Near-Optimal Constrained Padding for Object Retrievals with Dependencies

Pranay Jain
Duke University

Andrew C. Reed
United States Military Academy

Michael K. Reiter
Duke University

33rd USENIX Security Symposium
August 15, 2024

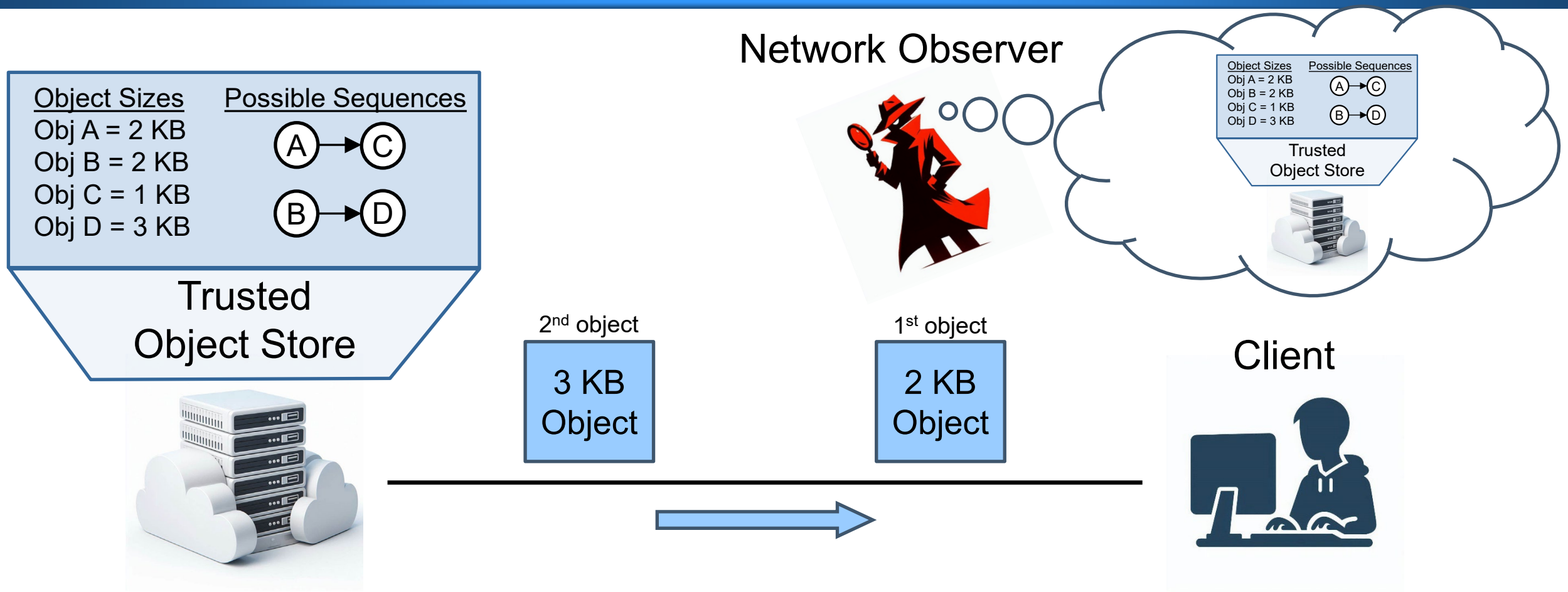
Agenda

- Objective
- Algorithm
- Evaluation
- Questions

Agenda

- Objective
- Algorithm
- Evaluation
- Questions

Objective: High Level



- **Client** has retrieved a sequence of objects from **Trusted Object Store**
- **Network Observer's** goal is to identify which objects were requested

All images generated by the authors using Bing Copilot Designer, August 5, 2024.

Objective: High Level

- Threat: A network observer with the following…
  - Capability: discern the sizes of sequentially-retrieved objects
  - Goal: identify which objects were retrieved
  - Knows:
    - every object's size
    - all possible sequences of object retrievals, and how often retrieved
    - the padding defense used by object store
- Trusted Object Store's Goal: Compute a padding scheme $\lceil \cdot \rceil$ that…
  1. Uses padding to best thwart the adversary
  2. Controls the per-object overhead due to padding

Note: The object store is only willing to pad objects, i.e., it will not insert decoy objects.

Objective: Our Approach

- **Objective**: Minimize $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$
  - $\mathbb{I}_\infty$ = Sibson mutual information of order infinity, also referred to as **min-capacity**[1] and **maximal leakage**[2]
  - S = random variable for an object's **identity**
  - Y = random variable for an object's **padded size**
  - $\longrightarrow$ denotes a sequence

Why did we choose $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$?

[1] and [2] advocate for this metric because:

a) $\mathbb{I}\left(\vec{S};\vec{Y}\right) \leq \mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ over all distributions of $\vec{S}$.

b) $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ upper-bounds an adversary's multiplicative gain in correctly guessing any function of $\vec{S}$ after observing $\vec{Y}$, over all distributions of $\vec{S}$.

- **Constraints**: For a given **max pad factor** c_{tgt} :
  - No object is padded by more than a factor of c_{tgt}
  - Each object is served in full

Note: it's possible for some objects to remain isolated in our setting

1. M. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith, "Measuring information leakage using generalized gain functions," *25th IEEE Computer Security Foundations*, Jun. 2012.
2. I. Issa, A. B. Wagner, and S. Kamath, "An operational approach to information leakage," *IEEE Transactions on Information Theory*, vol. 66, no. 3, Mar. 2020.

Agenda

- Objective
- Algorithm
- Evaluation
- Questions

Algorithm: Padding For Sequences (PFS)

- Design: a linear program named Padding For Sequences (PFS)

- Inputs:
  - S = the set of objects
  - $\vec{S}$ = the set of possible sequences
  - c_{tgt} = max padding factor per object

- Output:
  - A *memoryless* padding scheme $\lceil \cdot \rceil$ that minimizes an upper bound on $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ and does not violate c_{tgt} for any object

Agenda

- Objective
- Algorithm
- Evaluation
- Questions

Competitors: Overview

- BDK[3]
  - Target metric: $\mathbb{I}\left(\vec{S};\vec{Y}\right)$
- MVMD-D[4]
  - Target metric: ℓ-diversity
- PwoD[5]
  - Target metric: $\mathbb{I}_\infty(S;Y)$

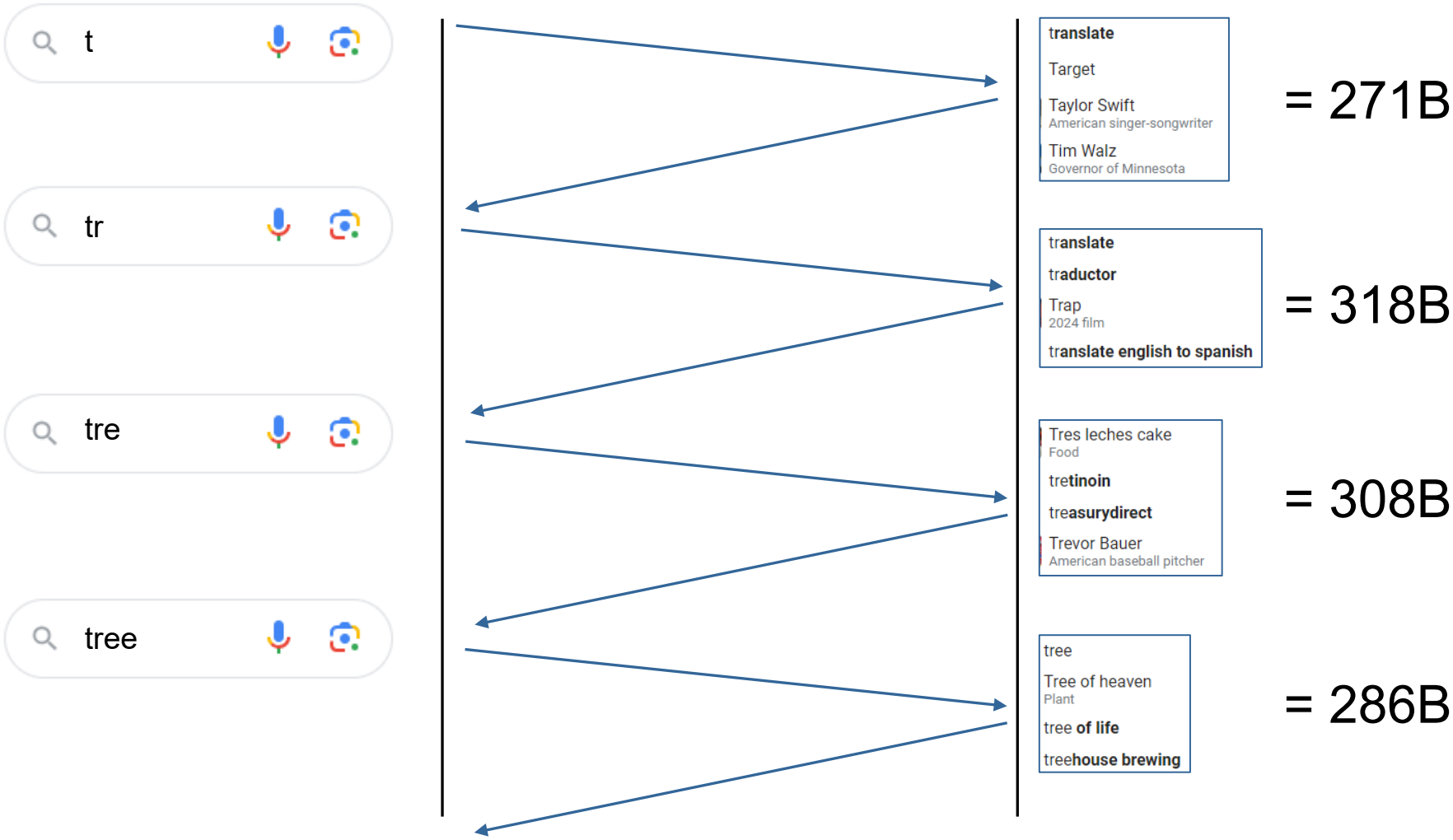
Inputs required per algorithm.

