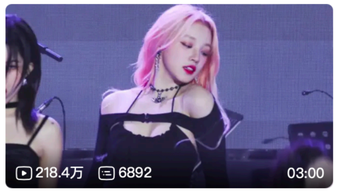
Encrypted Video Search: Scalable, Modular, and Content-similar

Yu Zheng, Heng Tian, Minxin Du, Chong Fu

Special thanks go to Sherman S.M. Chow
Reference: Jiafan Wang's slides on DBSec21; Sherman Chow's course slides

香港中文大學
The Chinese University of Hong Kong
信息工程學系
Department of Information Engineering

Popular Public Video Sharing



宋雨琦–TOMBOY 庆熙大学校庆 4k站子直拍 这场真的绝了！220526

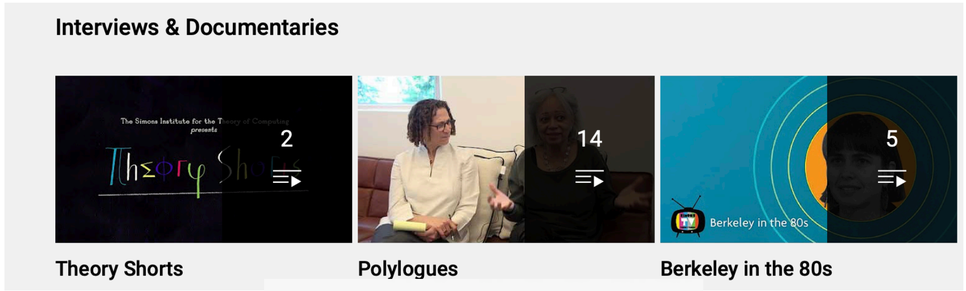
12万点赞 珍珠乌龙奶茶

《千与千寻》我唱国语 她唱日语–耳机听有惊喜

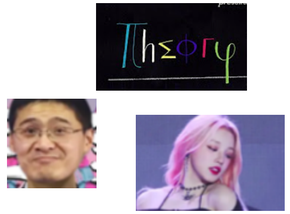
UP 张小伙isme · 5–28

《你没吃过的粽子》

睫毛弯弯，刑期加啊加



Share between targeted friends via public network

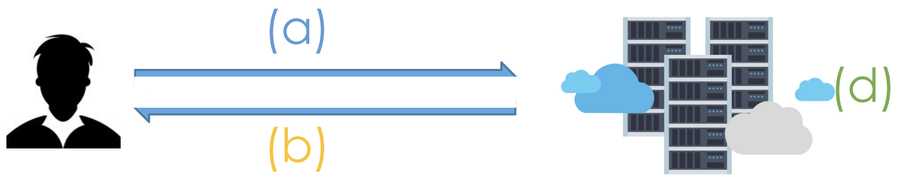


Sensitive information (Face, knowledge, identifiable records, etc)

Encrypted?
Searchable?

Trivial Retrieval of Encrypted Videos

- Download all *encrypted* videos (EVdio), then decrypt;
  - O(N) communication; N: #documents
  - If have storage for EVdio, why outsource them? 😱
    - O(ctxt) > O(Ptxt)
- Index EVdio, then download and decrypt
  - O(N) storage (Still large)



- Ideally, download n videos matching keyword w 😄
  - (a) |token|
  - (b) O(n) communication (n << N)
  - (d) Sub-linear search

