ImageProof: Enabling Authentication for Large-Scale Image Retrieval

Shangwei Guo[1] Jianliang Xu[1] Ce Zhang[1] Cheng Xu[1] Tao Xiang[2]

[1]Department of Computer Science, Hong Kong Baptist University

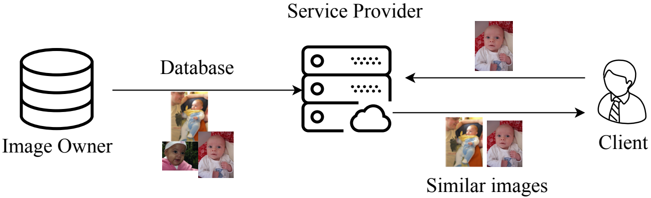
[2]College of Computer Science, Chongqing University

{csswguo,xujl,cezhang,chengxu}@comp.hkbu.edu.hk, txiang@cqu.edu.cn

ICDE 2019

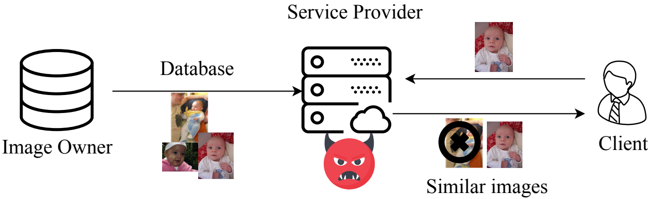
Background

- Content-based image retrieval (CBIR) has been widely used in business
- Data-as-a-Service (DaaS) enables companies to build and then outsource image retrieval systems to cloud platforms



Background

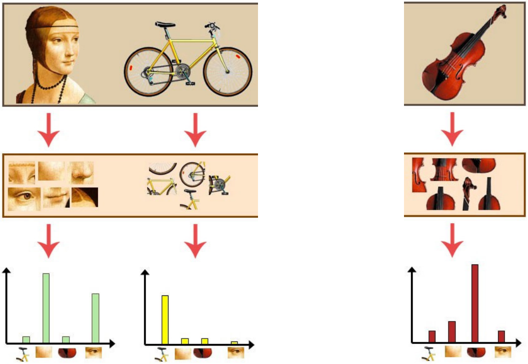
- Content-based image retrieval (CBIR) has been widely used in business
- Data-as-a-Service (DaaS) enables companies to build and then outsource image retrieval systems to cloud platforms



- **Security Threat**:
  - Query result integrity not guaranteed due to software/hardware malfunctions, hack attacks
  - Examples
    - Product image search
    - Medical image search

SIFT-Based Image Retrieval

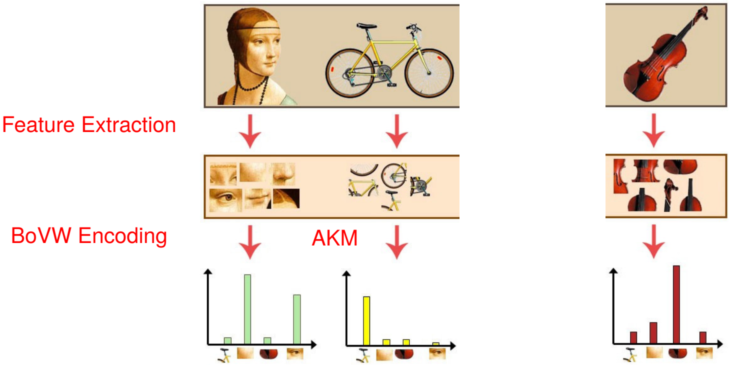
- Detect and extract local features using scale invariant feature transform (SIFT) and its variants



Feature Extraction

SIFT-Based Image Retrieval

- Detect and extract local features using scale invariant feature transform (SIFT) and its variants
- **Twe Steps**
  - Bag-of-visual-words (BoVW) encoding
    - Approximate k-means (AKM) using randomized k-d trees



