香港城市大學
City University of Hong Kong

SWAT
A System-Wide Approach to Tunable Leakage Mitigation in Encrypted Data Stores

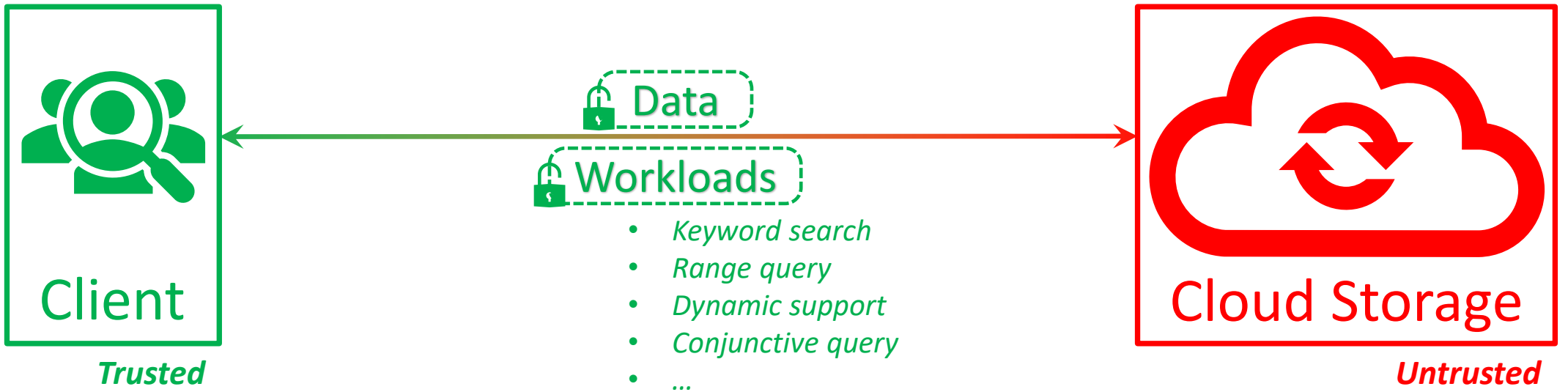
City University of Hong Kong

Leqian Zheng*, Lei Xu*, Cong Wang*, Sheng Wang^, Yuke Hu+, Zhan Qin+, Feifei Li^, Kui Ren+
*: City University of Hong Kong, ^: Alibaba Group, +: Zhejiang University
·: Nanjing University of Science & Technology

Encrypted Data Stores (EDS)

Outsource to cloud hosted data stores for **high availability, elasticity, cost-efficiency, scalability, ease-of-management, etc.**

Alibaba Cloud aws Azure Tencent Cloud



Potential Security Issues

Frequency attack against deterministic encrypted data

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	467xt2J23t
Patient 2	467xt2J23t
Patient 3	2wBrfQ7yn
Patient 4	467xt2J23t
Patient 5	ewgCgM20
Patient 6	467xt2J23t
Patient 7	467xt2J23t
Patient 8	467xt2J23t
Patient 9	zt1NEtbkn9
Patient 10	467xt2J23t
Patient 11	2wBrfQ7yn
Patient 12	467xt2J23t
Patient 13	j0th0LeIsP
Patient 14	467xt2J23t
Patient 15	zt1NEtbkn9
Patient 16	467xt2J23t
Patient 17	2wBrfQ7yn
Patient 18	467xt2J23t
Patient 19	467xt2J23t
Patient 20	2wBrfQ7yn

Potential Security Issues

Frequency attack against deterministic encrypted data

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	467xt2J23t
Patient 2	467xt2J23t
Patient 3	2wBrfQ7yn
Patient 4	467xt2J23t
Patient 5	ewgCgM20
Patient 6	467xt2J23t
Patient 7	467xt2J23t
Patient 8	467xt2J23t
Patient 9	zt1NEtbkn9
Patient 10	467xt2J23t
Patient 11	2wBrfQ7yn
Patient 12	467xt2J23t
Patient 13	j0th0LeIsP
Patient 14	467xt2J23t
Patient 15	zt1NEtbkn9
Patient 16	467xt2J23t
Patient 17	2wBrfQ7yn
Patient 18	467xt2J23t
Patient 19	467xt2J23t
Patient 20	2wBrfQ7yn

Sorted Histogram

# Days	Frequency
467xt2J23t	12
2wBrfQ7yn	4
zt1NEtbkn9	2
j0th0LeIsP	1
ewgCgM20	1

Potential Security Issues

Frequency attack against deterministic encrypted data

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	467xt2J23t
Patient 2	467xt2J23t
Patient 3	2wBrfQ7yn
Patient 4	467xt2J23t
Patient 5	ewgCgM20
Patient 6	467xt2J23t
Patient 7	467xt2J23t
Patient 8	467xt2J23t
Patient 9	zt1NEtbkn9
Patient 10	467xt2J23t
Patient 11	2wBrfQ7yn
Patient 12	467xt2J23t
Patient 13	j0th0LeIsP
Patient 14	467xt2J23t
Patient 15	zt1NEtbkn9
Patient 16	467xt2J23t
Patient 17	2wBrfQ7yn
Patient 18	467xt2J23t
Patient 19	467xt2J23t
Patient 20	2wBrfQ7yn

Sorted Histogram

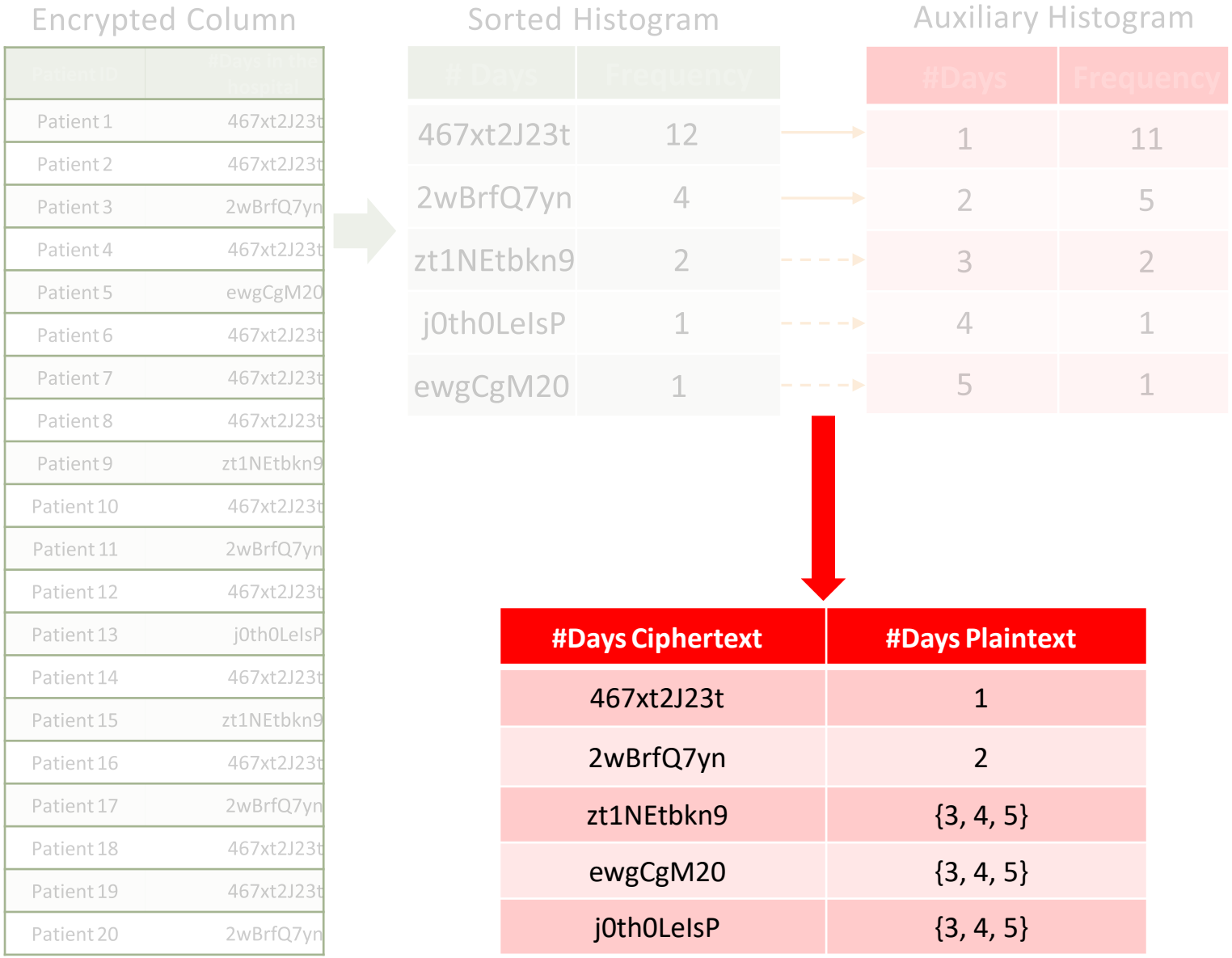
# Days	Frequency
467xt2J23t	12
2wBrfQ7yn	4
zt1NEtbkn9	2
j0th0LeIsP	1
ewgCgM20	1

