How to Search on Encrypted Data

SENY KAMARA

MICROSOFT RESEARCH

Encryption

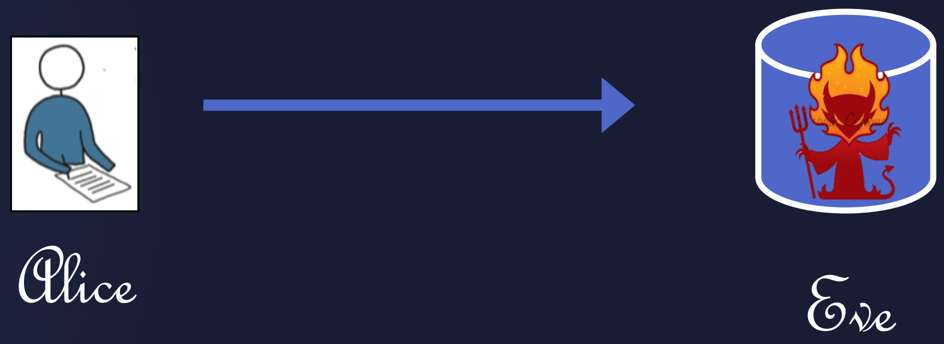
- $\text{Gen}(1^k) \Longmapsto K$
- $\text{Enc}(K, m) \Longmapsto C$
- $\text{Dec}(K, c) \Longmapsto m$

<u>Secure Communiation</u>

Encryption

- Gen(1^k) $\Longrightarrow$ K
- Enc(K, m) $\Longrightarrow$ C
- Dec(K, c) $\Longrightarrow$ m

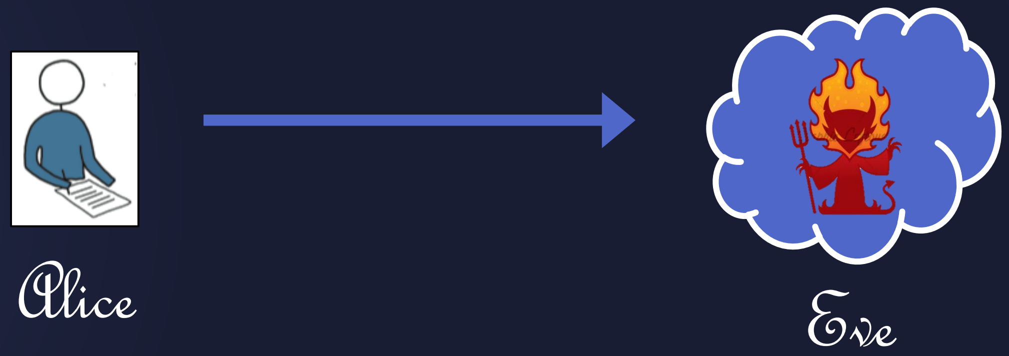
<u>Secure Storage</u>



Encryption

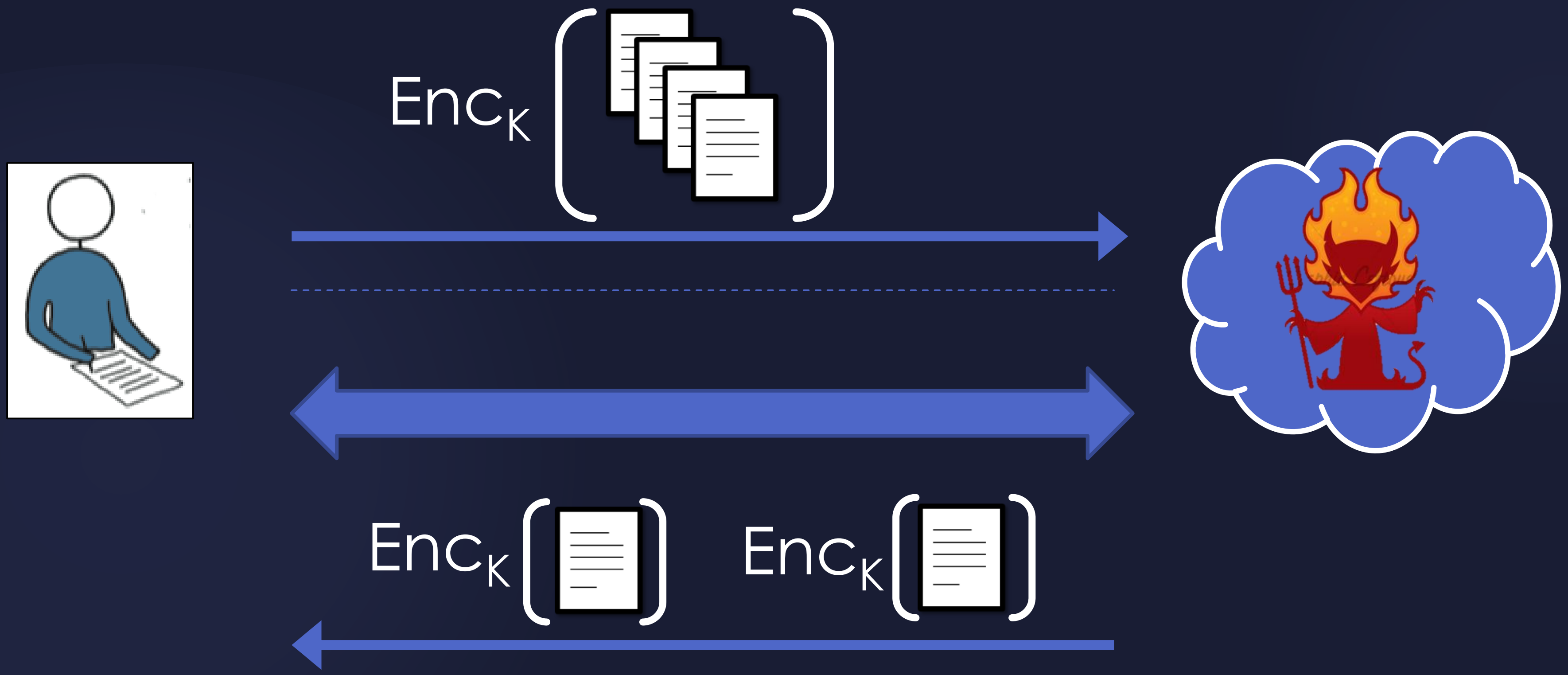
- $Gen(1^k) \Longmapsto K$
- $Enc(K, m) \Longmapsto C$
- $Dec(K, C) \Longmapsto m$

Secure Cloud Storage

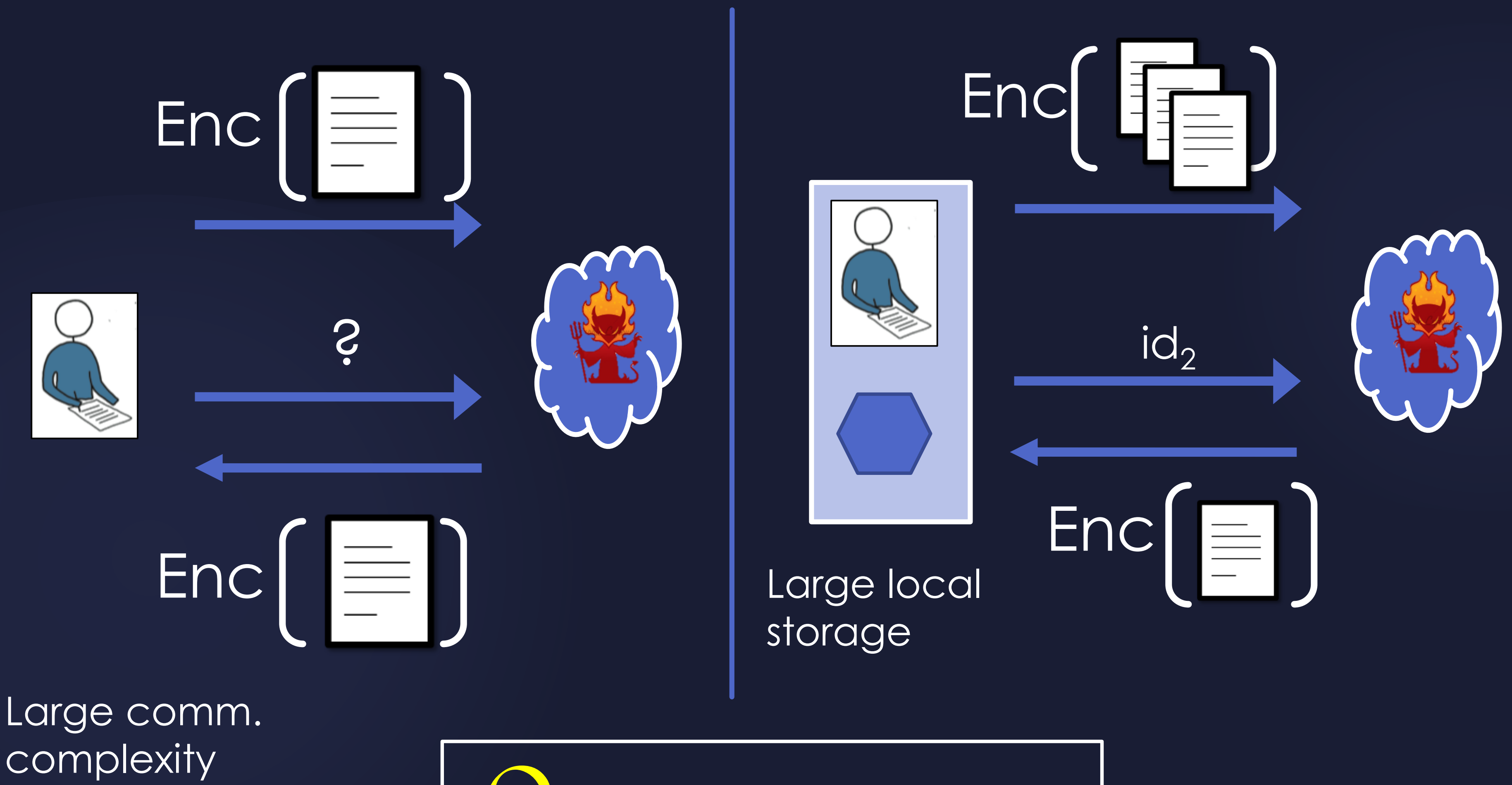


Encrypted Search

Encrypted Search



Two Simple Solutions

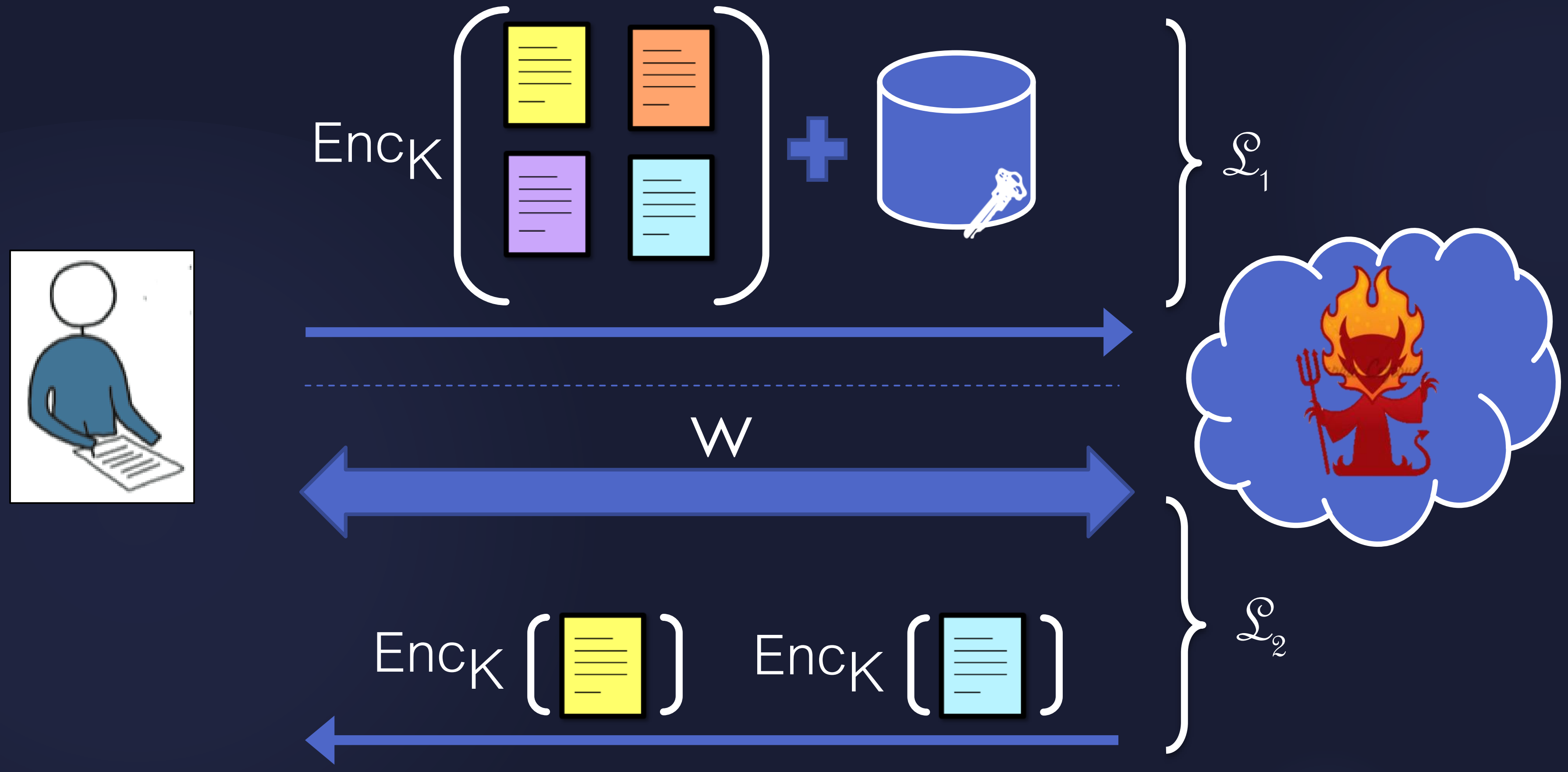


Q: can we do better?