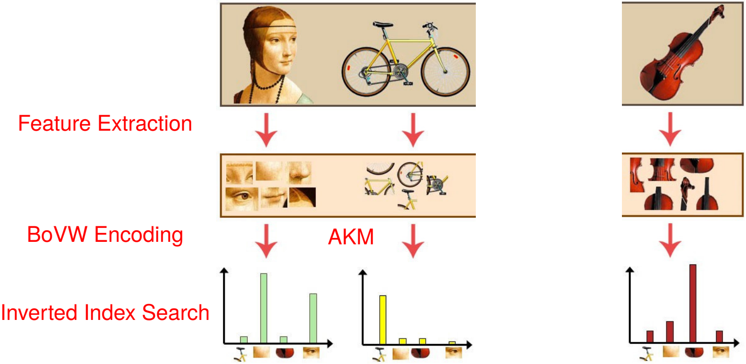
Feature Extraction

BoVW Encoding

AKM

SIFT-Based Image Retrieval

- Detect and extract local features using scale invariant feature transform (SIFT) and its variants
- **Twe Steps**
  - Bag-of-visual-words (BoVW) encoding
    - Approximate k-means (AKM) using randomized k-d trees
  - Inverted index search: search similar images with impact-ordered inverted index



Feature Extraction

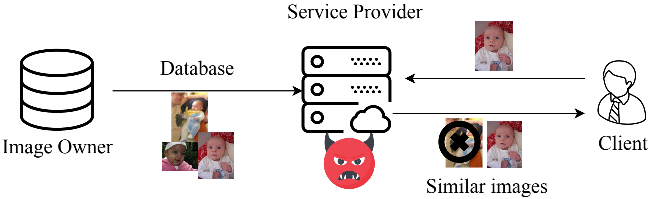
BoVW Encoding

AKM

Inverted Index Search

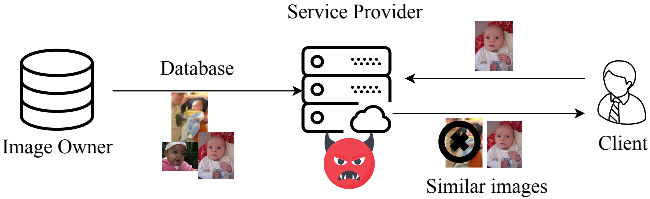
Problem Model

- Malicious threat model
- The service provider (SP) could return **incorrect** results (e.g., faked or low-ranked images)



Problem Model

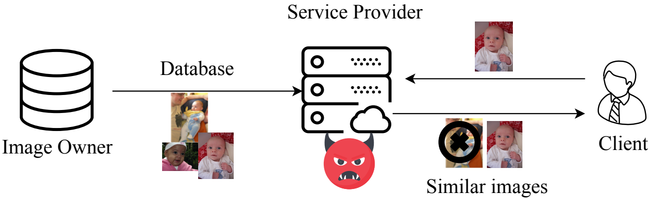
- Malicious threat model
- The service provider (SP) could return **incorrect** results (e.g., faked or low-ranked images)



- Query authentication for SIFT-based image retrieval and top-k query

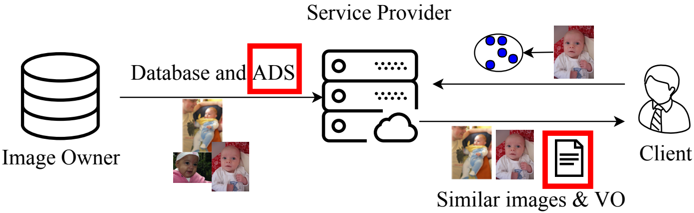
Problem Model

- Malicious threat model
- The service provider (SP) could return **incorrect** results (e.g., faked or low-ranked images)



