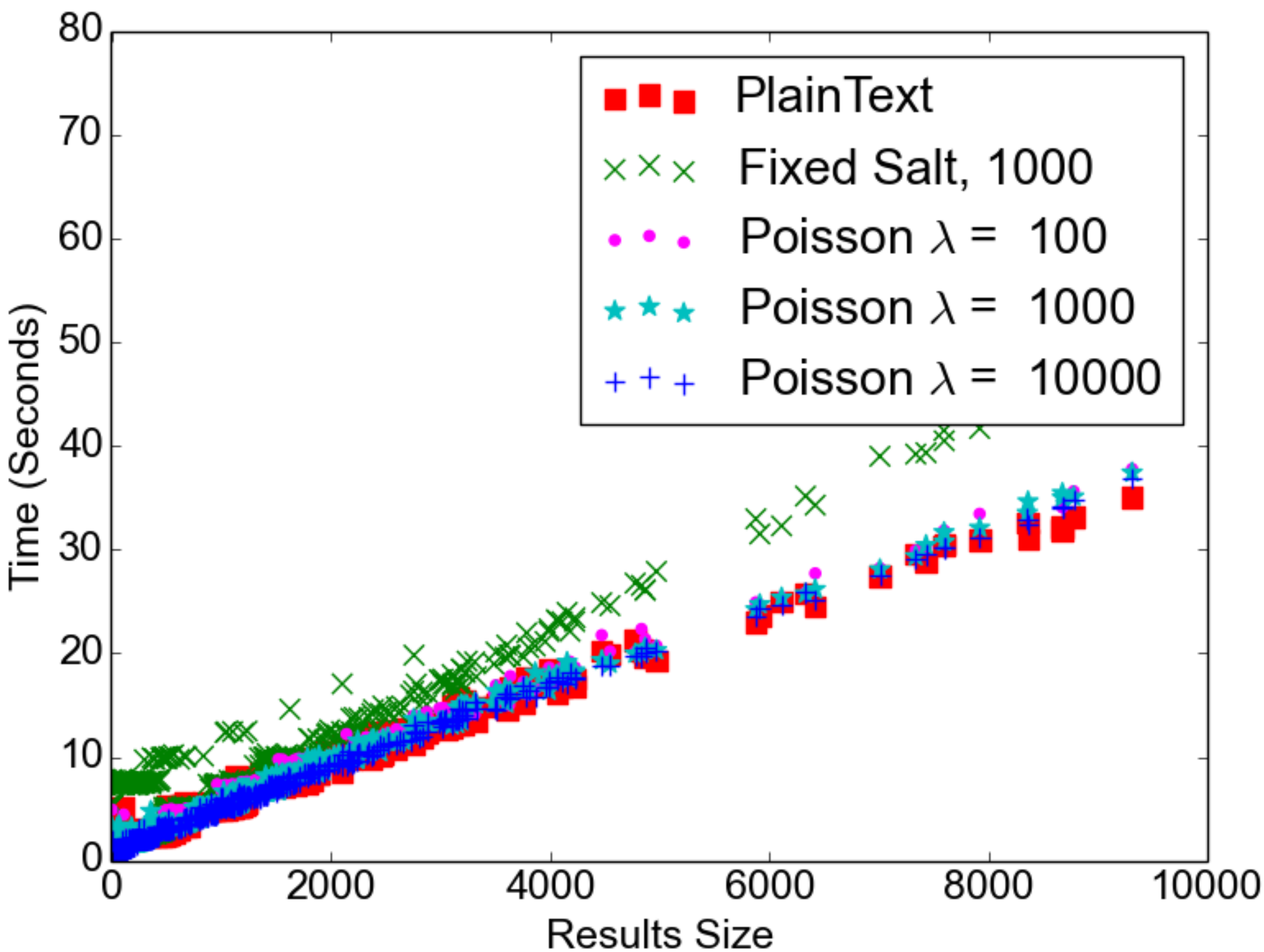
Con: Tags are now *buckets* representing multiple plaintexts