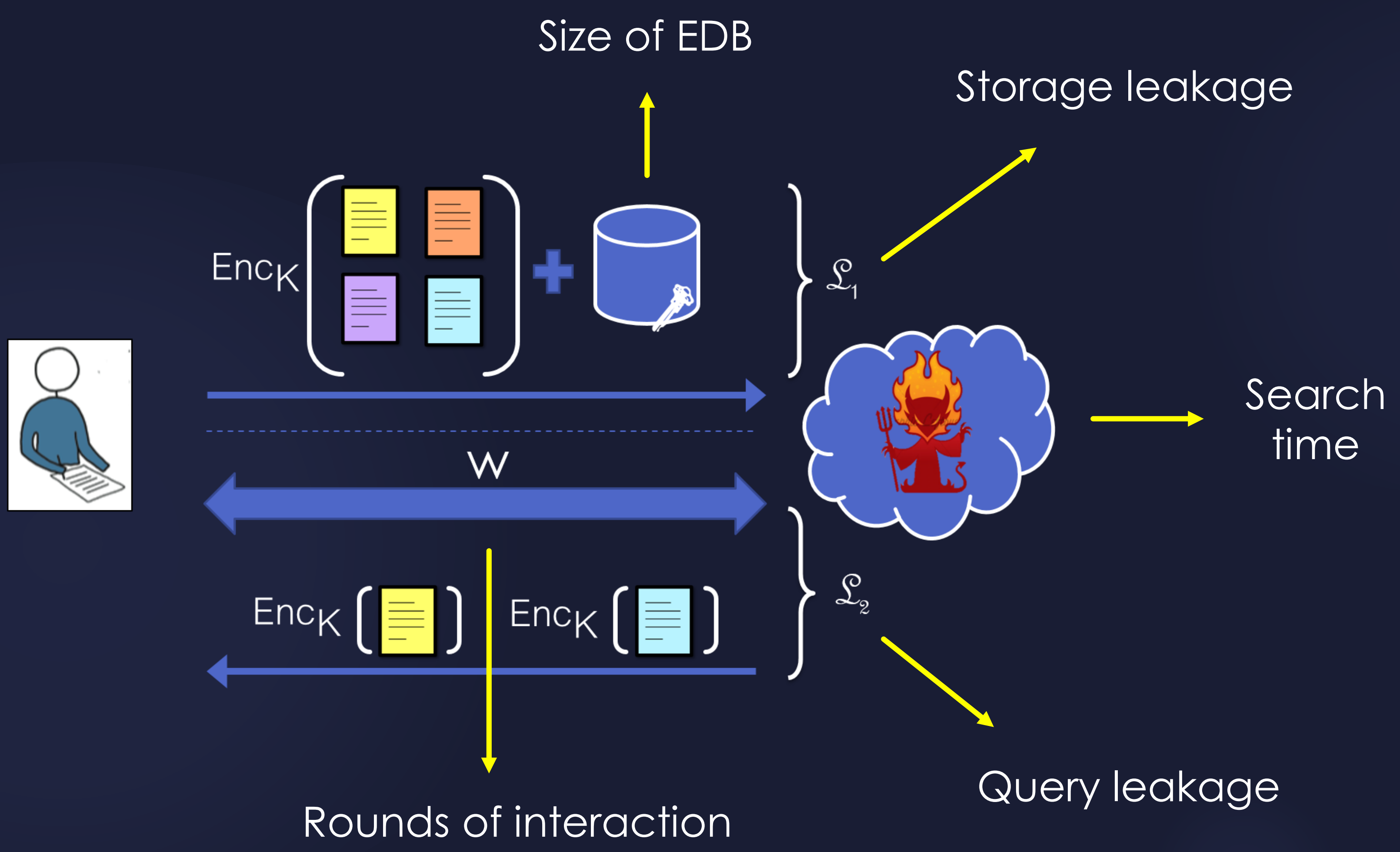
More Advanced Solutions

- Multi-Party Computation
  [Yao82, Goldreich-Micali-Wigderson87]
- Oblivious RAM
  [Goldreich-Ostrovsky92]
- Searchable symmetric encryption
  [Song-Wagner-Perrig01]
- Functional encryption
  [Boneh-di Crescenzo-Ostrovsky-Persiano06]
- Property-preserving encryption
  [Bellare-Boldyreva-O'Neill06]
- Fully-homomorphic encryption
  [Gentry09]

Encrypted Search



Encrypted Search



Property-Preserving Encryption

- Encryption that supports public tests
- Examples:
  - Deterministic encryption [Bellare-Boldyreva-O'Neill06]
  - Order-preserving encryption [Agrawal-Kiernan-Srikant-Xu04, Boldyreva-Chenette-Lee-O'Neill09]
  - Orthogonality-preserving encryption [Pandey-Rouselakis12]

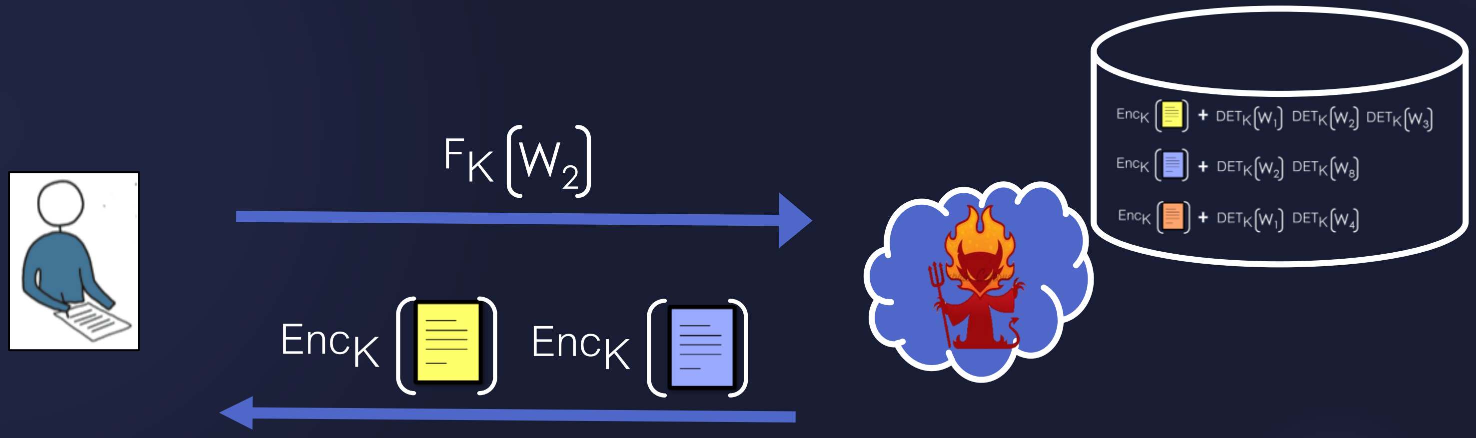
Deterministic Encryption

[Bellare-Boldyreva-O'Neill06]

- $\mathrm{Gen}(1^k) \Longrightarrow K = \langle K_1, K_2 \rangle$
- $\mathrm{DET}(K, w) \Longrightarrow \langle F_{K2}(w), F_{K1}(F_{K2}(w)) \oplus w \rangle$
- $\mathrm{Test}(c_1, c_2) \Longrightarrow c_1 = c_2$
- $\mathrm{Dec}(sk, c) \Longrightarrow F_{K_1}(c_1) \oplus c_2$

EDB:

- Enc_K [yellow document] + $\mathrm{DET}_K(w_1)$ $\mathrm{DET}_K(w_2)$ $\mathrm{DET}_K(w_3)$
- Enc_K [blue document] + $\mathrm{DET}_K(w_2)$ $\mathrm{DET}_K(w_8)$
- Enc_K [orange document] + $\mathrm{DET}_K(w_1)$ $\mathrm{DET}_K(w_4)$



DET-Based Solution

Security

- $\mathcal{L}_1$ leakage
  - #DB
  - equality
  - PK: DB*
- $\mathcal{L}_2$ leakage
  - access pattern
  - search pattern

Efficiency

- Search
  - Sub-linear in #DB
  - process EDB like DB
- Legacy

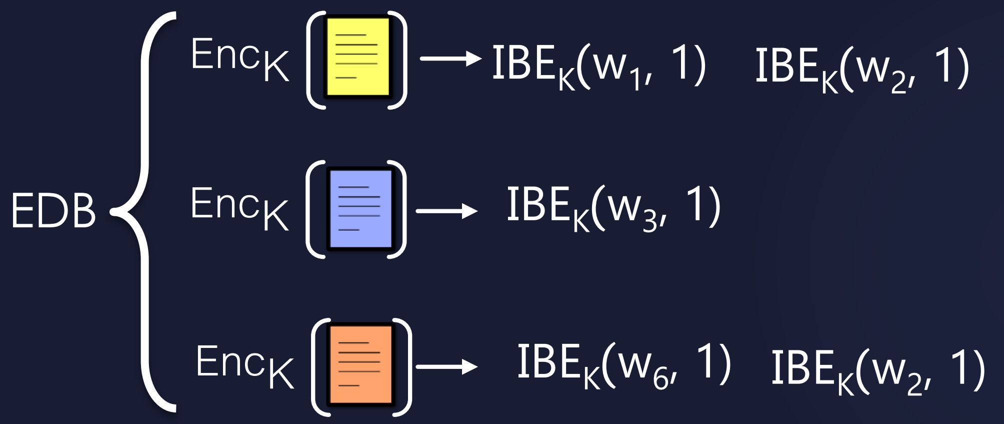
* Unless DB has high entropy

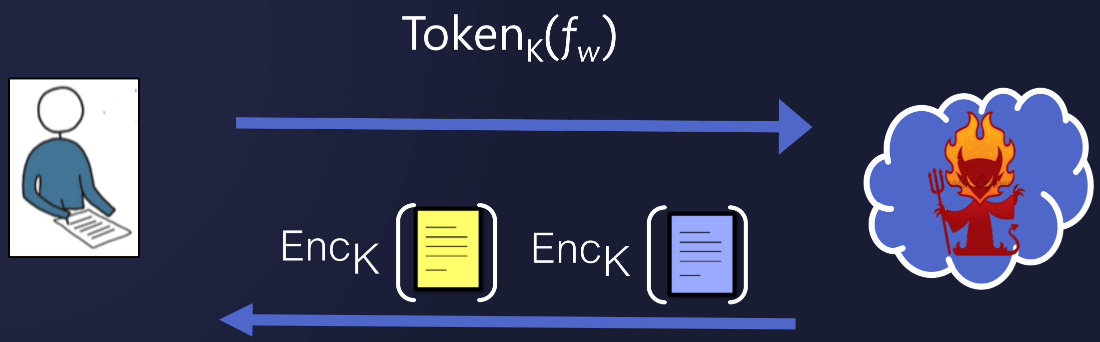
Functional Encryption

- Encryption that supports private tests
- Examples:
  - Identity-based encryption
    [Boneh-Franklin01, Boneh-diCrescenzo-Ostrovsky-Persiano06]
  - Attribute-based encryption
    [Sahai-Waters05]
  - Predicate encryption
    [Shen-Shi-Waters]

Identity-Based Encryption

- $Gen(1^k) \Longrightarrow K$
- $IBE(K, id, m) \Longrightarrow c$
- $Token(K, id') \Longrightarrow t$
- $Dec(t, c) \Longrightarrow m$ if $id = id'$





IBE-Based Solution

Security

- $\mathscr{L}_1$ leakage
  - #DB
  - ~~Equality~~
  - ~~PK: DB*~~
- $\mathscr{L}_2$ leakage
  - access pattern
  - PK: keyword*

Efficiency

- Slow search
  - Linear in #DB

* [Boneh-Raghunathan-Segev13]

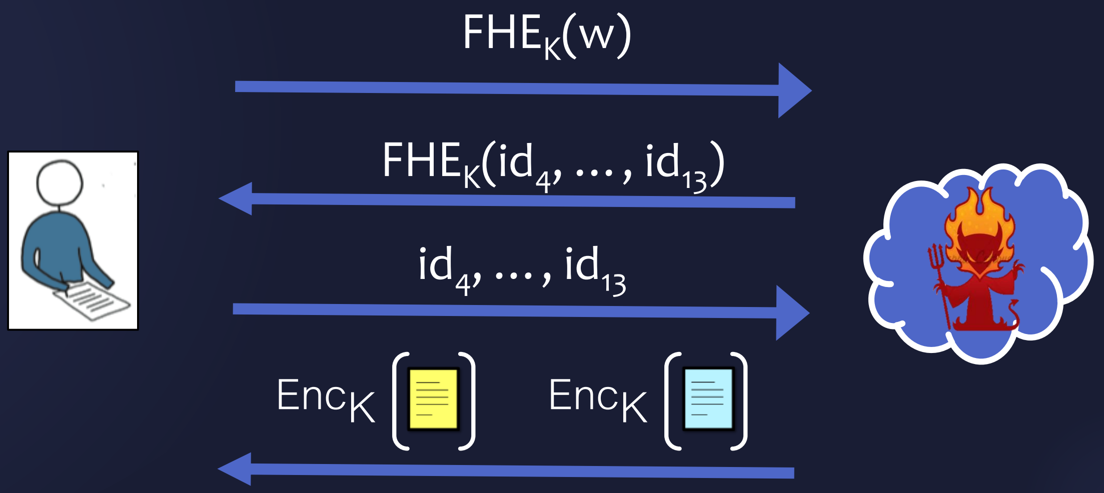
Homomorphic Encryption

- Encryption that supports computation
- Examples:
  - Fully-homomorphic encryption
    [Gentry09,…]
  - Somewhat homomorphic encryption
    [Boneh-Goh-Nissim05, …]

Homomorphic Encryption

- $\mathrm{Gen}(1^k) \Longrightarrow K$
- $\mathrm{Enc}(K, m) \Longrightarrow c$
- $\mathrm{Eval}(f, c_1, \ldots, c_n) \Longrightarrow c'$
- $\mathrm{Dec}(sk, c') \Longrightarrow f(\mathrm{Dec}(c_1), \ldots, \mathrm{Dec}(c_n))$

$\mathrm{EDB} = \mathrm{FHE}_K$



FHE-Based Solution (1)

Security

- $\mathcal{L}_1$ leakage
  - #DB
  - ~~Equality~~
  - ~~PK: DB*~~
- $\mathcal{L}_2$ leakage
  - access pattern
  - ~~PK: keyword~~

Efficiency

- Very slow search
  - Interactive (1 round)
  - Linear in |DB|

FHE-Based Solution (2)

Security

- $\mathcal{L}_1$ leakage
  - #DB
  - ~~Equality~~
  - ~~PK: DB*~~
- $\mathcal{L}_2$ leakage
  - ~~access pattern~~
  - ~~PK: keyword~~

Efficiency

- Very very slow search
  - ~~Interactive (1 round)~~
  - Linear in |Data|

Oblivious RAM

- Encryption that supports private reads and writes
- Examples:
  - Square-root scheme [Goldreich-Ostrovsky92]
  - Hierarchichal scheme [Goldreich-Ostrovsky]