- Query authentication for SIFT-based image retrieval and top-k query
- **Challenges**
  - Designing a query authentication scheme for a large and complex retrieval system is a big challenge in itself
  - The client usually has only limited storage, communication, and computation resources

Problem Model



- **Our Solution**:
  - Taking the advantage of the authenticated data structures (ADSs), the SP returns a verification object (VO) to prove
    - Soundness: The results must be the images which have not been tampered with
    - Completeness: The results include the k most similar images

Our Contributions

- Propose an efficient authentication scheme, ImageProof, for SIFT-based image retrieval with large or medium-sized codebooks

Our Contributions

- Propose an efficient authentication scheme, ImageProof, for SIFT-based image retrieval with large or medium-sized codebooks
- Two novel ADS components:
  - Merkle randomized k-d tree
  - Merkle inverted index with cuckoo filters

Our Contributions

- Propose an efficient authentication scheme, ImageProof, for SIFT-based image retrieval with large or medium-sized codebooks
- Two novel ADS components:
  - Merkle randomized k-d tree
  - Merkle inverted index with cuckoo filters
- Develop several optimization techniques to further reduce the costs of both the SP and the client

Preliminaries

- **Merkle Hash Tree**
    - An authenticated binary tree, enabling users to verify individual data objects without retrieving the entire database

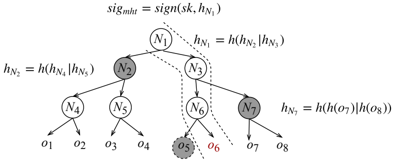

$sig_{mht} = sign(sk, h_{N_1})$
N_1
$h_{N_1} = h(h_{N_2}|h_{N_3})$
$h_{N_2} = h(h_{N_4}|h_{N_5})$
N_2
N_3
N_4
N_5
N_6
N_7
$h_{N_7} = h(h(o_7)|h(o_8))$
o_1
o_2
o_3
o_4
o_5
o_6
o_7
o_8

Figure 1: An example of a Merkle hash tree.

Preliminaries

- **Merkle Hash Tree**
  - An authenticated binary tree, enabling users to verify individual data objects without retrieving the entire database
- **Cuckoo Filter**
  - An efficient data structure for approximate set membership tests
    - Two hash values per item
    - Support delete operation

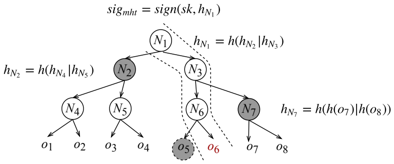

$sig_{mht} = sign(sk, h_{N_1})$
$h_{N_1} = h(h_{N_2}|h_{N_3})$
$h_{N_2} = h(h_{N_4}|h_{N_5})$
$h_{N_7} = h(h(o_7)|h(o_8))$
N_1
N_2
N_3
N_4
N_5
N_6
N_7
o_1
o_2
o_3
o_4
o_5
o_6
o_7
o_8

Figure 1: An example of a Merkle hash tree.

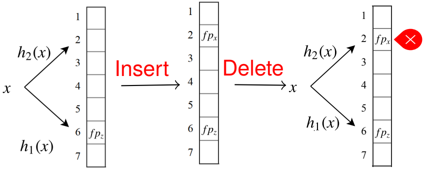
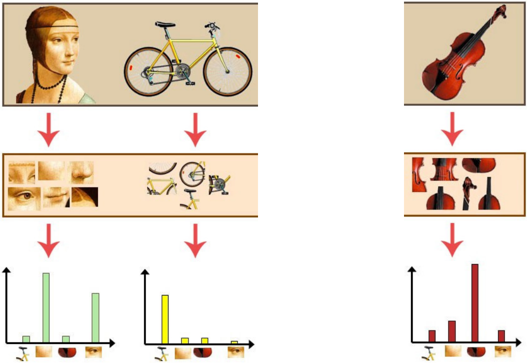


Figure 2: A cuckoo filter, two hash values per item.

