Encrypted databases

Tom Ristenpart
CS 6431

Outsourced storage settings

Client wants to store data up on Dropbox

- High availability, synch across devices
- Server includes much value-add functionality

  Keyword search (find all files with “Tom” in text)

  Deduplication   (if two files same, store only one copy)

  Thumbnail generation for images

Outsourced storage settings

Salesforce stores customer records for companies

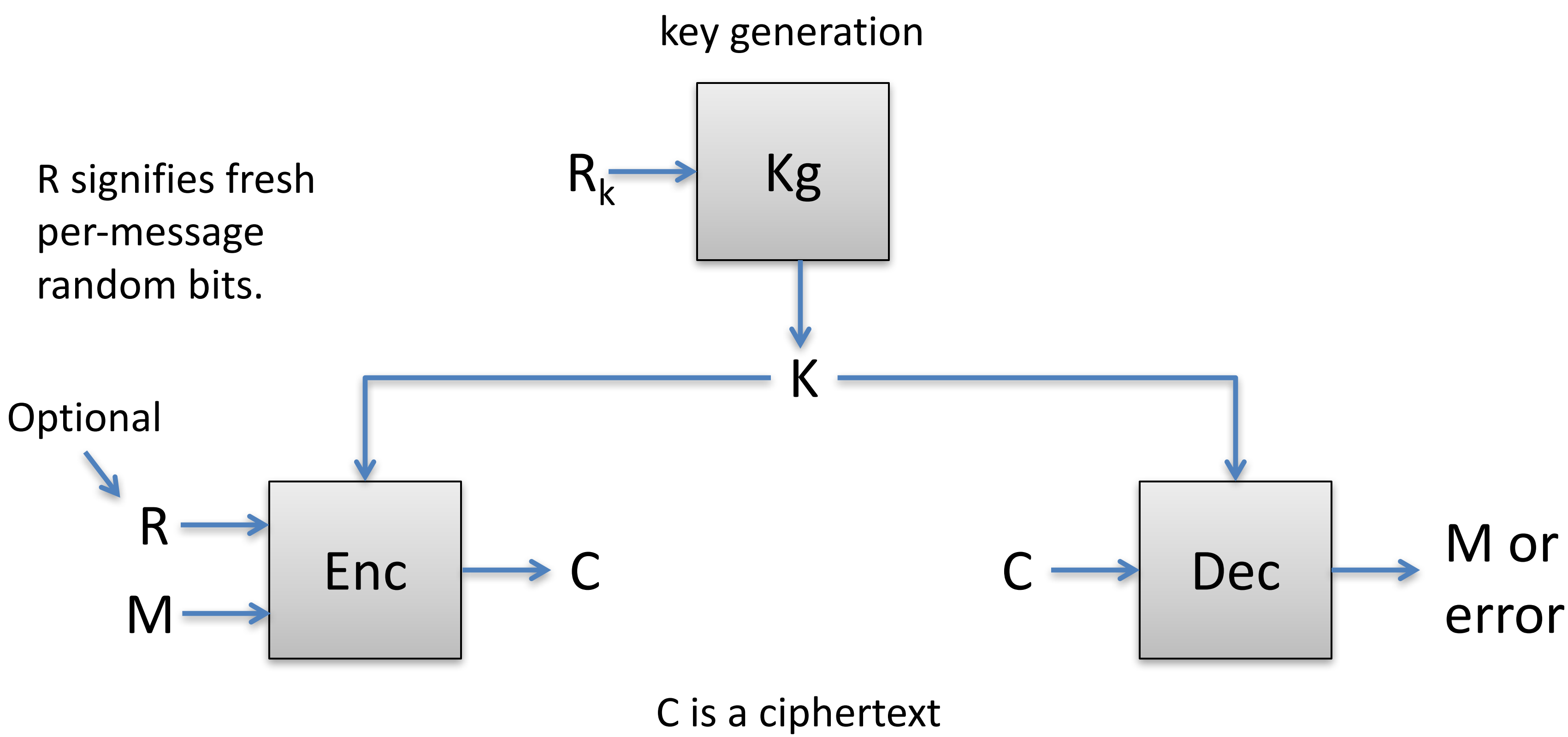
Name:	Clarisse	Comments:	Works in NYC office
Age:	22	Gender:	Female
Salary:	100,000	SSN:	555-31-4325

Much value-add server-side functionality

- Keyword search (find all records with name "Clarisse")
- Range queries (all records with 20 <= Age <= 30)
- Sorted lists (return records ordered by salary)

What security threats would one worry about?