Auxiliary Histogram

#Days	Frequency
1	11
2	5
3	2
4	1
5	1

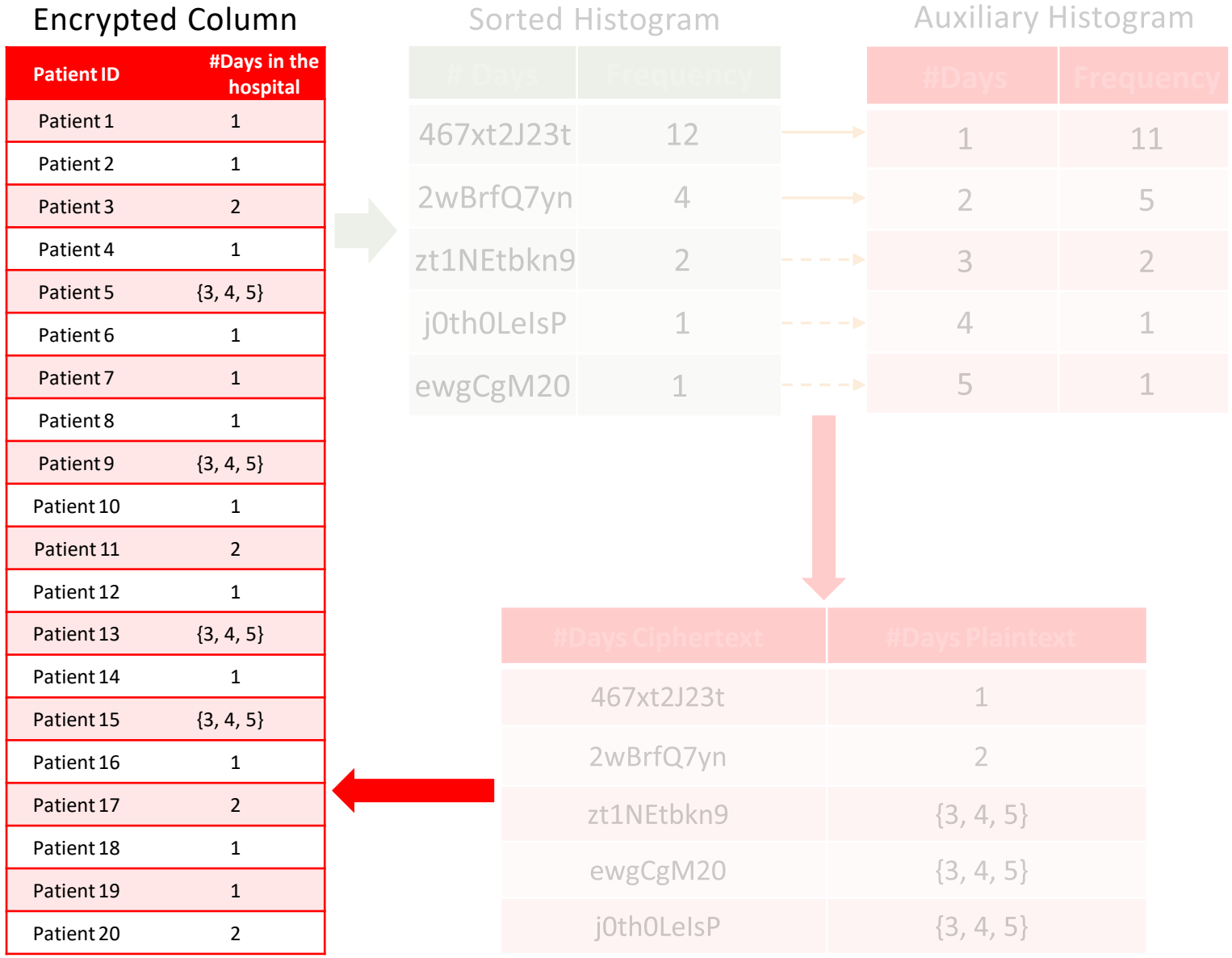
Potential Security Issues

Frequency attack against deterministic encrypted data



Potential Security Issues

Frequency attack against deterministic encrypted data

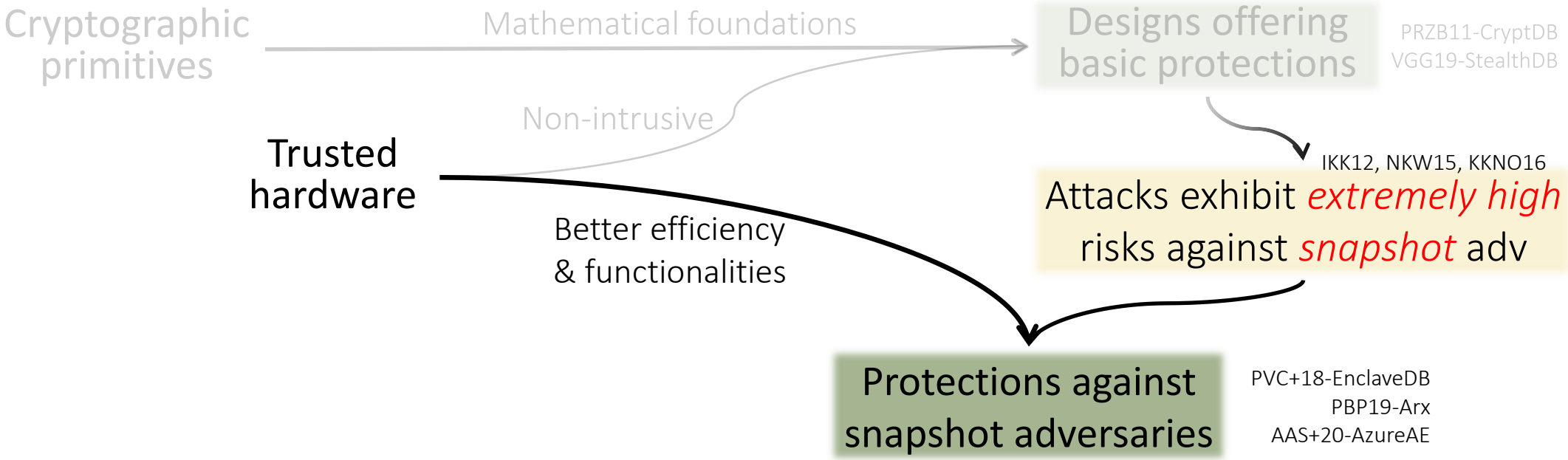


Commonly Used Crypto Primitives

- Property-preserving encryption[1, 2, 3, 4]
  - Deterministic Encryption
  - Order Preserving/Revealing Encryption
  - …

[1] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In CCS, 2015.
[2] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted data. In CCS, 2016.
[3] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In CCS, 2016.
[4] P. Grubbs, et al. Why Your Encrypted Database is not secure. In HotOS 2017.