Scheme Overview

- Ensure the integrity of query processing for each step

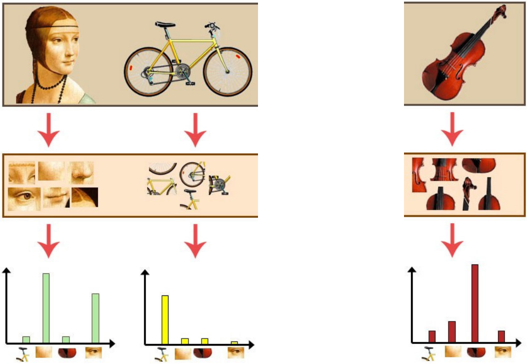


Authenticated BoVW Encoding

Authenticated Inverted Index Search

Scheme Overview

- Ensure the integrity of query processing for each step
- Two novel ADS components:
  - Merkle randomized k-d tree
  - Merkle inverted index with cuckoo filters



Merkle Randomized k-d Tree

Merkle Inverted Index

Merkle Randomlized k-d Tree (MRKD-tree)

- **ADS**
  - Internal nodes and leaf nodes

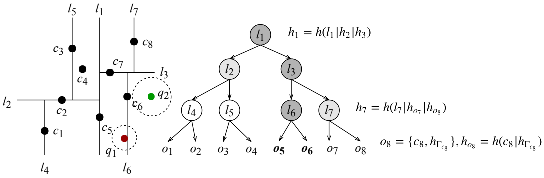


Figure 3: An example of the MRKD-tree and VO generation for query q_1, q_2.

Merkle Randomlized k-d Tree (MRKD-tree)

- **ADS**
  - Internal nodes and leaf nodes

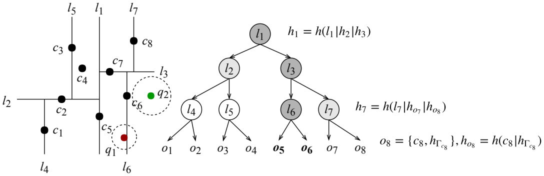

l_5
l_1
l_7
c_3
c_8
c_7
c_4
l_3
l_2
c_2
c_6
q_2
c_1
c_5
q_1
l_4
l_6
l_1
$h_1 = h(l_1|h_2|h_3)$
l_2
l_3
l_4
l_5
l_6
l_7
$h_7 = h(l_7|h_{o_7}|h_{o_8})$
o_1 o_2 o_3 o_4 o_5 o_6 o_7 o_8
$o_8 = \{c_8, h_{\Gamma_{c_8}}\}, h_{o_8} = h(c_8|h_{\Gamma_{c_8}})$

Figure 3: An example of the MRKD-tree and VO generation for query q_1, q_2.

- **Authenticated Query Processing**
  - Given a set of feature vectors, calculate the BoVW vector
  - Generate a single verification object (VO) for all feature vectors by maximizing the use of shared tree nodes

Merkle Inverted Index With Cuckoo Filters

- **ADS**
  - Each Merkle inverted list Γ_{c_i} consists of five components, i.e., the associated cluster c_i, the digest $h(\Theta_{c_i})$, the cluster weight w_{c_i}, the cuckoo filter Θ_i and its posting list

Table 1: An example of the Merkle inverted lists.

c_i	$h_{\Gamma_{c_i}}$	w_{c_i}	Θ_i	**Posting Lists**				
c_5	$h(2\sqrt{2}\vert h(\Theta_{c_5})\vert h_{pos_{5,1}})$	$2\sqrt{2}$	Θ_{c_5}	$\mapsto$	$\langle 1, 0.34, h_{pos_{5,1}} \rangle$	$\langle 3, 0.26, h_{pos_{5,2}} \rangle$	$\langle 4, 0.25, h_{pos_{5,3}} \rangle$	...
c_6	$h(\sqrt{2}\vert h(\Theta_{c_6})\vert h_{pos_{6,1}})$	$\sqrt{2}$	Θ_{c_6}	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$	$\langle 8, 0.32, h_{pos_{6,2}} \rangle$	$\langle 3, 0.28, h_{pos_{6,3}} \rangle$	...