Algorithm	Sets		Distributions			
	S	$\vec{S}$	E	$\mathbb{P}(S=s)$	$\mathbb{P}\left(\vec{S}_{1\ldots i}=\vec{s}_{1\ldots i}\right)$	$\mathbb{P}\left(\vec{S}_{i+1}=s' \middle\| \vec{S}_i=s\right)$
BDK	✓		✓	✓		✓
MVMD-D	✓	✓			✓	
PwoD	✓					
PFS	✓	✓				

3. M. Backes, G. Doychev, and B. Kopf, "Preventing side channel leaks in web traffic: A formal approach," *20th ISOC Network and Distributed System Security Symposium*, February 2013.
4. W. M. Liu, L. Wang, P. Cheng, K. Ren, S. Zhu, and M. Debbabi, "PPTP: Privacy-preserving traffic padding in web-based applications," *IEEE Transactions on Dependable and Secure Computing*, Nov-Dec 2014.
5. A. C. Reed and M. K. Reiter, "Optimally hiding object sizes with constrained padding," *IEEE Computer Security Foundations Symposium*, July 2023.

Dataset: Autocomplete

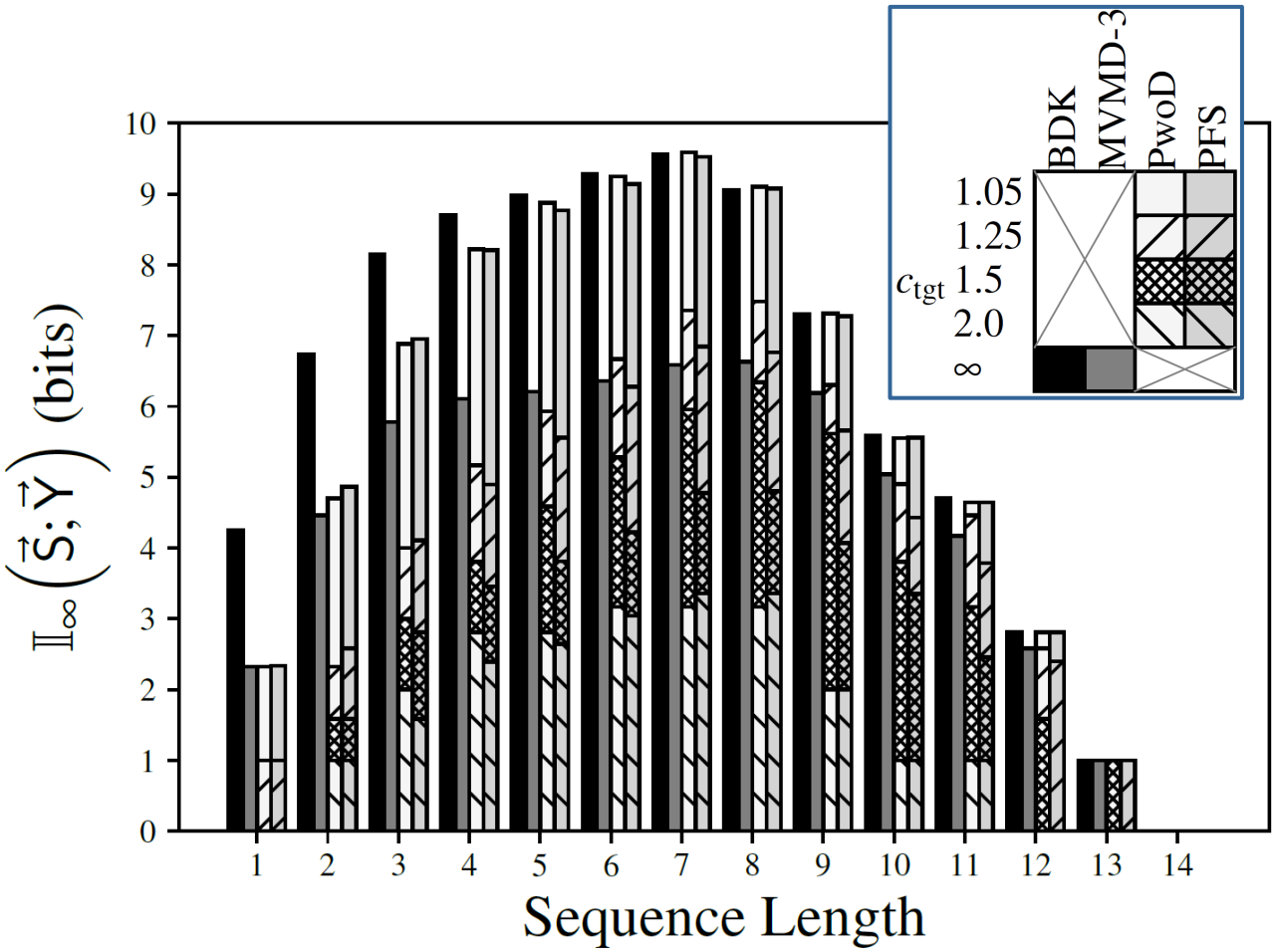
- 899 full words
- 3870 total sequences
- Models a user typing a word into the Google search bar and receiving suggested search terms after each character is typed.



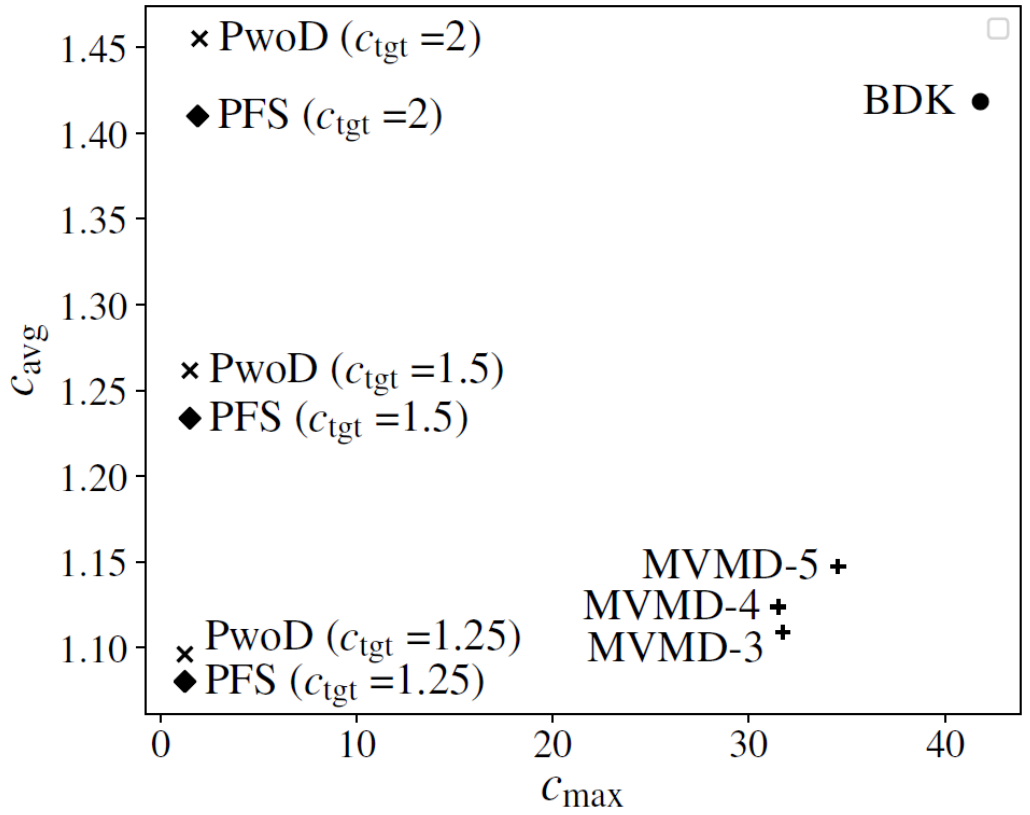
The Autocomplete sequence corresponding to the word “tree” is: (t, tr, tre, tree) = (271, 318, 308, 286)

Note: Screenshots taken Aug 8, 2024 and do not correspond to the sizes in the provided dataset. They are for illustrative purposes only.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

