Structured Encryption and Leakage Suppression

Tarik Moataz

Part I is a joint work with Seny Kamara and Olya Ohrimenko

Part II is a joint work with Seny Kamara
Structured Encryption (STE) [CK10]

\[\text{Setup} \quad 1^k, \text{DS} \quad \rightarrow \quad \text{tk} \quad \rightarrow \quad \text{ans} \quad \leftarrow \quad \text{Query} \quad \text{tk}, \text{EDS}\]

\[\text{tk} \quad \leftarrow \quad \text{Token} \quad \text{key}, q\]

\[\text{Answer} \quad \text{ans}\]
Structured Encryption [CK10]

Setup Leakage $\mathcal{L}_S$

Query Leakage $\mathcal{L}_Q$

Setup $\mathcal{E}_S$

Token $\mathcal{T}$, $q$

Answer $\mathcal{A}$

Setup $[1^k, \mathcal{D}_S]$

Query $[tk, \mathcal{E}_S]$

tk
Structured Encryption [CK10]

An STE scheme is \((\mathcal{L}_S, \mathcal{L}_Q)\)-secure if

- It reveals no information about the structure beyond \(\mathcal{L}_S\)
- It reveals no information about the structure and queries beyond \(\mathcal{L}_Q\)
Structured Encryption [CK10]

Applications

- Encrypted NoSQL Databases
- Encrypted Distributed Hash Tables
- Garbled Circuits
- Searchable Symmetric Encryption
- Network Provenance

Structured Encryption (STE)

- Encrypted Multi-maps, Encrypted Dictionaries, Encrypted Arrays, Encrypted Graphs...
Structured Encryption [CK10]
Structured Encryption Evolution

**Efficiency**
- ‘00: Linear per file [SWP00]
- ‘03: Linear [Goh03]
- ‘06: Optimal [CGKO06, CK10]
- ‘12: Dynamism [KPR12], [KP13], [CJJJKRS14]
- ‘14: I/O efficiency [CT14], [CJJJKRS14], [ANSS16], [DPP18], [ASS18]

**Expressiveness**
- ‘00: Single-keyword SSE [SWP00], [Goh03], [CGKO06], [CJJJKRS14]
- ‘06: Multi-user SSE [CGKO06], [JJKRS13], [PPY16], [HSWW18]
- ‘13: Boolean SSE [CJJJKRS13], [PKVK+14], [KM17]
- ‘14: Range SSE [PKVK+14], [FJKNRS15]
- ‘18: STE-based SQL [KM18]

**Security**
- ‘06: Leakage-parametrized security definitions [CGKO06]
- ‘12: Adv. models [KO12], [BFP16], [AKM18]
- ‘12: Attacks [IKK12], [CGPR15], [ZKP16], [KMNO16], [LMP18], [GLMP18]
- ‘14: Forward/Backward Security [SPS14], [Bost16], [LC17], [BMO17], [AKM18]
What about Leakage?
What about Leakage?

Cryptanalysis

[IKK12]

Measure

Suppression

[KMO18]
Def: Given a leakage profile, design attacks to recover the queries or the data under some assumptions

Goal: empirically learn the impact of a leakage pattern in real-world

Limitations: the gap between assumptions and reality can get wide
Measure

**Def:** Given a leakage profile, quantify (e.g., in bits) a specific leakage pattern

**Goal:** theoretically compare between leakage patterns

**Limitations:** (maybe) no possible total order (work in progress!)
Suppression

**Def:** Given a *leakage* profile, design a compiler or a transform to *suppress* a specific leakage pattern

**Goal:** develop tools to suppress various leakage patterns

**Limitations:** introducing some overhead
Part 1*

Suppressing Leakage

*joint work with Seny Kamara and Olya Ohrimenko

https://eprint.iacr.org/2018/551
Q: is there an existing approach to reduce leakage?
Existing Approaches

• ORAM Simulation [GO96, SvDSFRD13]

✓ • Generic
✓ • Small Leakage profile

✗ • Interactive
✗ • Efficiency

• Garbled RAM [LO13, GHLORW14]

• Custom Schemes [WNLCSSH14, BM16]
Q: are there more efficient ways to suppress leakage?
Background
Modeling Leakage

- **qe** : query equality
  - search pattern
- **d** : data identity
- **req** : response equality
- **rido** : response identity
  - access pattern
- **qlen** : query length
- **rlen** : response length
  - volume pattern
- **mqlen** : maximum query length
- **mrlen** : maximum response length
- **srlen** : sequence response length
- **ds** : data size
Background
Non-Repeating Sub-Pattern

