Security Analysis of MongoDB Queryable Encryption

Zichen Gui, Kenneth G. Paterson, and Tianxin Tang

ETH zürich

APPLIED CRYPTO GROUP

https://www.flaticon.com/authors/dinosoftlabs icons created by DinosoftLabs - Flaticon
https://www.flaticon.com/authors/freepik icons created by Freepik - Flaticon
https://www.flaticon.com/ icons created by Flat Icons - Flaticon

MongoDB® and QE?

Jobs at SpringField

“Nuclear Technician”

“CEO”

MongoDB database

Document collection

doc_id	document

1	doc_1 = {
	“Name”: “Homer Simpson”,
	“Job”: “Nuclear Technician”,
	“Address”: “742 Evergreen Terrace"
	}

2	doc_2 = {
	”Name”: “Lenny Leonard”,
	“Job”: “Nuclear Technician”,
	“Address”: “123 Evergreen Terrace”
	}

3	doc_3 = {
	“Name”: “Charles Montgomery Burns”,
	“Job”: “CEO”,
	“Address”: “1000 Mammon Street”
	}

MongoDB database 🔒

Document collection

```
doc_id   document
1        doc_1 = {
                "Name": "Homer Simpson",
                "Job": "Nuclear Technician",
                "Address": "742 Evergreen Terrace"
              }

2        doc_2 = {
                "Name": "Lenny Leonard",
                "Job": "Nuclear Technician",
                "Address": "123 Evergreen Terrace"
              }

3        doc_3 = {
                "Name": "Charles Montgomery Burns",
                "Job": "CEO",
                "Address": "1000 Mammon Street"
              }
```

Linear scan ↓

(Inverted) search index

field value	doc_id
"Nuclear Technician"	1, 2
"CEO"	3

Equality search on the "Job" field

MongoDB database 🔒 = QE

Document collection

```
doc_id   document
1        doc_1 = {
                   ”Name”: “Homer Simpson”,
                   “Job”: “Nuclear Technician”,
                   “Address”: “742 Evergreen Terrace"
                 }

2        doc_2 = {
                   ”Name”: “Lenny Leonard”,
                   “Job”: “Nuclear Technician”,
                   “Address”: “123 Evergreen Terrace”
                 }

3        doc_3 = {
                   ”Name”: “Charles Montgomery Burns”,
                   “Job”: “CEO”,
                   “Address”: “1000 Mammon Street”
                 }
```

