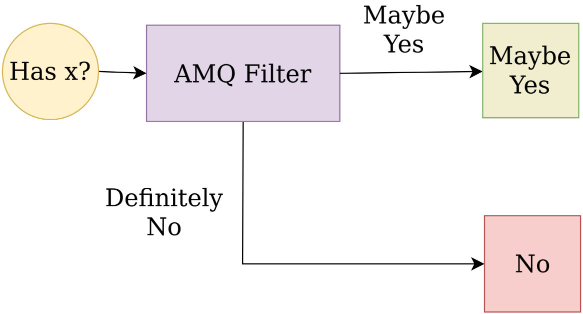
Tuning and Attacking Secure Learned Bloom Filters

Calvin Higgins

Department of Computer Science and Statistics
University of Rhode Island

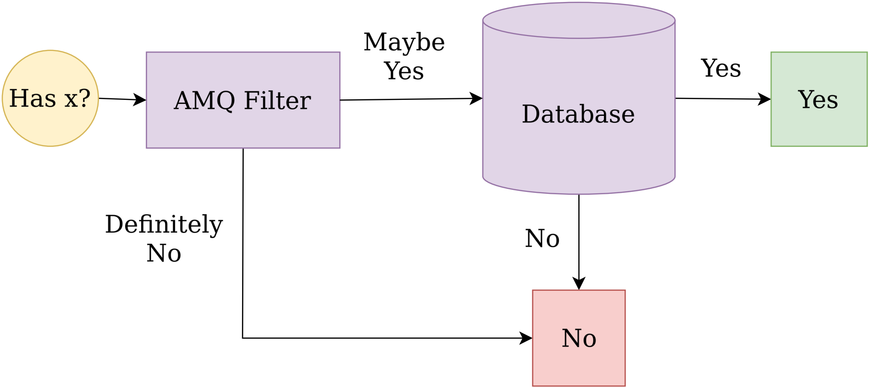
April 17, 2025

Approximate Membership Query (AMQ) Filters



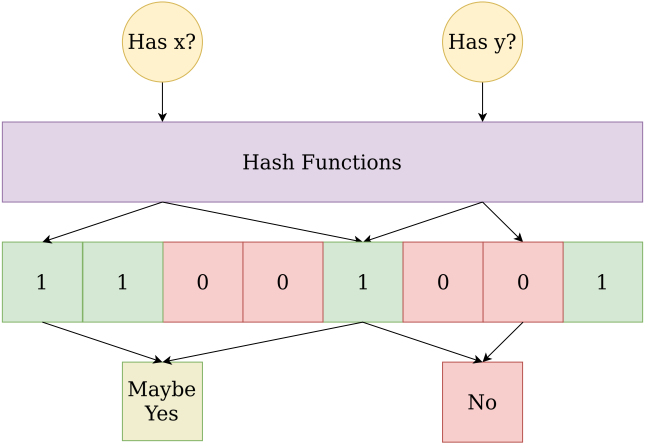
Idea: Check if $x \in S$ with false positive rate ϵ.

Approximate Membership Query (AMQ) Filters



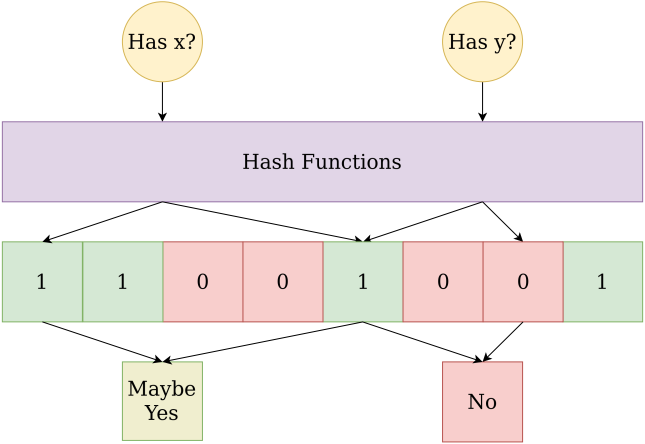
Motivation: Cheaply filter database queries to reduce load.