ORAM-Based Solution

- OStruct$(1^k, \text{Mem}) \Longmapsto K, \Omega$
- ORead$((K, i), \Omega)$
  $\Longmapsto (\text{Mem}[i], \perp)$
- OWrite$((K, i, v), \Omega)$
  $\Longmapsto (\perp, \Omega')$

EDB = OStruct

ORAM-Based Solution

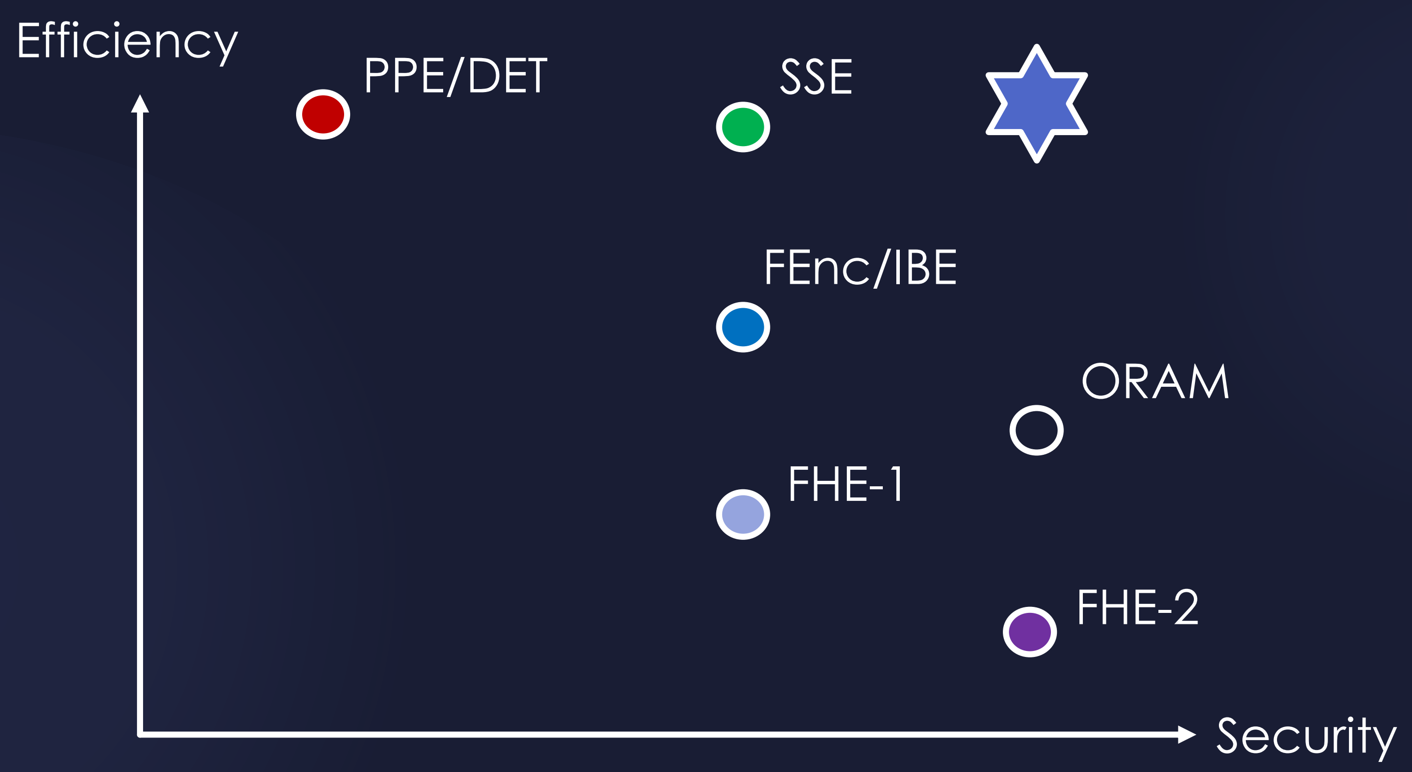
Security

- $\mathcal{L}_1$ leakage
  - #DB
  - ~~Equality~~
  - ~~PK: DB*~~
- $\mathcal{L}_2$ leakage
  - ~~access pattern~~
  - ~~PK: keyword~~

Efficiency

- Very slow search
  - 1 R/W = polylog(n) R+W

Tradeoffs



Searchable Symmetric Encryption

Searchable Symmetric Encryption

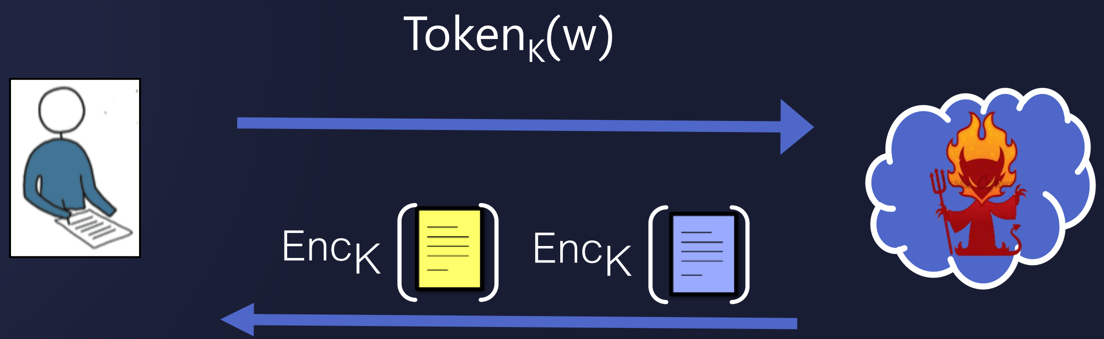
- Encryption that supports very slow search
  [Song-Wagner-Perrig01]
- Encryption that supports slow search
  [Song-Wagner-Perrig01, Goh03, Chang-Mitzenmacher05]
- Encryption that supports fast search
  [Curtmola-Garay-K.-Ostrovsky06]

- Very slow: linear in | Data |
- Slow: linear in #DB
- Fast: sub-linear in #DB

Searchable Encryption

- SSE(DB) $\Longmapsto$ (K, EDB)
- Token(K, w) $\Longmapsto$ t
- Search(EDB, t) $\Longmapsto$ $(id_1,\ldots,id_m)$
- Dec(K, c) $\Longmapsto$ m

EDB = SSE 



Security Definitions

- Security against chosen-keyword attack [Goh03,Chang-Mitzenmacher05,Curtmola-Garay-K.-OstrovskyO06]

CKA1: "Protects files and keywords even if chosen by adversary"

- Security against *adaptive* chosen-keywords attacks [Curtmola-Garay-K.-Ostrovsky06]

CKA2: "Protects files and keywords even if chosen by adversary, and even if chosen as a function of ciphertexts, index, and previous results"

Security Definitions

- Universal composability
  [Kurosawa-Ohtaki12, Canetti01]

UC: "Remains CKA2-secure even if composed arbitrarily"

CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

- *Simulation*-based definition
  - ``The EDB and tokens are simulatable given the leakage generated by an adversarially- and adaptively-chosen DB and queries''
  - Leakage
    - access pattern: pointers to (encrypted) files that satisfy search query
    - query pattern: whether a search query is repeated

CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

- *Game*-based definition
  - ``The EDBs and tokens generated from two adversarially- and adaptively-chosen DBs and query sequences with the same leakage are indistinguishable"
  - Leakage
    - access pattern: pointers to (encrypted) files that satisfy search query
    - query pattern: whether a search query is repeated

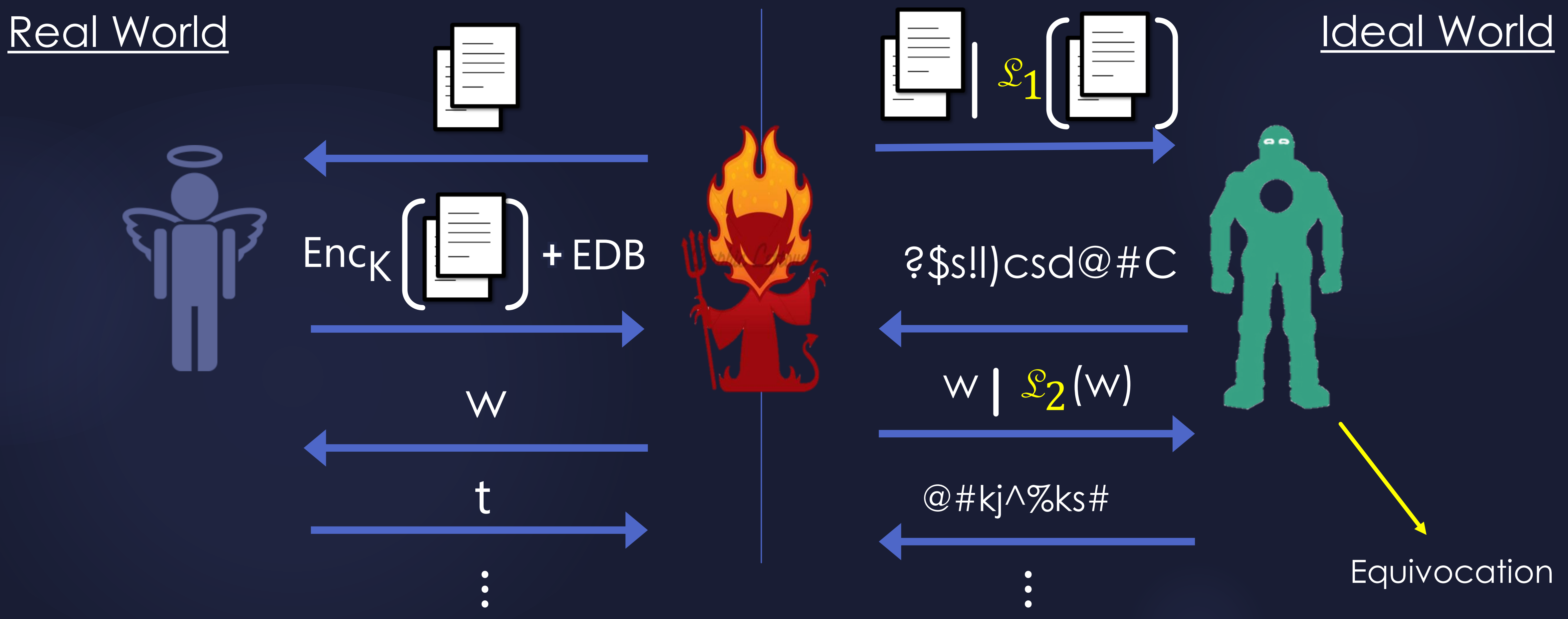
CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

- Simulation-based ⇒ Game-based
- Game-based ⇒ Simulation-based
  - If given leakage, one can efficiently sample plaintext docs and queries with same leakage profile
- Similar to results for functional encryption [O'Neill10, Boneh-Sahai-Waters11]

CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]



CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

- Simulator "commits" to encryptions before queries are made
  - requires equivocation and some form of non-committing encryption
- [Chase-K.10]
  - Lower bound on token length (simulation + w/o ROs)
    - ≈ [Nielsen02]
    - $\Omega(\lambda \cdot \log(n))$
      - n: # of documents
      - λ: max (over kw) # of documents w/ keyword
  - Lower bound on FE token length (simulation + w/o ROs)
    - Token proportional to maximum # of ciphertexts

Constructions

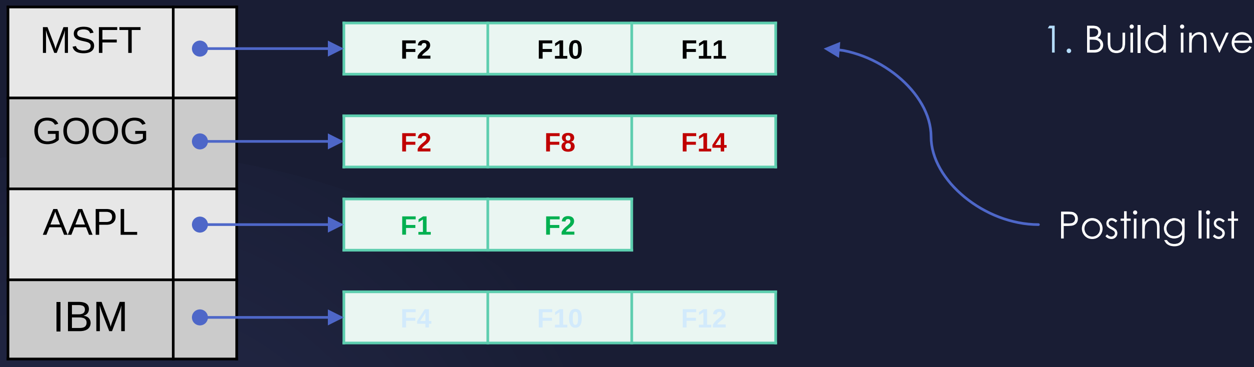
Searchable Symmetric Encryption

Scheme	Updates	Security	Search	Parallel	Queries
[SWP00]	No	CPA	O(|Data|)	**O(n/p)**	Single
[Goh03]	**Yes**	CKA1	O(#DB)	**O(n/p)**	Single
[CM05]	No	CKA1	O(#DB)	**O(n/p)**	Single
[CGKO06] #1	No	CKA1	**O(OPT)**	No	Single
[CGKO06] #2	No	**CKA2**	**O(OPT)**	No	Single
[CK10]	No	**CKA2**	**O(OPT)**	No	Single
[vLSDHJ10]	**Yes**	**CKA2**	O(log #W)	No	Single
[KO12]	No	**UC**	O(#DB)	No	Single
[KPR12]	**Yes**	**CKA2**	**O(OPT)**	No	Single
[KP13]	**Yes**	**CKA2**	**O(OPT·log(n))**	**$O(\frac{OPT}{p} \cdot \log(n))$**	Single
[CJJKRS13]	No	**CKA2**	**O(OPT)**	**Yes**	**Boolean**

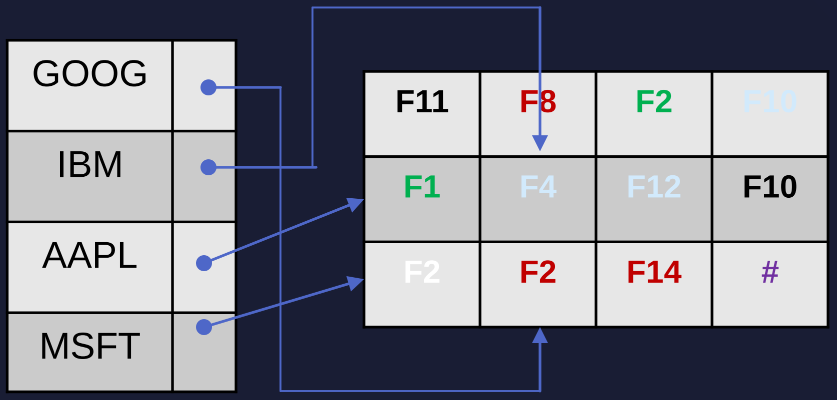
SSE-1

[Curtmola-Garay-K.-Ostrovsky06]