Linear scan

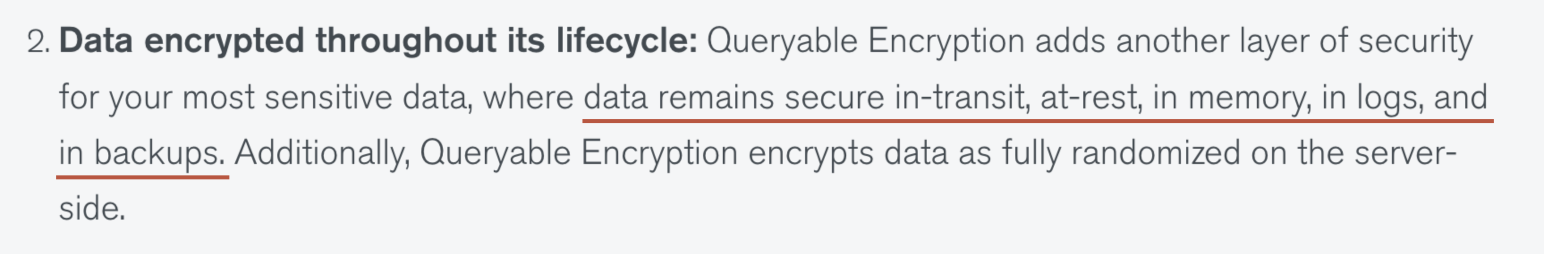
(Inverted) search index

field value	doc_id
“Nuclear Technician”	1, 2
“CEO”	3

Equality search on the “Job” field

MongoDB and QE

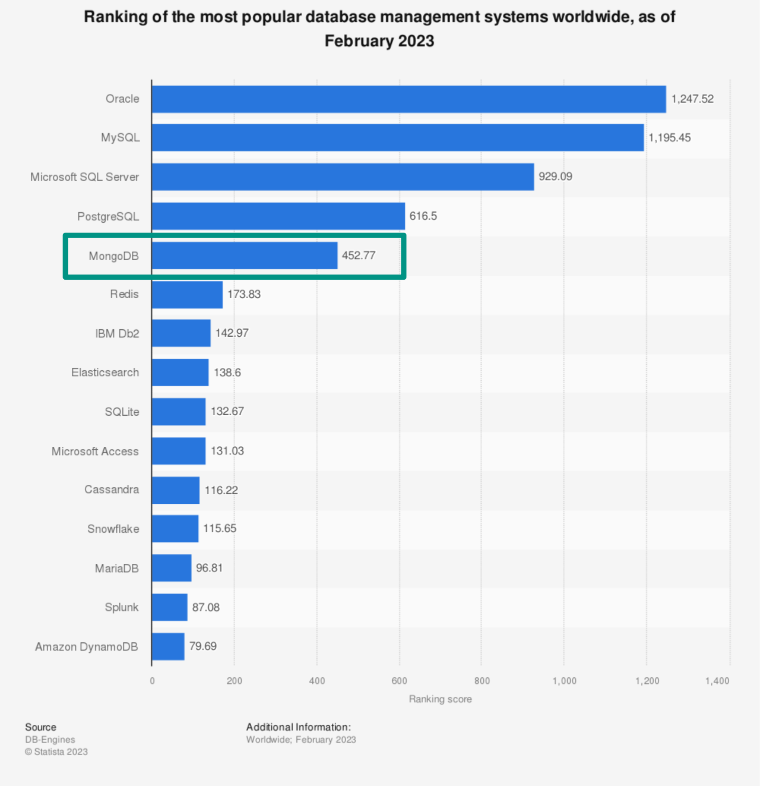
MongoDB claims that QE



2. **Data encrypted throughout its lifecycle:** Queryable Encryption adds another layer of security for your most sensitive data, where data remains secure in-transit, at-rest, in memory, in logs, and in backups. Additionally, Queryable Encryption encrypts data as fully randomized on the server-side.

https://www.mongodb.com/blog/post/mongodb-releases-queryable-encryption-preview

MongoDB and QE



Ranking of the most popular database management systems worldwide, as of February 2023

Source
DB-Engines
© Statista 2023

Additional Information:
Worldwide; February 2023

Business customers:

ebay BARCLAYS

Expedia SEGA

GE HealthCare

. . .

1: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/
2: https://www.mongodb.com/who-uses-mongodb

Are the security claims valid?

- QE is an instance of searchable encryption (SE) scheme.
- No security proof is available yet.

How does QE work?