Merkle Inverted Index With Cuckoo Filters

- **ADS**
  - Each Merkle inverted list Γ_{c_i} consists of five components, i.e., the associated cluster c_i, the digest $h(\Theta_{c_i})$, the cluster weight w_{c_i}, the cuckoo filter Θ_i and its posting list

Table 1: An example of the Merkle inverted lists.

c_i	$h_{\Gamma_{c_i}}$	w_{c_i}	Θ_i	**Posting Lists**				
c_5	$h(2\sqrt{2}\|h(\Theta_{c_5})\|h_{pos_{5,1}})$	$2\sqrt{2}$	Θ_{c_5}	$\mapsto$	$\langle 1, 0.34, h_{pos_{5,1}} \rangle$	$\langle 3, 0.26, h_{pos_{5,2}} \rangle$	$\langle 4, 0.25, h_{pos_{5,3}} \rangle$	...
c_6	$h(\sqrt{2}\|h(\Theta_{c_6})\|h_{pos_{6,1}})$	$\sqrt{2}$	Θ_{c_6}	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$	$\langle 8, 0.32, h_{pos_{6,2}} \rangle$	$\langle 3, 0.28, h_{pos_{6,3}} \rangle$	...

- **Authenticated Query Processing**
  - Find top-k most similar images and generate the VO of inverted index search
  - Ensure the integrity of top-k search with fewer postings with the help of cuckoo filters

Merkle Inverted Index With Cuckoo Filters

- **Main Idea**
  - Termination conditions:
    1. $s_k^L \geq S^U(Q, I)$, the upper bound of the similarity scores of the images popped, where s_k^L is the lower bound of the k-th similar score
    2. $s_k^L \geq$ the upper bound of the similarity scores of the images not popped

c_i	$h_{\Gamma_{c_i}}$	w_{c_i}	Θ_i	**Posting Lists**						
c_5	$h(2\sqrt{2}\|h(\Theta_{c_5})\|h_{pos_{5,1}})$	$2\sqrt{2}$	Θ_{c_5}	$\mapsto$	$\langle 1, 0.34, h_{pos_{5,1}} \rangle$	$\langle 3, 0.26, h_{pos_{5,2}} \rangle$	$\langle 4, 0.25, h_{pos_{5,3}} \rangle$	$\langle 10, 0.17, h_{pos_{5,4}} \rangle$	$\langle 7, 0.11, h_{pos5,5} \rangle$	...
c_6	$h(\sqrt{2}\|h(\Theta_{c_6})\|h_{pos_{6,1}})$	$\sqrt{2}$	Θ_{c_6}	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$	$\langle 8, 0.32, h_{pos_{6,2}} \rangle$	$\langle 3, 0.28, h_{pos_{6,3}} \rangle$	$\langle 6, 0.25, h_{pos_{6,4}} \rangle$	$\langle 4, 0.10, h_{pos_{6,5}} \rangle$	...

Merkle Inverted Index With Cuckoo Filters

- **Main Idea**
  - Termination conditions:
    1. $s_k^L \geq S^U(Q, I)$, the upper bound of the similarity scores of the images popped, where s_k^L is the lower bound of the k-th similar score
    2. $s_k^L \geq$ the upper bound of the similarity scores of the images not popped