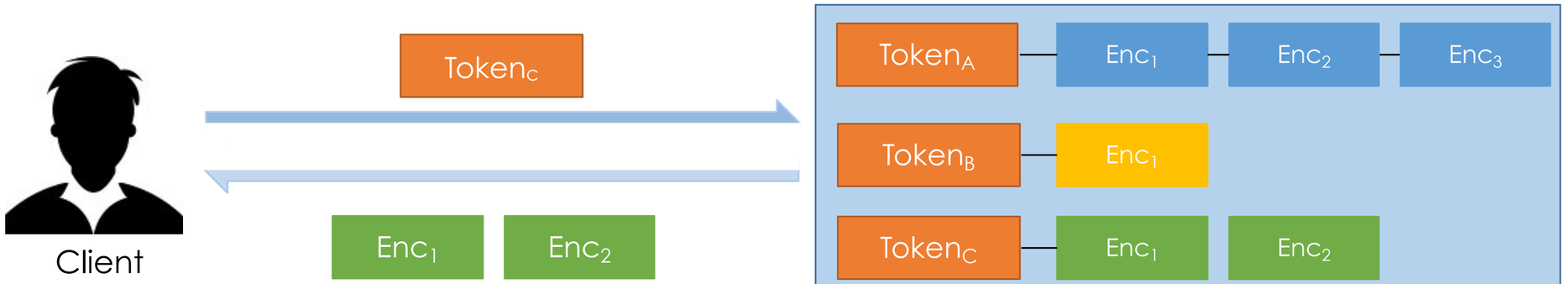
Symmetric Searchable Encryption (SSE)

- Client: upload EVdio to an *untrusted* server

- Client: send a token (of w)
- Server: search after obtaining the token

Which files contain keyword w ???

- Security (semi-honest)
  - Only allow a few Leakages (e.g., encryption size)
  - Adaptively secure, forward and backward secure



SSE Framework

Why Non-trivial to Integrate SSE and Video Retrieval

- Scalable for large database
  - Heavy computation on ctxt (>> computation on ptxt)
    - fully/somewhat HE: primitive operations (+ / *) only
  - Employ lightweight crypto. building on pseudorandom function (PRF)

- Content-similar videos to a given query
  - Meaningless to search a video exact to another one
  - Standard SSE tailored for textual data

- Modular design for easy adoption
  - The first framework for searching EVdio
  - Encapsulate algorithms/protocols
  - Strategically integrate SSE with video retrieval techniques