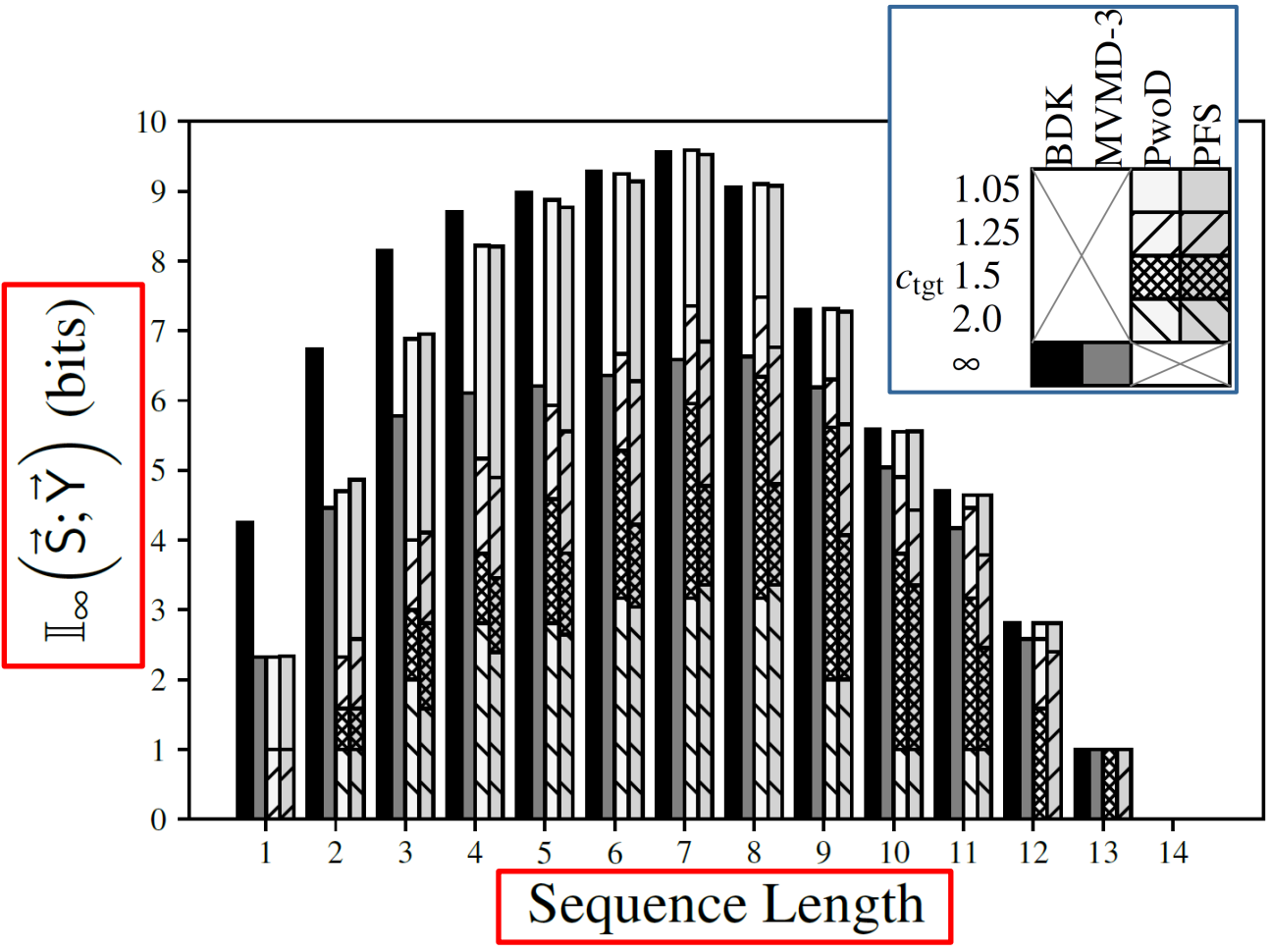


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

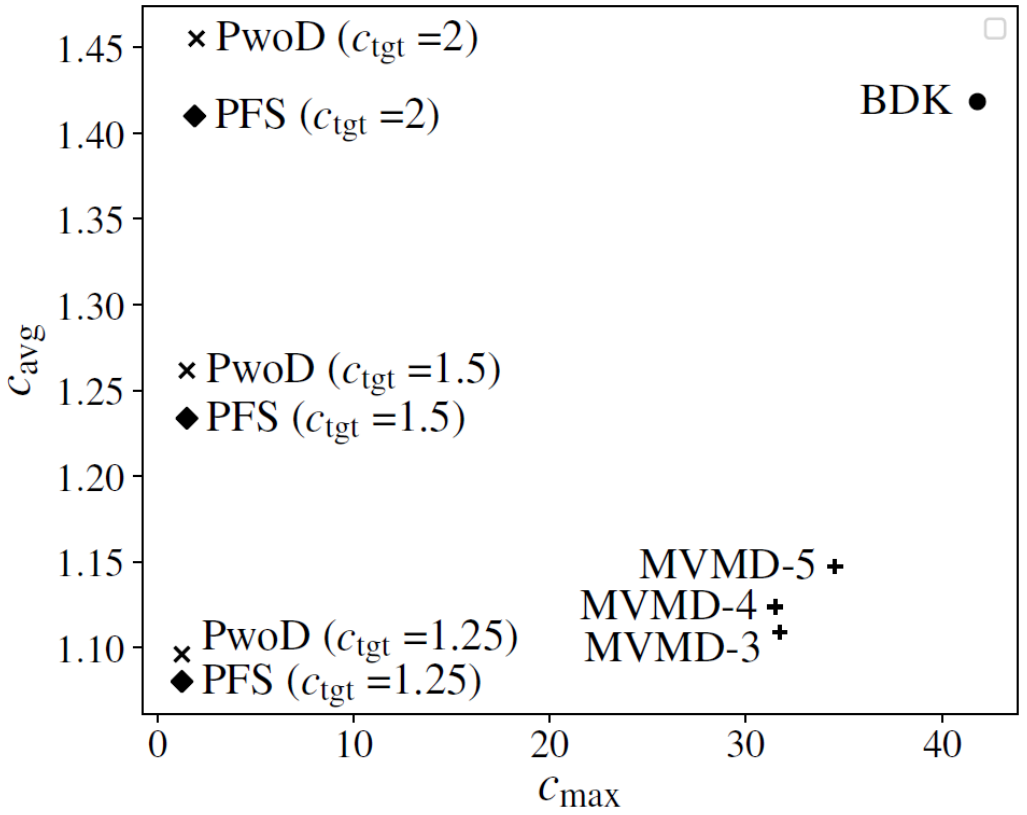


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

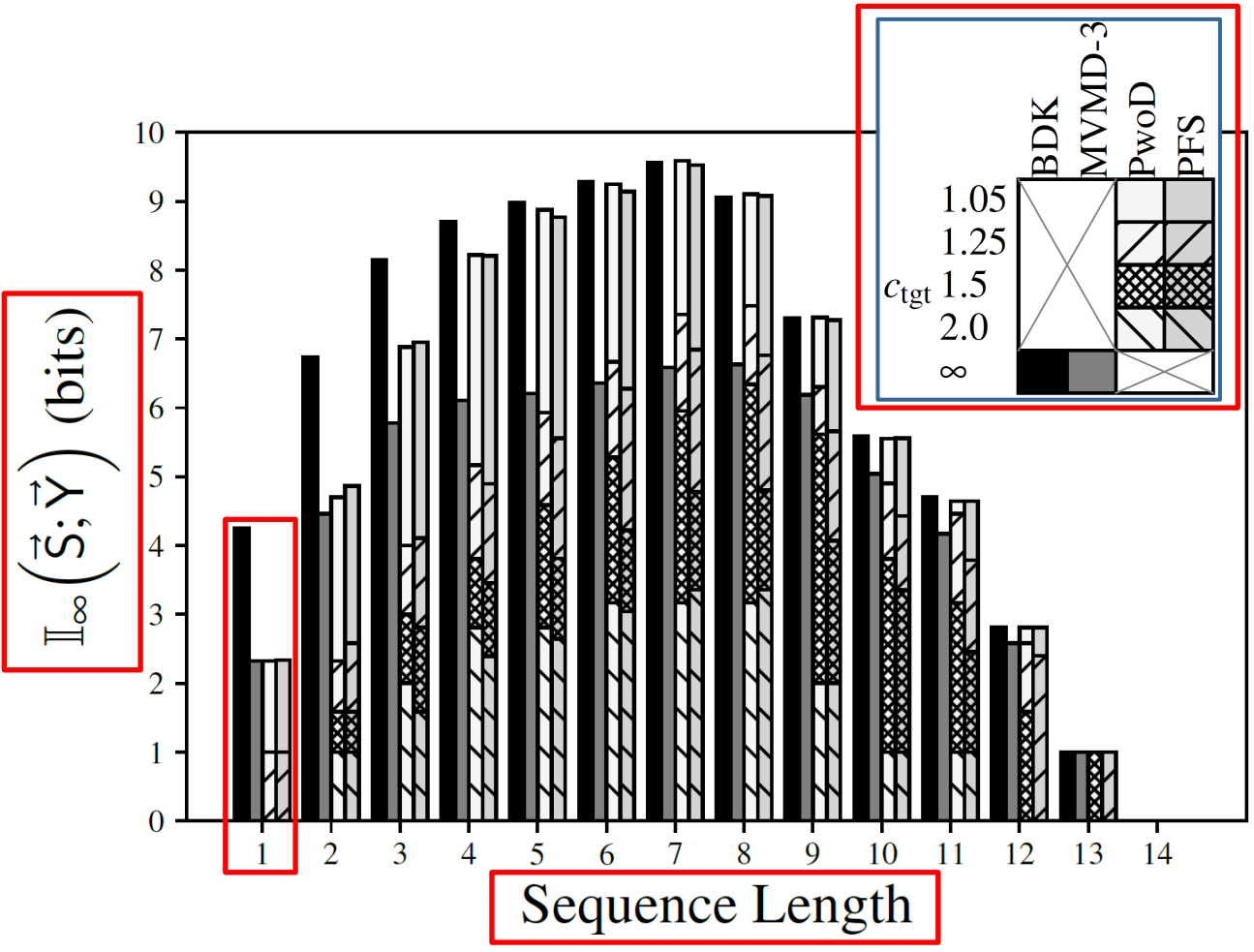


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

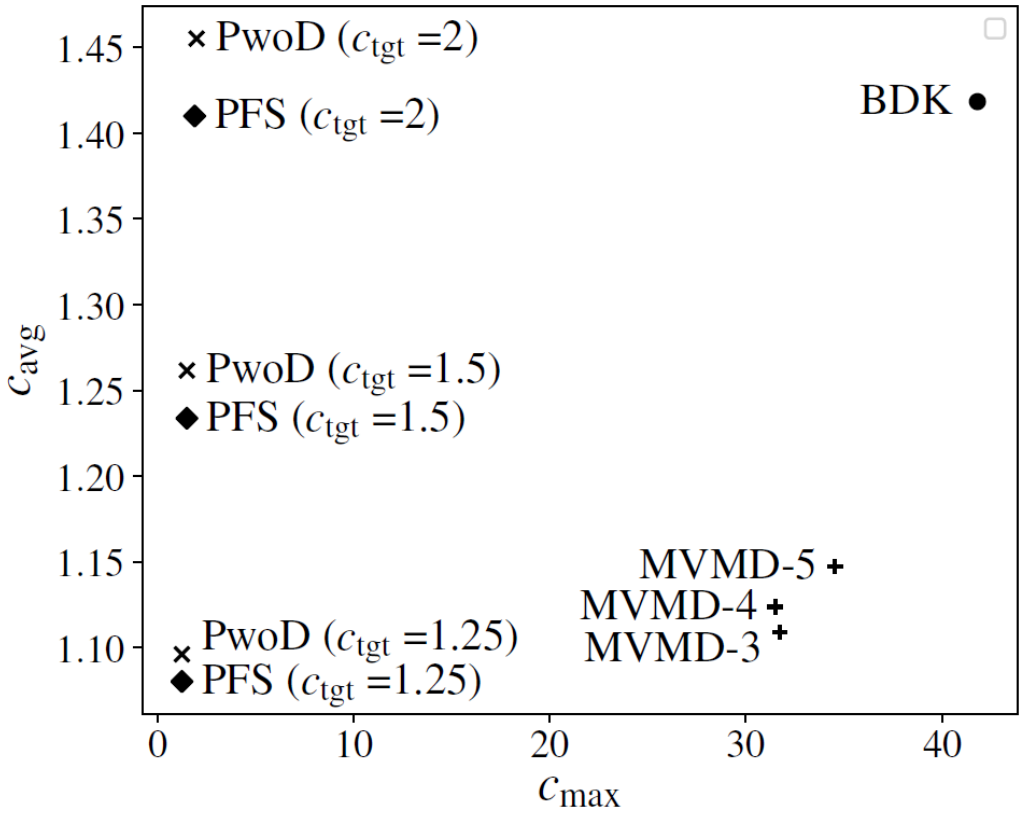


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

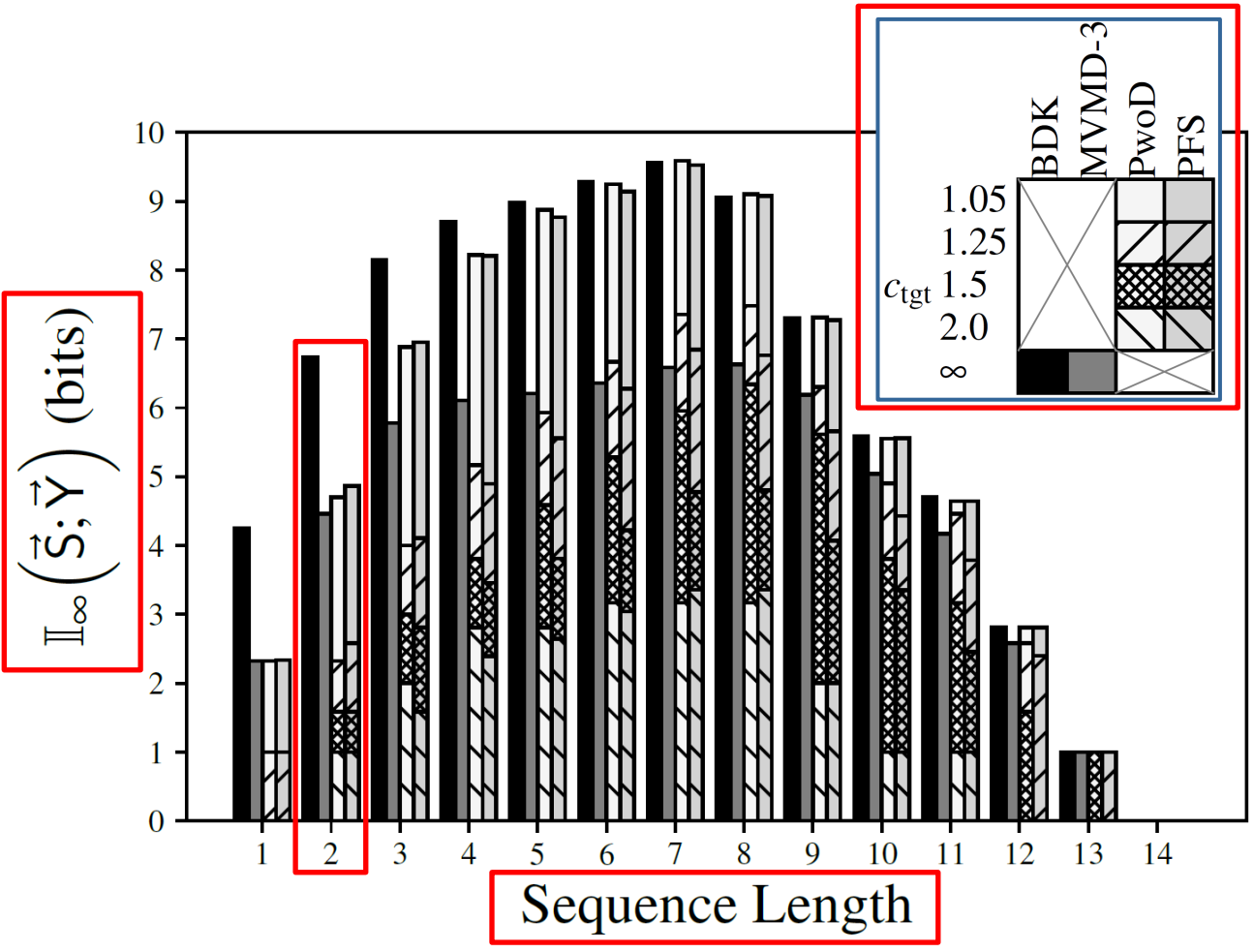