c_i	$h_{\Gamma_{c_i}}$	w_{c_i}	Θ_i		**Posting Lists**					
c_5	$h(2\sqrt{2}\|h(\Theta_{c_5})\|h_{pos_{5,1}})$	$2\sqrt{2}$	Θ_{c_5}	$\mapsto$	$\langle 1, 0.34, h_{pos_{5,1}} \rangle$	$\langle 3, 0.26, h_{pos_{5,2}} \rangle$	$\langle 4, 0.25, h_{pos_{5,3}} \rangle$	$\langle 10, 0.17, h_{pos_{5,4}} \rangle$	$\langle 7, 0.11, h_{pos5,5} \rangle$	...
c_6	$h(\sqrt{2}\|h(\Theta_{c_6})\|h_{pos_{6,1}})$	$\sqrt{2}$	Θ_{c_6}	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$	$\langle 8, 0.32, h_{pos_{6,2}} \rangle$	$\langle 3, 0.28, h_{pos_{6,3}} \rangle$	$\langle 6, 0.25, h_{pos_{6,4}} \rangle$	$\langle 4, 0.10, h_{pos_{6,5}} \rangle$	...

  - Estimate the similarity bounds using the cuckoo filters

Table 2: Example: the postings for $S(Q, 5)$.

Without cuckoo filter:

$$\begin{aligned} S^U(Q,5) &\mapsto \langle 5, 0.41, h_{pos_{6,1}} \rangle, \langle 4, 0.25, h_{pos_{5,3}} \rangle \\ S^L(Q,5) &\mapsto \langle 5, 0.41, h_{pos_{6,1}} \rangle \end{aligned}$$

Merkle Inverted Index With Cuckoo Filters

- **Main Idea**
  - Termination conditions:
    1. $s_k^L \geq S^U(Q, I)$, the upper bound of the similarity scores of the images popped, where s_k^L is the lower bound of the k-th similar score
    2. $s_k^L \geq$ the upper bound of the similarity scores of the images not popped

c_i	$h_{\Gamma_{c_i}}$	w_{c_i}	Θ_i		**Posting Lists**					
c_5	$h(2\sqrt{2}\|h(\Theta_{c_5})\|h_{pos_{5,1}})$	$2\sqrt{2}$	Θ_{c_5}	$\mapsto$	$\langle 1, 0.34, h_{pos_{5,1}} \rangle$	$\langle 3, 0.26, h_{pos_{5,2}} \rangle$	$\langle 4, 0.25, h_{pos_{5,3}} \rangle$	$\langle 10, 0.17, h_{pos_{5,4}} \rangle$	$\langle 7, 0.11, h_{pos5,5} \rangle$	...
c_6	$h(\sqrt{2}\|h(\Theta_{c_6})\|h_{pos_{6,1}})$	$\sqrt{2}$	Θ_{c_6}	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$	$\langle 8, 0.32, h_{pos_{6,2}} \rangle$	$\langle 3, 0.28, h_{pos_{6,3}} \rangle$	$\langle 6, 0.25, h_{pos_{6,4}} \rangle$	$\langle 4, 0.10, h_{pos_{6,5}} \rangle$	...

  - Estimate the similarity bounds using the cuckoo filters

Table 2: Example: the postings for $S(Q, 5)$.

Without cuckoo filter:	$S^U(Q,5)$	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle, \langle 4, 0.25, h_{pos_{5,3}} \rangle$
	$S^L(Q,5)$	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$
With cuckoo filter:	$S^U(Q,5)$	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$
	$S^L(Q,5)$	$\mapsto$	$\langle 5, 0.41, h_{pos_{6,1}} \rangle$

ImageProof

- **ADS Generation**
  - Build Merkle inverted lists $\{\Gamma_{c_i}\}$ and MRKD-trees $\{\mathcal{T}_i\}$