Symmetric encryption



Correctness: Dec(K , Enc(K,M,R)) = M with probability 1 over randomness used

IND-CPA security

[Bellare, Desai, Jokippii, Rogaway 97]

Adversary gets to submit messages to oracle

IND-CPA challenge game

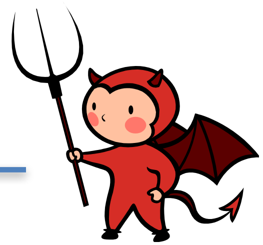
M0,M1

C’

M0’,M1’

C’

⋮



b'

<u>Encrypt(M0,M1)</u>
C <- Enc(K,Mb)
Ret C

Adversary outputs guess b’ of b
Wins if b’ = b

b is a uniformly sampled bit
and K is uniformly sampled key
Both hidden from adversary

Adversary must query with |M0| = |M1|

Security goal: Enc(K,M) doesn’t even leak single bit about M

Outsourced storage settings

Salesforce stores customer records for companies

Name:
Age:
Salary:

Comments:
Gender:
SSN:

What storage service provider functionalities broken?

- Field search (Find all records with Name = Alice)
- Keyword search
- Range queries (Find all people who make between 90k and 120k)
- Format problems (Age must be integer between 0 and 130)
- …

Instead:
Property-Revealing Encryption (PRE)

Salesforce stores customer records for companies

Name:
Age:
Salary:

Comments:
Gender:
SSN:

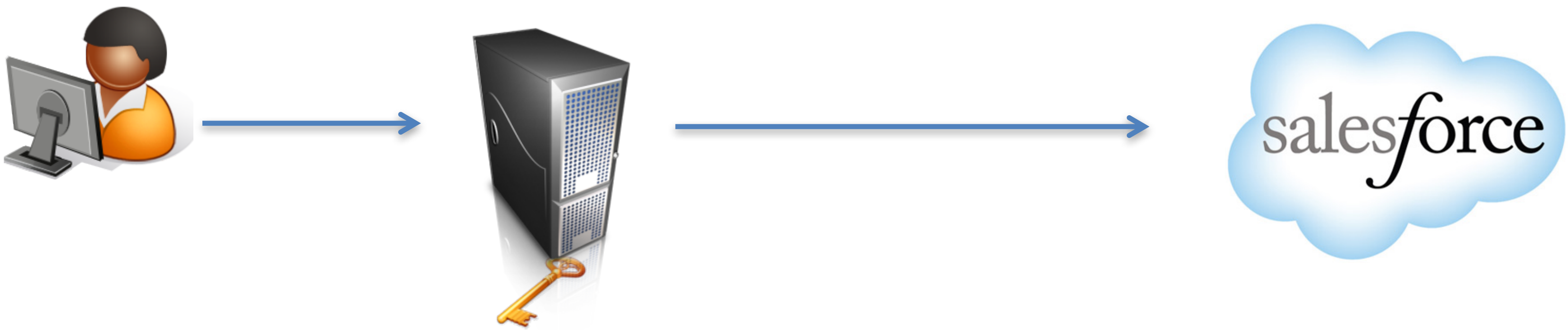
Encrypt data with special PRE schemes that leak just enough about plaintexts to perform some operations

Sometimes advertised as “computing over encrypted data” – metaphor is flawed

Property-revealing encryption

Problem	Crypto primitive	Description	Literature
Keyword search	Searchable symmetric encryption	Perform search over ciphertexts given encrypted search token	[Dawn, Song, Wagner 2000] [Curtmola et al. 2006] …
Equality search	Deterministic encryption	X = Y implies DET(X) = DET(Y)	[Rogaway Shrimpton 06]
Range queries	Order-preserving encryption	X > Y implies OPE(X) > OPE(Y)	[Boldyreva et al. 2009]
Range queries	Order-revealing encryption	X > Y implies Cmp(ORE(X),ORE(Y)) = 1	[Boldyreva et al. 11], [Boneh et al. 15]
Deduplication	Message-locked encryption (Convergent encryption)	Different user's encryptions of same plaintext give same ciphertext	[Douceur et al. 2002] [Bellare et al. 2013]
Format restrictions	Format-preserving encryption	Ciphertext has same format as plaintext	[Bellare et al. 2009]