• Non-repeating sub-pattern

\[
patt(\text{DS}, q_1, \cdots, q_t) = \begin{cases} 
\text{nrp}(\text{DS}, q_1, \cdots, q_t) & \text{if } q_i \neq q_j, \forall i, j \in [t] \\
\text{rp}(\text{DS}, q_1, \cdots, q_t) & \text{otherwise.}
\end{cases}
\]

• Example

\[
\text{qe}(\text{DS}, q_1, \cdots, q_t) = \begin{cases} 
\bot & \text{if } q_i \neq q_j, \forall i, j \in [t] \\
\text{rp}(\text{DS}, q_1, \cdots, q_t) & \text{otherwise.}
\end{cases}
\]
Leakage Suppression
Through Compilation

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{patt}_1, p_2 \times 2)) \]

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{patt}_1) \]
Suppressing Query Equality

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = \text{patt}) \]  \rightarrow  \text{Cache-Based Compiler (CBC)}  \rightarrow  \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{nrp})

\text{patt} = \begin{cases} \text{nrp} \\ \times \end{cases}
Leakage Suppression
Through Transformation

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{patt}_1) \]

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{patt}_1, \text{x})_3) \]

DS \quad \xrightarrow{\text{Transform}} \quad DS^* \quad \rightarrow \quad \text{STE}_{DS} \quad \rightarrow \quad EDS
• Cache-based Compiler (CBC)
  • suppresses the query equality and the repeating sub-pattern
  • induces an additive poly-log overhead
  • Requires a rebuildable STE
Rebuild Compiler (RBC)
  - makes any STE scheme **rebuildable**
  - preserves the scheme’s query efficiency
  - adds a super-linear rebuild cost
The problem boils down to reduce nrp of the base STE scheme
- Piggyback scheme (PBS)
  - hides the response length for non-repeating queries
  - introduces query latency
Square-Root ORAM [GO96]

1. Read the entire cache
2. Read the real block
3. Insert the block back in the cache

1. Read the entire cache
2. Read a dummy block
3. Insert the block back in the cache

Maximum size $\lambda$

Main memory

Rebuild after $\lambda$
Main memory is an encrypted array construction.
Accessing element is done deterministically through PRP evaluation.
Adversary learns if/when an access to the same element is repeated.
- Leaks query equality.
Reinterpreting the Square-Root Solution

- The cache is an encrypted dictionary data structure
  - Given a label, it outputs an element or $\perp$
- The cache is accessed in its entirety
  - Most trivial zero-leakage dictionary construction; therefore no query leakage
Reinterpreting the Square-Root Solution

Access(15)

Access Zero-Leakage Dictionary

Access Real or Dummy
Reinterpreting the Square-Root Solution
Reinterpreting the Square-Root Solution

• Requirements
  • EDS scheme has to be rebuildable
  • Data structure has to be extendable and safe
  • Base scheme has to have smaller non-repeating sub-pattern
Data Structure Extension

• \( \lambda \)-extension:
  • Extend the query space of the data structure with \( \lambda \) dummies
  • \( \forall q \in \overline{Q} \setminus Q \) s.t. \( Q \subseteq \overline{Q} \)

\[
\text{Query}\left(\overline{DS}, q\right) = \bot
\]

• Safe \( \lambda \)-extension:

\[
\mathcal{L}_S(\overline{DS}) \leq \mathcal{L}_S(\text{DS})
\]
\[
\mathcal{L}_Q(\overline{DS}, q) \leq \mathcal{L}_Q(\text{DS}, q)
\]
PBS: Data transformation

- Batch size (ex: $\alpha = 3$)
- Pad all responses to a multiple of $\alpha$
PBS Details

\[ \text{Setup}^{\text{PBS}} \left[ l_k, \right] \rightarrow \text{Multi-map MM} \rightarrow \text{Encrypted Dictionary EDX} \rightarrow \text{State} \]
Consider a sequence of labels $\mathbf{q} = (\ell_1, \ell_2)$

**Token**

1. $\ell_1$ has 2 batches
2. Instantiate a queue
3. Compute
4. Update queue
PBS Details
PBS Details

Token\textsuperscript{PBS}:

1. \(b_2\) has 1 batch
2. Update the queue
3. Compute
4. Update queue
PBS Details

