Encrypted Search Party

...

Fiona McCawley

Queryable Application Level Encryption

```
=# select * from users where dob > '1990-01-01';
 2  |  marc@effertz.name  |  1993-03-11
(1 row)
```

id	email	dob
1	cathey.blick@frami.com	1982-01-16
2	marc@effertz.name	1993-03-11
3	regenia@bosco-ullrich.co	1965-07-22

```
=# select * from users where dob > '1990-01-01';

( 0 rows )
```

id	email	dob
1	"AKNMFsXalXRAYh+kksGHHNR80vyl5iNp6fTAWRgOotU="	"dAWSvfRuRdoE6SRuq6IiLQ=="
2	"eHSPokL30i8JjXo8tpWv6ZE/UIgRTIJ7qEZo/hzm2D0="	"GXy/mGe8npLtjNEUtg0xlw=="
3	"hceJDprnKy+bqhNaC4w5b8MfLosirsNdJ9D8ATJ/tpg="	"dxSIpC1n7Rx2d9s4+GPdZg=="

Queryable Application Level Encryption

Encryption

Plaintext

Ciphertext

“Really sensitive data”

“lsMd1lOqTL1eK9asDJD/bA==”

Non Deterministic or Deterministic

Non Deterministic

encrypt("test@email.com") => "MWusCJjq6b5+iLxZKWoh0g=="

encrypt("test@email.com") => "xn74KQ/CdJPzRHs+Bl4DYg=="

 + IV (initialisation vector)

Deterministic

encrypt(“test@email.com”) => “MWusCJjq6b5+iLxZKWoh0g==”

encrypt(“test@email.com”) => “MWusCJjq6b5+iLxZKWoh0g==”

encrypt(“test@email.com”) => “MWusCJjq6b5+iLxZKWoh0g==”

Queryable Application Level Encryption

Application Level Encryption

Encryption at rest

Encryption in transit

Client

Cloud provider
(eg RDS instance)

Active Record Encryption

This guide covers encrypting your database information using Active Record.

After reading this guide, you will know:

- ✓ **How to set up database encryption with Active Record.**
- ✓ **How to migrate unencrypted data**
- ✓ **How to make different encryption schemes coexist**
- ✓ **How to use the API**
- ✓ **How to configure the library and how to extend it**

Chapters

Active Record supports application-level encryption. It works by declaring which attributes should be encrypted and seamlessly encrypting and decrypting them when necessary. The encryption layer sits between the database and the application. The application will access unencrypted data, but the database will store it encrypted.

1 Why Encrypt Data at the Application Level?

```
→  ActiveRecordEncryption git:(main) ✗ bin/rails db:encryption:init
Add this entry to the credentials of the target environment:

active_record_encryption:
  primary_key: Atu9ocMcaeUdxPh6Y9AUnCm7C4Gr8jNd
  deterministic_key: cFjxy9KvmCFptiJF7ICiHy9orNeUueV6
  key_derivation_salt: pzgf7d2FW8j01n1Sfhod1Nvy8do1RfQt

→  ActiveRecordEncryption git:(main) ✗ EDITOR='code --wait' rails credentials:edit
File encrypted and saved.
```