Lots of industry activity



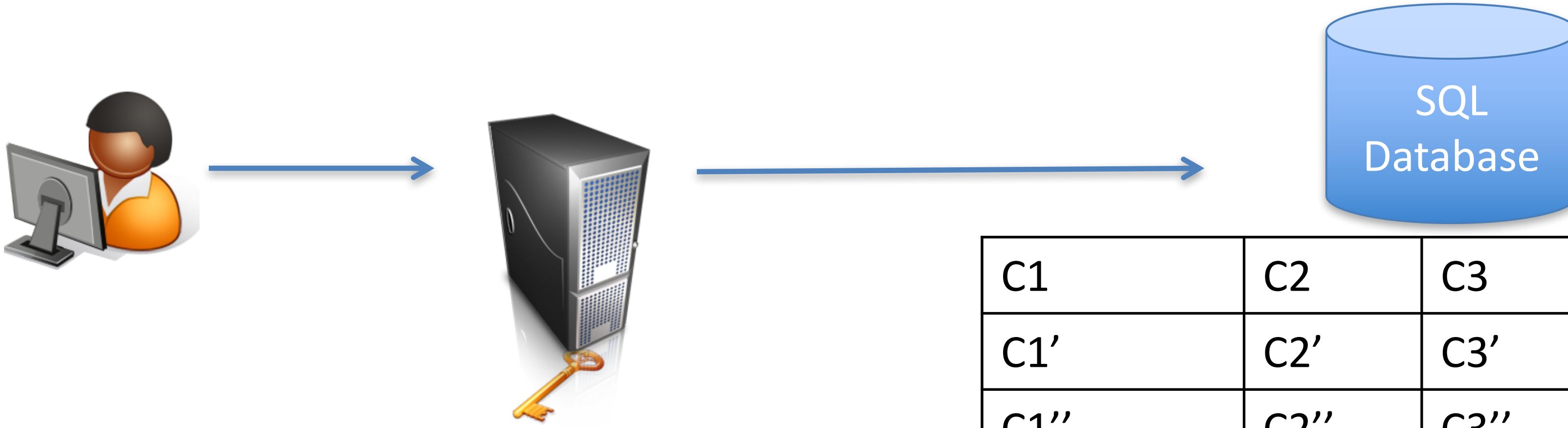
CipherCloud™
Trust in the Cloud

PerspecSys
Making the Public Cloud Private

skyhigh

Cloud access security brokers (CASB)

PREs in databases



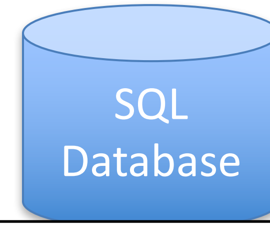
C1	C2	C3
C1’	C2’	C3’
C1’’	C2’’	C3’’

Encrypt by column

Name: C1 = DET(K1, “Alice”)
Age: C2 = OPE(K2, “22”)
Salary: C3 = OPE(K3, “100,000”)

PREs in databases

DET(K1, “Alice”)

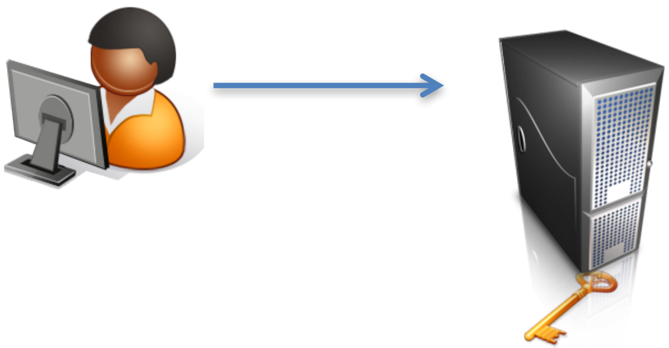


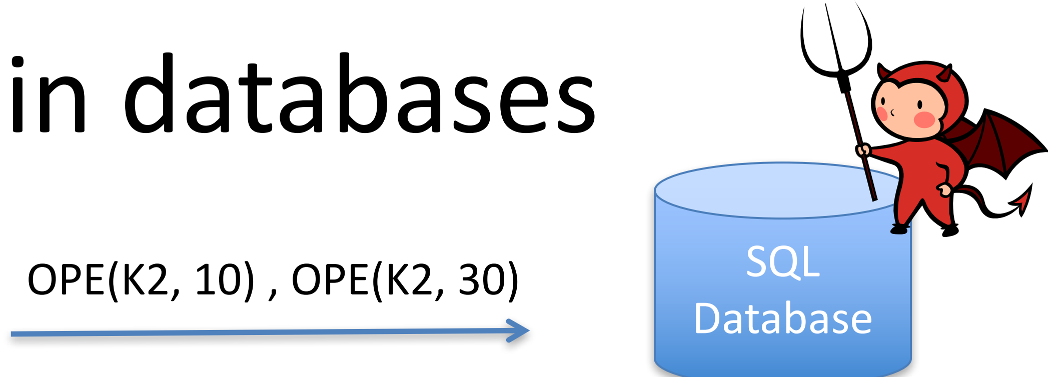
DET(K1, “Alice”)	OPE(K2, “22”)	C3
DET(K1, “Bob”)	OPE(K2, “29”)	C3’
DET(K1, “Alice”)	OPE(K2, “31”)	C3’’