State of Arts

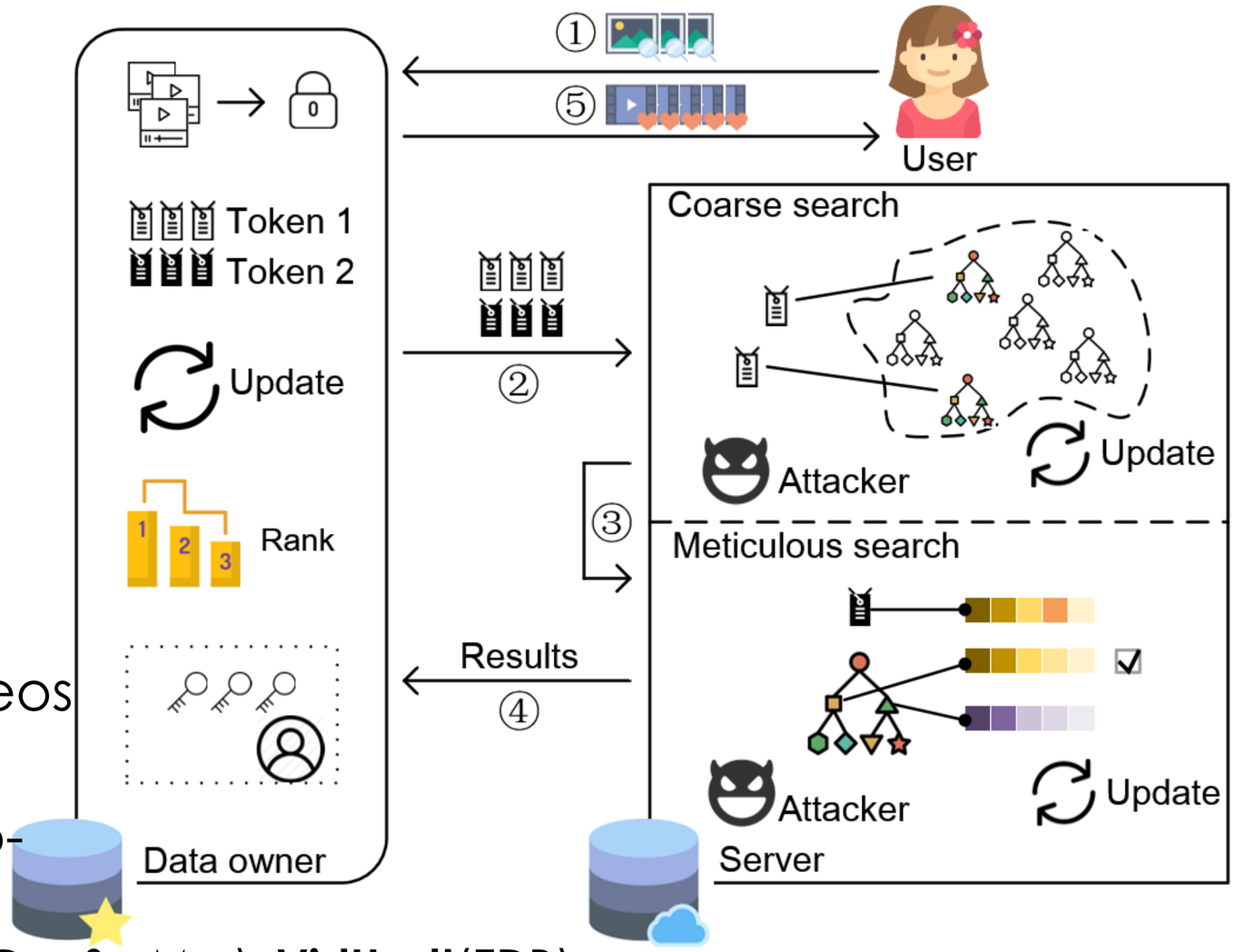
- Keyword SSE [KPR12][LC17]
  - Descriptive enough to match a video
  - Large #w
- Similarity search [LZW+18][WHD+18]
  - Search linearly over entire database
  - Heavy computing using homomorphic encryption
- ORAM-based SSE [GMP16]
  - Large communication overhead
- Weak security [LZW+18][WHD+18]
  - Leakage-abuse attack [CGPR15], File-injection attack [ZKP16]
  - Need forward/backward privacy