Get\textsuperscript{PBS}

Get\textsuperscript{EDX}
PBS Details

Token\textsuperscript{PBS}\left[\textbullet, \underbar{2}, \perp\right]:

1. Compute

\text{Token}\left[\textbullet, \overline{2}1\right] \rightarrow \overline{2}1

2. Update queue
PBS Latency

• The worst-case query sequence of size $t$ has latency

$$t \cdot \left( \frac{\max_{r \in \mathbb{R}_{DS}} |r|_w}{\alpha} - 1 \right)$$

• Real-world sequences have latency

$$\varepsilon \cdot t$$

with probability at least

$$1 - \exp \left( -2t \left( \varepsilon \cdot \frac{\alpha}{\max_{r \in \mathbb{R}_{DS}} |r|_w} \right)^2 \right)$$

where queries are drawn from a Zipf distribution and longer responses are mapped to less frequent labels
AZL Analysis

• Worst-case query complexity over $\lambda$ queries

$$\sum_{i=1}^{\lambda} T_{\mathrm{eds}}(q_i) + O \left( \lambda \cdot \max_{q \in q} |r|_w \cdot \log^2 \lambda \right) + O \left( \sum_{r \in R_{DS}} |r|_w \cdot \log^2 \#Q_{DS} \right)$$

• Comparison to ORAM simulation (Path-ORAM [SvDSFRD13])

$$T_{\mathrm{eds}}(q_1, \ldots, q_\lambda) = o \left( T_{\mathrm{tree}}(q_1, \ldots, q_\lambda) \right)$$

when

Natural Assumption:
If response lengths are power-law distributed

$$\sum_{r \in R_{DS}} |r|_w = o \left( \sum_{i=1}^{\lambda} B(q_i) \cdot \max_{r \in R_{DS}} |r|_w \right)$$
and

$$\lambda \cdot \max_{q \in q(DS, q)} |q(DS, q)|_w = o \left( \sum_{r \in R_{DS}} |r|_w \right)$$
Part 2*
Suppressing Volume

*joint work with Seny Kamara

https://eprint.iacr.org/2018/978
Leakage Suppression
Through Compilation

\[ \Lambda = \left( L_S, L_Q = \text{patt}_1, p \times \text{patt}_2 \right) \]

Compilation

\[ \Lambda' = \left( L_S, L_Q = \text{patt}_1 \right) \]
Leakage Suppression
Through Transformation

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{patt}_1) \]

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{patt}_1, \text{patt}_2)) \]
Q: is there any other approach to suppress leakage?
Suppression

- Black-box Compilation
- Data structure Transformation

against unbounded adversary

against bounded adversary
Computationally-Secure Leakage

Unbounded Adversary  vs.  Bounded Adversary
Leakage Suppression [KMO18] Through Transformation

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = (\text{patt}_1, \text{patt}^*) ) \]

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{patt}_1, \text{patt}_2) ) \]

\[ \text{patt}^* \simeq \perp \]
Q: can we suppress the response length pattern?
Background
Dictionary and Multi-Map data structures

• DXs map labels to values
  
  Dictionary DX

  \[
  \begin{align*}
  w_1 & \rightarrow id_1 \\
  w_2 & \rightarrow id_3 \\
  w_3 & \rightarrow id_2 
  \end{align*}
  \]

  • Get: \text{DX}[w_3] returns id_2

• MMs map labels to tuples
  
  Multi-map MM

  \[
  \begin{align*}
  w_1 & \rightarrow \{id_1, id_3, id_4\} \\
  w_2 & \rightarrow id_3 \\
  w_3 & \rightarrow \{id_2, id_4\} 
  \end{align*}
  \]

  • Get: \text{MM}[w_3] returns (id_2, id_4)
Background
Response Length Pattern (rlen) or Volume Pattern

\[ \mathcal{L}_Q = \left( \cdot, \text{rlen} \right) \quad \text{s.t.} \quad \text{rlen} \left( \text{DS}, q \right) = |\text{ans}| \]
Naive Approaches to Hide Volume
Through Naive Padding

**STE’\_MM**
\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{qeq}) \]

**STE\_MM**
\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{qeq, rlen})) \]

e.g., [CGKO06], [CK10], [CJJJKRS14]
Naive Approach to Hide Volume
Through Naive Padding

- Query complexity

\[ O(\max_{\ell \in \mathbb{L}_{MM}} \#MM[\ell]) \]