Simplified token generation

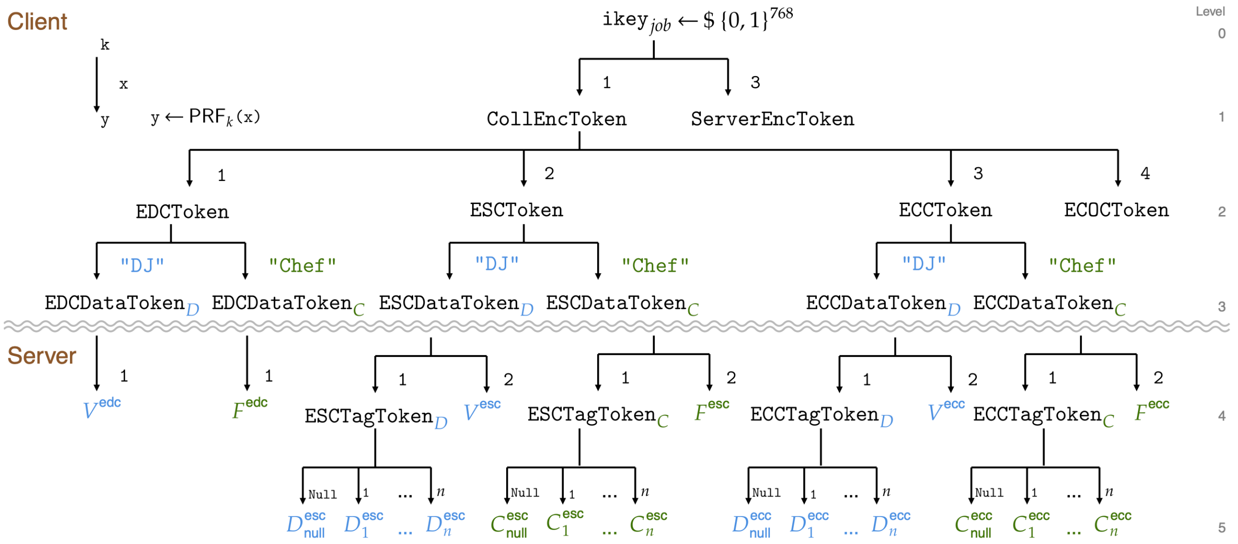

Client
k
x
y
$y \leftarrow \mathrm{PRF}_k(x)$
$\mathtt{ikey}_{job} \leftarrow \$ \{0,1\}^{768}$
Level
0
1
3
CollEncToken
ServerEncToken
1
1
2
3
4
EDCToken
ESCToken
ECCToken
ECOCToken
2
"DJ"
"Chef"
"DJ"
"Chef"
"DJ"
"Chef"
$\mathtt{EDCDataToken}_D$
$\mathtt{EDCDataToken}_C$
$\mathtt{ESCDataToken}_D$
$\mathtt{ESCDataToken}_C$
$\mathtt{ECCDataToken}_D$
$\mathtt{ECCDataToken}_C$
3
Server
1
1
1
2
1
2
1
2
1
2
V^{edc}
F^{edc}
$\mathtt{ESCTagToken}_D$
V^{esc}
$\mathtt{ESCTagToken}_C$
F^{esc}
$\mathtt{ECCTagToken}_D$
V^{ecc}
$\mathtt{ECCTagToken}_C$
F^{ecc}
4
Null
1
...
n
D^{esc}_{null} D^{esc}_1 ... D^{esc}_n
C^{esc}_{null} C^{esc}_1 ... C^{esc}_n
D^{ecc}_{null} D^{ecc}_1 ... D^{ecc}_n
C^{ecc}_{null} C^{ecc}_1 ... C^{ecc}_n
5

Figure 1: Simplified QE Token Derivation.

[GPT23]

Overview of QE (oversimplified)

Encrypted search index (ESC)

_id	value
$N_1^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \,\|\, 1)$	$\mathrm{Enc}_{K_2}(1)$
$N_2^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \,\|\, 2)$	$\mathrm{Enc}_{K_2}(2)$

N: “Nuclear Technician”

Omit doc3 (”CEO”) for simplication

Overview of QE (oversimplified)

Encrypted search index (ESC)

_id	value
$N_1^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \| \ 1)$	$\mathrm{Enc}_{K_2}(1)$
$N_2^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \| \ 2)$	$\mathrm{Enc}_{K_2}(2)$

N: “Nuclear Technician”

_id	encrypted document

$\mathrm{PRF}_{K_3}(1)$	edoc_1 = {
	“Name”: ***,
	“Job”: ***,
	“Address”: ***
	}

$\mathrm{PRF}_{K_3}(2)$	edoc_2 = {
	“Name”: ***,
	“Job”: ***,
	“Address”: ***
	}

ESC size

`Insert(doc₄)`

`Insert(doc₅)`

`Insert(doc₆)`

ESC

_id	value
N_1^{esc}	$\mathrm{Enc}_{K_2}(1)$
N_2^{esc}	$\mathrm{Enc}_{K_2}(2)$
N_3^{esc}	$\mathrm{Enc}_{K_2}(3)$
N_4^{esc}	$\mathrm{Enc}_{K_2}(4)$
N_5^{esc}	$\mathrm{Enc}_{K_2}(5)$

$N_1^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \| 1)$

$N_2^{esc} \leftarrow \mathrm{PRF}_{K_1}(\text{“Nuclear Technician”} \| 2)$

…

Compaction

ESC

_id	value
N_1^{esc}	$\text{Enc}_{K_2}(1)$
N_2^{esc}	$\text{Enc}_{K_2}(2)$

Compact →

ESC’