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

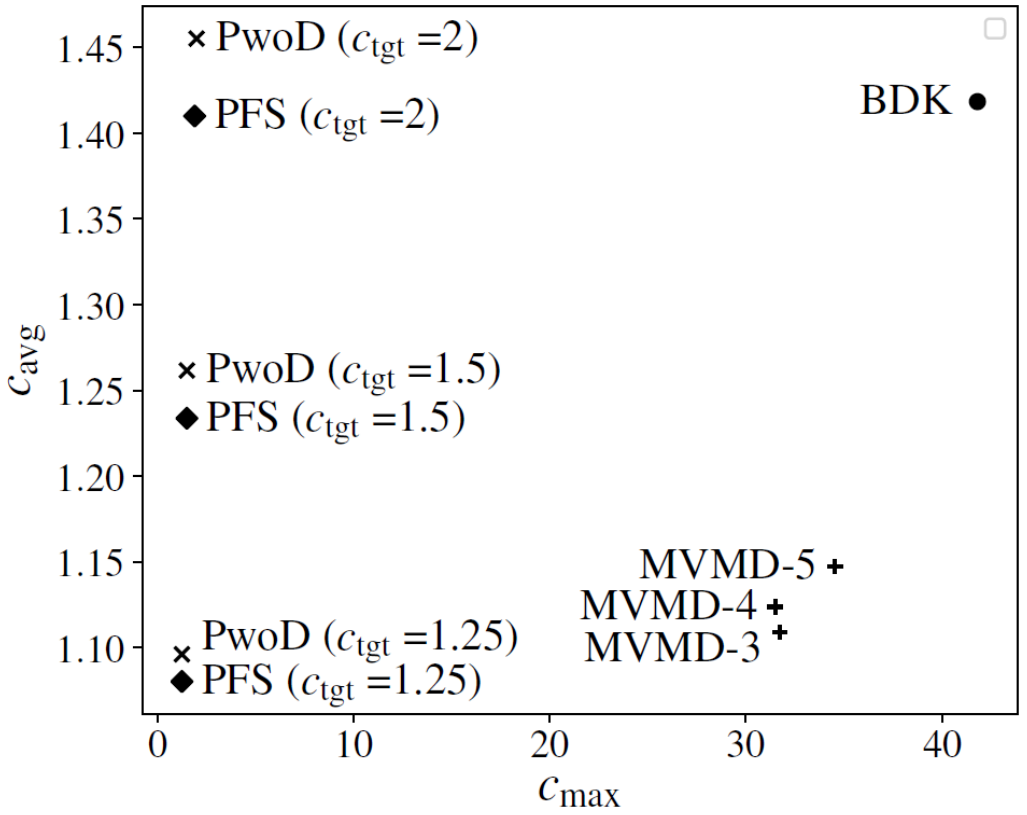


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

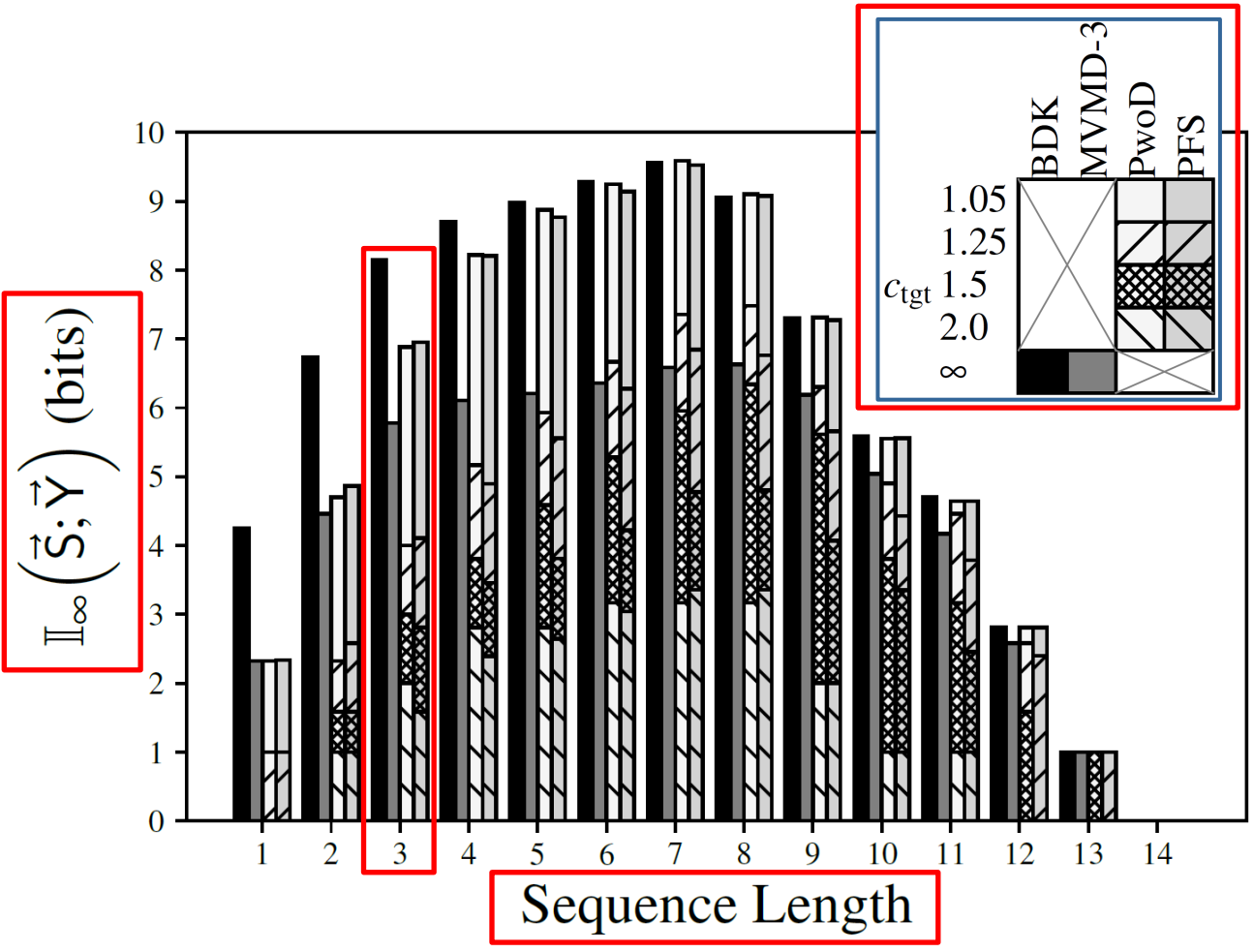


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

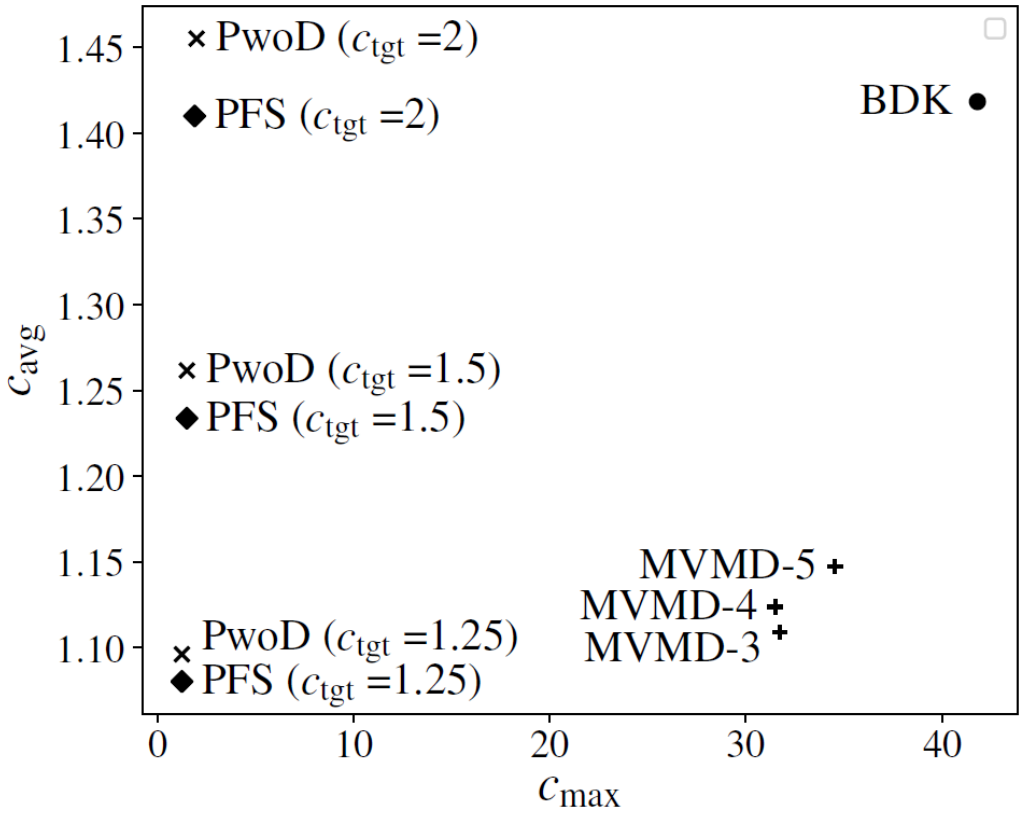


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

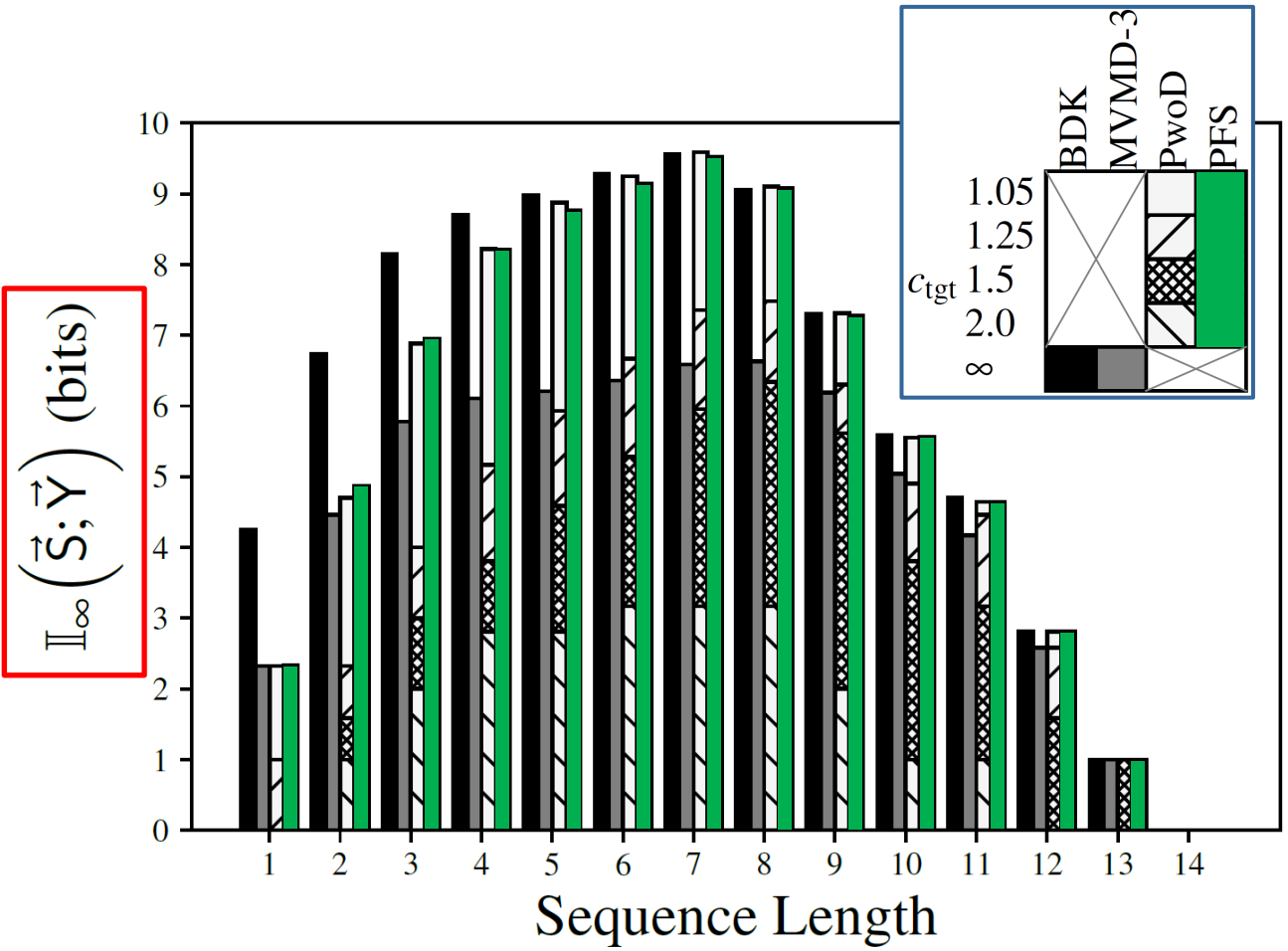