- Storage complexity

\[ O(\#L_{MM} \cdot \max_{\ell \in \mathbb{L}_{MM}} \#MM[\ell]) \]

- Non-interactive
Naive Approach to Hide Volume Through Leakage-Free Dictionary

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{qeq}) \]

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = \perp) \]

e.g., [GO96], [SCSL11], [SDSFRYD13]
Naive Approach to Hide Volume
Through Leakage-Free Dictionary (w/ [SDSFRYD13])

- Query complexity
  \[ O\left( \max_{\ell \in \mathbb{L}_{MM}} \#MM[\ell] \cdot \log^2 \left( \sum_{\ell \in \mathbb{L}_{MM}} \#MM[\ell] \right) \right) \]

- Storage complexity
  \[ O\left( \sum_{\ell \in \mathbb{L}_{MM}} \#MM[\ell] \right) \]

- Interactive
Q: can we achieve the best of both worlds?
Contributions

- Pseudo-Random Transform (PRT)
- Volume Hiding Multi-Map Encryption scheme (VLH)
- Densest-Subgraph Transform (DST)
- Advanced Volume Hiding Multi-Map Encryption scheme (AVLH)
- Dynamism
Pseudo-Random Transform (PRT)

- Pseudo-random function \( F : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^{\log \nu} \)
- Minimum response length \( \lambda \)
- Replace the response length of \( \ell \) by \( \lambda + F_K(\ell\|\#\text{MM}[\ell]) \)
  - Truncate if \( \lambda + F_K(\ell\|\#\text{MM}[\ell]) \leq \#\text{MM}[\ell] \)
  - Pad if \( \lambda + F_K(\ell\|\#\text{MM}[\ell]) > \#\text{MM}[\ell] \)
- Rank the response identities

E.g., \( \lambda = 1 \) and \( \nu = 3 \)

\[
F_K(w_1\|3) = 0 \\
F_K(w_2\|1) = 2 \\
F_K(w_3\|2) = 1
\]
Q: what about the number of truncations and storage overhead?
Pseudo-Random Transform (PRT)

Zipf-Distributed MM

A MM is Zipf-distributed if the $r^{th}$ response has length:

$$\frac{1}{r \cdot H_{\#M M, 1}} \cdot \sum_{\ell \in \#M M} \#M M[\ell]$$

- Common in real-world datasets [Zipf35], [CCKS07]
- Ex: Enron 0.5M emails (2004)
Pseudo-Random Transform (PRT) Analysis

• Let $\alpha$ be the storage reduction multiplicative factor

• If for $1/2 < \alpha < 1$, then with probability at least

  \[
  1 - \exp\left(-\#L_{MM} \cdot (2\alpha - 1)^2/8\right) \text{ the size of the MM is at most}
  \]

\[
\alpha \cdot \#L_{MM} \cdot \max_{\ell \in L_{MM}} \#MM[\ell]
\]

• $1 - \exp\left(-2\#L_{MM} \cdot \log^2(\#L_{MM})\right)$ the number of truncations is at most

\[
\frac{1}{\log(\#L_{MM})} \cdot \#L_{MM}
\]
Volume Hiding EMM (VLH)

Design

\[ \Lambda' = (\mathcal{L}_S, \mathcal{L}_Q = \text{qe}q) \]

Multi-map MM

\[
\begin{align*}
& w_1 \quad \text{id}_1 \quad \text{id}_3 \quad \text{id}_4 \\
& w_2 \quad \text{id}_3 \\
& w_3 \quad \text{id}_2 \quad \text{id}_4
\end{align*}
\]

Multi-map MM'

\[
\begin{align*}
& w_1 \quad \text{id}_1 \\
& w_2 \quad \text{id}_3 \quad \text{shadow} \\
& w_3 \quad \text{id}_2 \quad \text{id}_4
\end{align*}
\]

PRT Transform

\[ \text{STE}_{\text{MM}} \]

\[ \Lambda = (\mathcal{L}_S, \mathcal{L}_Q = (\text{qe}q, \text{rlen})) \]

e.g., [CGKO06], [CK10], [CJJJKRS14]
Volume Hiding EMM (VLH)
Analysis (with standard EMMs)

- **Query complexity** (worst-case)
  \[ O(\lambda + \nu) \]

- **Storage complexity**
  \[ O(\lambda \cdot \#L_{MM} + \sum_{\ell \in L_{MM}} n_{\ell}) \text{ s.t. } n_{\ell} \leftarrow [\nu] \]

  and w.h.p. when \( \frac{1}{2} < \alpha < 1 \)

  \[ O(\alpha \cdot (\nu - 1) \cdot \#L_{MM}) \]