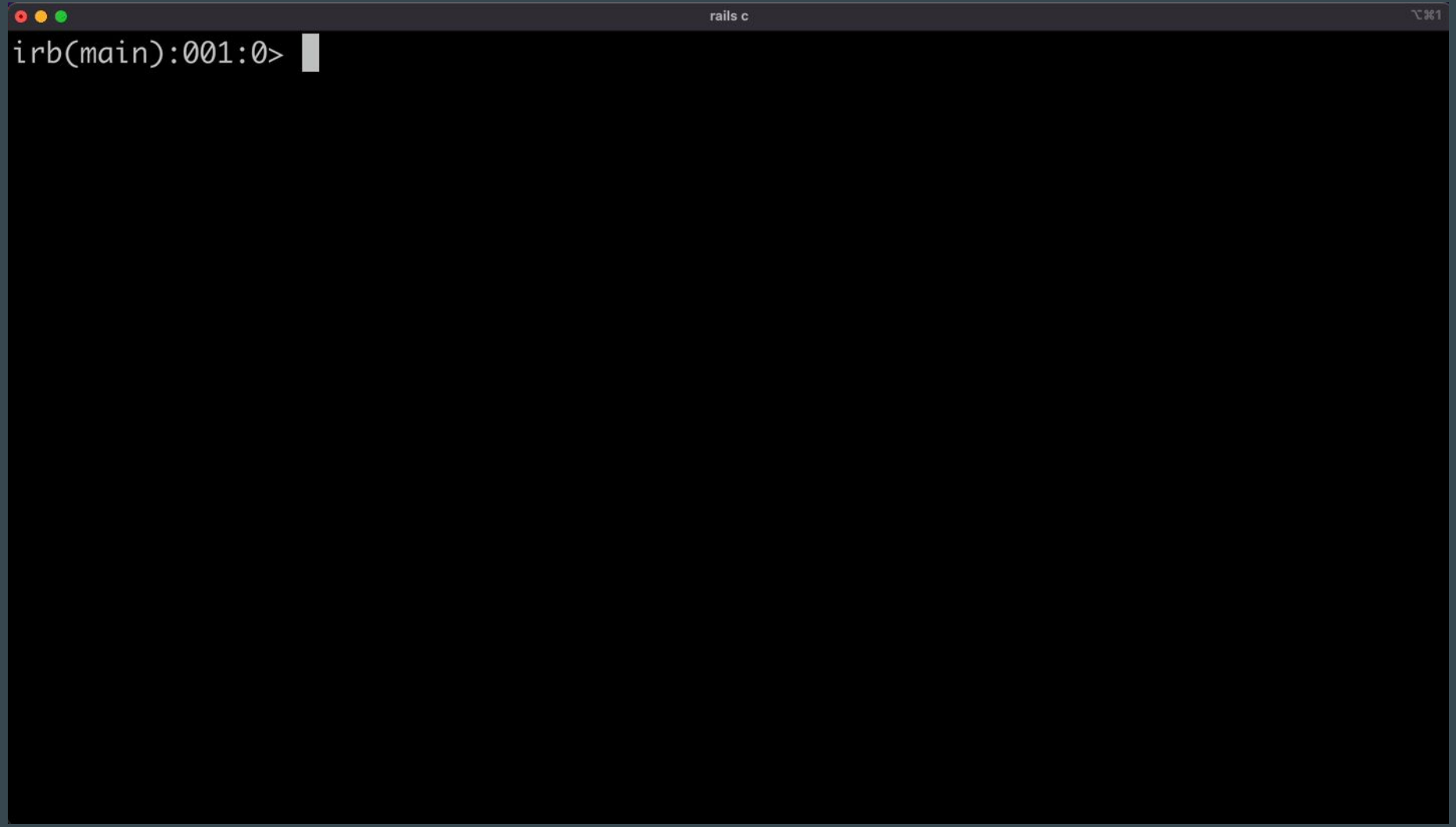
Schema

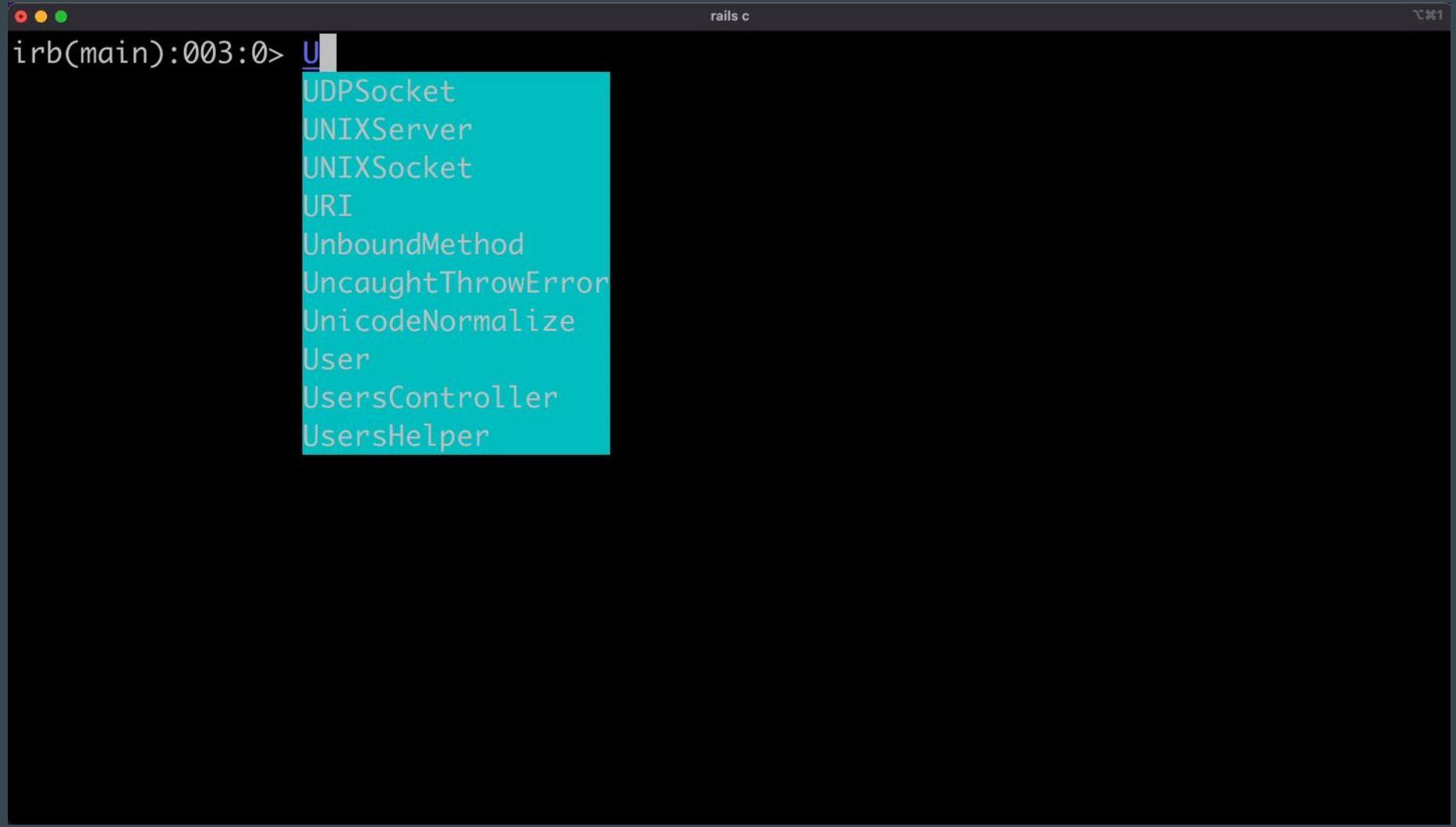
```ruby
ActiveRecord::Schema[7.0].define(version: 2023_02_02_043157) do
  # These are extensions that must be enabled in order to support this database
  enable_extension "plpgsql"

  create_table "users", force: :cascade do |t|
    t.string "email"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end
```