1. Build inverted/reverse index

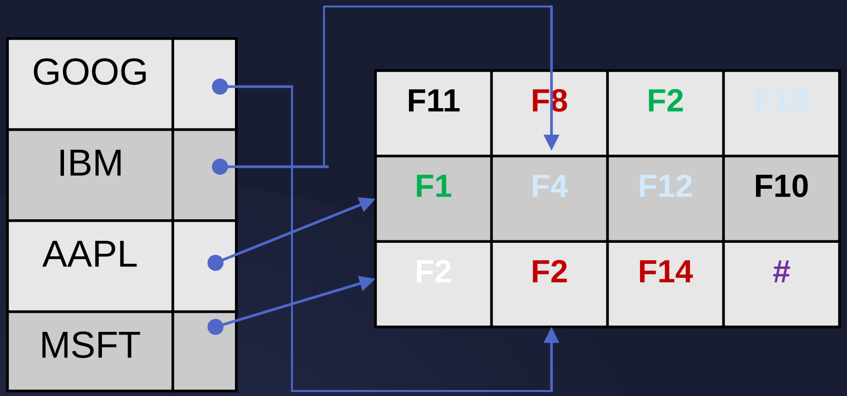


2. Randomly permute array & nodes



SSE-1

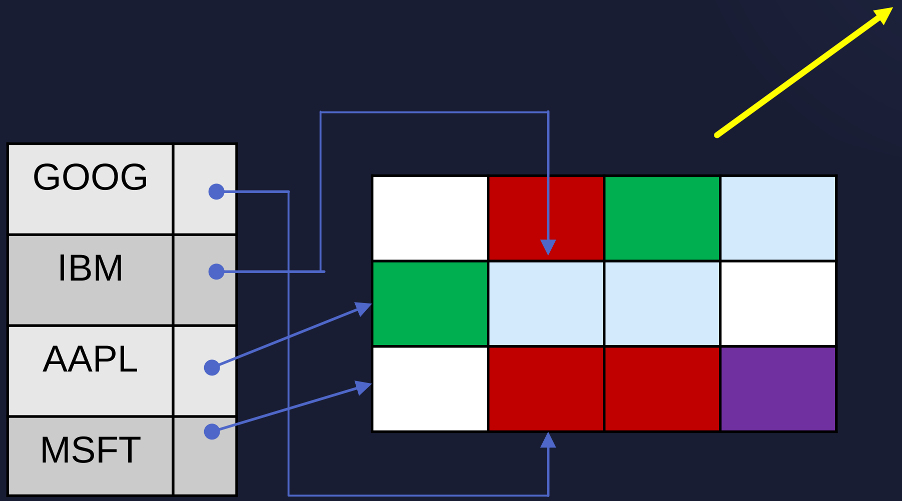
[Curtmola-Garay-K.-Ostrovsky06]



2. Randomly permute array & nodes

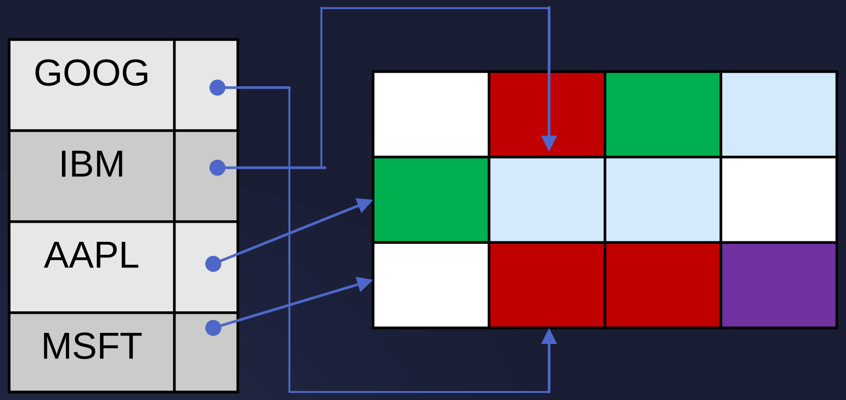
3. Encrypt nodes

CPA or Anonymous



SSE-1

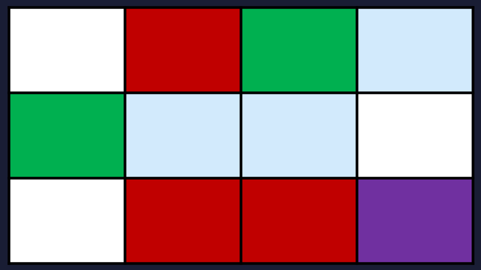
[Curtmola-Garay-K.-Ostrovsky06]



3. Encrypt nodes

4. “Hash” keyword & encrypt pointer

F_K(GOOG)	Enc_G(●, K)
F_K(IBM)	Enc_I(●, K)
F_K(AAPL)	Enc_A(●, K)
F_K(MSFT)	Enc_M(●, K)



Limitations of SSE-1

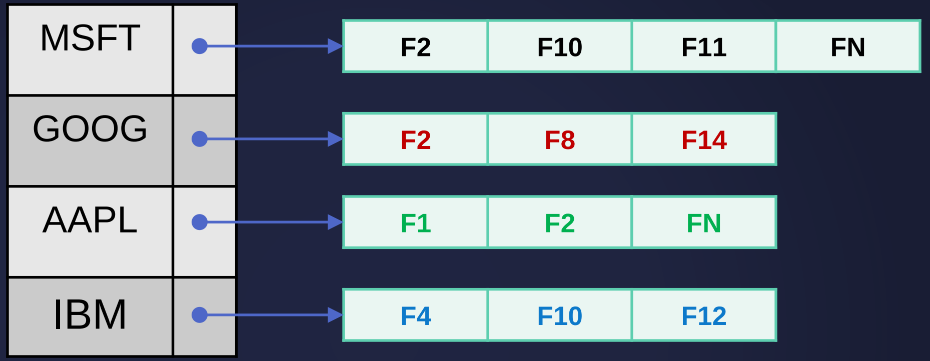
- Only CKA1-secure
  - addressed in [Chase-K.10]
- Only static
  - addressed in [K.-Papamanthou-Roeder12]
- High I/O complexity
  - addressed in [K.-Papamanthou13]
- Single keyword search
  - addressed in [Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13]

Making SSE-1 Adaptively Secure

- Idea #1 [Chase-K.-10]
  - replace general CPA encryption with standard PRF-based encryption
  - PRF-based encryption is non-committing
- Idea #2 [K.-Papamanthou-Roeder12]
  - PRF-based encryption not enough for dynamic data
    - Some add/delete patterns can make simulator commit to token before seeing outcome
    - Tokens must be equivocable (i.e., non-committing)
  - Use RO-based encryption

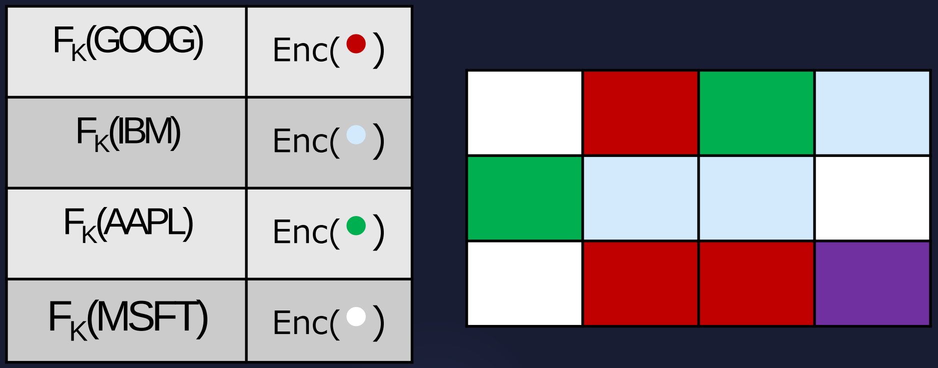
Making SSE-1 Dynamic

- Problem #1: Additions
  - given new file F_N = (AAPL, …, MSFT)
  - append node for F to list of every w_i in F



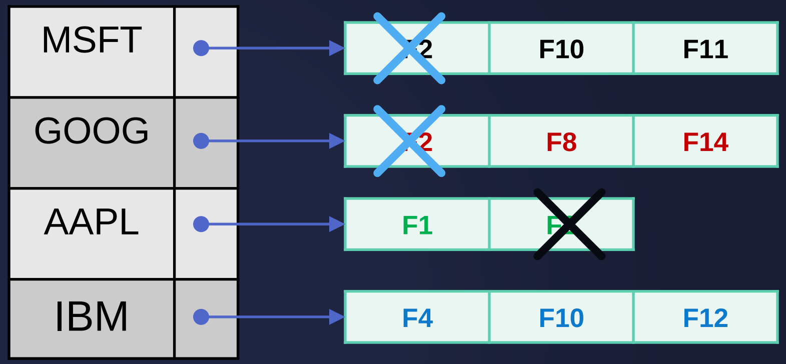
1. Over unencrypted index

2. Over encrypted index ???



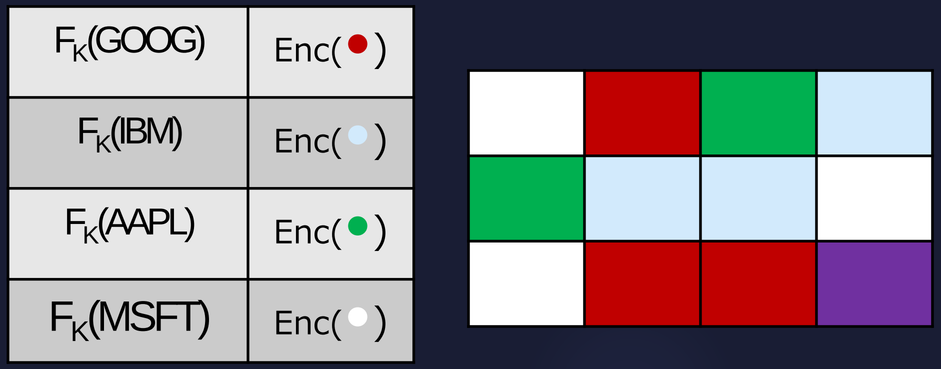
Making SSE-1 Dynamic

- Problem #2: Deletions
  - When deleting a file F_2 = (AAPL, …, MSFT)
  - delete all nodes for F_2 in every list



1. Over unencrypted index

2. Over encrypted index ???



Making SSE-1 Dynamic

- [K.-Papamanthou-Roeder12]
  - Idea #1
    - Memory management over encrypted data
    - Encrypted free list
  - Idea #2
    - PRF-based encryption is homomorphic
    - Pointer manipulation over encrypted data
  - Idea #3
    - deletion is handled using a “dual” SSE scheme
    - given deletion/search token for F_2 , returns pointers to F_2 ‘s nodes
    - then add them to the free list homomorphically

Making SSE-1 Boolean

- [Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13]
  - Use auxiliary (encrypted) data structure that stores labels for all (w, fid) pairs
  - Query SSE-1 data structure to receive (fid_1, …, fid_t) labels for w_1
  - Query auxiliary structure with labels for
    - (w_2, fid_1), …, (w_2, fid_t)
    - …
    - (w_q, fid_1), …, (w_q, fid_t)
  - Search is $O(t \cdot q)$ so optimize by using w_1's with small t

List intersection

State-of-the-art Implementation 2013

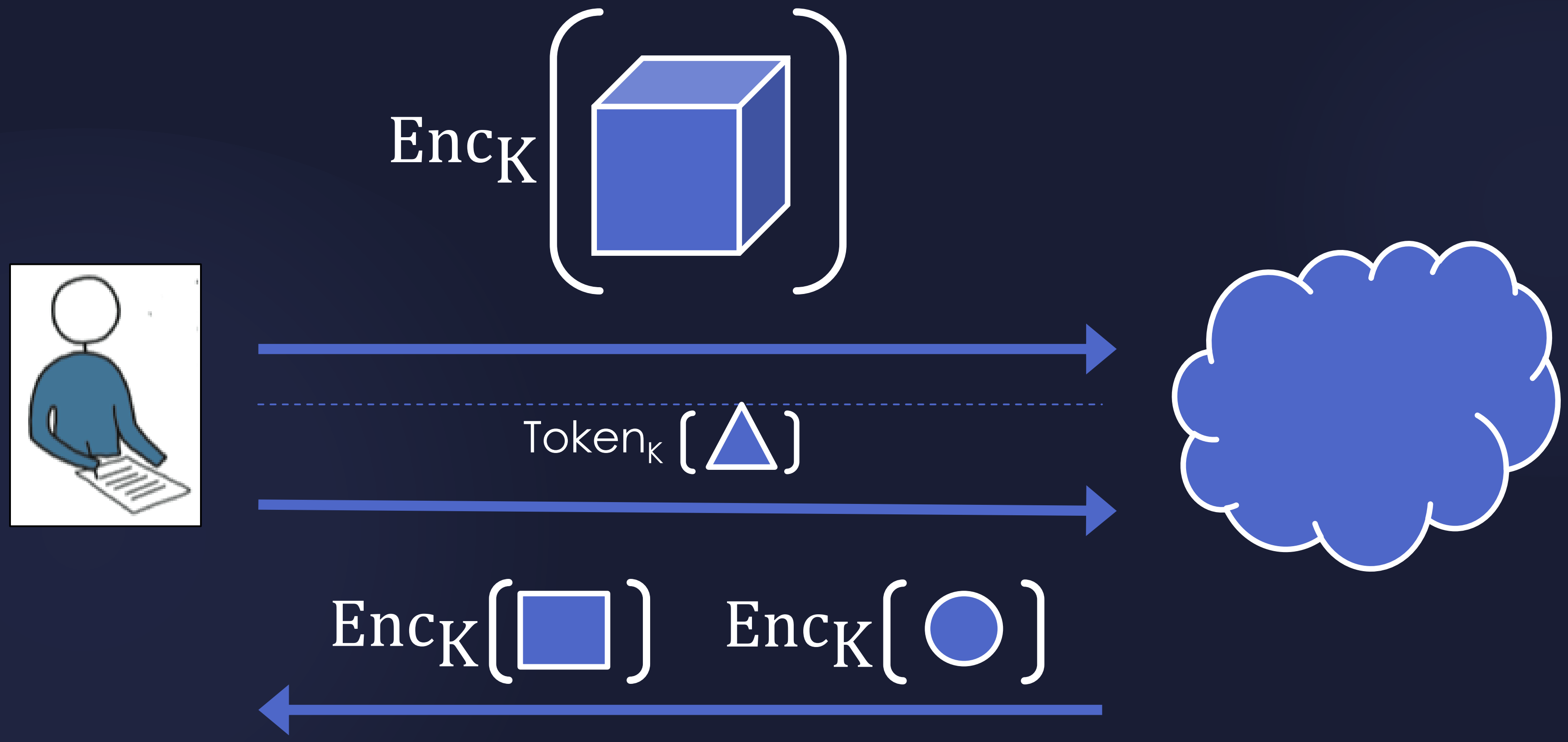
[Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13]

- 1.5 million emails & attachments
- EDB is 13 GB
- IBM Blade HS22
- Search for w_1 and w_2 less than .5 sec
  - w_1 in 1948 docs
  - w_2 in 1 million docs
- vs. cold MySQL 5.5
  - Single term: factor of .1 to 2 depending on term selectivity
  - Two terms: factor of .1 to ? depending on term selectivity
- vs. warm MySQL 5.5
  - slower by order of magnitude

Q: can we query other types of data?