- **Non-Interactive**

- **Lossy**
Densest-Subgraph Transform (DST)
Overview

• We view a MM as a bi-partite graph $G = \left( (\mathbb{L}_{MM}, B), E \right)$
  • top vertices: labels $\mathbb{L}_{MM}$
  • bottom vertices: bins $B$
• Given MM we build a Erdös-Rényi random graph
• All labels in MM have the same number of edges
• **Goal**: given a label, fetch the same number of bins
  ▶ reduce the load of the bin
Densest-Subgraph Transform (DST)

Details

Multi-map MM

State

Storage overhead

\[ O(\#L_{MM} \cdot \max_{\ell \in \mathbb{L}_{MM}} \#MM[\ell]) \]

Similar to Naive Padding
Densest-Subgraph Transform (DST)

Details

Edge Generation

\[
\begin{align*}
\text{rand}_{w_1} &\overset{$\$}{\leftarrow} \{0,1\}^k \\
F_K(\text{rand}_{w_1} || 1) &= 1 \\
F_K(\text{rand}_{w_1} || 2) &= 3 \\
F_K(\text{rand}_{w_1} || 3) &= 2 \\
\text{rand}_{w_2} &\overset{$\$}{\leftarrow} \{0,1\}^k \\
F_K(\text{rand}_{w_2} || 1) &= 2 \\
F_K(\text{rand}_{w_2} || 2) &= 3 \\
F_K(\text{rand}_{w_2} || 3) &= 4 \\
\text{rand}_{w_3} &\overset{$\$}{\leftarrow} \{0,1\}^k \\
F_K(\text{rand}_{w_3} || 1) &= 1 \\
F_K(\text{rand}_{w_3} || 2) &= 2 \\
F_K(\text{rand}_{w_3} || 3) &= 3
\end{align*}
\]

\[\mathcal{O}(\#\mathbb{I}_{\text{MM}}) \ll \mathcal{O}(\#\mathbb{I}_{\text{MM}} \cdot \max_{\ell \in \mathbb{I}_{\text{MM}}} \#\text{MM}[\ell])\]
Densest-Subgraph Transform (DST)

Details

• The output of DST is equal to:
  
  $\mathcal{F}_K(\text{rand}_{w_1} \| 1), \mathcal{F}_K(\text{rand}_{w_1} \| 2), \mathcal{F}_K(\text{rand}_{w_1} \| 3)$

• To fetch a keyword $w_1$, retrieve $\text{rand}_{w_1}$ from the state

• Compute bins’ identifiers $\mathcal{F}_K(\text{rand}_{w_1} \| 1), \mathcal{F}_K(\text{rand}_{w_1} \| 2), \mathcal{F}_K(\text{rand}_{w_1} \| 3)$

• Retrieve all the bins from the dictionary $\text{DX}$
Q: what about the load of a bin?
Densest-Subgraph Transform (DST)

Analysis

With probability at least $1 - \varepsilon$, the load of a bin is

$$\frac{N}{n} + \frac{\ln(1/\varepsilon)}{3} \left( 1 + \sqrt{1 + \frac{18N}{n \cdot \ln(1/\varepsilon)}} \right)$$

where $N = \sum_{\ell \in \mathbb{L}_{\text{MM}}} \#\text{MM}[\ell]$

The size of the transformed multi-map MM is $O(N)$

The size of the state is $O(\#\mathbb{L}_{\text{MM}}) \ll O(N)$
Advanced Volume-Hiding EMM (AVLH)

Setup (1)

Setup $[1^k, \text{Multi-map MM}]$

1. DST $[1^k, \text{Multi-map MM}] \rightarrow [\text{key}_1, \text{State}, \text{Graph G}]$

2. Graph $G$

Dictionary $DX$

$B_1 \rightarrow \text{id}_1, \text{id}_2$

$B_2 \rightarrow \text{id}_3, \text{id}_4$

$B_3 \rightarrow \text{id}_3$

$B_4 \rightarrow \text{id}_4$
Advanced Volume-Hiding EMM (AVLH)
Setup (2)