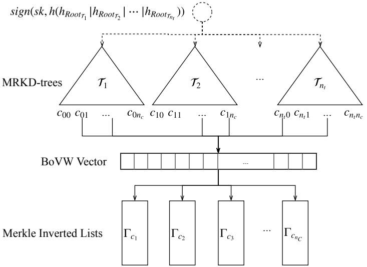

$sign(sk, h(h_{Root_{\mathcal{T}_1}}|h_{Root_{\mathcal{T}_2}}|\cdots|h_{Root_{\mathcal{T}_{n_t}}}))$
MRKD-trees
$\mathcal{T}_1$
$\mathcal{T}_2$
...
$\mathcal{T}_{n_t}$
c_{00} c_{01} ... c_{0n_c}
c_{10} c_{11} ... c_{1n_c}
c_{n_t0} c_{n_t1} ... $c_{n_tn_c}$
BoVW Vector
...
Merkle Inverted Lists
Γ_{c_1}
Γ_{c_2}
Γ_{c_3}
...
$\Gamma_{c_{n_C}}$

Figure 4: An overview of ADSs for ImageProof.

ImageProof

- **ADS Generation**
  - Build Merkle inverted lists $\{\Gamma_{c_i}\}$ and MRKD-trees $\{\mathcal{T}_i\}$
- **Authenticated Query Processing**
  - Search the top-k images and generate the VOs for both the BoVW encoding and the inverted index search
  - Send the VOs, together with the top-k results the client

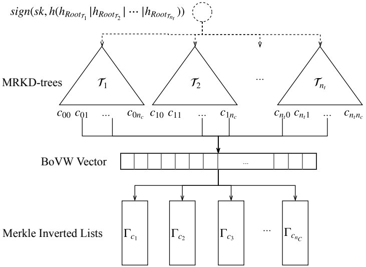
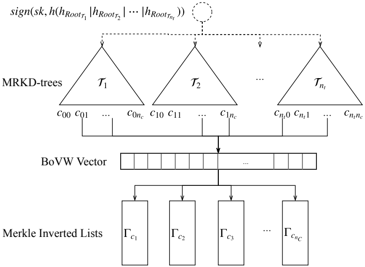

$sign(sk, h(h_{Root_{\mathcal{T}_1}} | h_{Root_{\mathcal{T}_2}} | \cdots | h_{Root_{\mathcal{T}_{n_t}}}))$
MRKD-trees
$\mathcal{T}_1$
$\mathcal{T}_2$
...
$\mathcal{T}_{n_t}$
c_{00} c_{01} ... c_{0n_c}
c_{10} c_{11} ... c_{1n_c}
c_{n_t0} c_{n_t1} ... $c_{n_tn_c}$
BoVW Vector
...
Merkle Inverted Lists
Γ_{c_1}
Γ_{c_2}
Γ_{c_3}
...
$\Gamma_{c_{n_C}}$

Figure 4: An overview of ADSs for ImageProof.

ImageProof

- **ADS Generation**
  - Build Merkle inverted lists $\{\Gamma_{c_i}\}$ and MRKD-trees $\{\mathcal{T}_i\}$
- **Authenticated Query Processing**
  - Search the top-k images and generate the VOs for both the BoVW encoding and the inverted index search
  - Send the VOs, together with the top-k results the client
- **Result Verification**
  - Check the integrity of image retrieval
  - Verify the integrity of raw image data

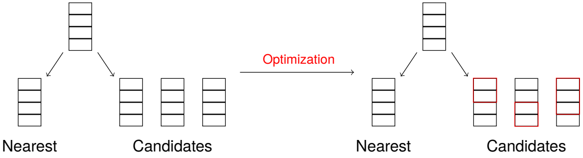


$sign(sk, h(h_{Root_{\mathcal{T}_1}} | h_{Root_{\mathcal{T}_2}} | \cdots | h_{Root_{\mathcal{T}_{n_t}}}))$
MRKD-trees
$\mathcal{T}_1$
$\mathcal{T}_2$
...
$\mathcal{T}_{n_t}$
c_{00} c_{01} ... c_{0n_c}
c_{10} c_{11} ... c_{1n_c}
c_{n_t0} c_{n_t1} ... $c_{n_tn_c}$
BoVW Vector
...
Merkle Inverted Lists
Γ_{c_1}
Γ_{c_2}
Γ_{c_3}
...
$\Gamma_{c_{n_C}}$

Figure 4: An overview of ADSs for ImageProof.