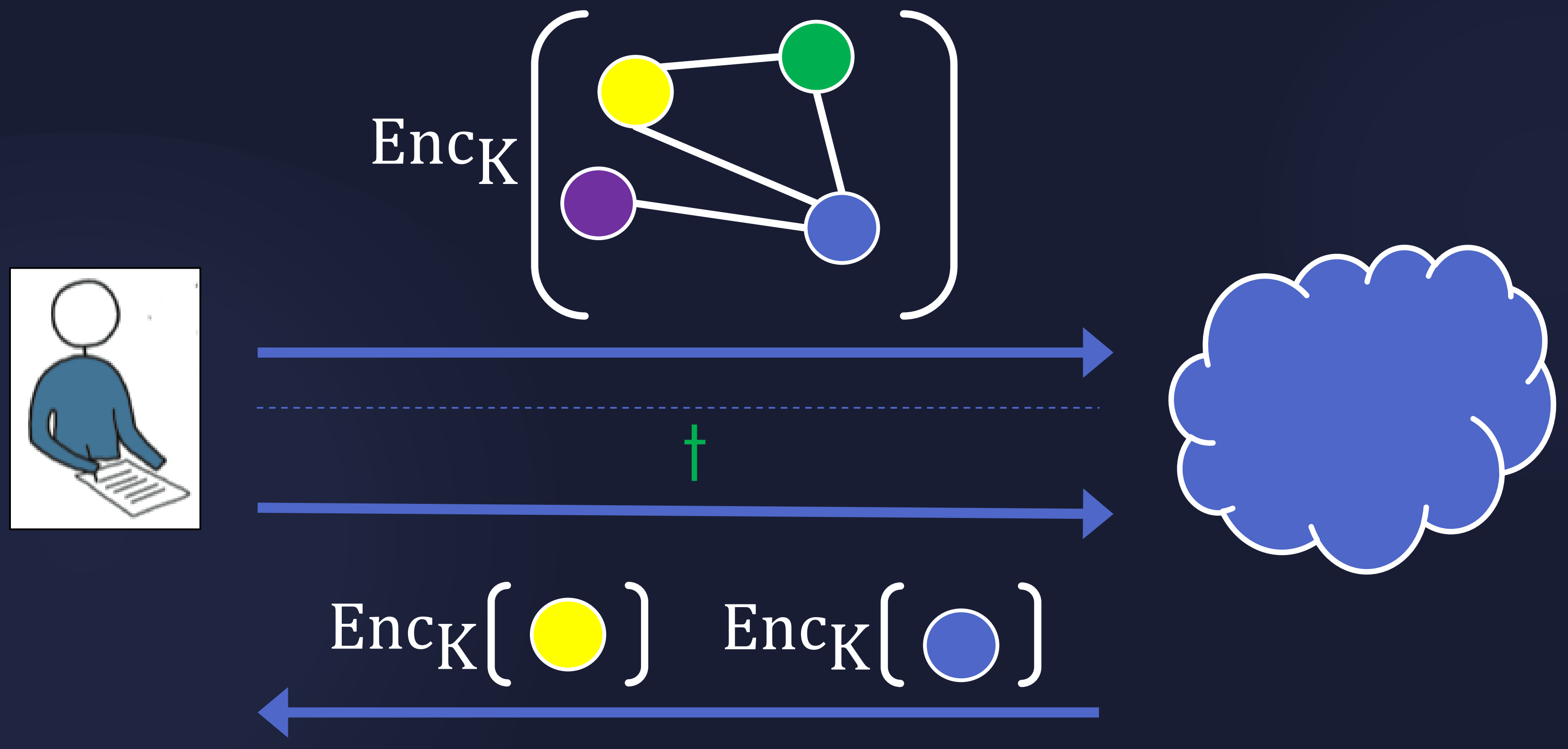
Structured Encryption

[Chase-K.10]

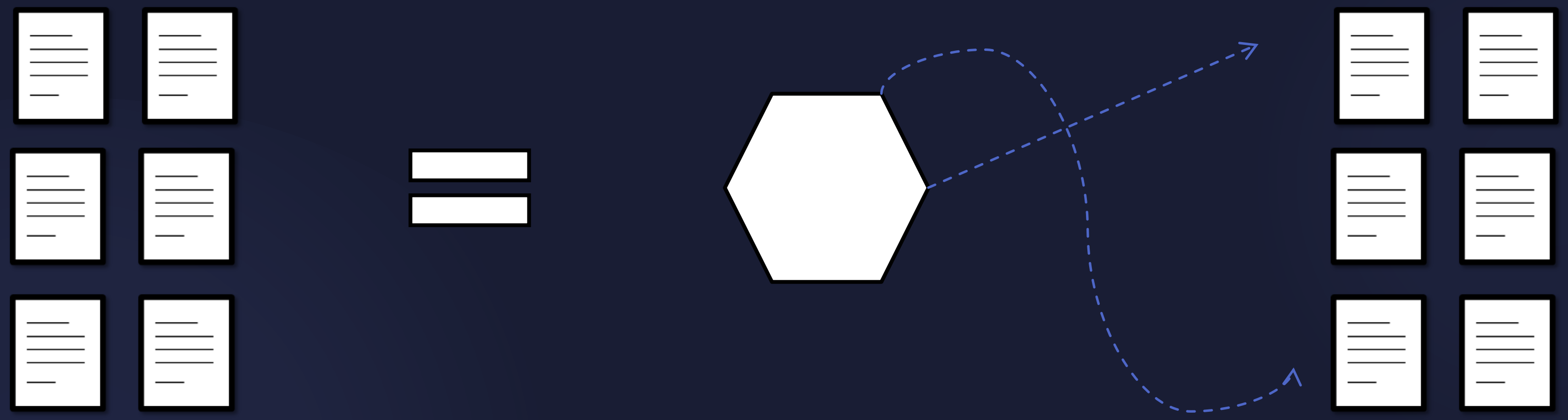


Structured Encryption

[Chase-K.10]