Perform *equality search* by querying DET encryption of keyword

Property revelation: $\mathrm{DET}(K1,X) = \mathrm{DET}(K1,Y) \Leftrightarrow X = Y$

PREs in databases





DET(K1, “Alice”)	OPE(K2, “22”)	C3
DET(K1, “Bob”)	OPE(K2, “29”)	C3’
DET(K1, “Alice”)	OPE(K2, “31”)	C3’’

Perform *equality search* by querying DET encryption of keyword

Property revelation: DET(K1,X) = DET(K1,Y) $\Leftrightarrow$ X = Y

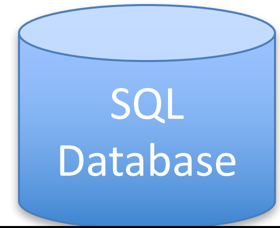
Perform *range search* by querying OPE encryption of end points

Property revelation: OPE(K2,X) < OPE(K2,Y) $\Leftrightarrow$ X < Y

What is revealed to adversarial server?

CryptDB

[Popa et al. 2011]



Name: Alice
Age: 22
Salary: 100,000

C1	C2	C3
C1’	C2’	C3’
C1’’	C2’’	C3’’

Encrypted “onions”

Name: C1 = Enc(K1’, DET(K1, “Alice”))
Age: C2 = Enc(K2’, OPE(K2, “22”))
Salary: C3 = Enc(K3’, OPE(K3, “100,000”))

At this point nothing is leaked to Database:

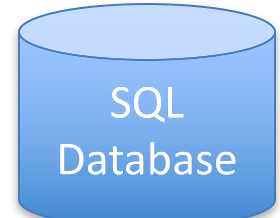
- Full symmetric encryption security given by Enc
- Repeats not visible, ordering not visible

CryptDB

[Ada Popa et al. 2011]

Name: Alice
Age: 22
Salary: 100,000

K1’ , DET(K1, “Alice”)



C1	C2	C3
C1’	C2’	C3’
C1’’	C2’’	C3’’

Perform equality search by:

- Send outer encryption key for searched-on-column
- Server decrypts that column
- Use DET to find rows corresponding to Alice as before

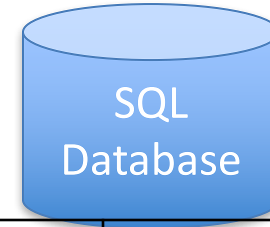
At this point equality patterns of first column leaked

CryptDB

[Ada Popa et al. 2011]

Name: Alice
Age: 22
Salary: 100,000

K2’ , OPE(K2, 10) , OPE(K2, 30)



DET(K1, “Alice”)	C2	C3
DET(K2, “Bob”)	C2’	C3’
DET(K3, “Alice”)	C2’’	C3’’

Perform range search by:

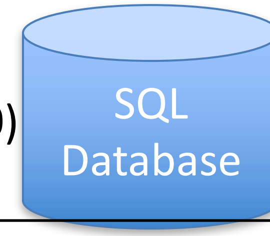
- Send outer encryption key for searched-on-column
- Server decrypts that column
- Use OPE range query mechanism from before

CryptDB

[Ada Popa et al. 2011]

Name: Alice
Age: 22
Salary: 100,000

K2’ , OPEnc(K2, 10) , OPEnc(K2, 30)



DET(K1, “Alice”)	OPE(K2, “22”)	C3
DET(K1, “Bob”)	OPE(K2, “29”)	C3’
DET(K1, “Alice”)	OPE(K2, “31”)	C3’’

Perform range search by:

- Send outer encryption key for searched-on-column
- Server decrypts that column
- Use order preservation of OPEnc to find rows in range

At this point equality patterns of first and second column leaked
Ordering information on second column leaked

Standard PRE use and CryptDB

- CryptDB: lots more engineering details
  - Other types of queries, other kinds of onions
  - How to decompose SQL queries into appropriate levels of encrypted data
  - Showed that some columns need not be decrypted in some workloads
  - Does not work with legacy storage systems (e.g., SalesForce)
- "Standard" PRE use (no onions) easier to deploy. This is what is used in practice currently

How damaging is the leakage?

What is known about security?