Image-to-Video Encrypted Search Framework

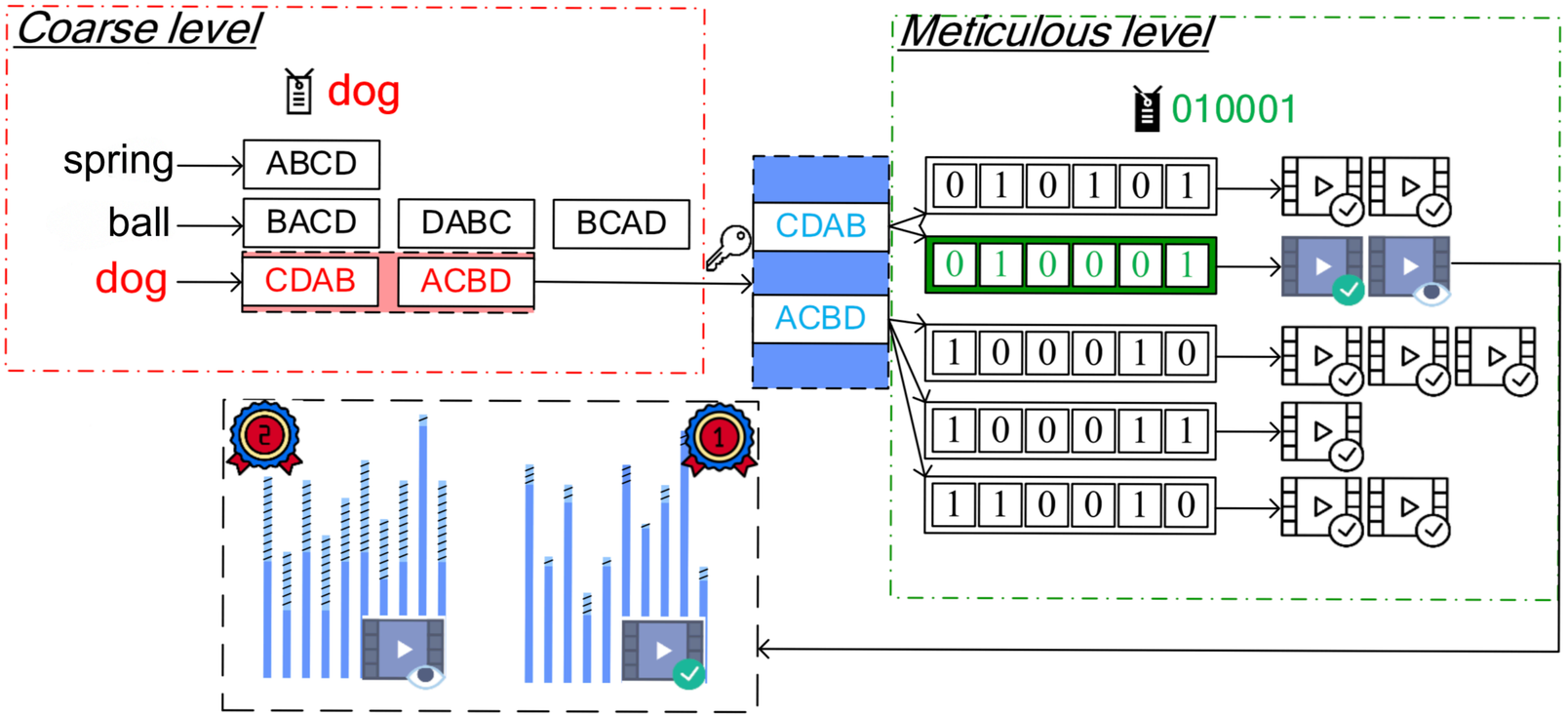
- User, Data owner, Server
- Hierarchical search
  - Coarse: general topics
  - Meticulous: visual similarity
- Rank for top-k similar videos

- Database →Multiple subsets
  - Identified by topics
- Subsets → Multiple visual representations to group videos

- New syntax: secure image-to-video framework (SItVF)
- **VidStp**(**D**); **HichSrch**(*q*; **EDB**); **Rank**(**D***q*, *SmMm*); **VidUpdt**(**EDB**)



An Illustrative Example for SItVF



- *Given q* → subset identifiers → file identifies (matching similar videos)
- *Ranking: measuring similarity between feature representations*

Security of Our Framework

- Security for standard SSE — adaptive security
- SItVF security: adopt adaptive security twice
  - Both coarse/meticulous searches satisfy adaptive security
  - No extra leakage when subsets are located
- Correctness of SItVF
  - Correct subset identifies + correct top-k similar videos

Modularization for Various Instantiations

- **VidStp(D)**
  - **W**←**SemExtra**(**D**), **G**←**VisExtra**(**D**), **R**$_i$←**VisRepGen**(**G**,D$_i$)
  - ((K, K*, σ, σ*); **EDB**)←**Setup**(λ,**DB**;$\perp$)
- **HichSrch**(q;**EDB**)
  - (**w**$_q$, **r**$_q$)←**WRquery**(q)
  - ((σ'**DB**(w$_q$));**EDB'**)←**CoarSearch**(K,σ,w$_q$; **EDB**)
  - ((σ'*,**DB**(r$_q$));**EDB'***)←**MetiSearch**(K*,σ*,r$_q$; **EDB'**
- **Rank**(**D**$_q$,SmMm)
  - Sim=SmMm(D$_q$,**G**$_q$); **D**$_q^{\text{top-}k}$=**Sort**(**D**$_q$,Sim)
- **VidUpdt**(**EDB**)
  - While op: (σ',σ'*;**EDB'**)←**Update**(K,K*,σ,σ*,op,in; **EDB**)