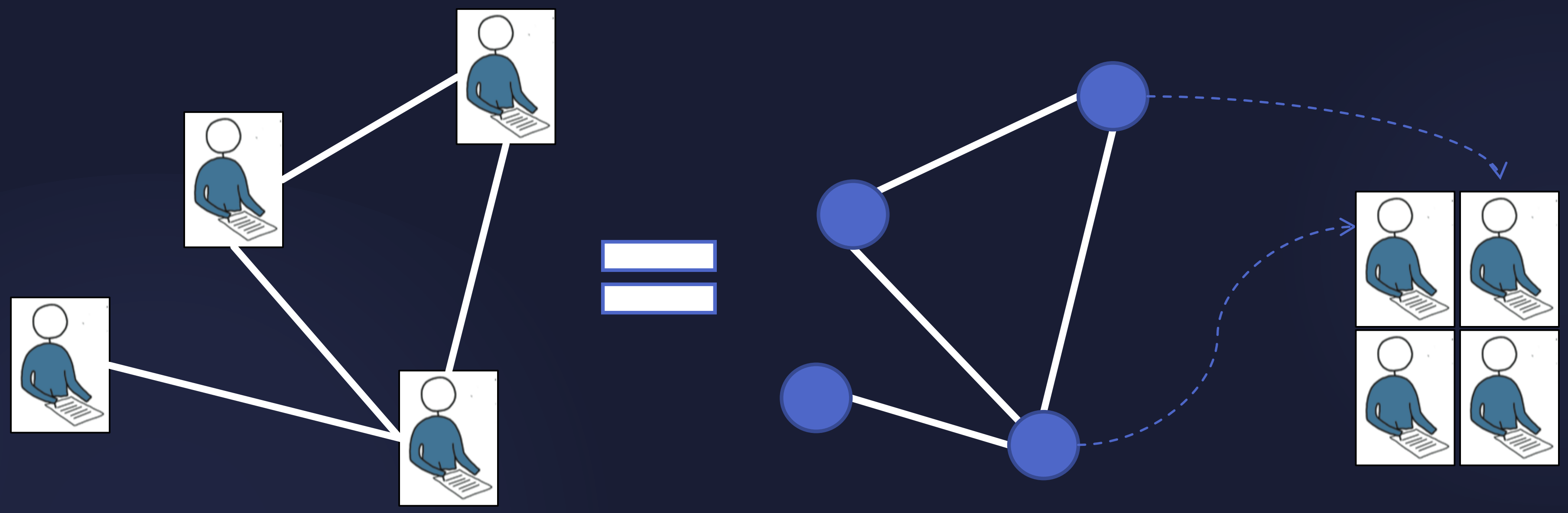


Structured Data



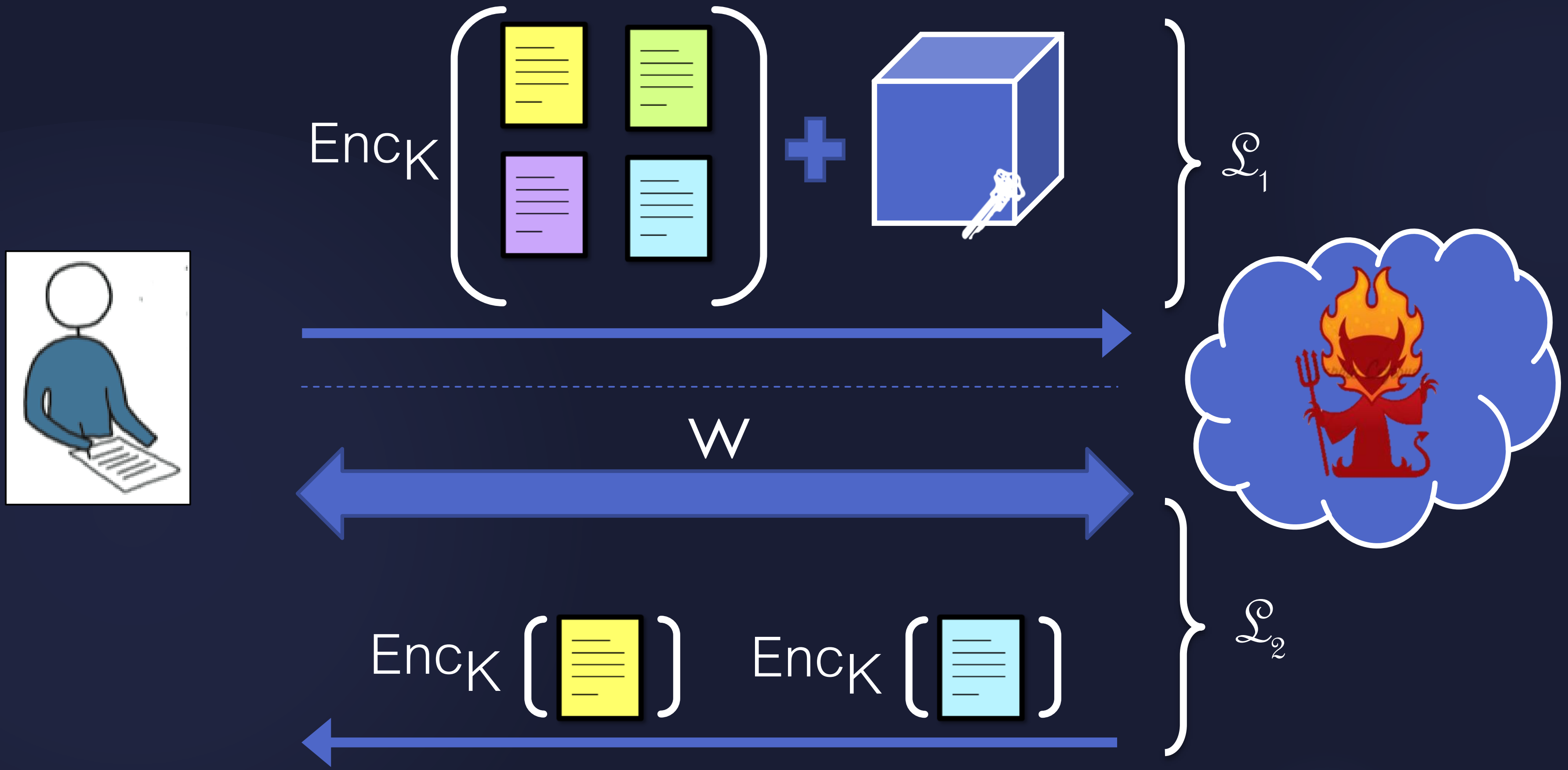
- Email archive = Index + Email text

Structured Data

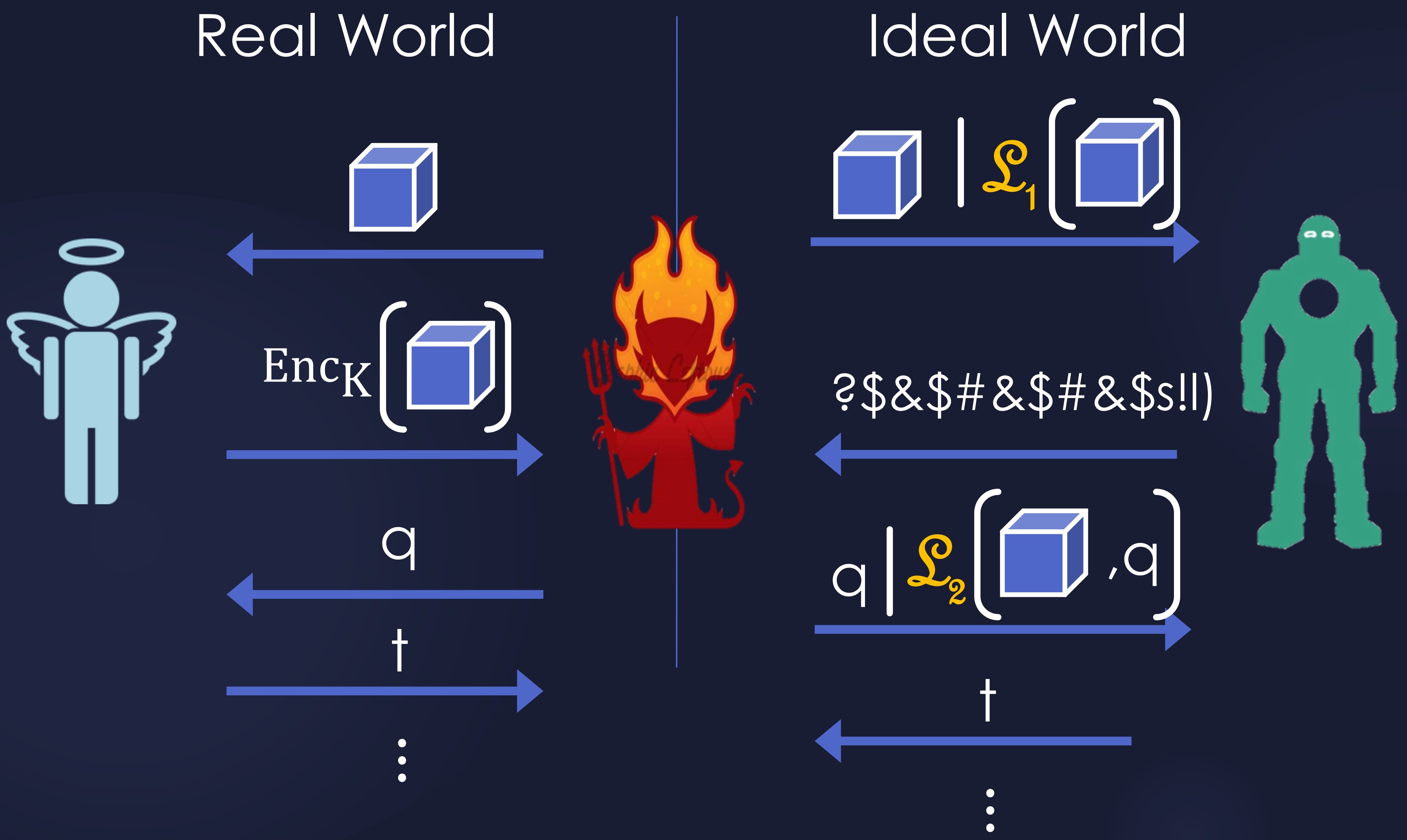


- Social network = Graph + Profiles

Structured Encryption



CQA2-Security

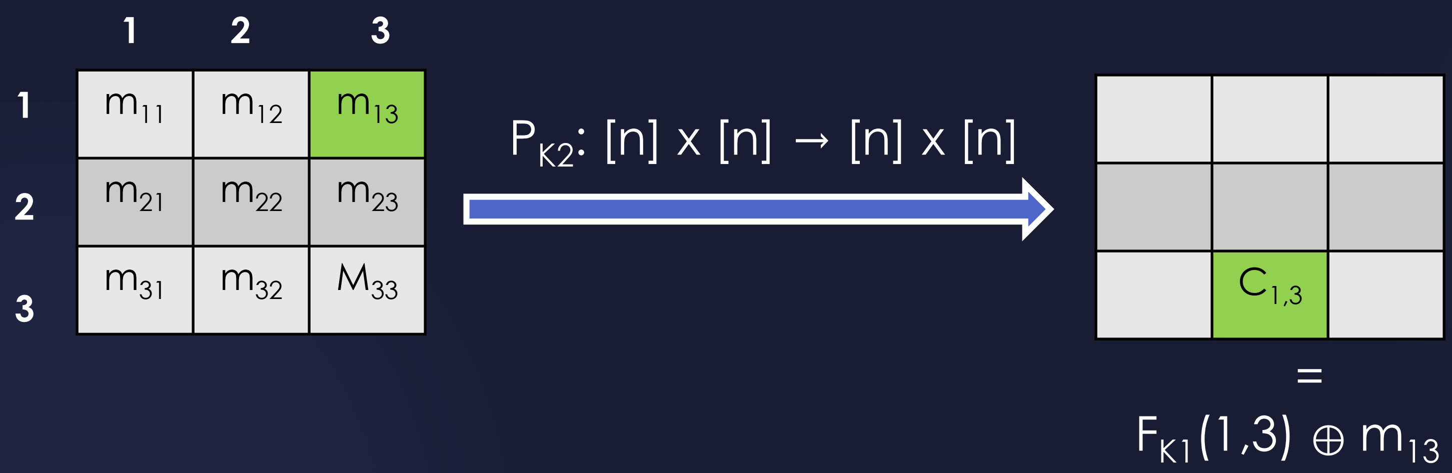


Constructions

[Chase-K.10]

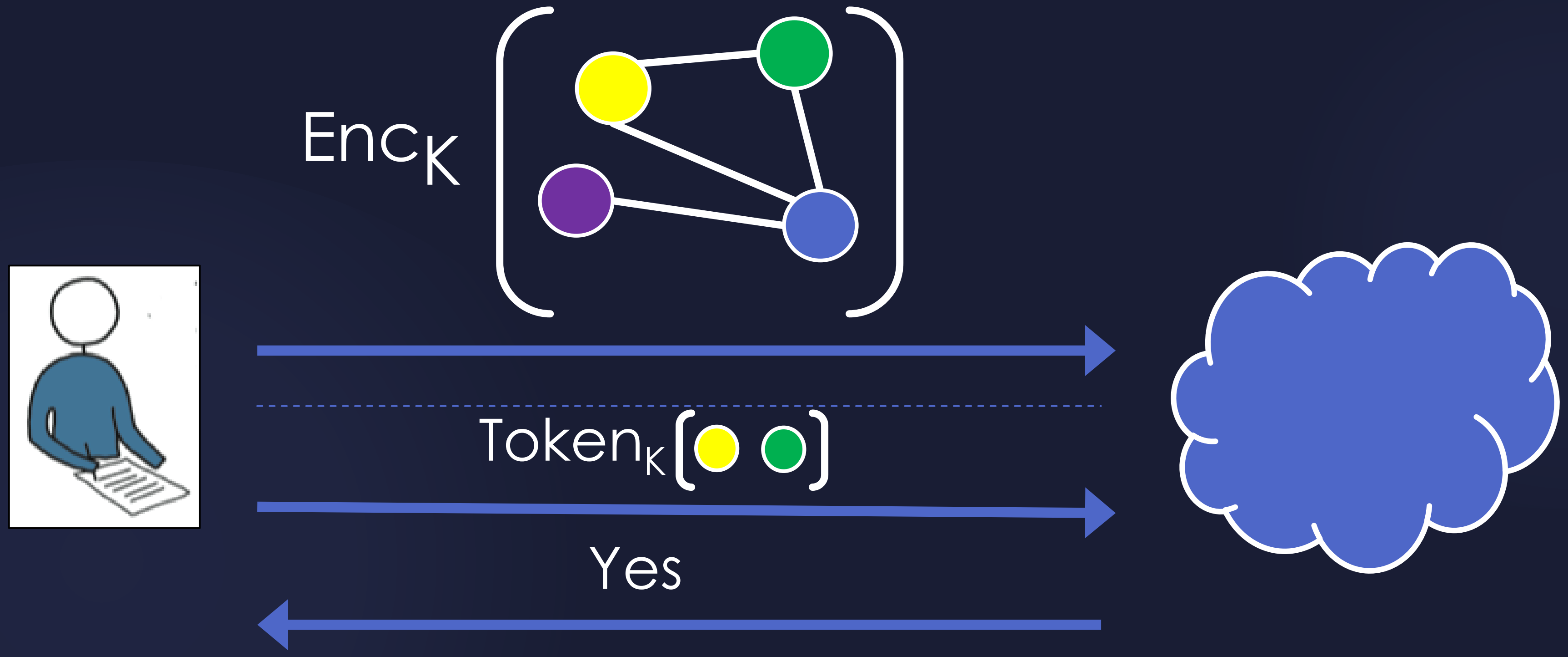
- 1-D Matrix encryption with lookup queries
- 2-D Matrix encryption with lookup queries [K.-Wei13]
- Graph encryption with adjacency queries
- Graph encryption with neighbor queries
- Web graph encryption with focused subgraph queries

Matrix Encryption



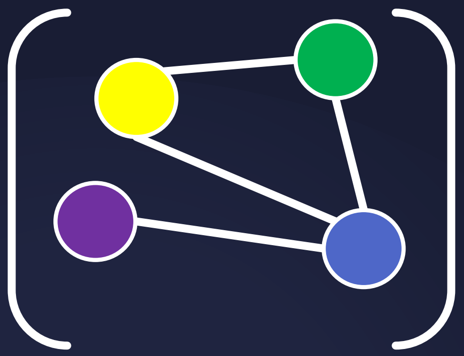
- Encrypt: permute + PRF-based encryption
- Search: $\text{Token}_K(1,3) = F_{K1}(1,3), P_{K2}(1,3)$

Graph Encryption + Adj. Queries



Graph Encryption + Adj. Queries

Enc_K

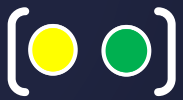


= Matrix-Enc(M_G) =

=

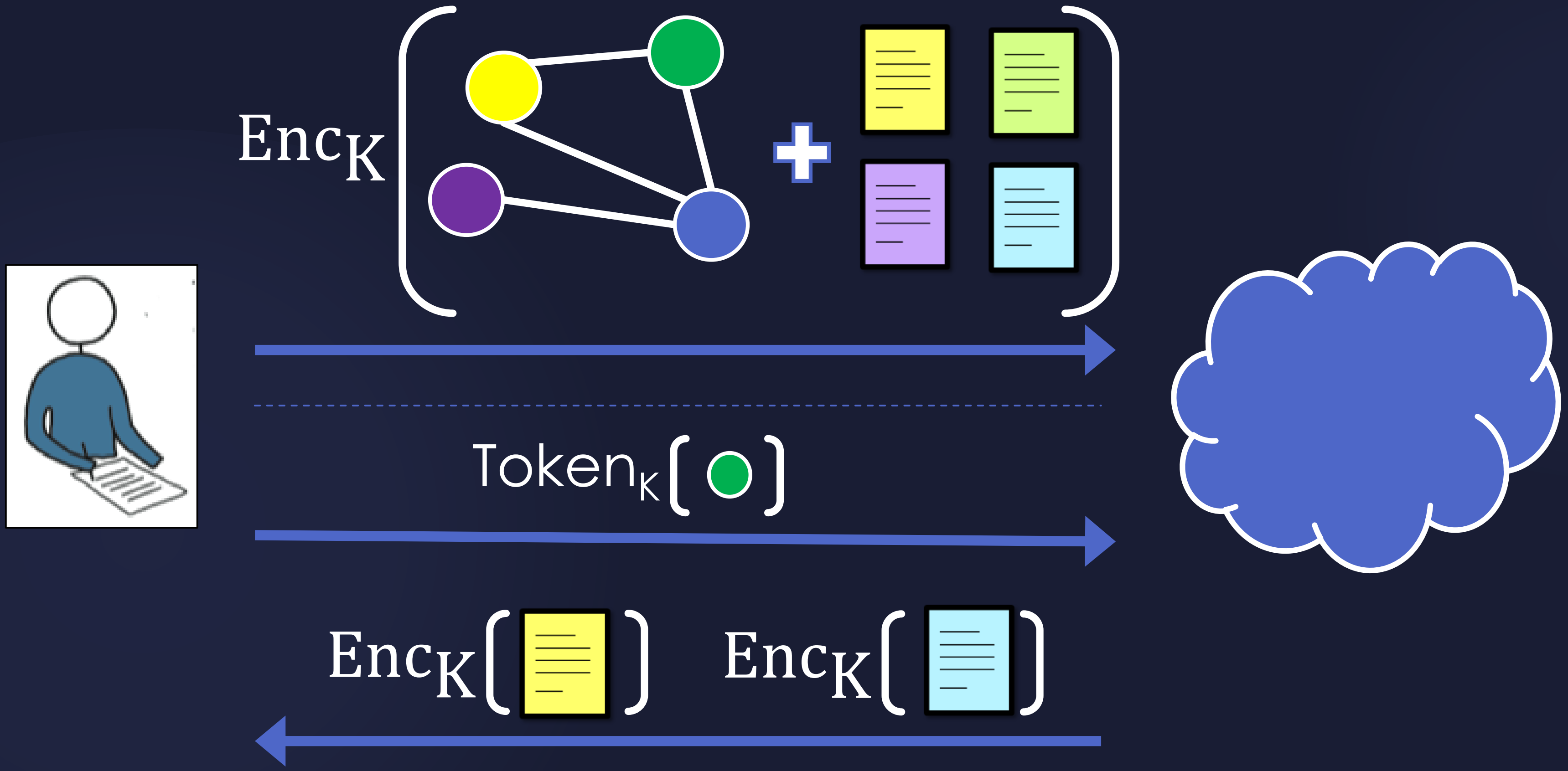
$F_{K1}(1,3) \oplus 1/0$

$Token_K$

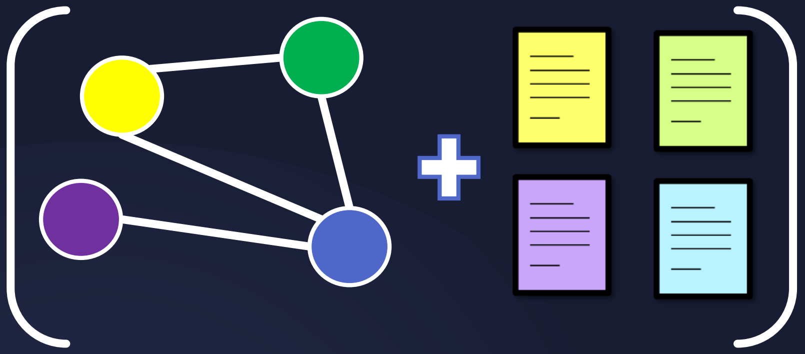


= Matrix-Lookup(N_i, N_j) = $Token_K(1,3) = F_{K1}(1,3), P_{K2}(1,3)$

Graph Encryption + Neigh. Queries



Graph Encryption + Neigh. Queries

Enc_K  $=$ $SSE(N_1, \ldots, N_n)$

↓

$N_n = (N_3, \ldots, N_{12})$

$Token_K$[] $=$ Search (N_i)