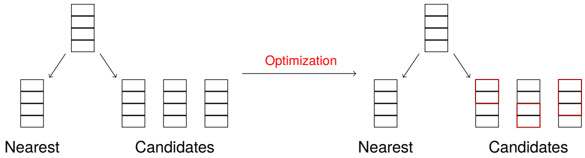
Optimization

- Compressing nearest neighbor candidates



Optimization

- Compressing nearest neighbor candidates



- Frequency-grouped inverted index

Component	Value
c_i	c_5
$h_{\Gamma^f_{c_i}}$	$h(2\sqrt{2}\vert h(\Theta_{c_5})\vert h^f_{pos_{5,1}})$
w_{c_i}	$2\sqrt{2}$
Θ_{c_i}	Θ_{c_5}
Posting List	$\langle 1, \mathbf{0.34}, h_{pos_{5,1}}\rangle$ $\langle 3, \mathbf{0.26}, h_{pos_{5,2}}\rangle$ $\langle 4, \mathbf{0.25}, h_{pos_{5,3}}\rangle$ $\ldots$

$\xrightarrow{\text{Optimization}}$

Component	Value
c_i	c_5
$h_{\Gamma^f_{c_i}}$	$h(2\sqrt{2}\vert h(\Theta_{c_5})\vert h^f_{pos_{5,1}})$
w_{c_i}	$2\sqrt{2}$
Θ_{c_i}	Θ_{c_5}
Posting List	$\langle \mathbf{4}, (1, 33.3; 10, 66.6), h_{pos^f_{5,1}}\rangle$ $\langle \mathbf{5}, (3, 54.4), h_{pos^f_{5,2}}\rangle$ $\langle \mathbf{3}, (4, 33.9; 7, 77.1; 2, 94.3), h_{pos^f_{5,3}}\rangle$ $\ldots$

Performance Evaluation

- **Experimental Setup**
  - Dataset: MirFlickr1M
  - Algorithms
    - **Baseline**: The scheme that combines the proposed MRKD-trees without sharing nodes and the authenticated inverted index search in PVLDB2008
    - **ImageProof**: The proposed scheme
    - **Optimized**: The optimized ImageProof

BoVW Performance

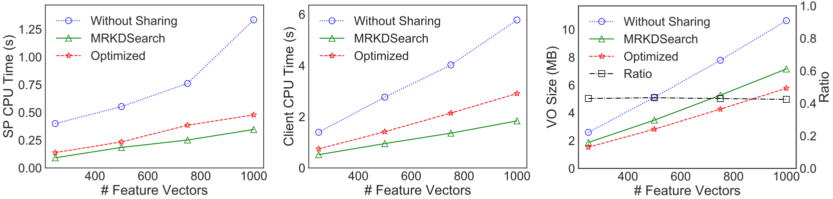


Figure 5: BoVW performance as the number of feature vectors increases.

Overall Performance

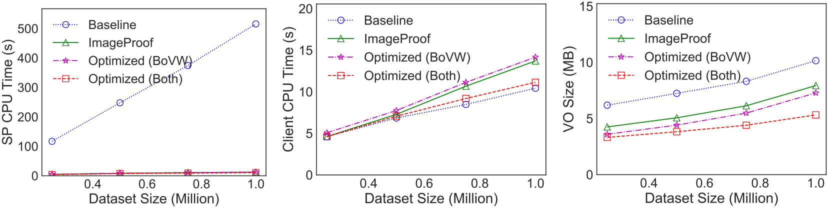


Figure 6: Overall performance as dataset size increases.

Summary

- Focus on the query authentication problem in SIFT-based image retrieval
- Two authenticated data structures (ADSs) for both BoVW encoding and inverted index search
- Extensive experiments on real-world image dataset

Thanks

Q&A