EMPIRICAL EVALUATION

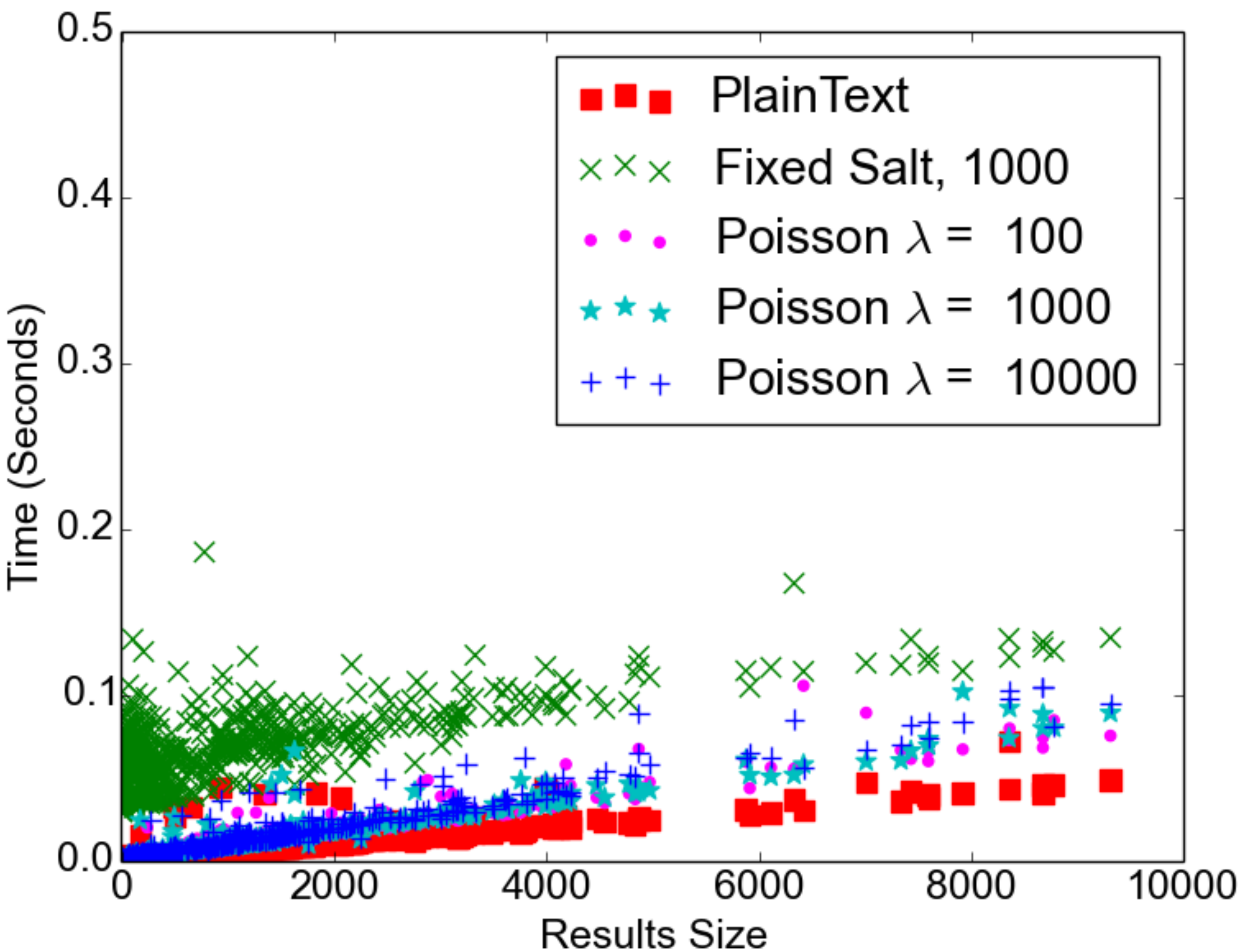
Experimental Procedure

- Used SPARTA testing framework from MIT-LL
  - Generated synthetic databases
    - 1M, 10M records
  - Generated synthetic queries
    - SELECT … FROM table WHERE column = value;
    - Return up to 10k matching records
- Ran queries on real SQL databases
  - Google Compute Engine
  - Local Postgres server

Performance: Cold Cache



Performance: Warm Cache



Conclusion

- WRE Contributions
  - Easy to deploy
  - Secure against most common threats
  - Performance close to plaintext

- Future Work / Open Problems
  - Security for queries? For access pattern?
  - Security for multiple (correlated) columns?
  - Range queries?