_id	value
N_{null}^{esc}	$\text{Enc}_{K_2}(2)$

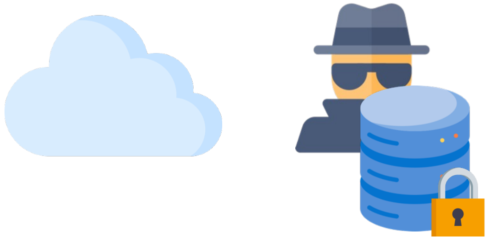
K_1, K_2

“Security” of QE

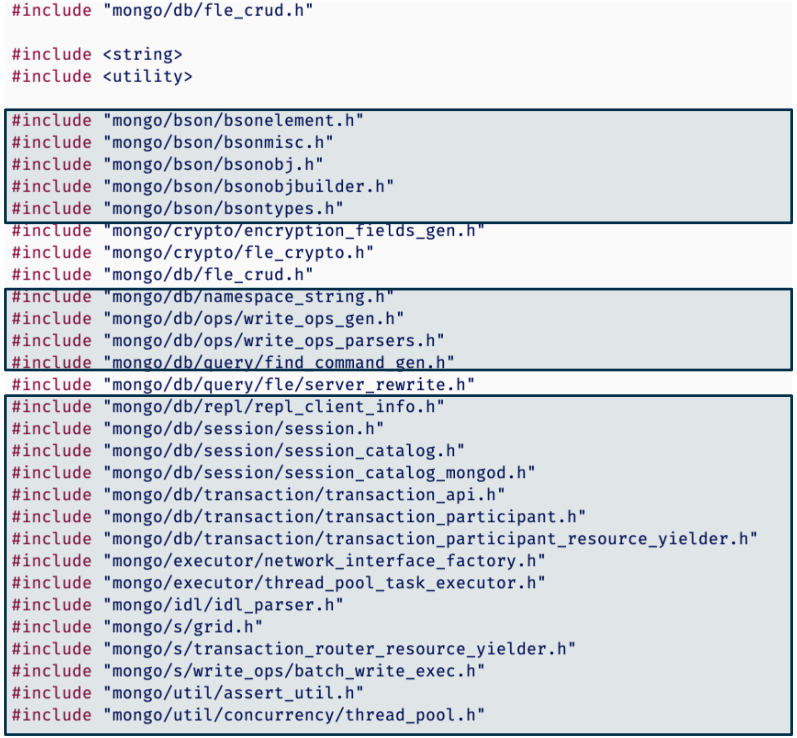
👍 QE satisfies snapshot security of searchable encryption.

“Security” of QE

👍 QE satisfies snapshot security of searchable encryption.



How was QE implemented in MongoDB?



```
#include "mongo/db/fle_crud.h"

#include <string>
#include <utility>

#include "mongo/bson/bsonelement.h"
#include "mongo/bson/bsonmisc.h"
#include "mongo/bson/bsonobj.h"
#include "mongo/bson/bsonobjbuilder.h"
#include "mongo/bson/bsontypes.h"
#include "mongo/crypto/encryption_fields_gen.h"
#include "mongo/crypto/fle_crypto.h"
#include "mongo/db/fle_crud.h"
#include "mongo/db/namespace_string.h"
#include "mongo/db/ops/write_ops_gen.h"
#include "mongo/db/ops/write_ops_parsers.h"
#include "mongo/db/query/find_command_gen.h"
#include "mongo/db/query/fle/server_rewrite.h"
#include "mongo/db/repl/repl_client_info.h"
#include "mongo/db/session/session.h"
#include "mongo/db/session/session_catalog.h"
#include "mongo/db/session/session_catalog_mongod.h"
#include "mongo/db/transaction/transaction_api.h"
#include "mongo/db/transaction/transaction_participant.h"
#include "mongo/db/transaction/transaction_participant_resource_yielder.h"
#include "mongo/executor/network_interface_factory.h"
#include "mongo/executor/thread_pool_task_executor.h"
#include "mongo/idl/idl_parser.h"
#include "mongo/s/grid.h"
#include "mongo/s/transaction_router_resource_yielder.h"
#include "mongo/s/write_ops/batch_write_exec.h"
#include "mongo/util/assert_util.h"
#include "mongo/util/concurrency/thread_pool.h"
```