Model

```ruby
class User < ApplicationRecord
  encrypts :email
end
```

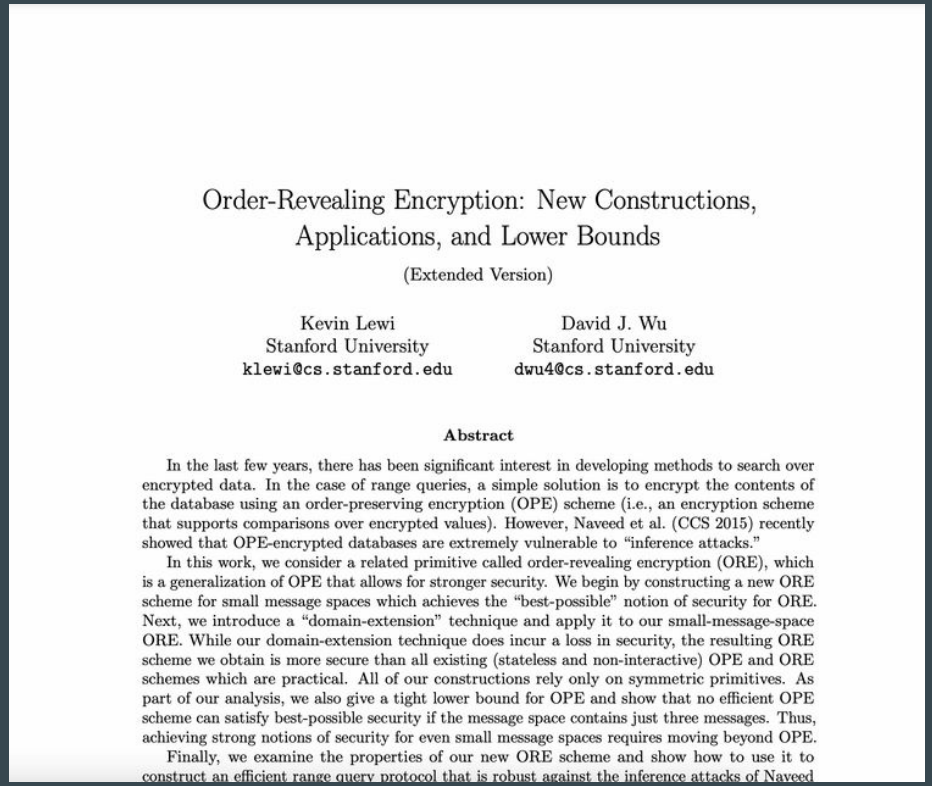

rails c
⌥⌘1
irb(main):001:0>


rails c
⌥⌘1
irb(main):003:0> U
UDPSocket
UNIXServer
UNIXSocket
URI
UnboundMethod
UncaughtThrowError
UnicodeNormalize
User
UsersController
UsersHelper

```
irb(main):005:0> User.create!(email: "fi@test.com")
```

Queryable Application Level Encryption

Order Revealing Encryption (ORE)



Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds

(Extended Version)

Kevin Lewi
Stanford University
klewi@cs.stanford.edu

David J. Wu
Stanford University
dwu4@cs.stanford.edu

Abstract

In the last few years, there has been significant interest in developing methods to search over encrypted data. In the case of range queries, a simple solution is to encrypt the contents of the database using an order-preserving encryption (OPE) scheme (i.e., an encryption scheme that supports comparisons over encrypted values). However, Naveed et al. (CCS 2015) recently showed that OPE-encrypted databases are extremely vulnerable to "inference attacks."

In this work, we consider a related primitive called order-revealing encryption (ORE), which is a generalization of OPE that allows for stronger security. We begin by constructing a new ORE scheme for small message spaces which achieves the "best-possible" notion of security for ORE. Next, we introduce a "domain-extension" technique and apply it to our small-message-space ORE. While our domain-extension technique does incur a loss in security, the resulting ORE scheme we obtain is more secure than all existing (stateless and non-interactive) OPE and ORE schemes which are practical. All of our constructions rely only on symmetric primitives. As part of our analysis, we also give a tight lower bound for OPE and show that no efficient OPE scheme can satisfy best-possible security if the message space contains just three messages. Thus, achieving strong notions of security for even small message spaces requires moving beyond OPE.