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

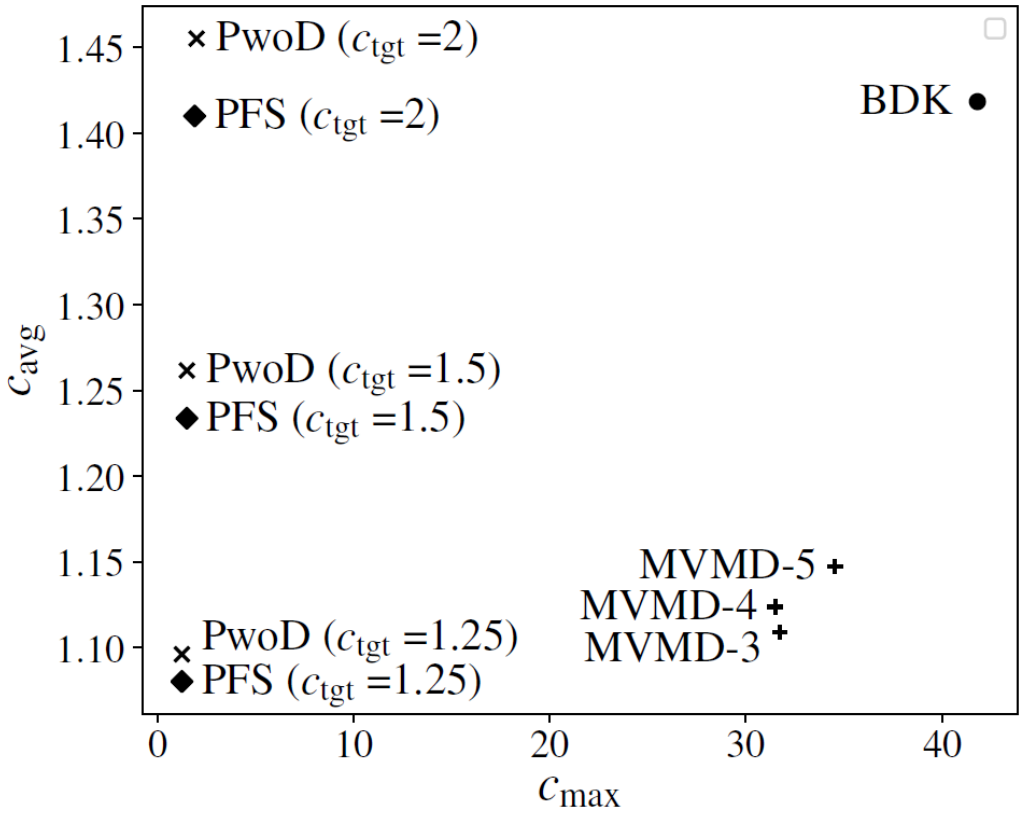


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

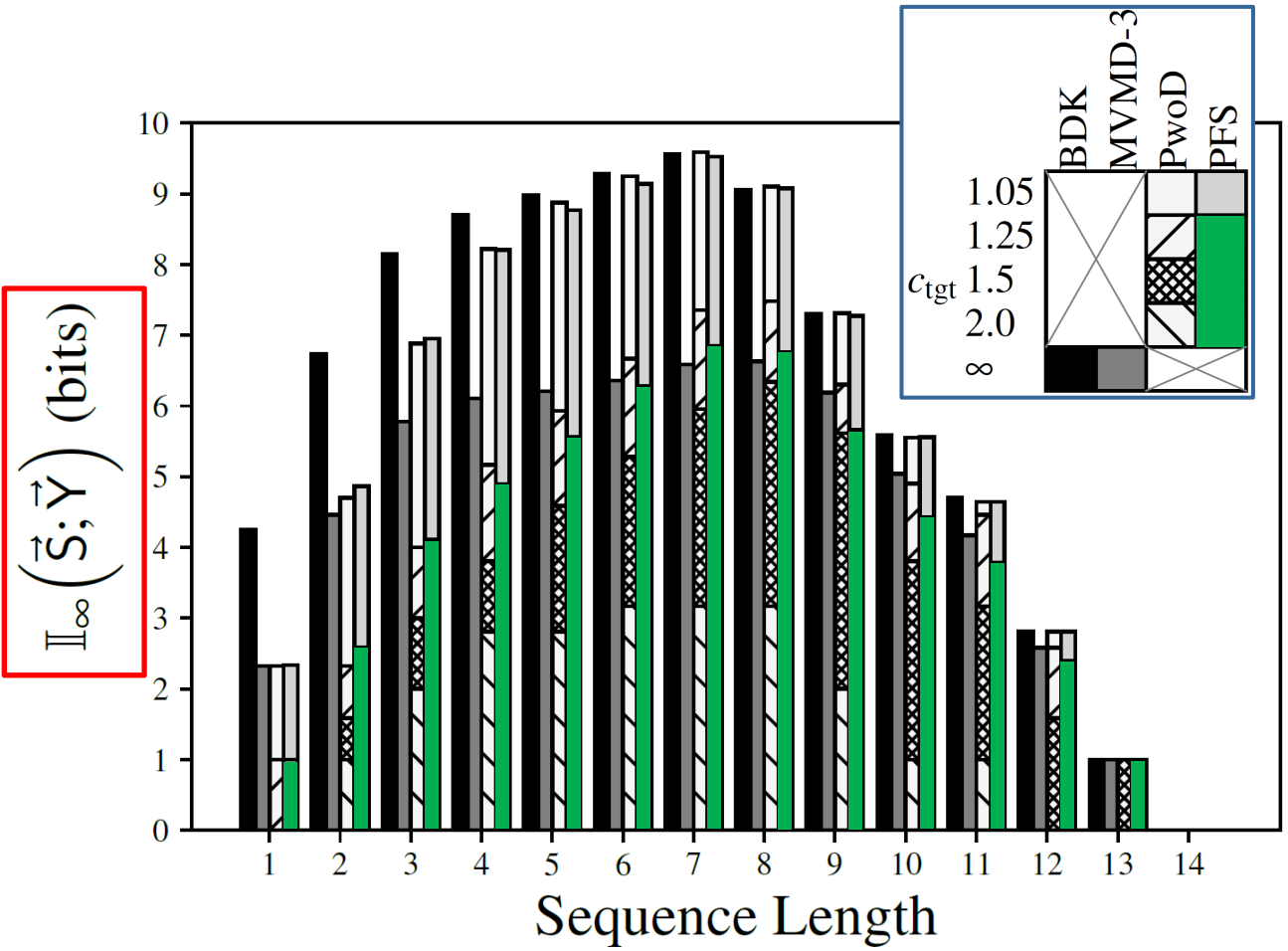


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

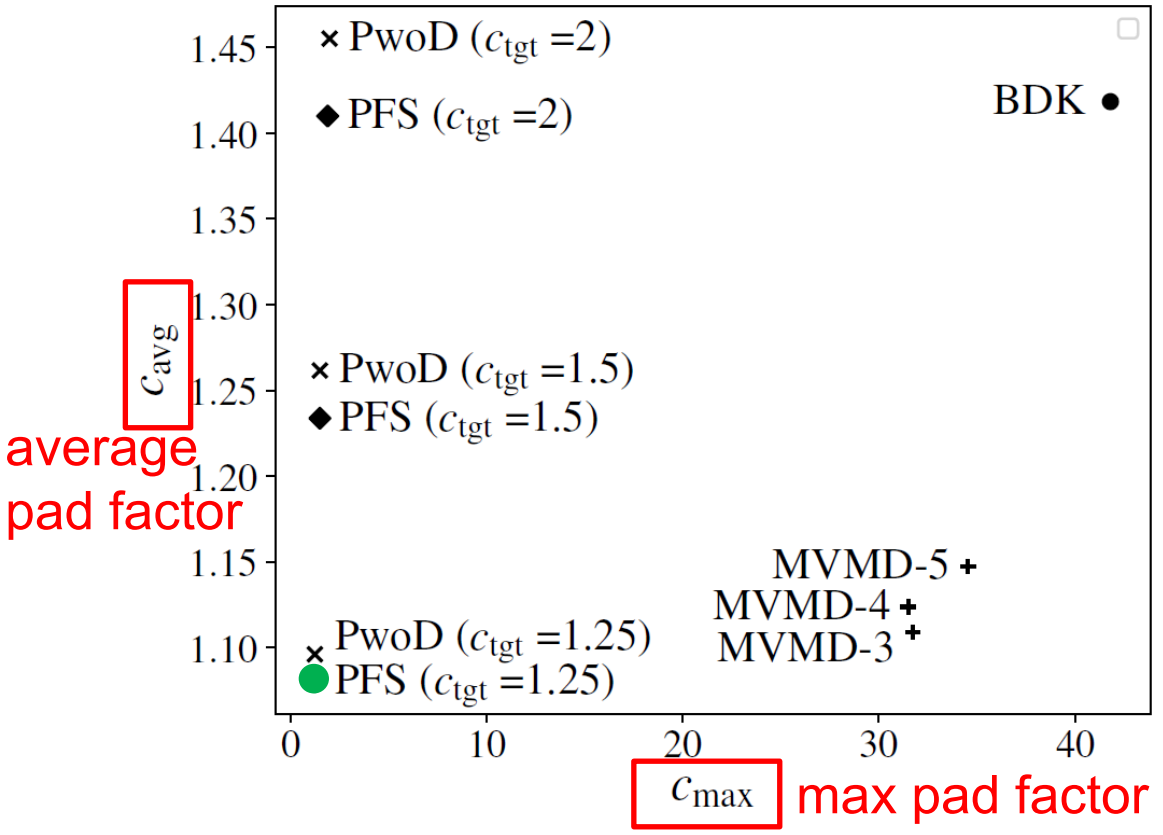


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

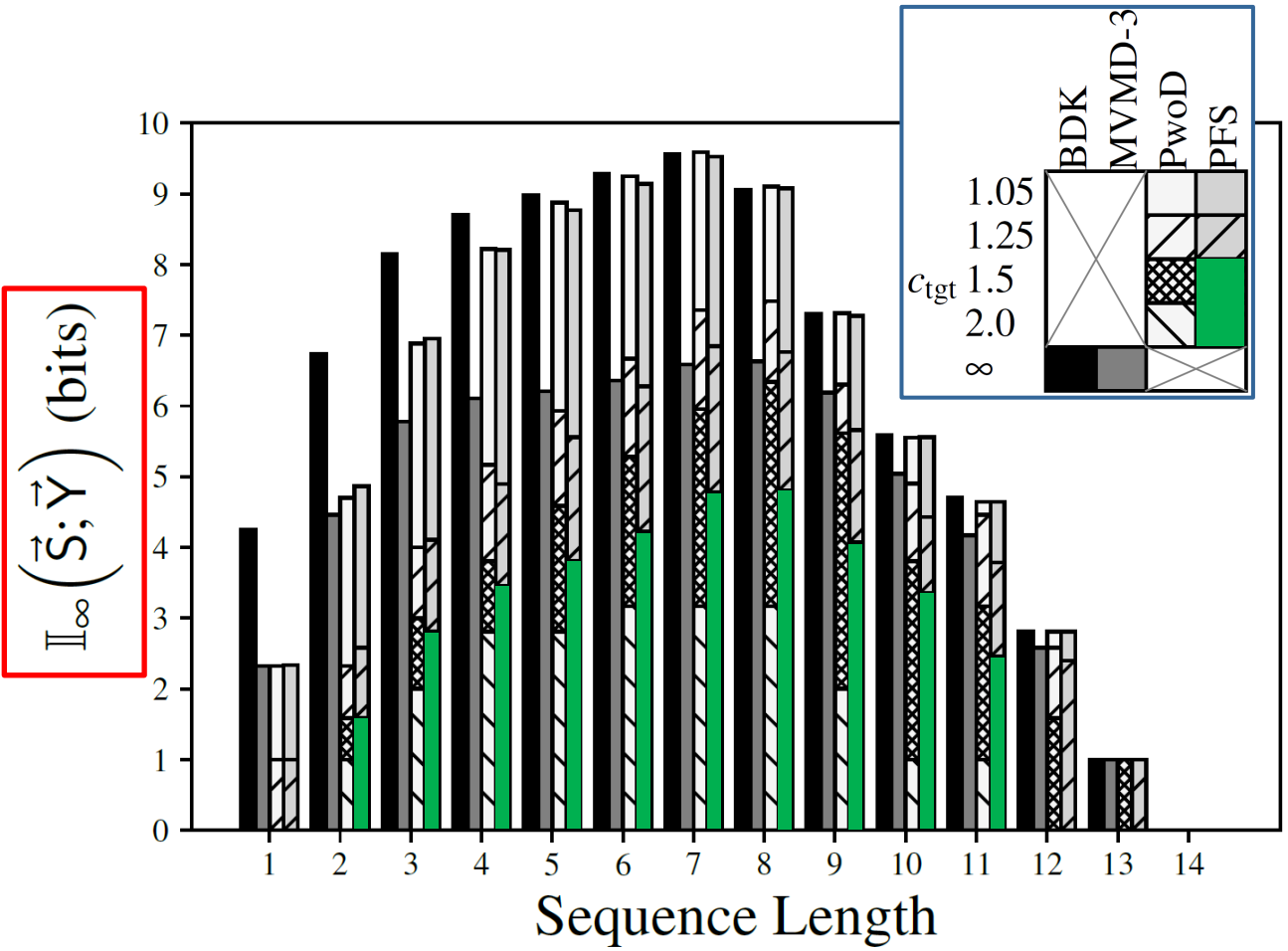