Cooking speciality whatever you expect

Method	Common Use	Work
Keyword split	texts, emails	[3, 7, 32, 36]
Traditional hash	texts, images	

Method	Recomm. Type
Feature extract	images, audios
Deep hashing	images, audios, vid
Cross-modal	text↔image/video
	image↔video

Table 2: Multimedia Process

Scheme	Comp-S	Comp-U
Dsse [32]	$O(n_w)$	-
Moneta [3]	$O(a_w \log N + \log^3 N)$	$O(\log^2 N)$
Diana [3]	$O(a_w)$	$O(\log a_w)$
Mitra [7]	$O(a_w)$	$O(1)$
Orion [7]	$O(n_w \cdot \log^2 N)$	$O(\log^2 N)$
Horus [7]	$O(n_w \cdot \log d_w \cdot \log N)$	$O(\log^2 N)$
Sd [17]	$O(a_w + \log N)$	$O(\log^3 N)$
Qos [17]	$O(n_w \log i_w + \log^2 \lvert W \rvert)$	$O(\log^3 N)$

Scheme	Comm-S	Comm-U
Moneta [3]	$\widetilde{O}(a_w \log N + \log^3 N)$	$\widetilde{O}(\log^3 N)$
Diana [3]	$O(n_w + d_w \cdot \log a_w)$	$O(1)$
	(a_w)	$O(1)$
	$\cdot \log^2 N)$	$O(\log^2 N)$
	$\cdot \log^2 N)$	$O(\log^2 N)$
	$+ \log N)$	$O(\log^3 N)$
	$_w + \log^2 \lvert W \rvert)$	$O(\log^3 N)$

nmunication of Prevailing SSE
er of document-keyword pairs,

Measurement	Formula	E.g.
Hamming Distance	$\sum_i U_i \oplus V_i$	[38]
Jaccard Coefficient	$\lvert \mathbf{U} \cap \mathbf{V} \rvert / \lvert \lvert \mathbf{U} \rvert + \lvert \mathbf{V} \rvert - \lvert \mathbf{U} \cap \mathbf{V} \rvert \rvert$	Sec. 5
Cosine Similarity	$(\sum_i U_i V_i)/(\sqrt{\sum_i U_i^2} \cdot \sqrt{\sum_i V_i^2})$	[25]
Dice's Coefficient	$2\lvert \mathbf{U} \cap \mathbf{V} \rvert / (\lvert \mathbf{U} \rvert + \lvert \mathbf{V} \rvert)$	[18]
Euclidean Distance		

Table 4

Security Type	Leakage Function	
	$\mathcal{L}_{\text{Search}}$	$\mathcal{L}_{\text{Update}}$
FP	$(\text{sp}(w), \text{UpHist}(w))$	$\mathcal{L}'(\text{op}, \text{ind})$
FP + BP Type-I	$\mathcal{L}''(\text{TimeDB}(w), a_w)$	$\mathcal{L}'(\text{op})$
FP + BP Type-II	$\mathcal{L}''(\text{TimeDB}(w), \text{Updates}(\mathbf{w}))$	$\mathcal{L}'(\text{op}, w)$
FP + BP Type-III	$\mathcal{L}''(\text{TimeDB}(w), \text{DelHist}(\mathbf{w}))$	$\mathcal{L}'(\text{op}, w)$

Table 5: Leakages for Updatable Database. FP is forward privacy, BP is backward privacy. $\mathcal{L}'$, $\mathcal{L}''$ are stateless functions. TimeDB(w) denotes all timestamps of updating w excluding deleted ones. Updates(w) returns all timestamps of update queries to w. DelHist(w) returns all deletion operations to w and timestamp of the inserted entry it removes.