Finally, we examine the properties of our new ORE scheme and show how to use it to construct an efficient range query protocol that is robust against the inference attacks of Naveed

ORE Ciphertext

LEFT CT < = > RIGHT CT

The life of a plaintext in ORE land

ORE land

Domain

0 1 2 3

ORE land

97 98 99 100

ORE land

0 1 2 3

Domain

0 1 2 3

<=>

 = 0

Offset

Left CT

0

Comparison results

Right CT

[0 , 1 , 1 , 1]

Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =1	1	[-1, 0, 1, 1]
 =2	2	[-1, -1, 0, 1]
 =3	3	[-1, -1, -1, 0]

Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =1	1	[-1, 0, 1, 1]
 =2	2	[-1, -1, 0, 1]
 =3	3	[-1, -1, -1, 0]

Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =2	2	[-1, -1, 0, 1]

Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =2	2	[-1, -1, 0, 1]
 =2	2	[-1, -1, 0, 1]

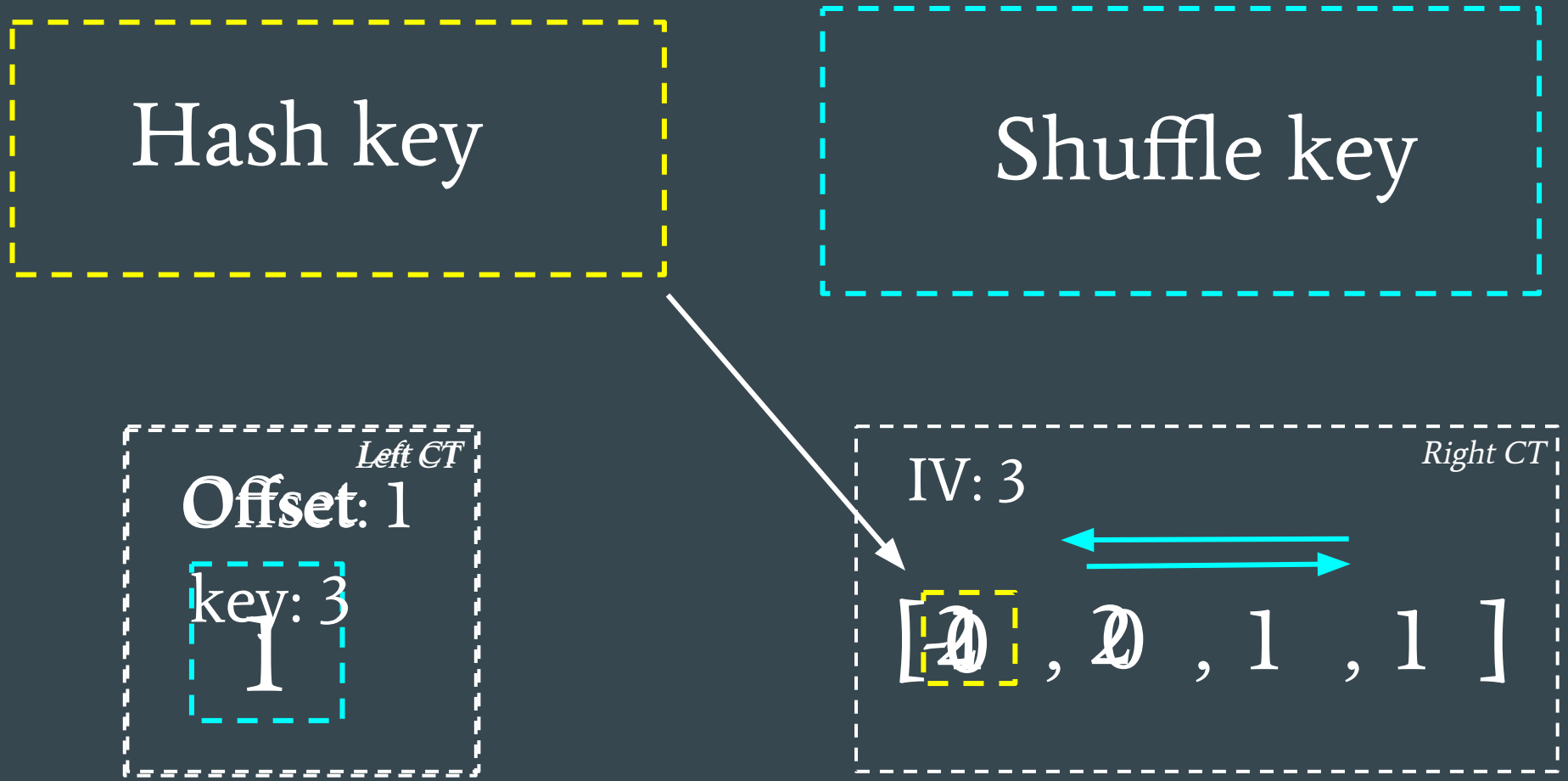
Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =1	1	[-1, 0, 1, 1]
 =2	2	[-1, -1, 0, 1]
 =3	3	[-1, -1, -1, 0]