"fle_crud_mongod.cpp"

Implementing core read and write functionality used by QE

24 out of 29 are native MongoDB library headers

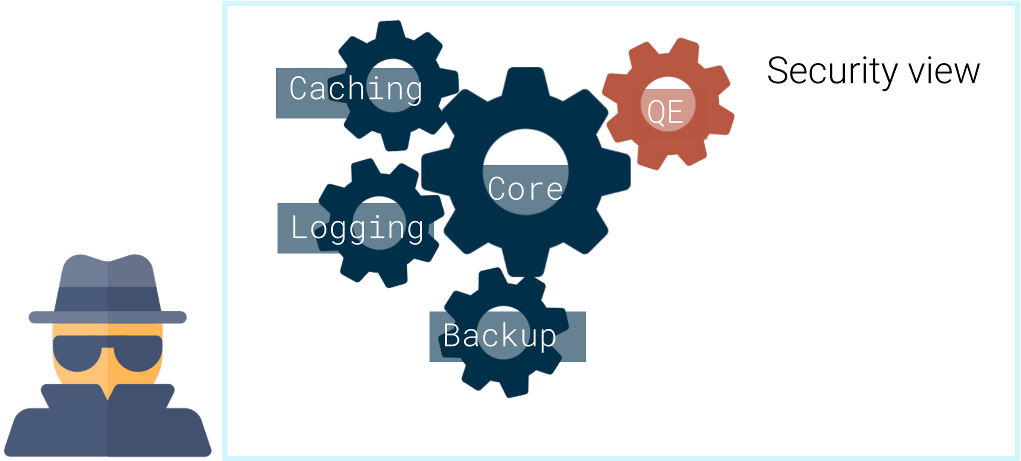
System integration of QE

- QE is built using native MongoDB operations.
- QE interacts with other MongoDB system components.
- MongoDB has adopted a cost-effective approach integrating QE: incurring minimal changes to the existing system.

System integration of QE

System integration of QE

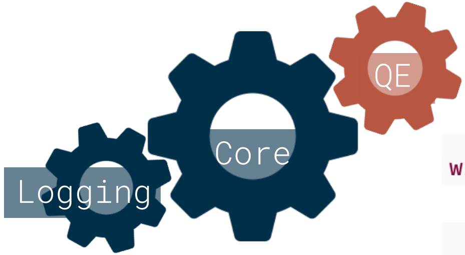
System integration of QE



Security view

Grubbs et al. [GRS17]

Logging system



OpLog: data consistency in deployment

QE command

```
write_ops::FindAndModifyCommandReply processFLEFindAndModify(
```

```
    uassert(6371800,
            "Encrypted index operations are only supported on replica sets",
            repl::ReplicationCoordinator::get(opCtx->getServiceContext())->getReplicationMode() ==
                repl::ReplicationCoordinator::modeReplSet);
```

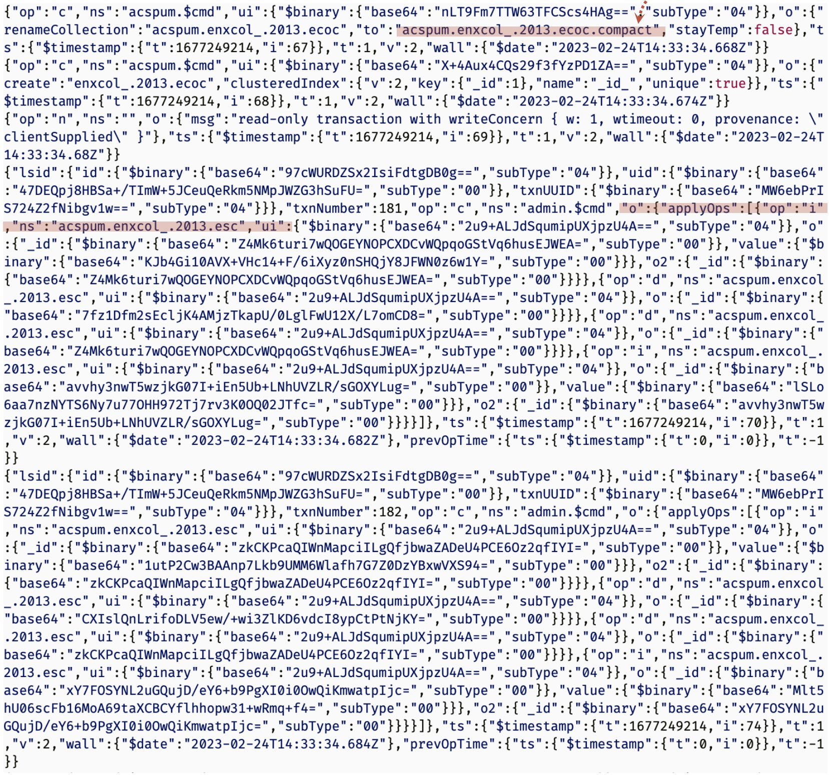
Interacts with MongoDB's OpLog for replication