No instantiation is perfect, but rational one exists for particularity.

It is effected by media types, computational resource, communication tolerance, client storage, and desirable security.

Our Basic Instantiation

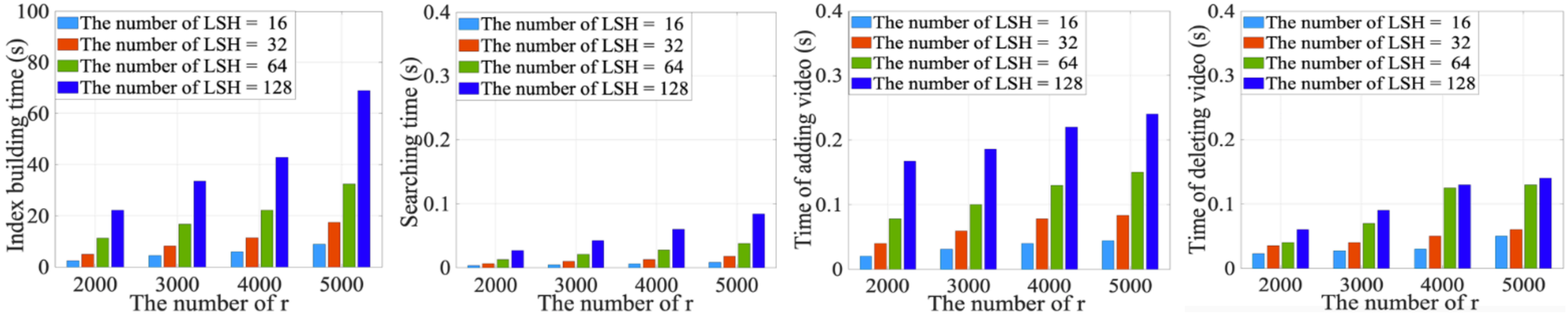
- Low computation + adaptive security

- SIFT[L04], BoVW[CDF+04], LSH[IM98] → **W** and **R**
  - **W**: semantic keyword set; **R**: visual representation set
- DSSE[KPR12] → first coarse search, then meticulous search
  - Standard PRF
- Jaccard similarity[WSS+14] → ranking for top-*k*

- Search complexity with $O(\sum_{\#\mathbf{I}_q} \#\mathbf{D}_r)$
- Storage with $O(\#\mathbf{I}+\#\mathbf{W}+\sum_{\#\mathbf{I}} \#\mathbf{D}_\mathbf{I}+\#\mathbf{R})$

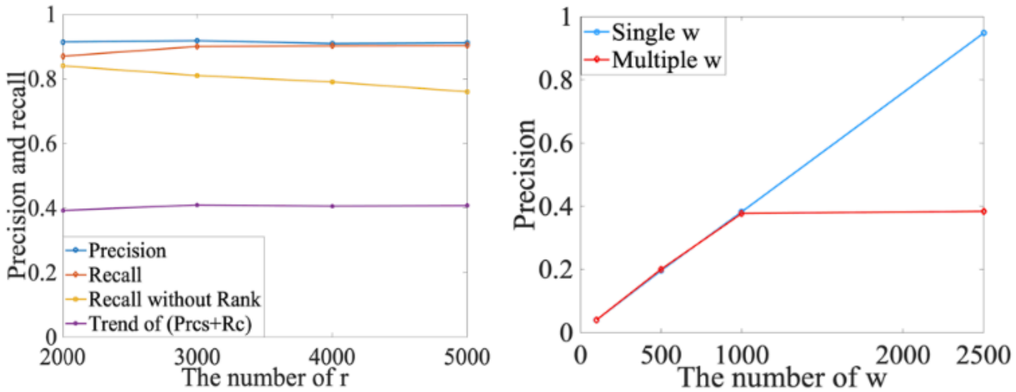
Experimental Results for Basic Instantiation

Figure 10: Time of Basic Instantiation



(a) Index Building Time (b) Search Time (c) Time of Adding Videos (d) Time of Deleting Videos