Bloom Filters (BFs) [2]



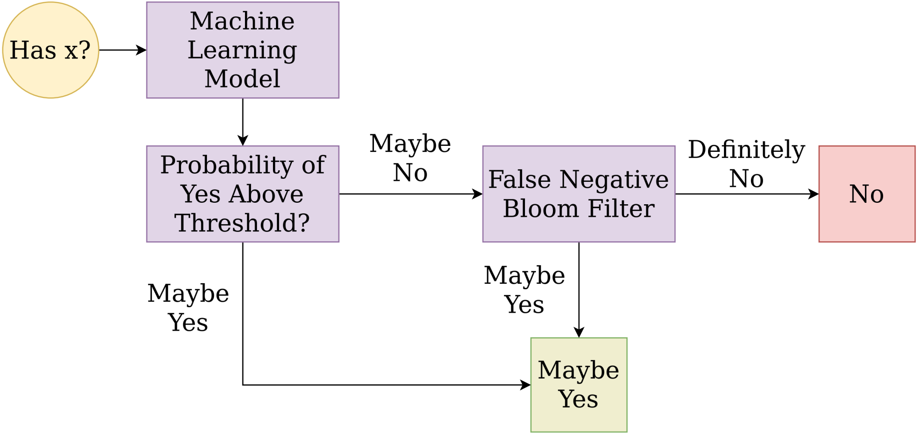
Idea: Output $x \in S$ iff x maps to all ones.

Bloom Filters (BFs) [2]



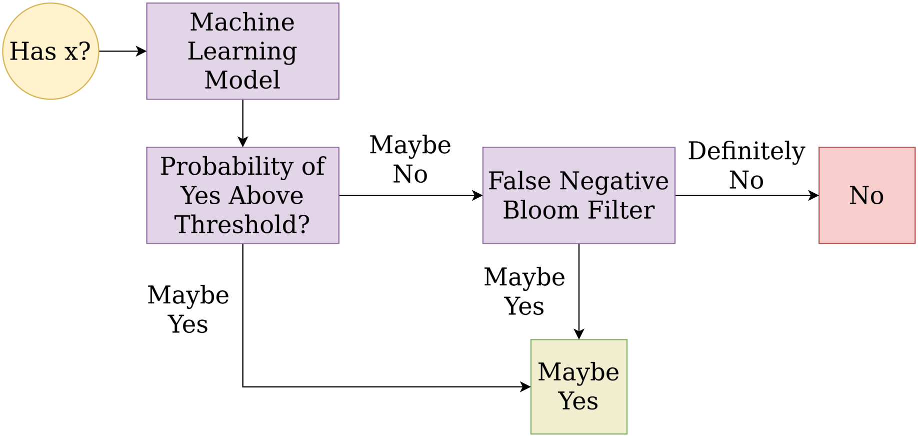
Parameters: The number of hash functions and bits.

Learned Bloom Filters (LBFs) [3]



Idea: Try machine learning model and then eliminate false negatives.

Learned Bloom Filters (LBFs) [3]



Parameters: The threshold and Bloom filter false positive rate.

Threat Model [1] [4]

Adversarial Objective

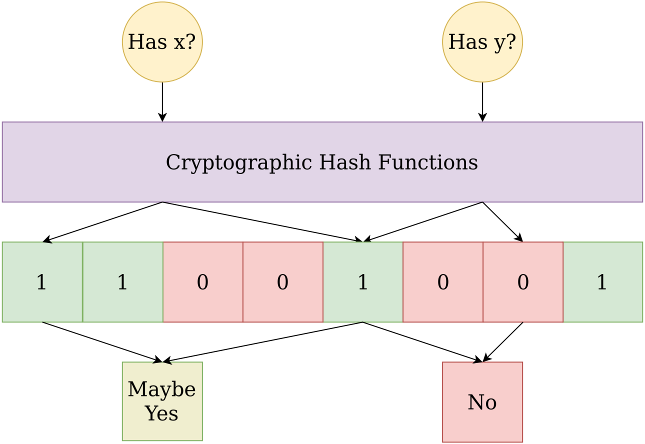
Create **new** false positives to reduce database performance.