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

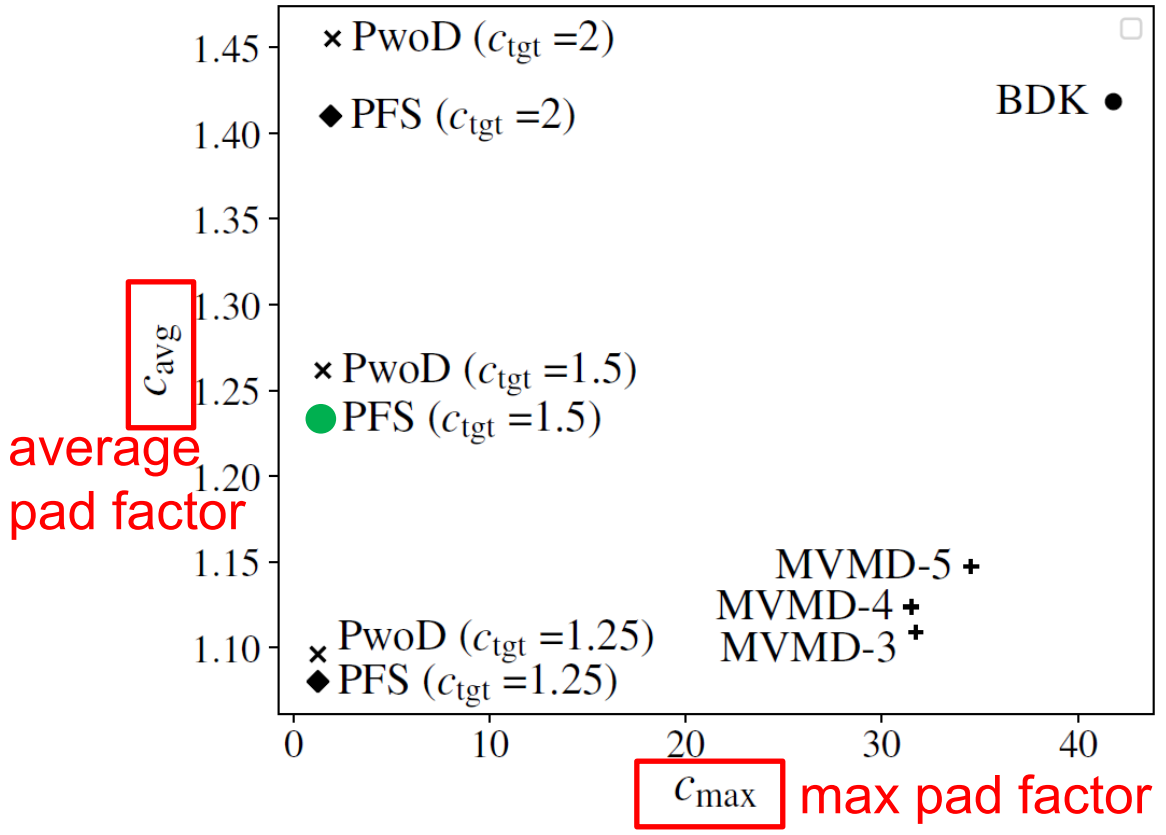


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

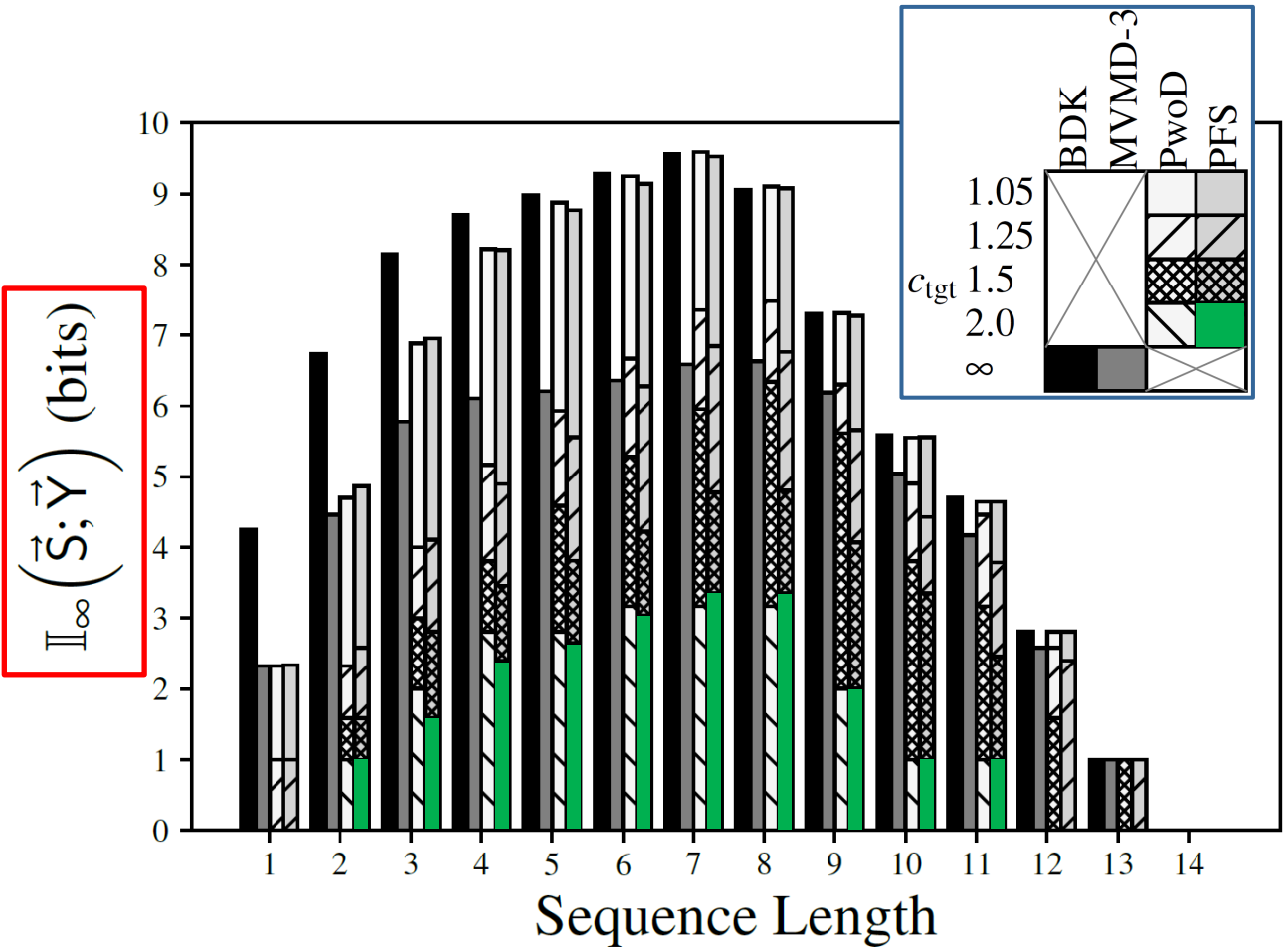


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

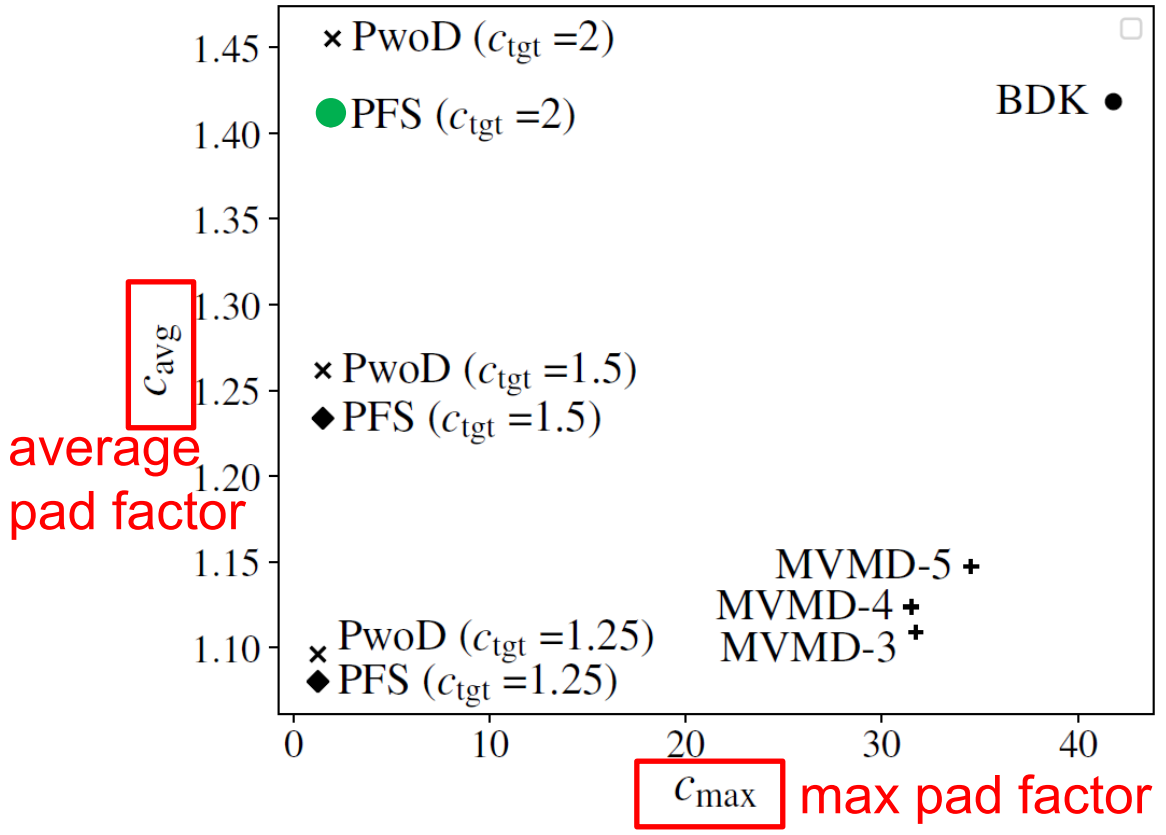


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$

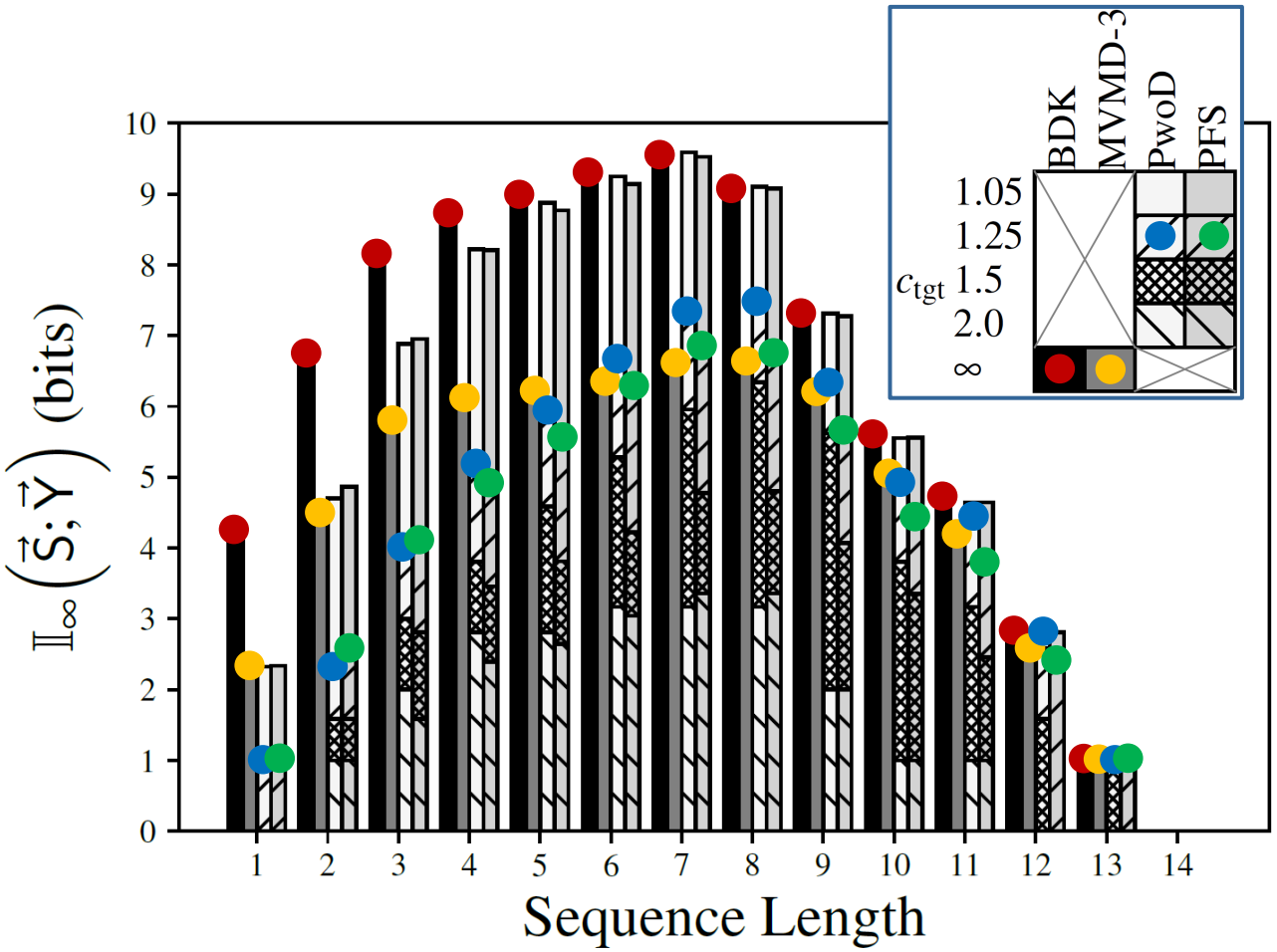


Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.