Evolution of EDS



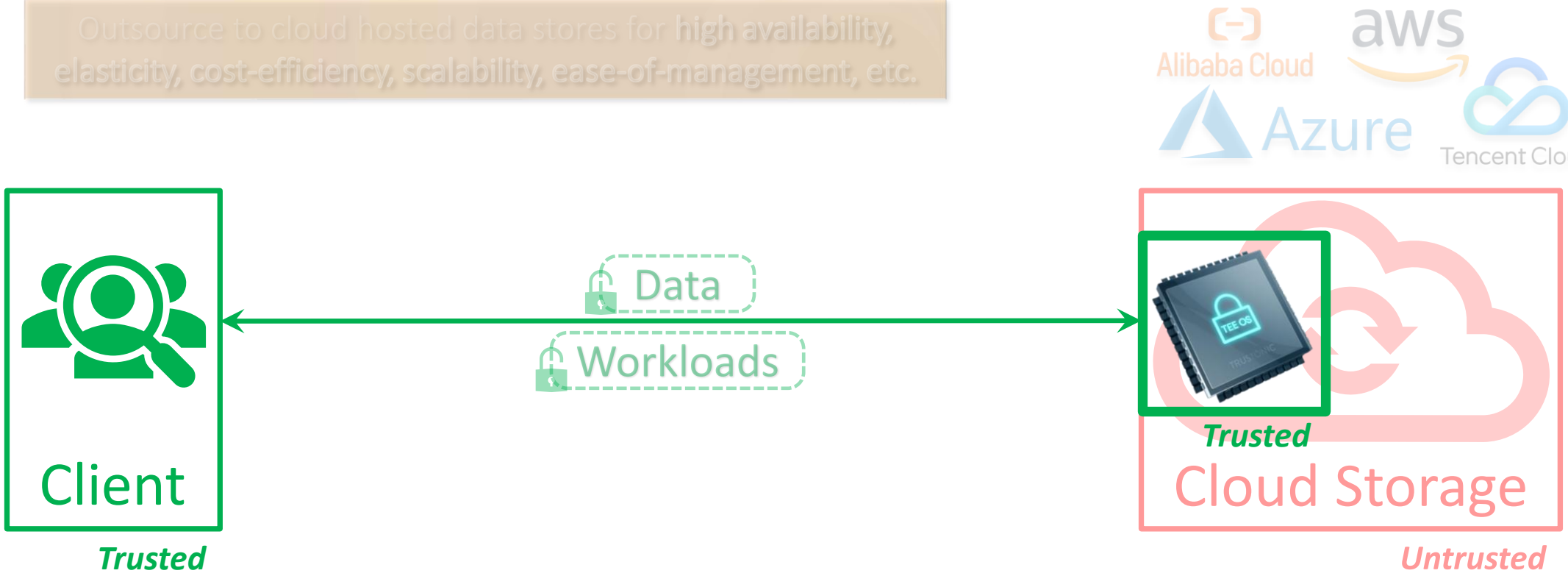
Existing Industrial Products

- Alibaba: Operon w/ Intel SGX, FPGA
- AWS: Searchable encryption via truncated hash
- Azure: Always Encrypted w/ Intel SGX
- Huawei: FE-in-GuassDB w/ ARM TrustZone, Intel SGX
- MongoDB: Queryable Encryption via structured encryption

[1] S. Wang, et al. Operon: an encrypted database for ownership-preserving data management. VLDB 2022.
[2] https://docs.aws.amazon.com/database-encryption-sdk/latest/devguide/searchable-encryption.html
[3] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves?view=sql-server-ver16
[4] J. Zhu, et al. Full Encryption: An end to end encryption mechanism in GaussDB. VLDB 2021.
[5] https://www.mongodb.com/docs/manual/core/queryable-encryption/

Trusted Execution Environment[1,2,3,4]

Outsource to cloud hosted data stores for high availability, elasticity, cost-efficiency, scalability, ease-of-management, etc.



[1] AMD Secure Encrypted Virtualization (SEV)
[2] ARM TrustZone / Confidential Compute Architecture (CCA)
[3] Intel Trust Domain Extensions (TDX) / Software Guard Extensions (SGX)
[4] RISC-V MultiZone / Keystone TEE

Potential Security Issues

Frequency attack exploiting the memory access pattern

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	VPULIzkl5F
Patient 2	7XXAlpbJTV
Patient 3	84Y8ZwNKai
Patient 4	jHhWHQZw1u
Patient 5	gTaDFvILCJ
Patient 6	Uuryvjhq0u
Patient 7	lKyFNh4fFp
Patient 8	XUuFSnprkr
Patient 9	WQu1zTGTjS
Patient 10	mHkP79wnlm
Patient 11	9mf1r6SN4Z
Patient 12	l4vlwsF1ft
Patient 13	RrCbyZjVCp
Patient 14	bbNiBWEwL6
Patient 15	PeiqkdYbmY
Patient 16	0SGlLaOmDp
Patient 17	dxp7gpTbod
Patient 18	0X328dRTTo
Patient 19	s9chRJt6l6
Patient 20	UqCD3c4N0y

Potential Security Issues

Frequency attack exploiting the memory access pattern

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	VPULIzkl5F
Patient 2	7XXAlpbJTV
Patient 3	84Y8ZwNKai
Patient 4	jHhWHQZw1u
Patient 5	gTaDFvILCJ
Patient 6	Uuryvjhq0u
Patient 7	lKyFNh4fFp
Patient 8	XUuFSnprkr
Patient 9	WQu1zTGTjS
Patient 10	mHkP79wnlm
Patient 11	9mf1r6SN4Z
Patient 12	l4vlwsF1ft
Patient 13	RrCbyZjVCp
Patient 14	bbNiBWEwL6
Patient 15	PeiqkdYbmY
Patient 16	0SGlLaOmDp
Patient 17	dxp7gpTbod
Patient 18	0X328dRTTo
Patient 19	s9chRJt6l6
Patient 20	UqCD3c4N0y

```
SELECT id FROM table
WHERE #days = ‘4daiE’;
```

Query 1

Potential Security Issues

Frequency attack exploiting the memory access pattern

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	VPULIzkl5F
Patient 2	7XXAlpbJTV
Patient 3	84Y8ZwNKai
Patient 4	jHhWHQZw1u
Patient 5	gTaDFvILCJ
Patient 6	Uuryvjhq0u
Patient 7	lKyFNh4fFp
Patient 8	XUuFSnprkr
Patient 9	WQu1zTGTjS
Patient 10	mHkP79wnlm
Patient 11	9mf1r6SN4Z
Patient 12	l4vlwsF1ft
Patient 13	RrCbyZjVCp
Patient 14	bbNiBWEwL6
Patient 15	PeiqkdYbmY
Patient 16	0SGlLaOmDp
Patient 17	dxp7gpTbod
Patient 18	0X328dRTTo
Patient 19	s9chRJt6l6
Patient 20	UqCD3c4N0y

```
SELECT id FROM table
WHERE #days = ‘sDfg1’;
```

Query 2

Potential Security Issues

Frequency attack exploiting the memory access pattern

Encrypted Column

Patient ID	#Days in the hospital
Patient 1	VPULIzkl5F
Patient 2	7XXAlpbJTV
Patient 3	84Y8ZwNKai
Patient 4	jHhWHQZw1u
Patient 5	gTaDFvILCJ
Patient 6	Uuryvjhq0u
Patient 7	lKyFNh4fFp
Patient 8	XUuFSnprkr
Patient 9	WQu1zTGTjS
Patient 10	mHkP79wnlm
Patient 11	9mf1r6SN4Z
Patient 12	l4vlwsF1ft
Patient 13	RrCbyZjVCp
Patient 14	bbNiBWEwL6
Patient 15	PeiqkdYbmY
Patient 16	0SGlLaOmDp
Patient 17	dxp7gpTbod
Patient 18	0X328dRTTo
Patient 19	s9chRJt6l6
Patient 20	UqCD3c4N0y

Sorted Histogram

Query	Result IDs	Frequency
1	1, 2, 4, 6, 7, 8, 10, 12, 14, 18, 19	12
2	3, 11, 17, 20	4
3	9, 15	2
4	5	1
5	13	1

Query 3, 4, 5

Potential Security Issues

Leakage-Abuse Attacks

- M. Islam, et al., *Access pattern* disclosure on searchable encryption: Ramification, attack and mitigation. NDSS 2012.
- D. Cash, et al., *Leakage-abuse* attacks against searchable encryption. CCS 2015.
- M. Lacharité, et al., Improved Reconstruction Attacks on Encrypted Data Using Range *Query Leakage*. S&P 2018.
- P. Grubbs, et al. Pump up the volume: Practical database reconstruction from *volume leakage* on range queries. CCS 2018.
- P. Grubbs, et al. Learning to reconstruct: Statistical learning theory and *encrypted database attacks*. S&P 2019.
- Z. Gui, et al. Encrypted databases: New *volume attacks* against range queries. CCS 2019.
- L. Blackstone, et al. Revisiting *leakage abuse* attacks. NDSS 2020.
- E. Kornaropoulos, et al. The state of the uniform: *Attacks* on encrypted databases beyond the uniform query distribution. S&P 2020.
- E. Kornaropoulos, et al. Response-hiding encrypted ranges: Revisiting security via *parametrized leakage-abuse attacks*. S&P 2021.
- S. Oya, et al. Hiding the access pattern is not enough: Exploiting *search pattern leakage* in searchable encryption. USENIX Security 2021.
- M. Damie, et al. A highly accurate *query recovery attack* against searchable encryption using non-indexed documents. USENIX Security 2021.
- J. Ning, et al. LEAP: *leakage-abuse attack* on efficiently deployable, efficiently searchable encryption with partially known dataset. CCS 2021.
- C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala. DP-Sync: Hiding *update patterns* in secure outsourced databases with differential privacy. SIGMOD, 2021.
- S. Oya and F. Kerschbaum. IHOP: improved statistical query recovery against searchable symmetric encryption through quadratic optimization. USENIX Security 2022
- S. Lambregts, et al. VAL: *Volume and Access Pattern* Leakage-Abuse Attack with Leaked Documents. ESORICS 2022.
- Z. Gui, et al. Rethinking Searchable Symmetric Encryption. S&P 2023.