Adversarial Capabilities

The adversary has

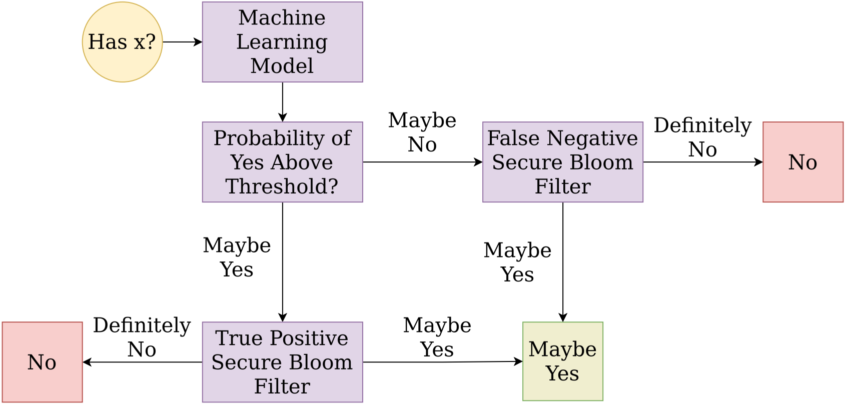
1. Polynomial time
2. Query access to the AMQ filter
3. White-box access to the AMQ filter
4. Total control over the set S

Secure Bloom Filters (SBFs) [4]

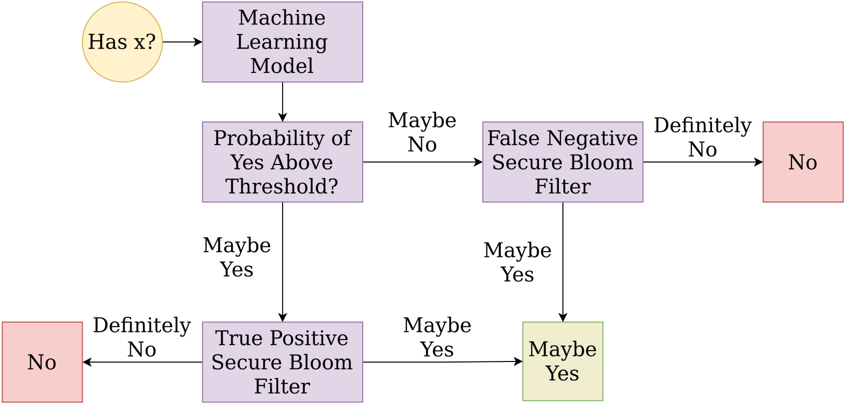


Idea: Replace hash functions with cryptographic hash functions.

Secure Learned Bloom Filters (SLBFs) [1]



Idea: Double-check machine learning model with secure Bloom filters.



Parameters: The threshold and secure Bloom filter false positive rates.

Tuning Secure Learned Bloom Filters (SLBFs)

False Positive Rate [1]

SBLFs have an expected false positive rate of

$$\epsilon = \mathrm{M}_{\mathrm{FPR}}\mathrm{TP}_{\mathrm{FPR}} + \mathrm{M}_{\mathrm{TNR}}\mathrm{FN}_{\mathrm{FPR}}$$

where M is the machine learning model, TP is the true positive filter and FN is the false negative filter.

Memory Footprint

SLBFs require

$$m = -\frac{n}{\ln(2)^2}\left(\mathrm{M}_{\mathrm{TPR}}\ln(\mathrm{TP}_{\mathrm{FPR}}) + \mathrm{M}_{\mathrm{FNR}}\ln(\mathrm{FN}_{\mathrm{FPR}})\right) + \mathrm{M}_m + 2\lambda$$

bits of memory.