Figure 12: Searching Results



(a) Precision and Recall (LSH=32) (b) Searching with w

- CPU 64-bit 3.4GHz
- Search time: << 0.1s
- Precision: 90+%

Our Advanced Instantiation

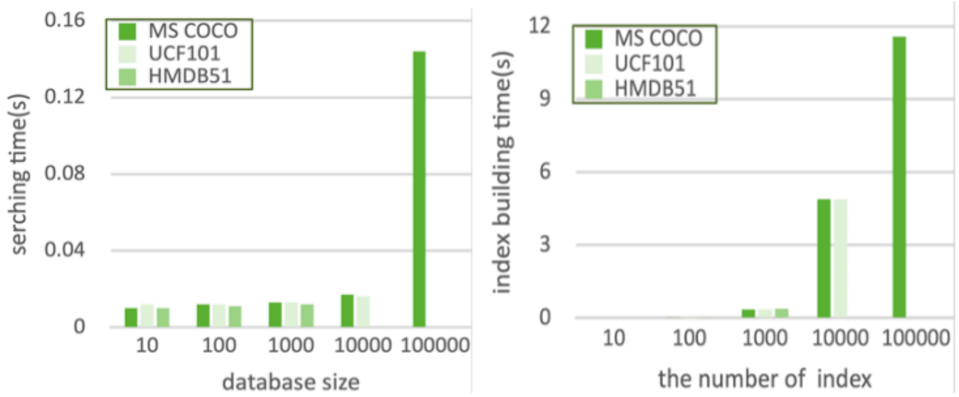
- Better similarity + forward and backward privacy (strong enough)

- CSQ[YWZ+20] → **W** and **R**
- DSSE[BMO17] → first coarse search, then meticulous search
  - Range-constrained PRF

- Computational complexity
- $O(a_w+a_w*a_r)$ for search; $O(\log(a_r)+\log(a_w)*\log(a_r))$ for update
  - $a_@$: #inserted entries matching @

- Communication
  - $O(n_w+d_w*\log(a_w)+d_r*\log(a_r))$ for search; $O(1)$ for update
    - d: #deleted entries; n: |search results|

Experimental Results for Advanced Instantiation



(a) Search Time (b) Index Building Time

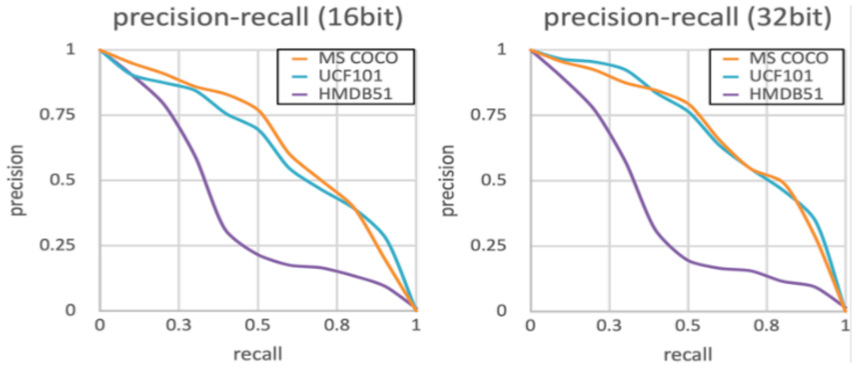
Size of D		10	100	1k	10k	100k
MS COCO	Add	0.017	0.018	0.024	0.023	0.035
	Del	0.010	0.011	0.014	0.015	0.024
UCF101	Add	0.017	0.019	0.024	0.027	n/a
	Del	0.011	0.012	0.016	0.016	n/a
HMDB51	Add	0.014	0.015	0.025	n/a	n/a
	Del	0.010	0.010	0.013	n/a	n/a