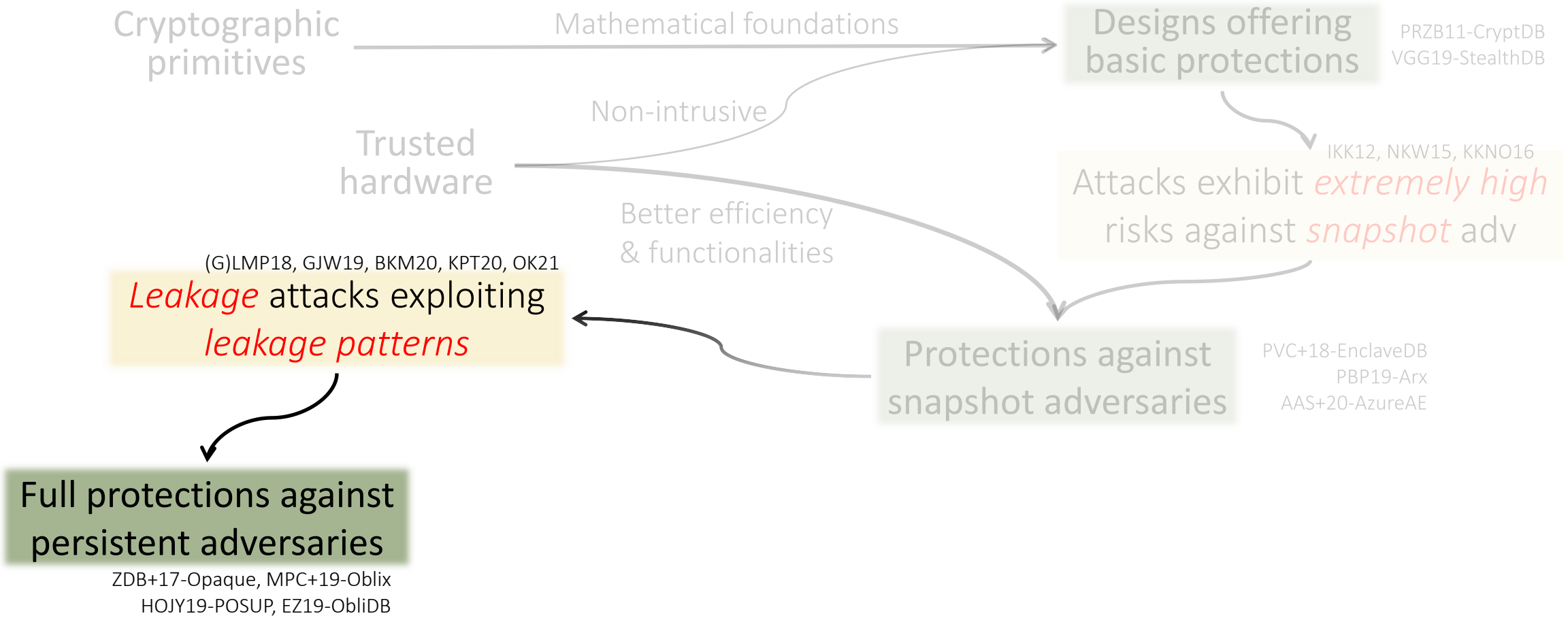
Potential Security Issues

- Property-preserving encryption[1, 2, 3, 4]
- Workload leakage
  - Access pattern
  - Volume pattern
  - Order pattern
  - Query correlation pattern
  - Operation timestamp pattern

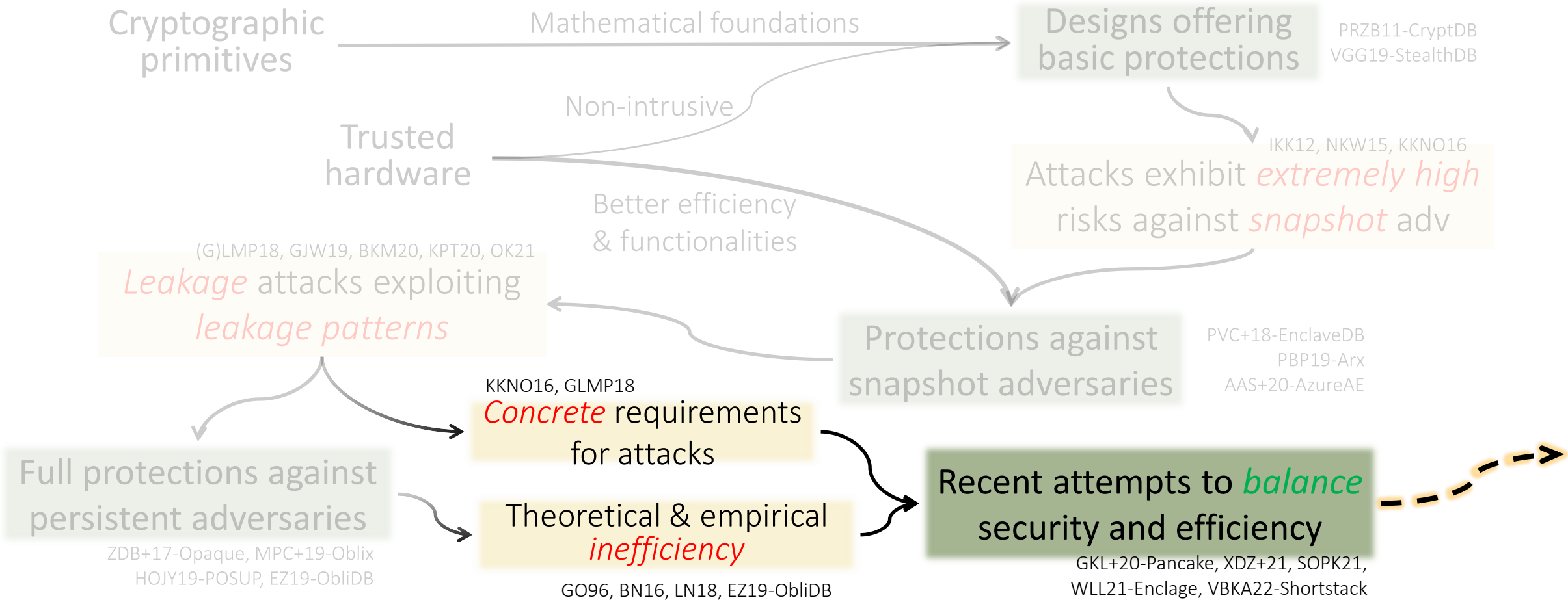
Persistent adversaries

Passively observe how workloads are executed

The Evolution of EDS



The Evolution of EDS

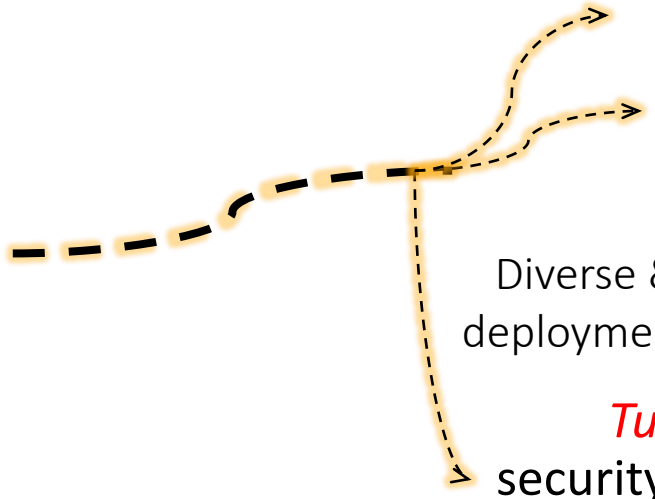


Motivation

- Full leakage suppression introduces *substantial* overheads

Recent attempts to *balance* security and efficiency

GKL+20-Pancake, XDZ+21, SOPK21, WLL21-Enclage, VBKA22-Shortstack



Diverse & complex deployment scenarios

Tunable security-efficiency trade-offs

Motivation

- Full leakage suppression introduces *substantial* overheads
- Some works protect only *subsets of* leakage patterns
  - ObliDB[1]: access pattern ✓, volume pattern ✗
  - Volume-hiding solutions[2,3,4]: volume pattern ✓, query equality pattern ✗
  - Pancake[5]: access pattern ≠✓, query correlation pattern ✗, volume pattern ✗