Tuning Secure Learned Bloom Filters (SLBFs)

Optimal Secure Bloom Filter False Positive Rates

The optimal true positive and false negative filter false positive rates are

$$\mathsf{TP}^*_{\mathsf{FPR}} = \epsilon \frac{\mathsf{M}_{\mathsf{TPR}}}{\mathsf{M}_{\mathsf{FPR}}} \qquad \mathsf{FN}^*_{\mathsf{FPR}} = \epsilon \frac{\mathsf{M}_{\mathsf{FNR}}}{\mathsf{M}_{\mathsf{TNR}}}$$

where M is the machine learning model, TP is the true positive filter and FN is the false negative filter.

Proof Sketch

Express $\mathsf{FN}_{\mathsf{FPR}}$ in terms of $\mathsf{TP}_{\mathsf{FPR}}$. Set $\frac{\partial m}{\partial \mathsf{TP}_{\mathsf{FPR}}} = 0$ and solve for $\mathsf{TP}^*_{\mathsf{FPR}}$. Substitute to find $\mathsf{FN}^*_{\mathsf{FPR}}$.

Tuning Secure Learned Bloom Filters (SLBFs)

Optimal Threshold

The optimal threshold τ^* minimizes

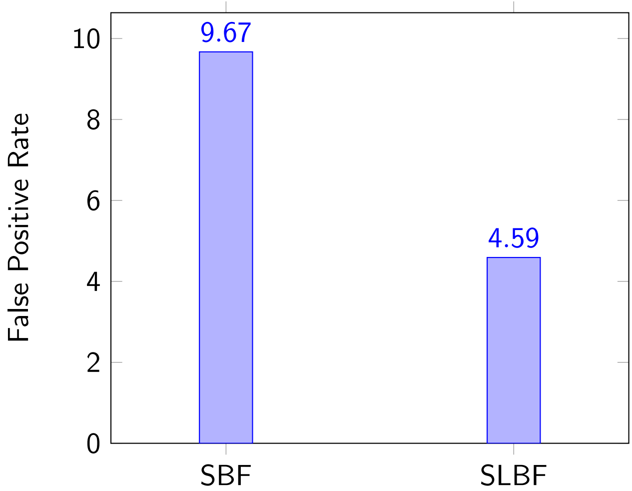
$$C(\tau) = -\mathsf{M}_{\mathsf{TPR}} \ln\left(\frac{\mathsf{M}_{\mathsf{TPR}}}{\mathsf{M}_{\mathsf{FPR}}}\right) - \mathsf{M}_{\mathsf{FNR}} \ln\left(\frac{\mathsf{M}_{\mathsf{FNR}}}{\mathsf{M}_{\mathsf{TNR}}}\right)$$

subject to

$$\max\left\{\epsilon\frac{\mathsf{M}_{\mathsf{TPR}}}{\mathsf{M}_{\mathsf{FPR}}}, \epsilon\frac{\mathsf{M}_{\mathsf{FNR}}}{\mathsf{M}_{\mathsf{TNR}}}\right\} \leq \epsilon_{\mathsf{max}}$$

and can be found in $\Theta(n)$ time with dynamic programming!

Tuning Secure Learned Bloom Filters (SLBFs)



SLBFs outperform SBFs under a fixed memory budget!

Attacking Secure Learned Bloom Filters (SLBFs)

Experimental Setup:

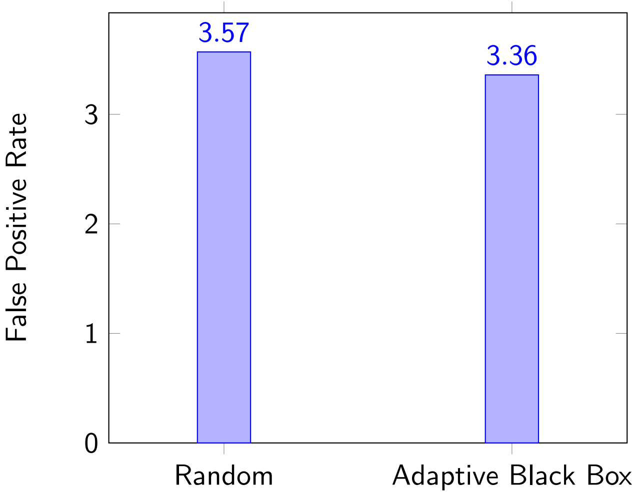
1. Truncate/pad $\sim$ 600K benign/malicious URLs to 128 bytes.
2. Train a shallow Transformer encoder for URL classification.
3. Compute optimal SLBF parameter values.
4. Construct a SLBF ($\epsilon = 0.05$, $\epsilon_{\text{max}} = 0.2$) from the set of malicious URLs.