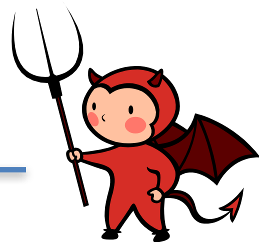
Cryptographic theory offers proofs of security for OPE, DET

Preservation means can’t achieve IND-CPA definitions

Solution:
Weaken security notions until one can show achievability

IND-CPA security for DET

Adversary gets to submit messages to oracle



b'

M0,M1

C’

M0’,M1’

C’

⋮

Challenge game

Encrypt(M0,M1)
C <- Enc(K,Mb)
Ret C

Adversary outputs guess b’ of b
Wins if b’ = b

b is a uniformly sampled bit
and K is uniformly sampled key
Both hidden from adversary

Adversary must query with |M0| = |M1|

Security goal: Enc(K,M) doesn’t leak anything about plaintexts

IND-CPA security for DET

Adversary gets to submit messages to oracle

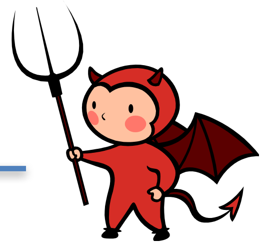
M0,M1

C'

M0',M1'

C'

⋮



b'

Challenge game

<u>Encrypt(M0,M1)</u>

C <- DET(K,Mb)

Ret C

Adversary outputs guess b' of b

Wins if b' = b

b is a uniformly sampled bit and K is uniformly sampled key

Both hidden from adversary

Adversary must query with |M0| = |M1|

and Mb,Mb', … all distinct for b = 0,1

Security goal: Enc(K,M) doesn't leak anything *but* plaintext equality

IND-OCPA security for OPE

[Boldyreva, Chenette, Lewi, O'Neill 09]

Adversary gets to submit messages to oracle

IND-OCPA challenge game

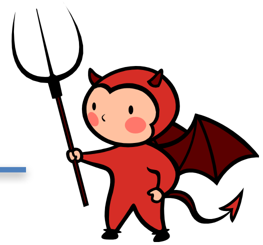
M0,M1

C'

M0',M1'

C'

⋮



b'

Encrypt(M0,M1)
C <- OPE(K,Mb)
Ret C

Adversary outputs guess b' of b
Wins if b' = b

b is a uniformly sampled bit
and K is uniformly sampled key
Both hidden from adversary

Adversary must query with |M0| = |M1|
and Mb,Mb', … all distinct for b = 0,1
and M0 < M0' iff M1 < M1' for all pairs of queries

Security goal: Enc(K,M) doesn't leak anything *but* plaintext equality and order relationships

Achieving IND-OCPA hard

Scheme must have mutable ciphertexts (even w/ stateful clients & interactivity), otherwise require exponentially large ciphertexts

[Popa et al. 2011]

This degrades the deployability of IND-OCPA schemes.
They aren't currently used

Instead: weaker security goals such as indistinguishability from a random order-preserving function

[BCLO 09] give clever scheme provably secure under this definition
[BCO 11] show that ciphertexts leak at least ***approximately half the plaintext***

Leakage-abuse attacks (LAAs)

Question: what can attackers learn from leakage of PRE schemes?

Searchable encryption	[Islam, Kuzu, and Kantarcioglu 2012] [Cash, Grubbs, Perry, Ristenpart 2015] [Zhang, Katz, Papamanthou 2016] [Wright, Kamara 2016]
DET, OPE, ORE	[Naveed, Kamara, Wright 2015] [Durak, DuBuisson, Cash 2016] [Grubbs et al. 2016]

[Naveed, Kamara, Wright 2015]

NKW paper

Ciphertext-only inference attacks:
- get access to database
- Auxiliary information about plaintext distribution

DET(K1, “Alice”)	OPE(K2, “22”)
DET(K1, “Bob”)	OPE(K2, “29”)
DET(K1, “Alice”)	OPE(K2, “31”)

Sorting attack against “dense” OPE columns

OPE(K2, “3”)
OPE(K2, “1”)
OPE(K2, “2”)

OPE(K2, “1”)	1
OPE(K2, “2”)	2
OPE(K2, “3”)	3

Already reveals everything for some column types (e.g., binary attributes)

Frequency analysis

DET(K1, “Alice”)
DET(K1, “Bob”)
DET(K1, “Alice”)
DET(K1, “Alice”)
DET(K1, “Alice”)
DET(K1, “Bob”)