Table 9: Time of Addition and Deletion in Advanced Instantiation

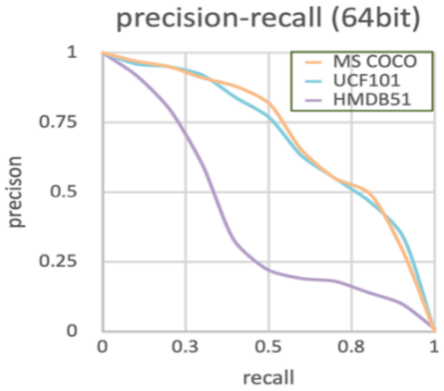
Time(s)	16-bit hash		32-bit hash		64-bit hash	
	Add	Del	Add	Del	Add	Del
MS COCO	0.010	0.010	0.023	0.018	0.032	0.024
UCF101	0.008	0.007	0.017	0.014	0.022	0.015
HMDB51	0.006	0.006	0.013	0.011	0.020	0.013

Table 8: Addition/Deletion Time for Advanced Instantiation

Figure 14: Precision and Recall in Advanced Instantiation



(a) 16-bit Deep Hash (b) 32-bit Deep Hash



(c) 64-bit Deep Hash

Dataset	16-bit hash	32-bit hash	64-bit hash
MS COCO	0.747	0.795	0.833
HMDB51	0.834	0.872	0.872

Table 6: Mean Average Precision for Video Preprocessing

- GPU GeForce RTX1080Ti

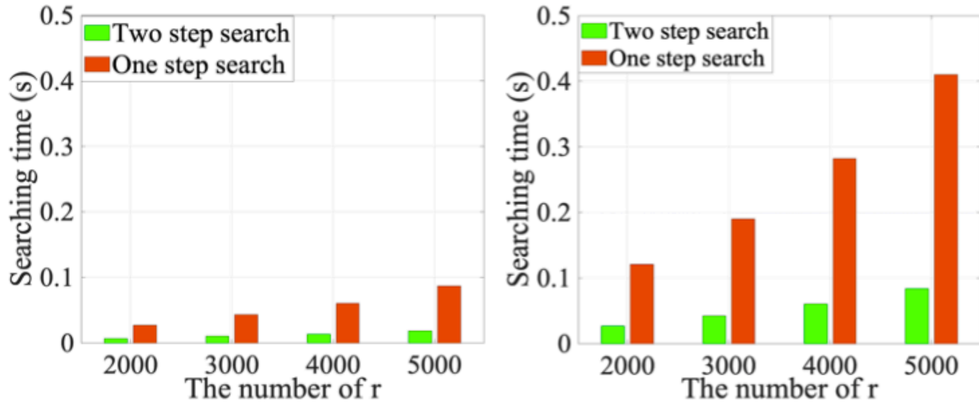
Comparison with [KPR12][LZW+18]

- Keyword SSE
  - [KPR12] for textual messages (CCS)
  - Avoid linear scan of searching

Figure 13: Examples of Search Results

(a) Basic Instantiation (b) Advanced Instantiation (c) Advanced Instantiationw

Figure 11: Comparison of One/Two-Step Search



(a) Time Comparison (LSH=32) (b) Time Comparison (LSH=128)

- Similarity search
  - [LZW+18] for images (Infocom)
  - Additionally support Test⊈Train
  - Much higher accuracy (75% for 128-bit hashing reported by [LZW+18])
  - Motivate further works that integrate deep hashing and SSE

Test⊂Train	Test=Train	Test⊈Train	Counterpart [38]
100.00%	100.00%	96.71%	75%

Table 7: Precision Comparison

Concluding Remarks

- The first framework for searching over EVdio
  - Using SSE twice for searching semantics and visuals

Paper here

- Generic and modular for integrating up-to-date primitives/techniques
  - New syntax; potential ingredients
- Two concrete instantiations for balancing efficiency and security
  - Extensive experimental evaluations
  - Security proof

We expect our work can build the bridge and stimulate further research in processing encrypted data and video retrieval.