Attacking Secure Learned Bloom Filters (SLBFs)

Adaptive Black Box Attack:

1. Create a synthetic SLBF
    1. Generate a random set of URLs.
    2. Label the URLs with the SLBF.
    3. Train a deep Transformer encoder on the labeled URLs.
2. Create adversarial examples
    1. Generate a new random set of URLs.
    2. Embed the URLs with the embedding layer.
    3. Optimize the URL embeddings with Adam to have positive class label.
    4. Unembed the URL embeddings with maximum cosine similarity.
    5. Remove any previously queried URLs with an AMQ filter.
    6. Return the optimized URLs.

Attacking Secure Learned Bloom Filters (SLBFs)



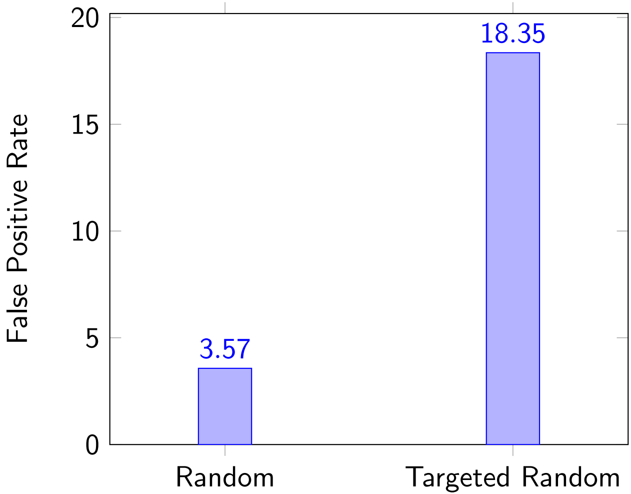
No significant difference! ($p = 0.1532, \chi^2 = 266$)

Attacking Secure Learned Bloom Filters (SLBFs)

Targeted Random Attack:

1. Generate a random set of URLs.
2. Label the random URLs with the learned model.
3. Split the URLs into two sets by label.
4. If the true positive filter has higher false positive rate, take the positive URL set, otherwise take the negative URL set.
5. Remove any previously queried URLs with an AMQ filter.
6. Return the optimized URLs.

Attacking Secure Learned Bloom Filters (SLBFs)



Significant but the defense succeeded ($\epsilon_{\mathbf{max}} = 0.2$)!

Summary of Contributions

Contributions:

1. Implemented Secure Bloom filters
2. Implemented Secure Learned Bloom filters
3. Derived optimal parameters for Secure Learned Bloom filters
    1. Closed-form solutions for $\mathrm{FN}_{\mathrm{FPR}}$ and $\mathrm{TP}_{\mathrm{FPR}}$
    2. Efficient dynamic programming algorithm for τ
4. Evaluated the robustness of Secure Learned Bloom filters against
    1. Adaptive black box attack
    2. Targeted randomized attack

References

[1] Allison Bishop and Hayder Tirmazi. "Adversary Resilient Learned Bloom Filters". In: (2025). arXiv: 2409.06556.

[2] Burton H. Bloom. "Space/time trade-offs in hash coding with allowable errors". In: *Commun. ACM* 13.7 (July 1970), pp. 422–426. ISSN: 0001-0782. DOI: 10.1145/362686.362692. URL: https://doi.org/10.1145/362686.362692.

[3] Tim Kraska et al. "The Case for Learned Index Structures". In: (2018).

[4] Moni Naor and Yogev Eylon. "Bloom Filters in Adversarial Environments". In: (2019). DOI: 10.1145/3306193.