[1] S. Eskandarian, et al. ObliDB: Oblivious query processing for secure databases. In VLDB, 2019.
[2] S. Kamara and T. Moataz. Computationally volume-hding structured encryption. In EuroCrypt, 2019.
[3] S. Patel, et al. Mitigating leakage in secure cloud-hosted data structures: Volume-hding for multi-maps via hashing. In CCS, 2019.
[4] G. Amjad, et al. Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption. In PETs, 2023
[5] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

Motivation

- Full leakage suppression introduces *substantial* overheads
- Some works protect only *subsets of* leakage patterns
  - ObliDB[1]: access pattern ✓, volume pattern ✗
  - Volume-hiding solutions[2,3,4]: volume pattern ✓, query equality pattern ✗
  - Pancake[5]: access pattern ✓, query correlation pattern ✗, volume pattern ✗

Consider system-wide leakage

[1] S. Eskandarian, et al. ObliDB: Oblivious query processing for secure databases. In VLDB, 2019.
[2] S. Kamara and T. Moataz. Computationally volume-hding structured encryption. In EuroCrypt, 2019.
[3] S. Patel, et al. Mitigating leakage in secure cloud-hosted data structures: Volume-hding for multi-maps via hashing. In CCS, 2019.
[4] G. Amjad, et al. Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption. In PETs, 2023
[5] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

Motivation

- Full leakage suppression introduces *substantial* overheads
- Some works protect only *subsets of* leakage patterns
- *Uniformed* security notion applied over *various* workloads
  - Adore[1]: differentially obliviousness → table joins?

[1] L. Qin, et al. **Adore: Differentially oblivious relational database operators. In VLDB, 2022**.
[2] D. Bogatov, et al. Epsolute: Efficiently querying databases while providing differential privacy. In CCS, 2021.

Motivation

- Full leakage suppression introduces *substantial* overheads
- Some works protect only *subsets of* leakage patterns
- *Uniformed* security notion applied over *various* workloads
  - Adore[1]: differentially obliviousness → table joins?

Identify nuanced privacy requirements in different workloads

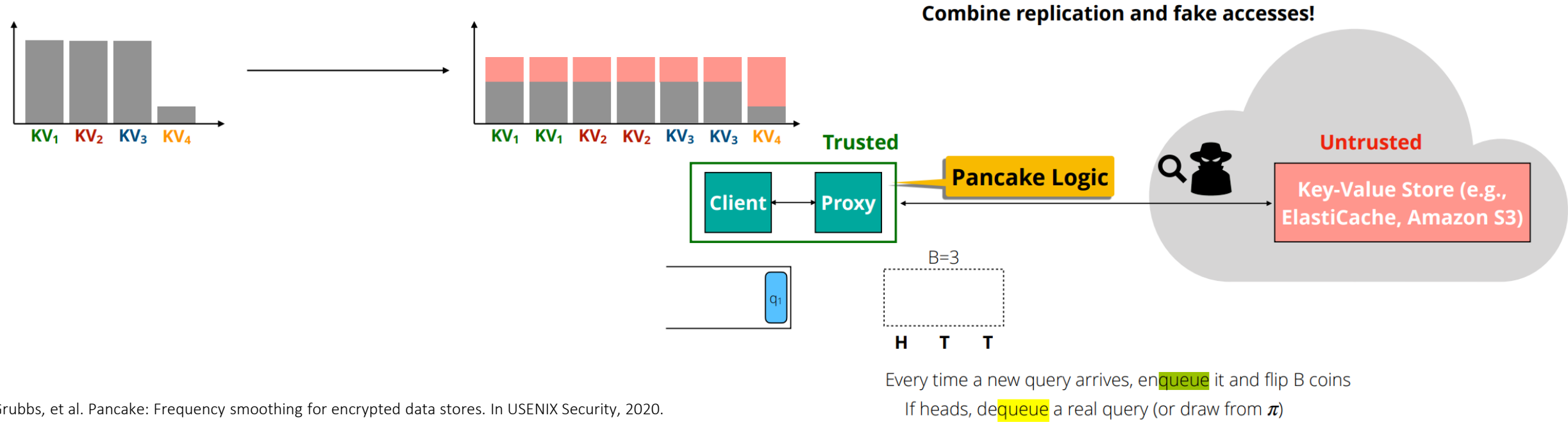
[1] L. Qin, et al. **Adore: Differentially oblivious relational database operators. In VLDB, 2022**.
[2] D. Bogatov, et al. Epsolute: Efficiently querying databases while providing differential privacy. In CCS, 2021.

SWAT

- Progressive solutions
  - Key-value workload
  - Range-query workload
  - Dynamic workload

SWAT – Key-Value Workload

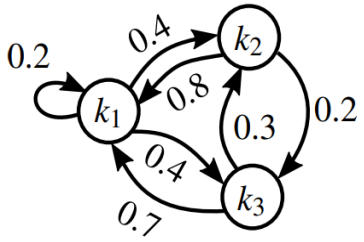
- Leakage – memory access pattern
  - Access frequency → Pancake[1]

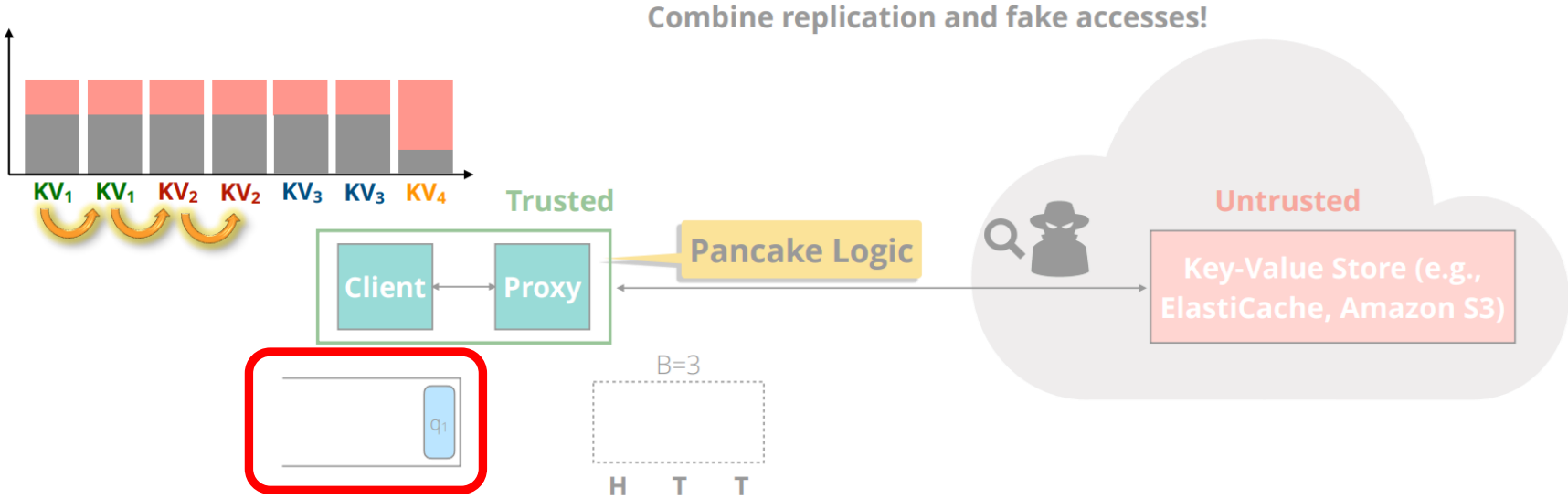


[1] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

SWAT – Key-Value Workload

- Leakage – memory access pattern
  - Access frequency → Pancake[1]
  - Query correlation pattern