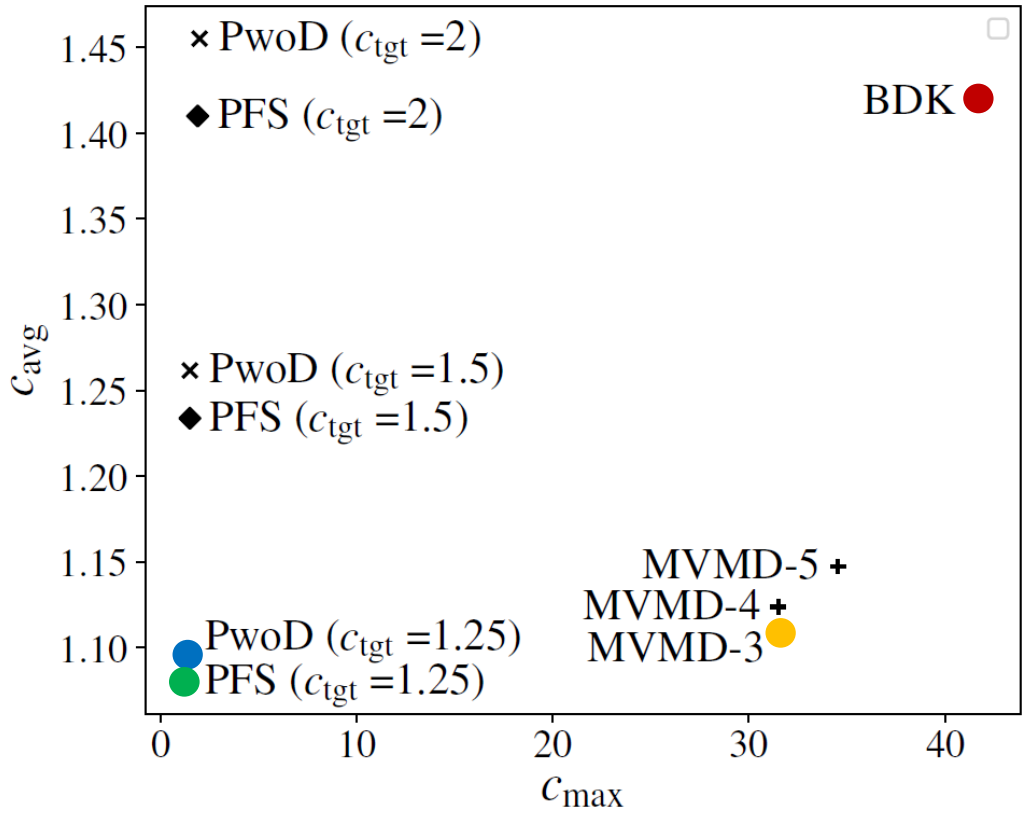


Padding overhead factors for each padding algorithm.

Evaluation: Autocomplete - $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$



Comparing all algorithms using $\mathbb{I}_\infty\left(\vec{S};\vec{Y}\right)$ as the privacy metric.



Padding overhead factors for each padding algorithm.

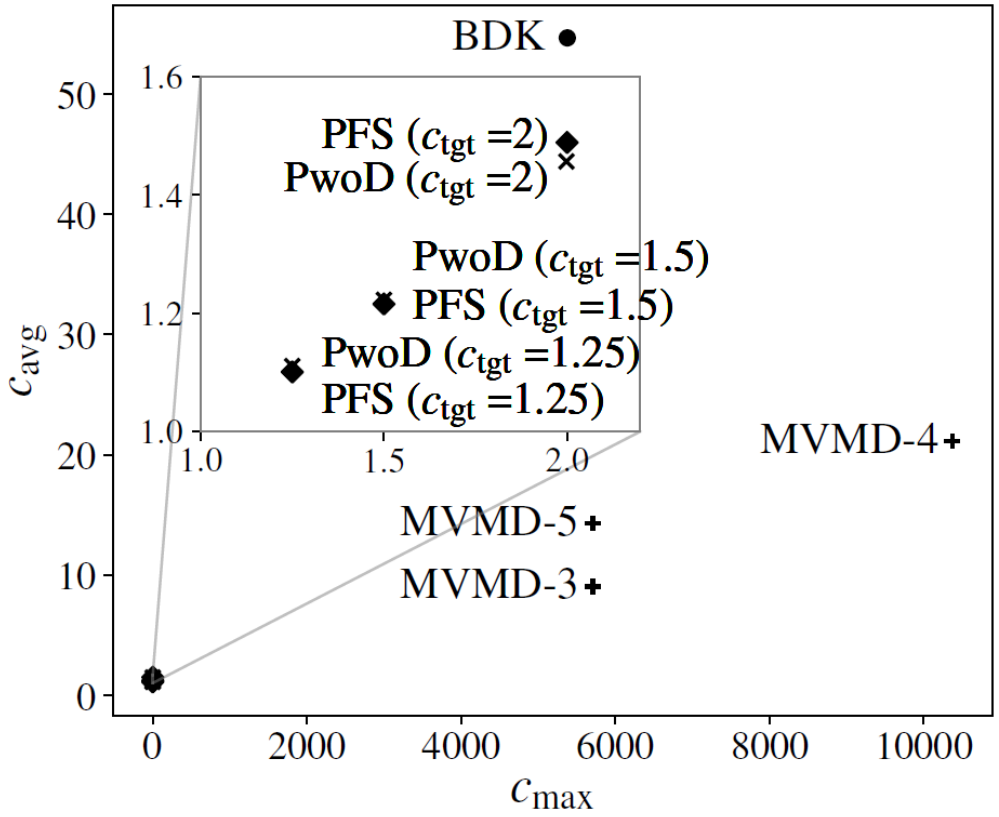
Additional Material in the Paper…