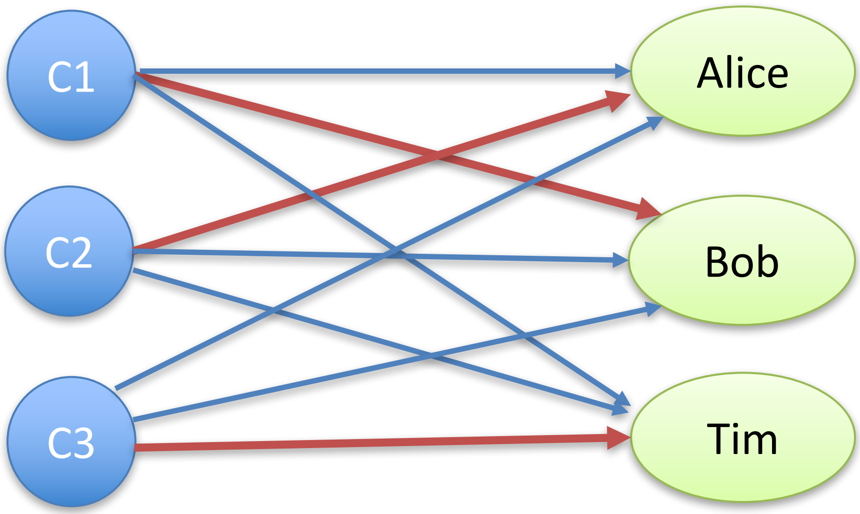
Auxiliary data is plaintext dataset distributed similarly to target. Histogram:

Alice 15
Bob 3

Match most frequent ciphertext to most frequent auxiliary value
Match next most frequent to next most frequent, etc.

Technique is thousands of years old; applied most often to substitution ciphers
But: DET is just a substitution cipher over large alphabet

Graph viewpoint: DET



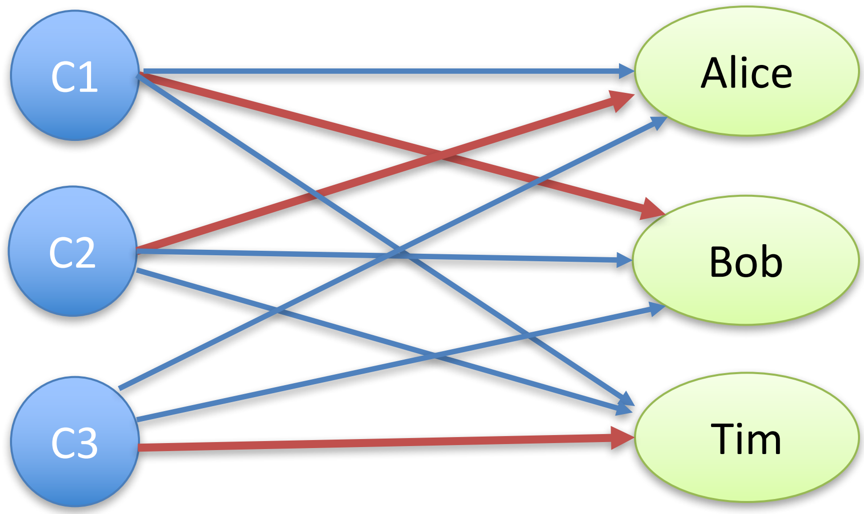
Bipartate graph problem: find matching of ciphertexts, names that minimizes total weight

$$\text{Weight}(C,\text{Name}) = |\text{Freq}(C) - \text{Freq}(\text{Name})|$$

Equivalent to frequency analysis

Graph viewpoint: OPE

[Grubbs et al. 2016]



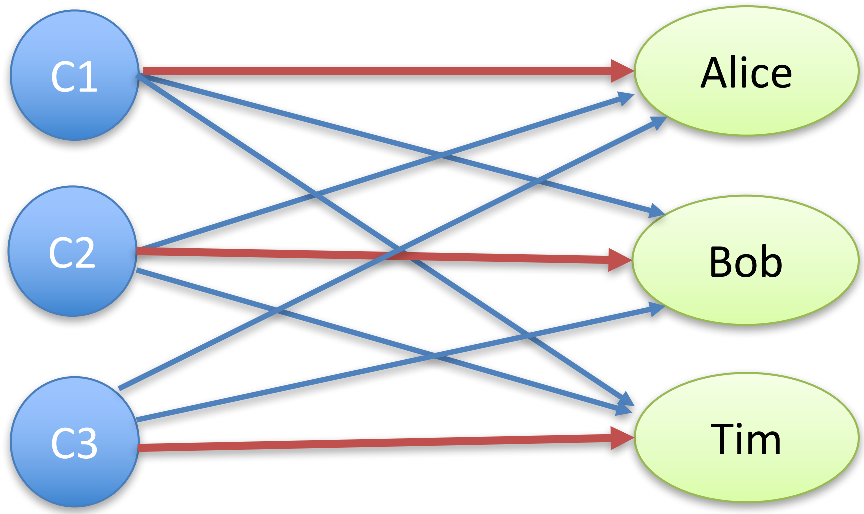
Bipartate graph problem: find *non-crossing* matching of ciphertexts, names that minimizes total weight

Weight(C,Name) = |Freq(C) – Freq(Name)|

This takes into account ordering constraints

Graph viewpoint: OPE

[Grubbs et al. 2016]