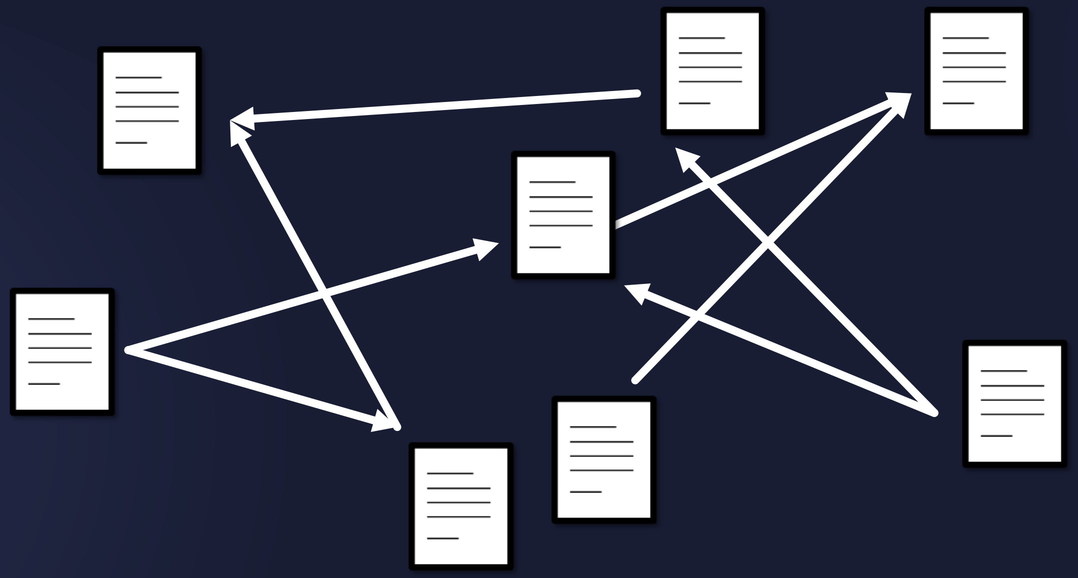
Complex Queries

Labeled Graph Encryption + FSQs

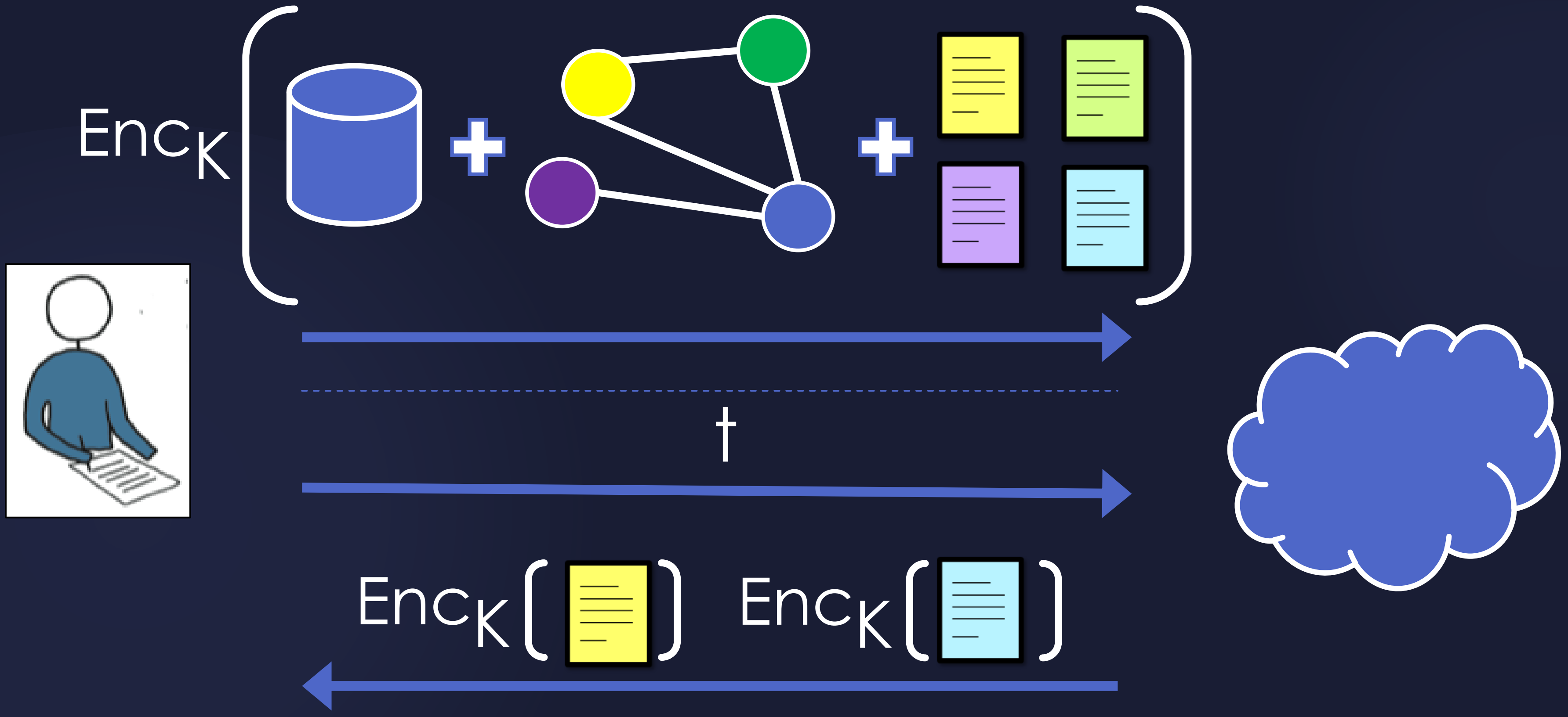
- Labeled graphs
  - mix text and graph structure
  - Web graphs: pages + hyperlinks
  - Graph DBs: patient information + relationships
  - Social networks: user information + friendships
- Focused subgraph queries on web graphs
  - Kleinberg's HITS algorithm [Kleinberg99]
- Focused subgraph queries on graph DBs
  - Find patients with symptom X and anyone related to them
- Focused subgraph queries on social networks
  - Find users that like product X and all their friends

Focused Subgraph Queries

Crypto



Labeled Graph Encryption + FSQs



Labeled Graph Encryption + FSQs

- Naïve approach
  - Encrypt text with SSE
  - Encrypt graph with Graph Enc w/ NQ
  - does not work!
- Combine schemes
  - Chaining technique

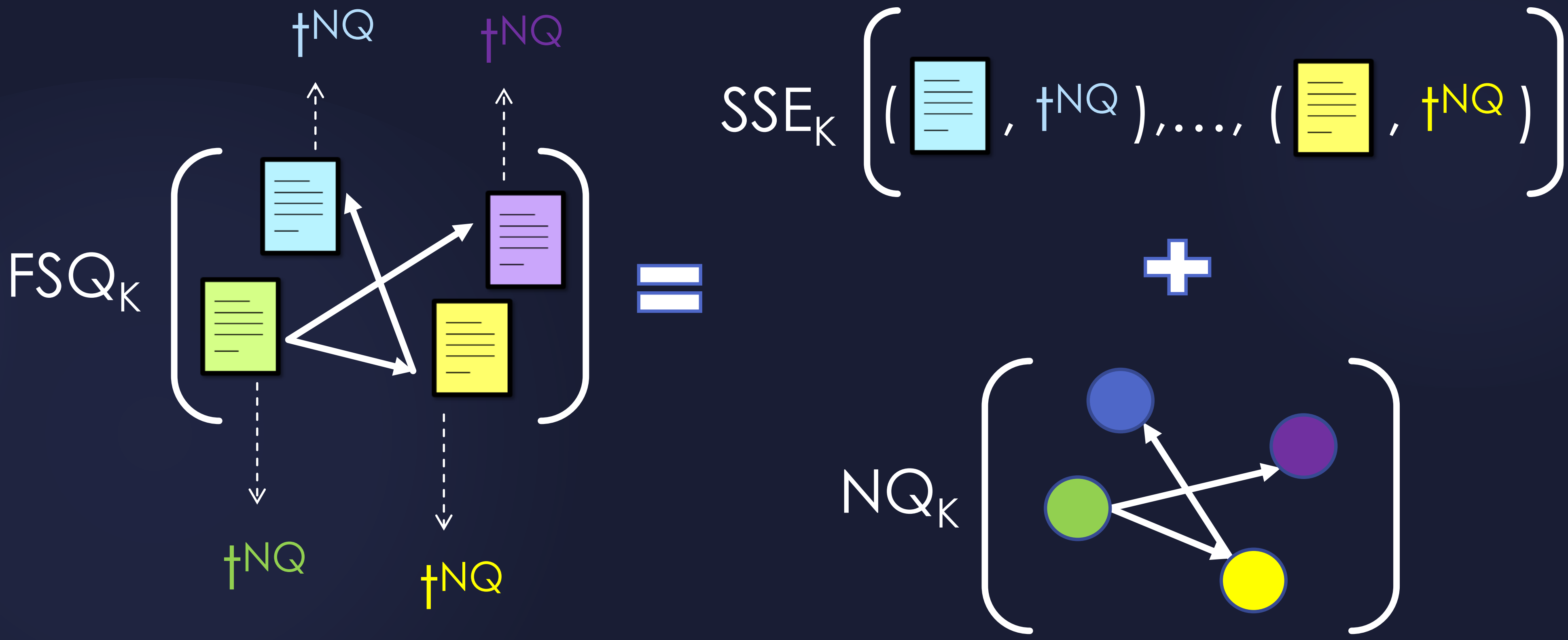
Chaining

- Best explained with example…
- Requires associative structured encryption
  - message space consists of pairs of
    - data items
    - arbitrary strings (semi-private data)
  - Query answer consists of pairs of
    - pointers to data items
    - associated string

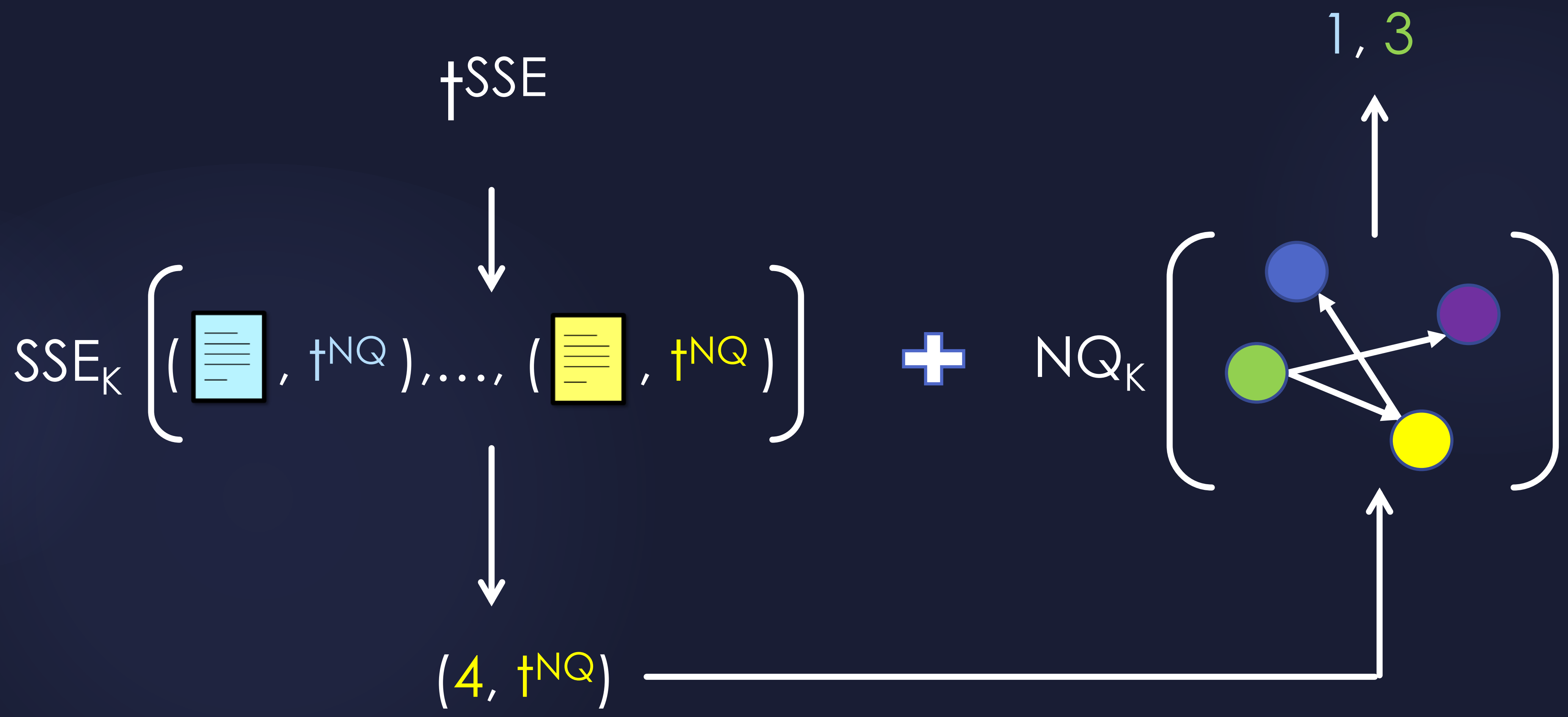
Chaining

- Constructions
  - [Curtmola-Garay-K.-Ostrovsky06] #1: is associative but only CKA1-secure
  - [Curtmola-Garay-K.-Ostrovsky06] #2: is CKA2-secure but not associative
  - [Chase-K.10]: SSE that is associative and CKA2-secure

Labeled Graph Encryption + FSQs



Labeled Graph Encryption + FSQs



Applications

Limitations of Secure Outsourcing

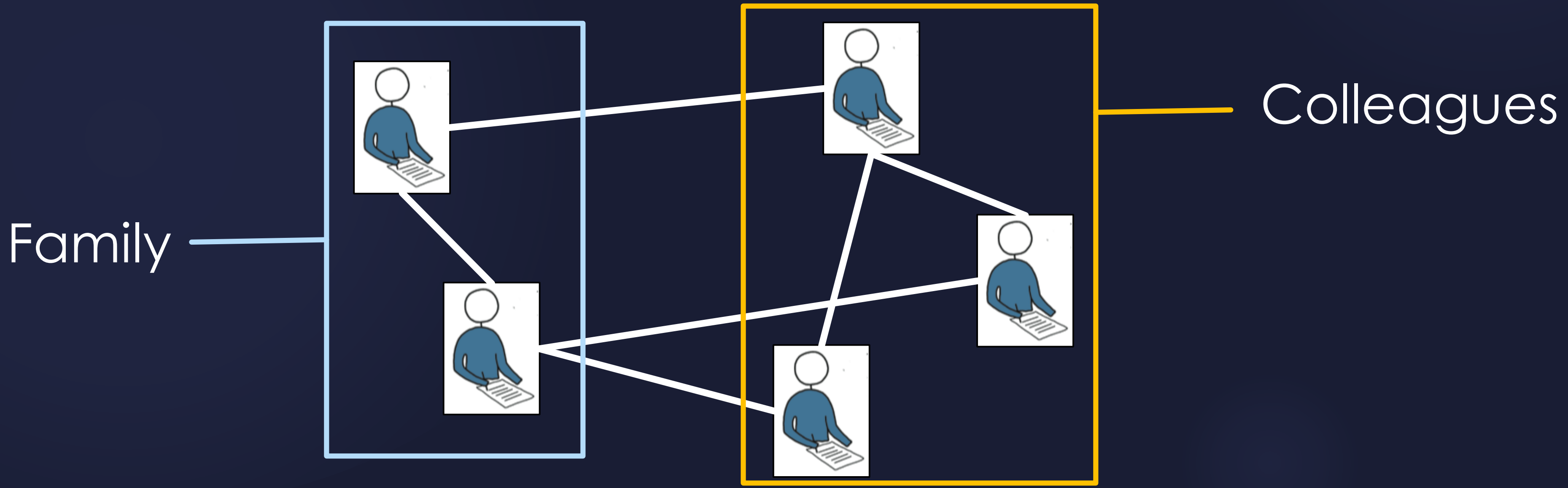
- 2PC & FHE don't scale to massive datasets (e.g., Petabytes)

Q: do we give up security completely?