“fle_crud_mongod.cpp”

What does the 🪵 look like ?

Raw OpLog

compact



```
{"op":"c","ns":"acspum.$cmd","ui":{"$binary":{"base64":"nLT9Fm7TTW63TFCScs4HAg==","subType":"04"}},"o":{"
renameCollection":"acspum.enxcol_.2013.ecoc","to":"acspum.enxcol_.2013.ecoc.compact","stayTemp":false},"t
s":{"$timestamp":{"t":1677249214,"i":67}},"t":1,"v":2,"wall":{"$date":"2023-02-24T14:33:34.668Z"}}
{"op":"c","ns":"acspum.$cmd","ui":{"$binary":{"base64":"X+4Aux4CQs29f3fYzPD1ZA==","subType":"04"}},"o":{"
create":"enxcol_.2013.ecoc","clusteredIndex":{"v":2,"key":{"_id":1},"name":"_id_","unique":true}},"ts":{"
$timestamp":{"t":1677249214,"i":68}},"t":1,"v":2,"wall":{"$date":"2023-02-24T14:33:34.674Z"}}
{"op":"n","ns":"","o":{"msg":"read-only transaction with writeConcern { w: 1, wtimeout: 0, provenance: \"
clientSupplied\" }"},"ts":{"$timestamp":{"t":1677249214,"i":69}},"t":1,"v":2,"wall":{"$date":"2023-02-24T
14:33:34.68Z"}}
{"lsid":{"id":{"$binary":{"base64":"97cWURDZSx2IsiFdtgDB0g==","subType":"04"}},"uid":{"$binary":{"base64"
:"47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=","subType":"00"}},"txnUUID":{"$binary":{"base64":"MW6ebPrI
S724Z2fNibgv1w==","subType":"04"}}},"txnNumber":181,"op":"c","ns":"admin.$cmd","o":{"applyOps":[{"op":"i"
,"ns":"acspum.enxcol_.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o"
:{"_id":{"$binary":{"base64":"Z4Mk6turi7wQOGEYNOPCXDCvWQpqoGStVq6husEJWEA=","subType":"00"}},"value":{"$b
inary":{"base64":"KJb4Gi10AVX+VHc14+F/6iXyz0nSHQjY8JFWN0z6w1Y=","subType":"00"}}},"o2":{"_id":{"$binary":
{"base64":"Z4Mk6turi7wQOGEYNOPCXDCvWQpqoGStVq6husEJWEA=","subType":"00"}}}},{"op":"d","ns":"acspum.enxcol
_.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{
"base64":"7fz1Dfm2sEcljK4AMjzTkapU/0LglFwU12X/L7omCD8=","subType":"00"}}}},{"op":"d","ns":"acspum.enxcol_
.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{"
base64":"Z4Mk6turi7wQOGEYNOPCXDCvWQpqoGStVq6husEJWEA=","subType":"00"}}}},{"op":"i","ns":"acspum.enxcol_.
2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{"b