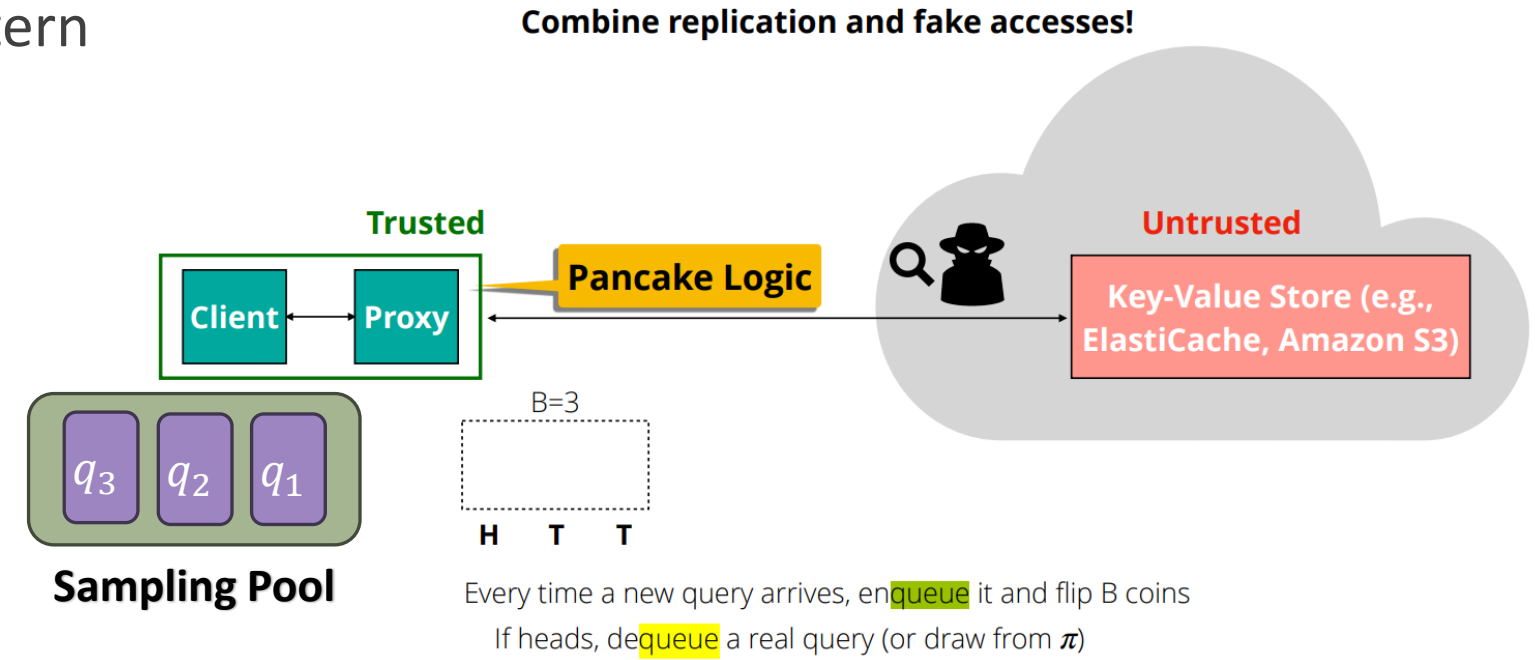




[1] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

SWAT – Key-Value Workload

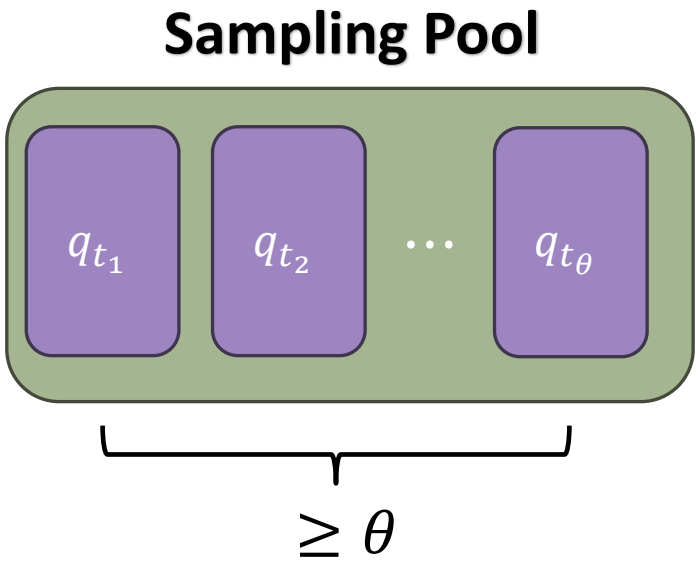
- Leakage – memory access pattern
  - Access frequency → Pancake[1]
  - Query correlation pattern



[1] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

SWAT – Key-Value Workload

- Leakage – memory access pattern
  - Access frequency → Pancake[1]
  - Query correlation pattern – θ-query decorrelation



Independency $\Longrightarrow$ Zero correlation

[1] P. Grubbs, et al. Pancake: Frequency smoothing for encrypted data stores. In USENIX Security, 2020.

SWAT – Range-Query Workload

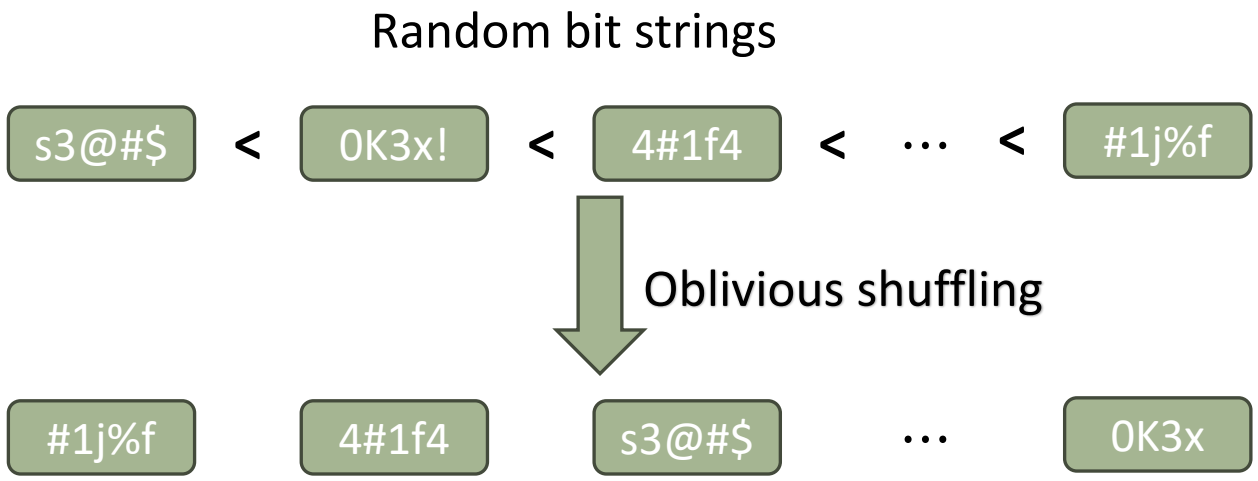
- Leakage
  - Data storage: order leakage

Random bit strings

s3@#$ < 0K3x! < 4#1f4 < … < #1j%f

SWAT – Range-Query Workload

- Leakage
  - Data storage: order leakage

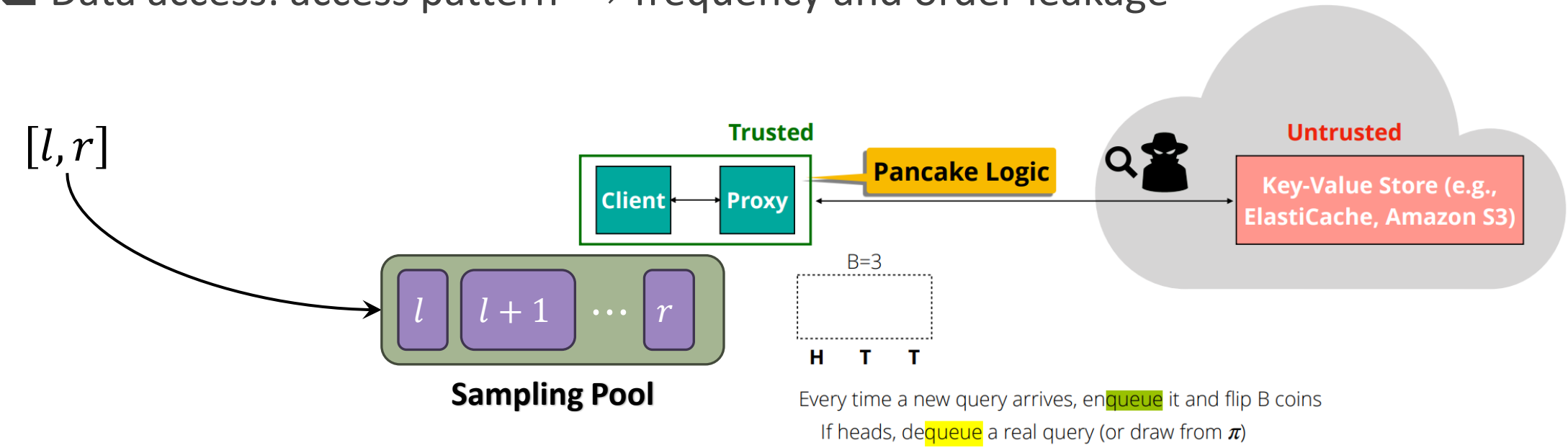


SWAT – Range-Query Workload

- Leakage
  - Data storage: order leakage
  - Data access: access pattern ⟹ frequency and order leakage