PRF key

(Pseudo Random Function)

PRP key

(Pseudo Random Permutation)


Hash key
Shuffle key
Left CT
Offset: 1
key: 3
]
IV: 3
Right CT
[0 , 0 , 1 , 1]

Plaintext	Left CT	Right CT
 =0	0	[0, 1, 1, 1]
 =1	1	[-1, 0, 1, 1]
 =2	2	[-1, -1, 0, 1]
 =3	3	[-1, -1, -1, 0]

Plaintext	Left CT		Right CT	
	Key	Offset	Encryptions	IV
 =0	0	2	[2, 1, 1, 3]	1
 =1	3	3	[1, 2, -3, 1]	2
 =2	1	1	[3, 1, -1, -4]	0
 =3	2	0	[1, -3, -4, -1]	3

Left CT
Key: 0
Offset: 2

1	[2, 1, 1, 3]
2	[1, 2, -3, 1]
0	[3, 1, -1, -4]
3	[1, -3, -4, 1]

Client

Database

```
=# select * from users where dob > '1990-01-01';
 2  |  marc@effertz.name  |  1993-03-11
(1 row)
```

id	email	dob
1	cathey.blick@frami.com	1982-01-16
2	marc@effertz.name	1993-03-11
3	regenia@bosco-ullrich.co	1965-07-22

```
=# select * from users where dob > '1990-01-01';

( 0 rows )
```

id	email	dob
1	"AKNMFsXalXRAYh+kksGHHNR80vyl5iNp6fTAWRgOotU="	"dAWSvfRuRdoE6SRuq6IiLQ=="
2	"eHSPokL30i8JjXo8tpWv6ZE/UIgRTIJ7qEZo/hzm2D0="	"GXy/mGe8npLtjNEUtg0xlw=="
3	"hceJDprnKy+bqhNaC4w5b8MfLosirsNdJ9D8ATJ/tpg="	"dxSIpC1n7Rx2d9s4+GPdZg=="

```
=# select * from users where dob_ore > {offset: 2, key: 3};

 2  |  marc@effertz.name  |  1993-03-11
( 1 row )
```

id	email	dob	dob_ore
1	“AKNMF80vyl5iNp6fTAWRgOotU=”	“dAWSvfRuRdoE6SRuq6IiLQ==”	*Right CT* iv: 1 [2, 1, 1, 3]
2	“eHSPokL30RTIJ7qEZo/hzm2D0=”	“GXy/mGe8npLtjNEUtg0xlw==”	*Right CT* iv: 1 [2, 1, 1, 3]
3	“hceJDprnKirsNdJ9D8ATJ/tpg=”	“dxSIpC1n7Rx2d9s4+GPdZg==”	*Right CT* iv: 1 [2, 1, 1, 3]

gem install toy-ore

@saucerlike

github.com/fimac