ase64":"avvhy3nwT5wzjkG07I+iEn5Ub+LNhUVZLR/sGOXYLug=","subType":"00"}},"value":{"$binary":{"base64":"lSLo
6aa7nzNYTS6Ny7u77OHH972Tj7rv3K0OQ02JTfc=","subType":"00"}}},"o2":{"_id":{"$binary":{"base64":"avvhy3nwT5w
zjkG07I+iEn5Ub+LNhUVZLR/sGOXYLug=","subType":"00"}}}}]},"ts":{"$timestamp":{"t":1677249214,"i":70}},"t":1
,"v":2,"wall":{"$date":"2023-02-24T14:33:34.682Z"},"prevOpTime":{"ts":{"$timestamp":{"t":0,"i":0}},"t":-1
}}
{"lsid":{"id":{"$binary":{"base64":"97cWURDZSx2IsiFdtgDB0g==","subType":"04"}},"uid":{"$binary":{"base64"
:"47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=","subType":"00"}},"txnUUID":{"$binary":{"base64":"MW6ebPrI
S724Z2fNibgv1w==","subType":"04"}}},"txnNumber":182,"op":"c","ns":"admin.$cmd","o":{"applyOps":[{"op":"i"
,"ns":"acspum.enxcol_.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o"
:{"_id":{"$binary":{"base64":"zkCKPcaQIWnMapciILgQfjbwaZADeU4PCE6Oz2qfIYI=","subType":"00"}},"value":{"$b
inary":{"base64":"1utP2Cw3BAAnp7Lkb9UMM6Wlafh7G7Z0DzYBxwVXS94=","subType":"00"}}},"o2":{"_id":{"$binary":
{"base64":"zkCKPcaQIWnMapciILgQfjbwaZADeU4PCE6Oz2qfIYI=","subType":"00"}}}},{"op":"d","ns":"acspum.enxcol
_.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{
"base64":"CXIslQnLrifoDLV5ew/+wi3ZlKD6vdcI8ypCtPtNjKY=","subType":"00"}}}},{"op":"d","ns":"acspum.enxcol_
.2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{"
base64":"zkCKPcaQIWnMapciILgQfjbwaZADeU4PCE6Oz2qfIYI=","subType":"00"}}}},{"op":"i","ns":"acspum.enxcol_.
2013.esc","ui":{"$binary":{"base64":"2u9+ALJdSqumipUXjpzU4A==","subType":"04"}},"o":{"_id":{"$binary":{"b
ase64":"xY7FOSYNL2uGQujD/eY6+b9PgXI0i0OwQiKmwatpIjc=","subType":"00"}},"value":{"$binary":{"base64":"Mlt5
hU06scFb16MoA69taXCBCYflhhopw31+wRmq+f4=","subType":"00"}}},"o2":{"_id":{"$binary":{"base64":"xY7FOSYNL2u
GQujD/eY6+b9PgXI0i0OwQiKmwatpIjc=","subType":"00"}}}}]},"ts":{"$timestamp":{"t":1677249214,"i":74}},"t":1
,"v":2,"wall":{"$date":"2023-02-24T14:33:34.684Z"},"prevOpTime":{"ts":{"$timestamp":{"t":0,"i":0}},"t":-1
}}
```