SWAT – Range-Query Workload

- Leakage
  - Data storage: order leakage
  - Data access: access pattern ⟹ frequency and order leakage



SWAT – Range-Query Workload

- Leakage
  - Data storage: order leakage
  - Data access: access pattern ⟹ frequency and order leakage
  - Data transition: volume and timestamp leakage

SWAT – Range-Query Workload

- Leakage
  - Data storage: order leakage
  - Data access: access pattern ⟹ frequency and order leakage
  - Data transition: volume and timestamp leakage

SWAT – Range-Query Workload

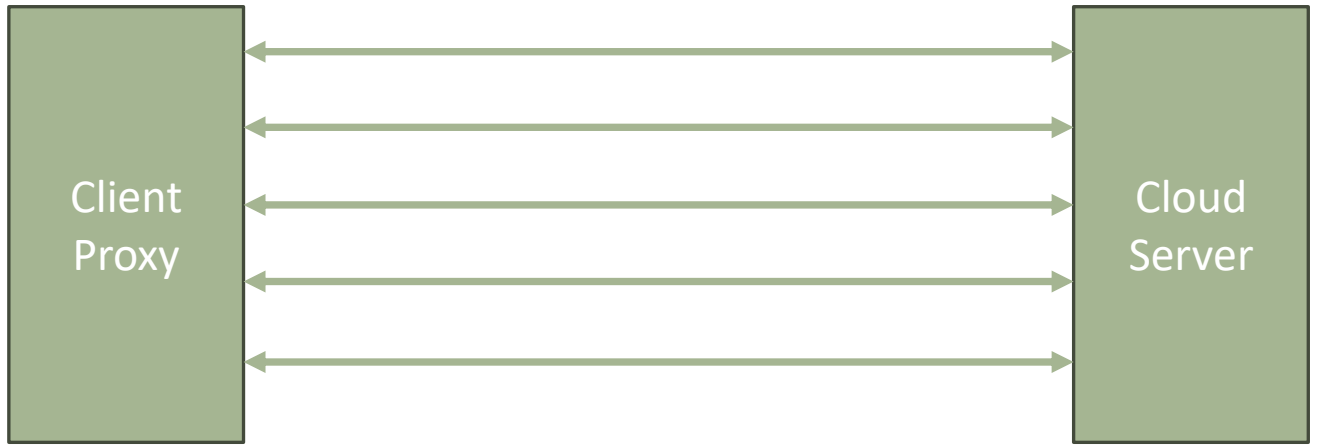
- ❑ Leakage
- ❑ Efficiency
  - ❑ Terrible amount of data transferred

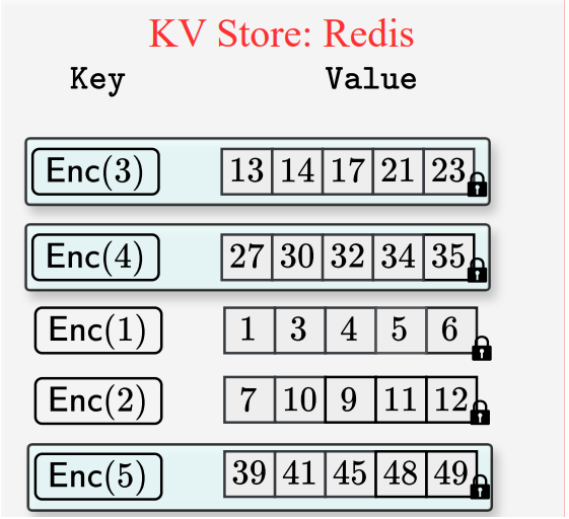
SWAT – Range-Query Workload

- ❑ Leakage
- ❑ Efficiency
  - ❑ ~~Terrible amount of data transferred~~
  - ❑ Acceptable bandwidth w/ pay-by-bandwidth billing model

SWAT – Range-Query Workload

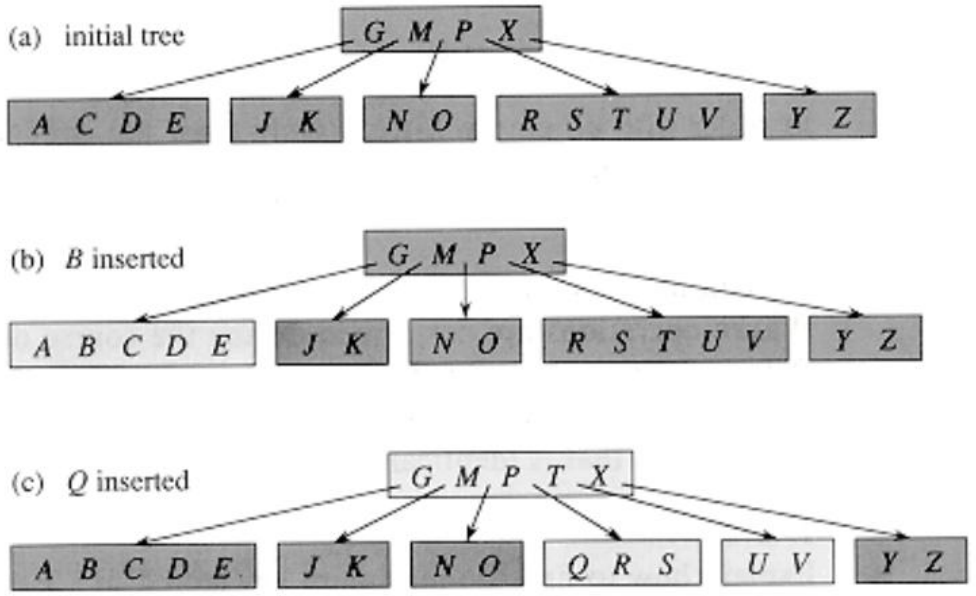
- Leakage
- Efficiency
  - Acceptable bandwidth
  - Less roundtrip





SWAT – Dynamic Workload

- Leakage
  - Access pattern



[1] T. Cormen, et al. Introduction to Algorithms.

SWAT – Dynamic Workload

- Leakage
- Approach
  - k-binomial transformation[1]
  - k-way differentially oblivious[2] merge

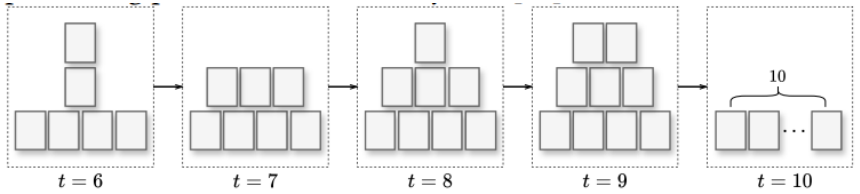


Figure 2: An example of 3-binomial transform. At each step t, layers from top to bottom hold $\binom{D_1}{1}$, $\binom{D_2}{2}$, and $\binom{D_3}{3}$ items, respectively, resulting in a total of t items. We have $D_1 = 2$, $D_2 = 3$, and $D_3 = 4$ for $t = 9$. Inserting one more element will trigger the destruction of all three layers and rebuild them into a new one in the third layer, with $D_1 = 0$, $D_2 = 1$, $D_3 = 5$.