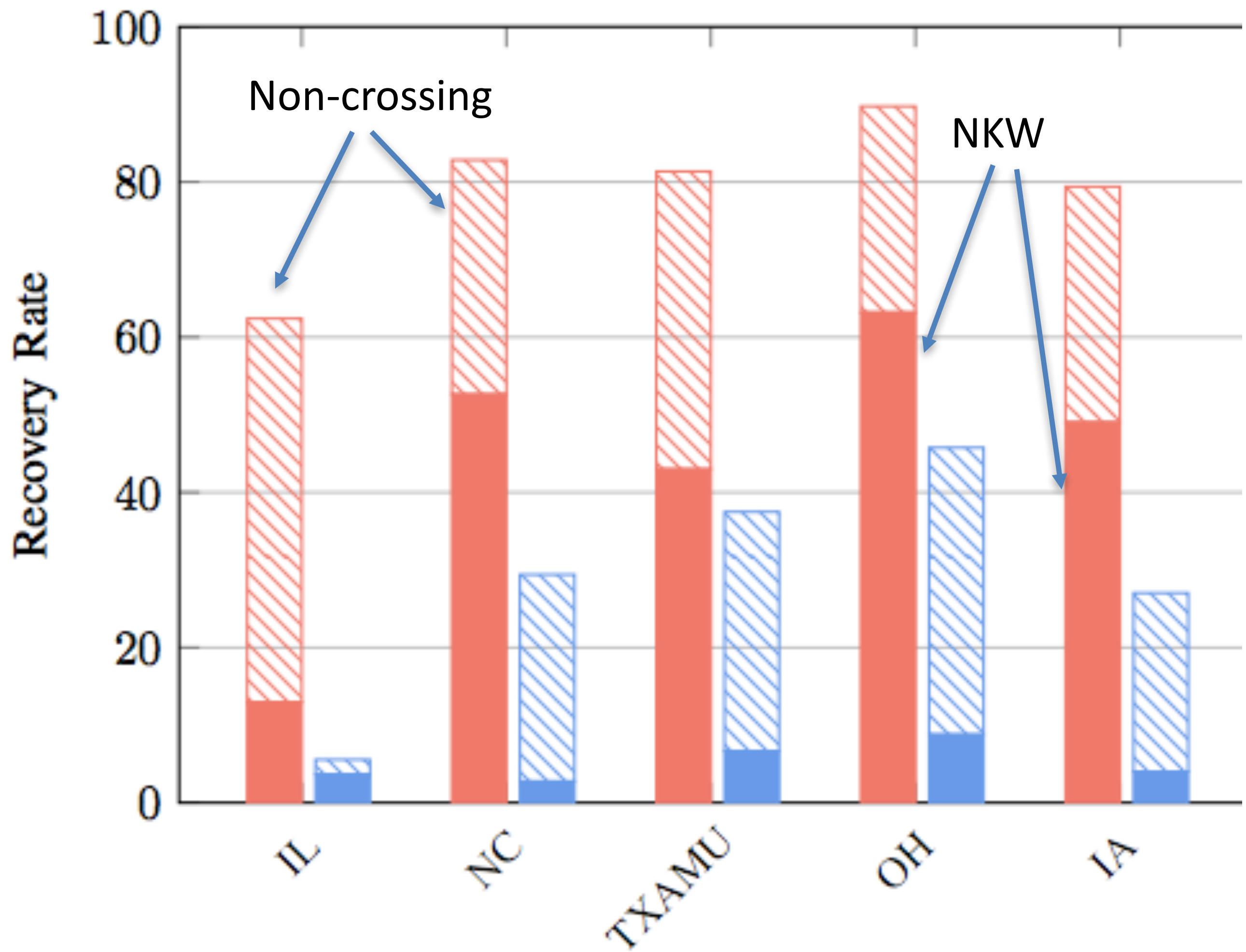
Bipartate graph problem: find *non-crossing* matching of ciphertexts, names that minimizes total weight

$$\text{Weight}(C,\text{Name}) = |\text{Freq}(C) - \text{Freq}(\text{Name})|$$

This takes into account ordering constraints

Better version called multinomial w/ more principled weighting (stay tuned)

Non-crossing vs. NKW attack on first and last names



First names: red Last names: blue

[Grubbs et al. 2016]