insert

What does the 🪵 say?

leakage :(

What does the 🪵 say?

How we extract the leakage from OpLog

Insert(doc$_1$)

Insert(doc$_2$)

Insert(doc$_3$)

Compact()

{ "Delete": "ESC"
"_id": N_1^{esc},
"txnid": "211" }

{ "Delete": "ESC"
"_id": N_2^{esc},
"txnid": "211" }

{ "Insert": "ESC"
"_id": N_{null}^{esc},
"txnid": "211" }

(N_1^{esc}, N_2^{esc})

{ "Delete": "ESC"
"_id": C_1^{esc},
"txnid": "212" }

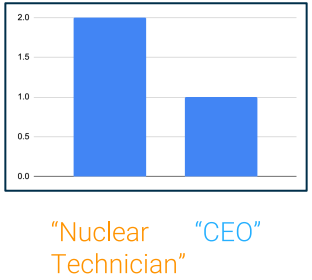
{ "Insert": "ESC"
"_id": C_{null}^{esc},
"txnid": "212" }

(C_1^{esc})

Inference attack

- ○ The leakage we have extracted corresponds to frequency and correlation leakage.
- ○ Auxiliary information
- ○ New inference attack techniques based on Gui et al. [GPP21]

$edoc_1$, $edoc_2$ contains the same field value
$edoc_3$ has a different one



```
{
  "Name": "Homer Simpson",
  "Job": "Nuclear Technician",
  "Class of Jobs": "Technician"
  "Address": "742 Evergreen Terrace"
}
```

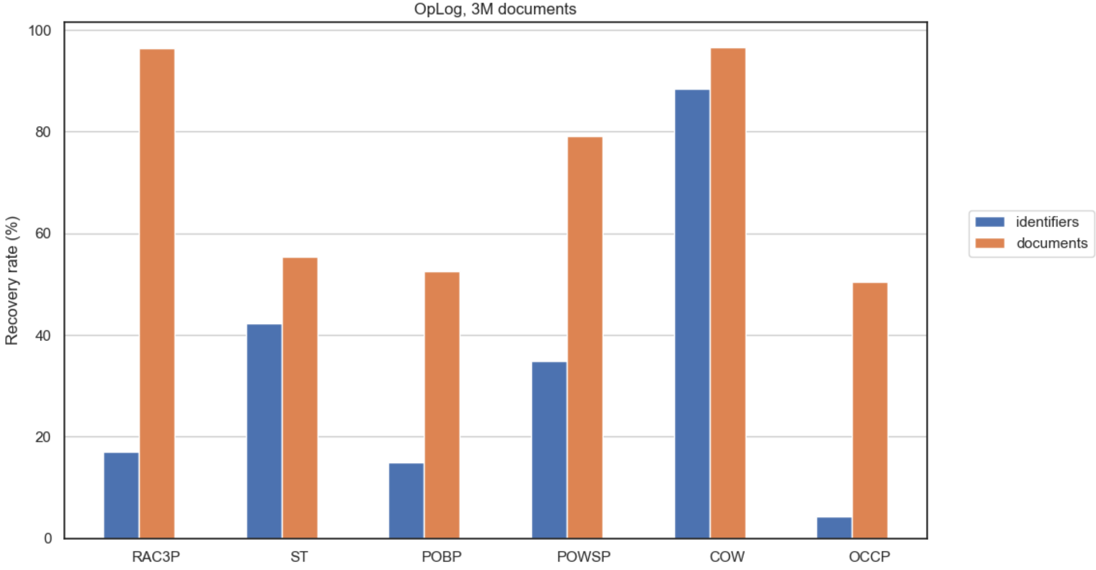
Experimental validation

- Auxiliary information: ACS (American Community Survey micro-data) 2012
- Recovery target: ACS 2013
- Simulated leakage
- Artifact available!

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

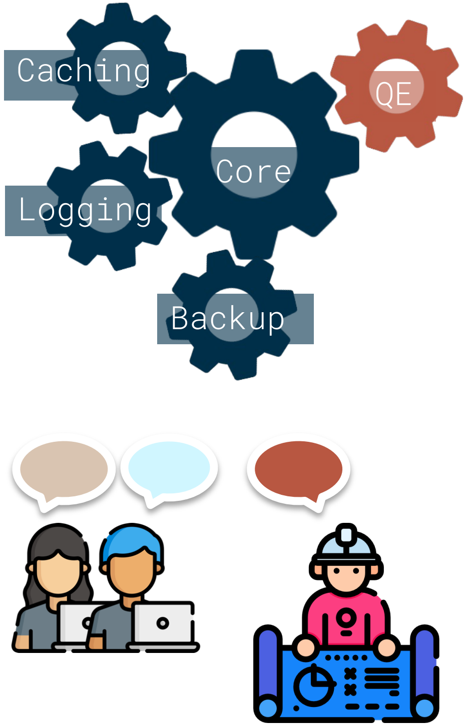


How and when to get this “leaky” OpLog?

- OpLog is stored on the server’s file system.
- After **Compact()**

Takeaways

System integration of searchable encryption schemes is challenging!



Paper & artifact

[GPT23] Zichen Gui, Kenneth G. Paterson, and Tianxin Tang

(tianxin.tang@inf.ethz.ch)

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

References

- [GRS17] Paul Grubbs, Thomas Ristenpart, Vitaly Shmatikov, Why Your Encrypted Database Is Not Secure
- [GPP21] Zichen Gui, Kenneth G Paterson, Sikhar Patranabis, Rethinking Searchable Symmetric Encryption