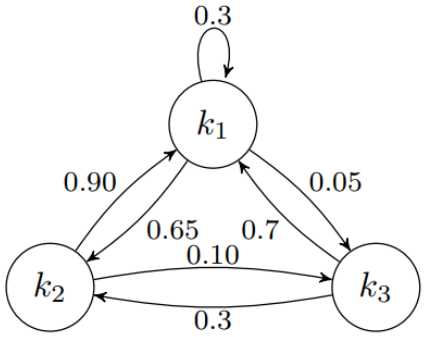
Definition 2. *We say that a dynamic outsourced data store* Π *is* (ε, δ)*-differentially oblivious with respect to updates (a.k.a* DO_{update}*-ODDS) if for any data store* $\mathcal{D}$ *and any two query-consistent neighboring operational sequences* $\mathsf{ops} \sim \mathsf{ops}'$*, and any possible set of memory access patterns* S *(adapted from [18]):*

$$\Pr[\mathcal{AP}_{\Pi}(\mathcal{D}, \mathsf{ops}) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{AP}_{\Pi}(\mathcal{D}, \mathsf{ops}') \in S] + \delta.$$

[1] J. Bentley, et al. Decomposable Searching Problems I: Static-to-Dynamic Transformation. J. Algorithms 1 (1980).
[2] T.-H. Chan, et al. Foundations of Differentially Oblivious Algorithms. SODA' 19.

Evaluations

- θ-query decorrelation



Evaluations

- θ-query decorrelation
- Storage & bandwidth costs

	$\mathcal{C}$ MB	$\mathcal{S}$ GB
StealthDB	0	3.81
$\mathcal{E}$psolute	28.9	12.0
SWAT	0.18	7.65

TABLE I: Storage usage in default setting.

σ \ n	10^5	10^6	10^7
0.1%	6	12	118
0.2%	6	24	236
0.5%	8	58	586

TABLE II: Bandwidth costs (MB) with various σ and n.

Evaluations

- ❑ θ-query decorrelation
- ❑ Storage & bandwidth costs
- ❑ Performance against baseline approaches

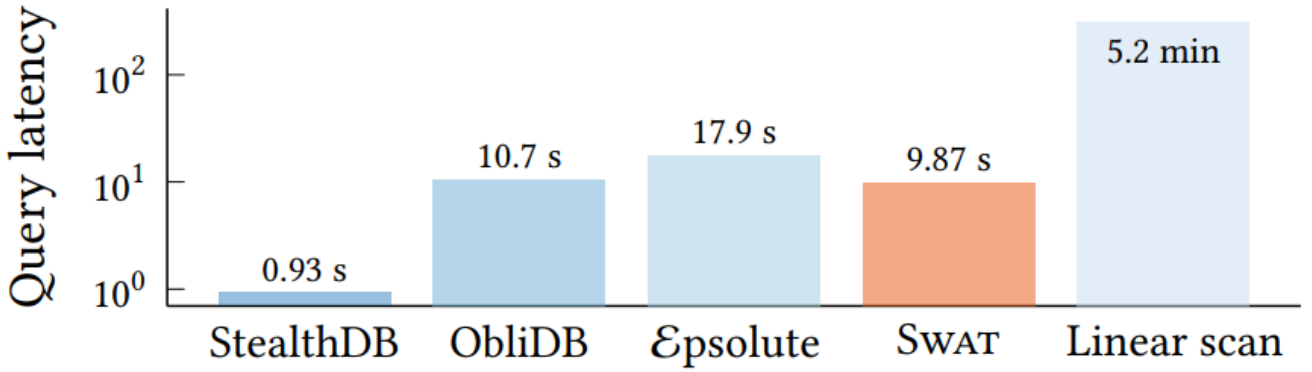
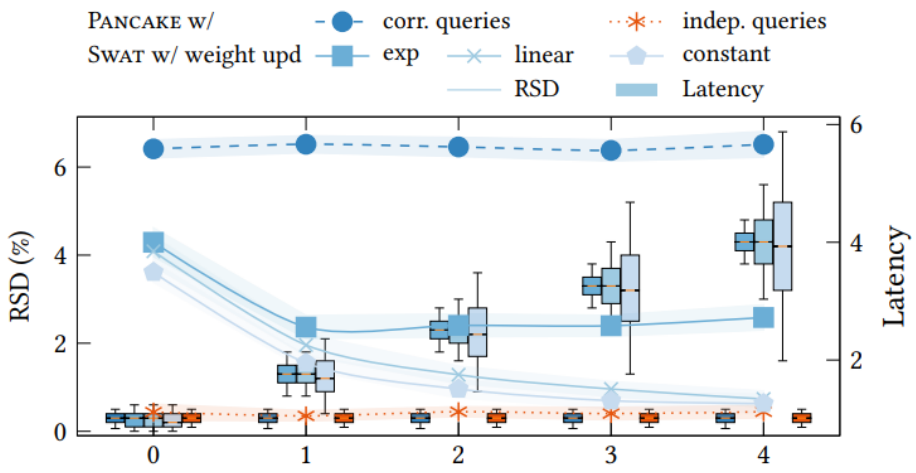
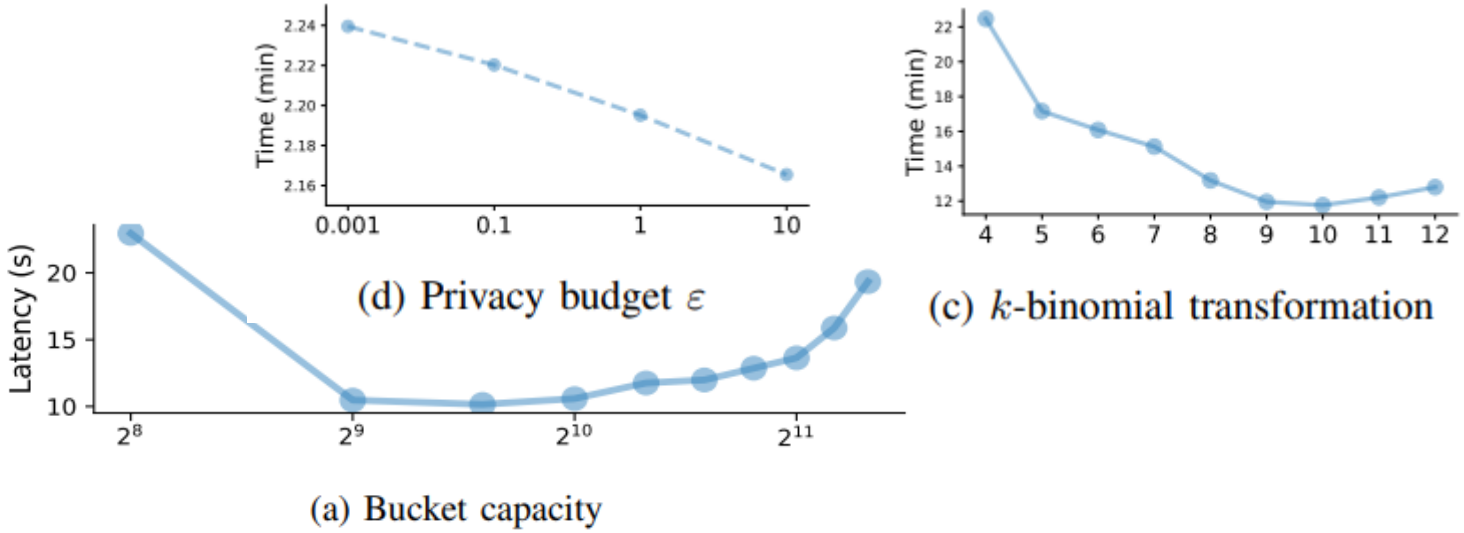


Figure 6: *Range-query* systems under the default setting. The security strength increases from left to right.

Evaluations

- ❑ θ-query decorrelation
- ❑ Storage & bandwidth costs
- ❑ Performance against baseline approaches
- ❑ Parameters for “tunable” efficiency/privacy





Conclusions

- New *security models* to capture varying security requirements in key-value, range-query, and dynamic workloads
- An efficient construction, *SWAT*, progressively enables these workloads while *provably* mitigating system-wide leakage with tunable privacy-efficiency trade-offs
- Implement and *evaluate* SWAT over an extensive set of settings with detailed result compilation

香港城市大學
City University of Hong Kong

Conclusions

- New *security models* to capture varying security requirements in key-value, range-query, and dynamic workloads
- An efficient construction, SWAT, progressively enables these workloads while *provably* mitigating system-wide leakage with tunable privacy-efficiency trade-offs
- Implement and *evaluate* SWAT over an extensive set of settings with detailed result compilation

Thank you!

Questions?

SWAT – System Prototype