- Compare PFS against BDK and MVMD-D using *their* metrics
- Additional datasets used for evaluation
- Precision-Recall tests that model a network adversary
- Faster alternative to PFS named Padding For Graphs (PFG)

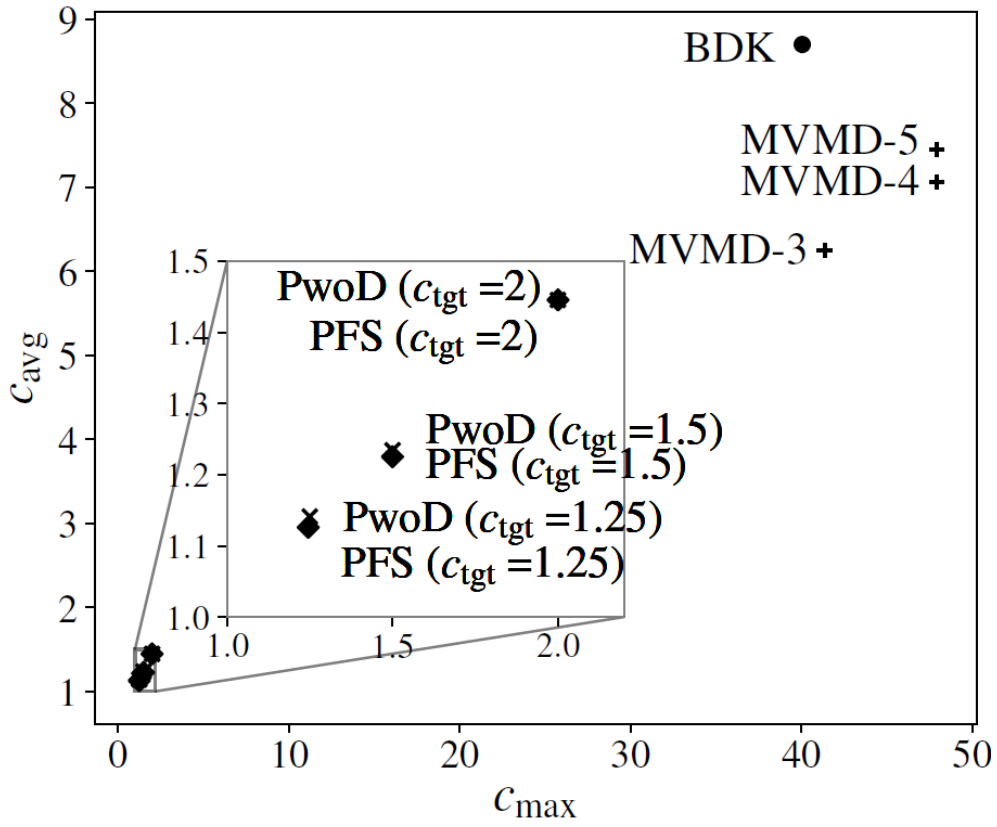
Questions?

Source Code and Datasets available at:
https://doi.org/10.5281/zenodo.13119687

Backup Slide: Padding Overhead



Linode



Wikipedia