Taking account of additional leakage

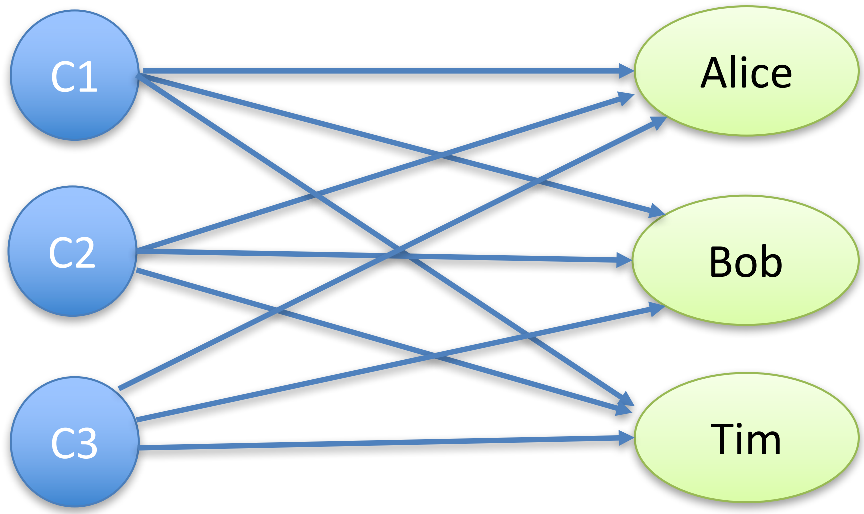
Recall that most OPE schemes reveal more information than just order, frequency
BCLO scheme (the one used most widely in practice) leaks ~half of plaintext bits

BCO '11 gave way to compute value m_c which is midpoint of range of size sqrt(N) that has 99% chance of including plaintext

Plaintext	Ciphertext	m_c
michael	cyrzjipnouushzh	michaekypfbkfr
david	aenpse cevvpkmr	david jwbvhec
robert	emlqrnycvblqqnd	robert lwyeorr
john	ccnnczzzpruvjhd	johmzzzysfbunn
james	bzkxrq gzortby	james zyovtq
daniel	aelfspocabjdvjc	daniel jgaginu
richard	ekrzjmjhjxykbba	richardkmfnwwx
jose	ccqrlzzziozokby	josdzzzxvfruqg
mark	cwmlfzzzjxhlklh	marjzzzxzqyduv
christopher	zokwwbrbibyouo	christotnqfolw

[Grubbs et al. 2016]

Can prune graph problem using this additional leakage



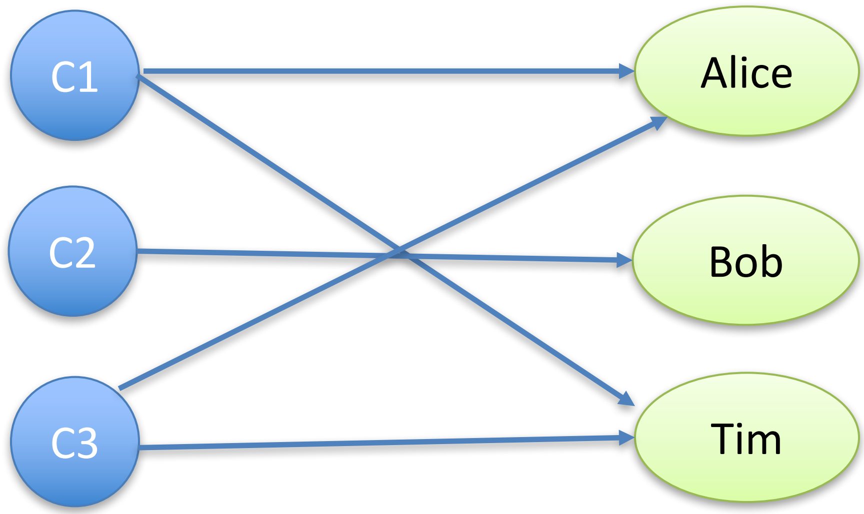
Bipartate graph problem: find *non-crossing* matching of ciphertexts, names that minimizes total weight

$$\text{Weight}(C,\text{Name}) = |\text{Freq}(C) - \text{Freq}(\text{Name})|$$

This takes into account ordering constraints

[Grubbs et al. 2016]

Can prune graph problem using this additional leakage



Bipartate graph problem: find *non-crossing* matching of ciphertexts, names that minimizes total weight

$$\text{Weight}(C,\text{Name}) = |\text{Freq}(C) - \text{Freq}(\text{Name})|$$

This takes into account ordering constraints

[Grubbs et al. 2016]

LAAs Fallout

Lots of open questions on how to improve already damaging attacks

Unclear if, and if so where, PREs can still be used securely

In practice PREs are still the only option in many important contexts
- otherwise: must download to process encrypted data

Scheme(s)	First names	Last names
Kerschbaum [27]	26%	6%
Popa et al. [36], Kerschbaum [28]	**84%**	38%
BCLO [12, 13]	**99%**	**97%**
CLWW [18]	**98%**	**75%**
BCLO + CLWW [18]	**85%**	44%
Baseline Guessing	4%	1%