Setup $[1^k, \text{Multi-map MM}]$

3. EDX.Setup $[1^k, \text{Dictionary DX}]$ \rightarrow $[\text{Key}_2, \text{Dictionary DX}]$

Output $[\text{Key}_1, \text{Key}_2, \text{State}]$

\begin{itemize}
\item $w_1 \rightarrow \text{rand}_{w_1}$
\item $w_2 \rightarrow \text{rand}_{w_2}$
\item $w_3 \rightarrow \text{rand}_{w_3}$
\item $B_1 \rightarrow \text{id}_1 \rightarrow \text{id}_2$
\item $B_2 \rightarrow \text{id}_3 \rightarrow \text{id}_4$
\item $B_3 \rightarrow \text{id}_3$
\item $B_4 \rightarrow \text{id}_4$
\end{itemize}
Advanced Volume-Hiding EMM (AVLH)

Token

Token $\left[ \begin{array}{c}
\text{key} \\
\text{State} \\
W_1
\end{array} \right]$

1. Fetch $\text{rand}_{w_1}$ from State

2. Compute $t = \left( F_K(\text{rand}||i) \right)_{i \in [3]}$

3. for each identifier $i$ in $t$ add to $tk$

$\text{EDX.Token} \left[ \begin{array}{c}
\text{key} \\
i
\end{array} \right] \rightarrow tk_t$

Output $tk$
Advanced Volume-Hiding EMM (AVLH)

Query

Query \[ tk \]

1. for each sub-token \( tk_i \) in \( tk \)

\[
EDX.\text{Query} \left[ tk_i, \text{Dictionary DX} \right] \rightarrow ct_i
\]

Output

\[ ct = (ct_1, ct_2, ct_3) \]
Advanced Volume Hiding EMM (VLH)
Analysis ([CGKO06])

• **Query complexity** w.h.p.

\[ O\left(t \cdot \frac{N}{\#_{\text{MM}} \cdot \text{polylog}(\#_{\text{MM}})}\right) \]

where \( t \) is the maximum length and

\[ N = \sum_{\ell \in \mathbb{L}_{\text{MM}}} \#_{\text{MM}[\ell]} \]

• **Storage complexity** w.h.p.

\[ O(N) \]

• **Non-Interactive**

• **Non-Lossy**
Densest-Subgraph Transform (DST)

Improving Storage

Erdős-Rényi graph

Erdős-Rényi graph with planted dense subgraph

Found applications in public-key cryptography [ABW10] and computational complexity of financial products [ABBG11]
Densest-Subgraph Transform (DST)
Improving Storage

Multi-map MM

Concentrated MM: labels with non-empty intersection

id₂ and id₄ constitute the concentrated part

Add the concentrated part only once to the graph

Result: Reduce the load of bins
Densest-Subgraph Transform (DST)

Analysis

With probability at least $1 - \varepsilon$, the load of a bin is

$$\frac{N - N_{DS}}{n} + \frac{\ln(1/\varepsilon)}{3} \left(1 + \sqrt{1 + \frac{18(N - N_{DS})}{n \cdot \ln(1/\varepsilon)}}\right)$$

where $N_{DS}$ is the size of the concentrated part.

Instead of

$$\frac{N}{n} + \frac{\ln(1/\varepsilon)}{3} \left(1 + \sqrt{1 + \frac{18N}{n \cdot \ln(1/\varepsilon)}}\right)$$
Takeaways
Takeaways

• Introduce a new direction in encrypted search

• A general framework that suppresses the search pattern

• First solution to hide response-length pattern (volume pattern)

• A general compiler that makes any STE scheme rebuildable

• First scheme to leak at most the sequence response length (very hard to exploit)

• The first scheme that leaks (nothing)

• Introduces a new tradeoff: query latency vs. security
Takeaways

• Volume pattern has been recently leveraged as an attack vector [KKNO16], [GLMP18]

• Without trivial naive padding, hiding volume is extremely hard

• Hiding volume is an important step for leakage suppression

• The first non-trivial schemes that hide the volume pattern

  • VLH based on a new lossy pseudo-random transform (PRT)

  • AVLH based on a new non-lossy densest-subgraph transform (DST)
Takeaways

• Leveraging computational assumptions to suppress leakage
  
  • Intuitively it is hard to hide volume information theoretically without padding
  
  • Get around this leveraging computational assumptions
  
  • first to do so for any pattern, and for volume in particular
  
  • possibility to leverage computational assumptions to suppress other patterns
  
• Introducing a new tradeoff: correctness vs. security

• Hiding volume can help thwart many existing attacks: [IKK12], [CGPR15], [KKNO16], [LMP18], [GLMP18], [LMP19]
Thank you!