Controlled Disclosure

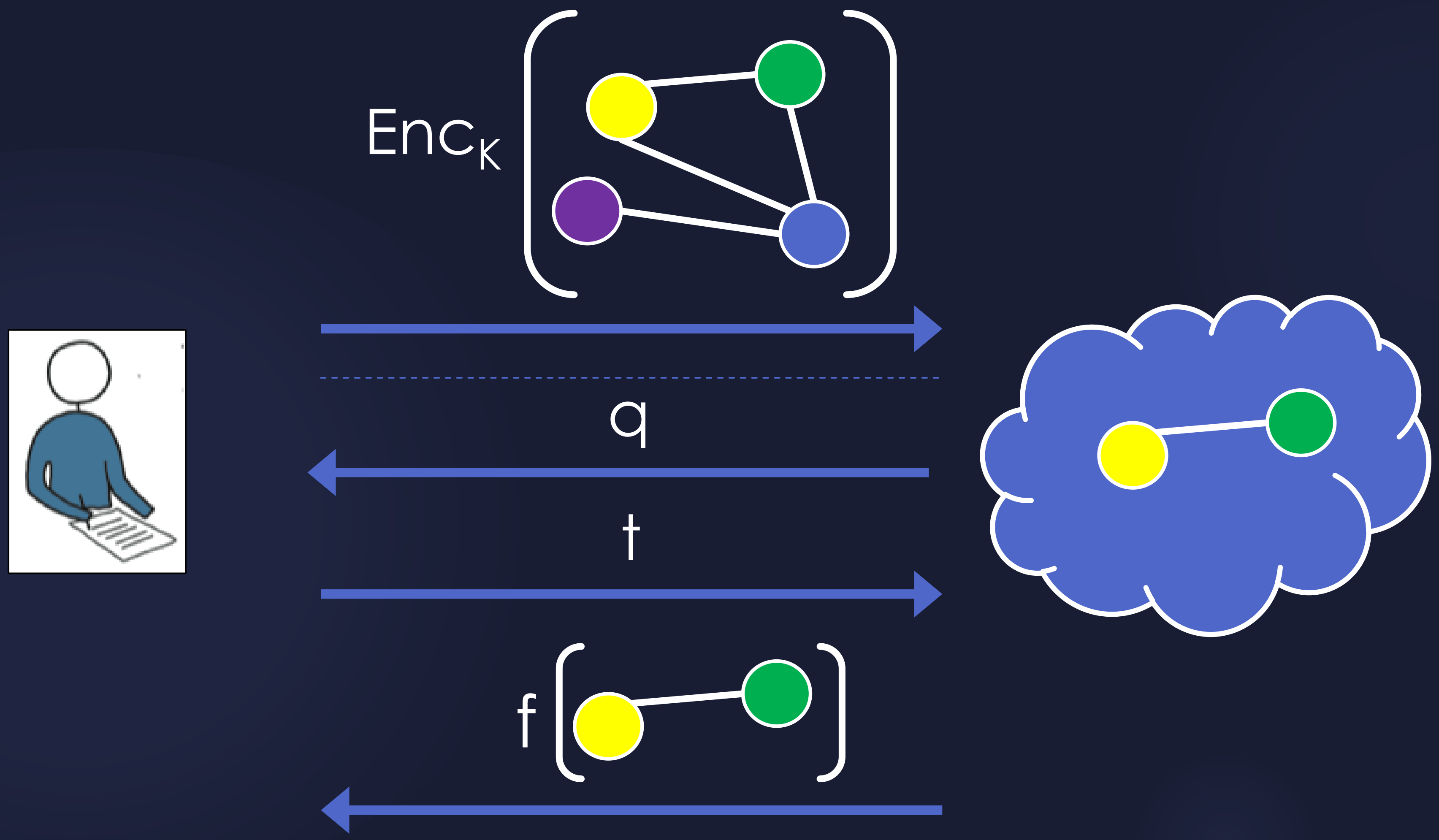
[Chase-K.10]

- Compromise
  - reveal only what is *necessary* for the server's computation
- Local algorithms
  - Don't need to ``see'' all their input
  - e.g., simulated annealing, hill climbing, genetic algorithms, graph algorithms, link-analysis algorithms, …



Controlled Disclosure

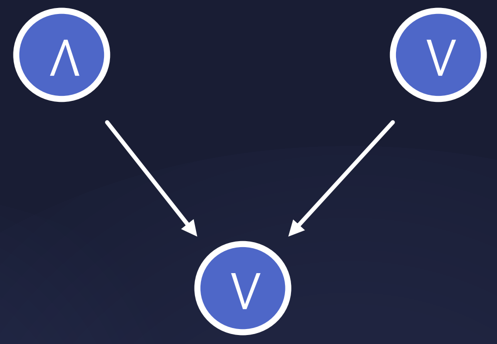
[Chase-K.10]

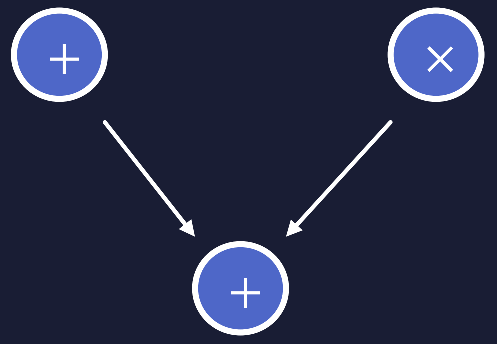


Garbled Circuits

- Two-party computation [Yao82]
- Server-aided multi-party computation [K.-Mohassel-Raykova12]
- Covert multi-party computation [Chandran-Goyal-Sahai-Ostrovsky07]
- Homomorphic encryption [Gentry-Halevi-Vaikuntanathan10]
- Functional encryption [Seylioglu-Sahai10]
- Single-round oblivious RAMs [Lu-Ostrovsky13]
- Leakage-resilient OT [Jarvinen-Kolesnikov-Sadeghi-Schneider10]
- One-time programs [Goldwasser-Kalai-Rothblum08]
- Verifiable computation [Gennaro-Gentry-Parno10]
- Randomized encodings [Applebaum-Ishai-Kushilevitz06]

Circuits





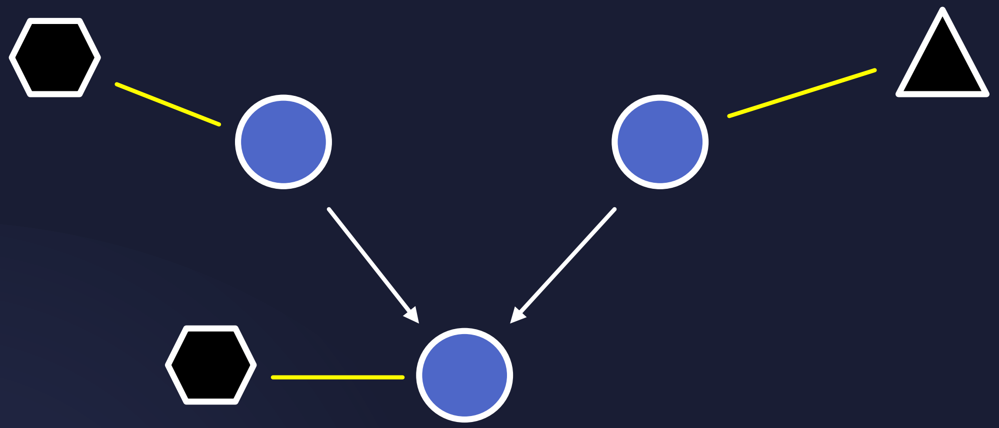
Boolean circuits

- [Yao82]: public-key techniques
- [Lindell-Pinkas09]: double encryption
- [Naor-Pinkas-Sumner99]: hash functions
- [Bellare-Hoang-Rogaway12]: dual-key ciphers

Arithmetic circuits

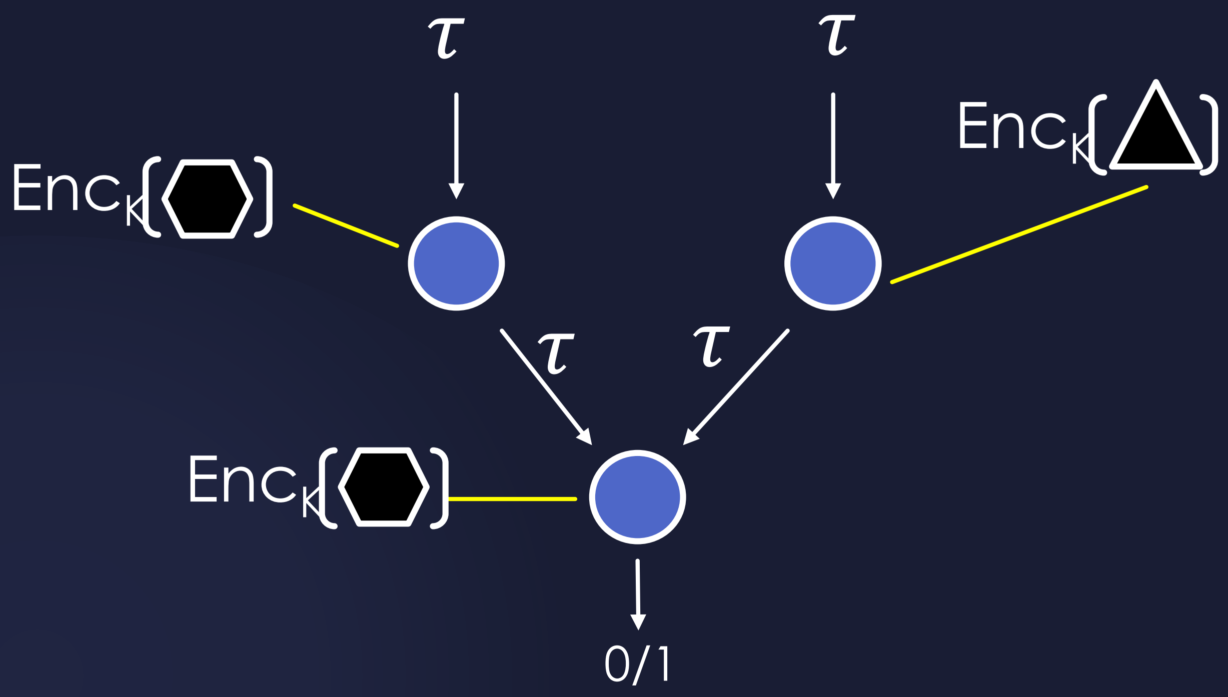
- [Applebaum-Ishai-Kushilevitz12]: affine randomized encodings

Structured Circuits



- Efficient for "structured problems"
  - Search, graphs, DFAs, branching programs

How to Garble a Structured Circuit



- Correctness
  - Encrypt data structures
  - Associativity (store & release tokens)
  - Dimensionality (merge tokens)
- Security
  - CQA1 enc ⇒ SIM1 & UNF1 garbling
  - CQA2 enc ⇒ SIM2 & UNF2 garbling

Observations

- Associativity
  - [Curtmola-Garay-K.-Ostrovsky06]: CQA1 & CQA2 inverted index encryption
  - [Chase-K.10]: CQA2 matrix, graph & labeled graph encryption
- Dimensionality
  - All previously-known constructions are 1-D
  - [K.-Wei13]: 2-D matrix encryption from 1-D matrix encryption + synthesizers
- Yao garbled gate $\Longleftrightarrow$ 2-D associative CQA1 matrix encryption scheme

Secure Two-Party Graph Computation

- Are and friends?
- Who are 's friends?
- Find the friends of anyone who likes my product
- Find the friends of anyone with disease X

Conclusions

Summary

- Various ways to search on encrypted data
  - PPE, FE, ORAM, FHE, SSE
- Searchable encryption
  - Best tradeoffs between security and efficiency
  - Very fast search
  - Updates
  - Boolean queries
  - Parallel and I/O-efficient search
- **Caveats**
  - Leaks (controlled) information
  - *We don't really understand what we're leaking*

What's Next?

- Framework for understanding leakage
- Concrete leakage attacks
  - Exploiting access pattern [Islam-Kuzu-Kantarcioglu12]
    - attack is NP-complete but can work in practice depending on auxiliary knowledge
  - Exploiting search pattern [Liu-Zhu-Wang-Tan13]
- Countermeasures to leakage

What's Next?

- More interesting search
  - SQL [Ada Popa-Redfield-Zeldovich-Balakrishnan11]
  - Ranked search [Chase-K.10]
  - Graph algorithms (web graphs, graph databases) [Chase-K.10]
- Techniques
  - abstractions & compilers/transformation
  - Auxiliary structures [K.-Papamanthou-Roeder12, Cash et al.13]
  - Chaining [Chase-K.10]
  - Homomorphic encryption [K.-Papamanthou-Roeder12]
- Verifiable search
  - [Bennabas-Gennaro-Vahlis12, K.-Papamanthou-Roeder12, Kurosawa-Ohtaki13]

What's Next?

- Generalizations
  - Structured encryption [Chase-K.10]
- Connections
  - Garbled circuits [K.-Wei13]
- Applications
  - Secure two-party computation [K.-Wei13]
  - Anonymous database queries [Jarecki-Jutla-Krawczyk-Rosu-Steiner13]
  - Controlled disclosure [Chase-K.10]

The End