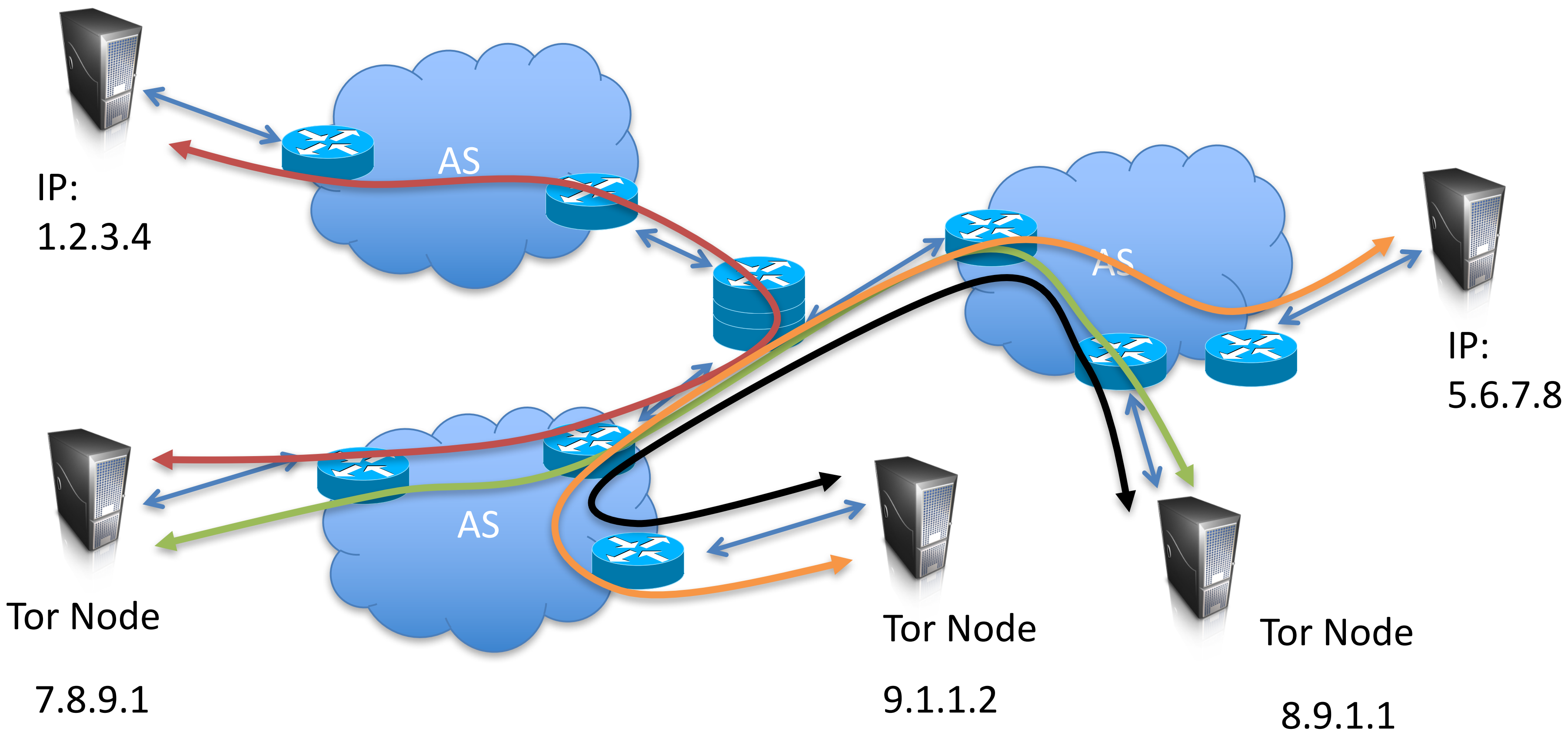
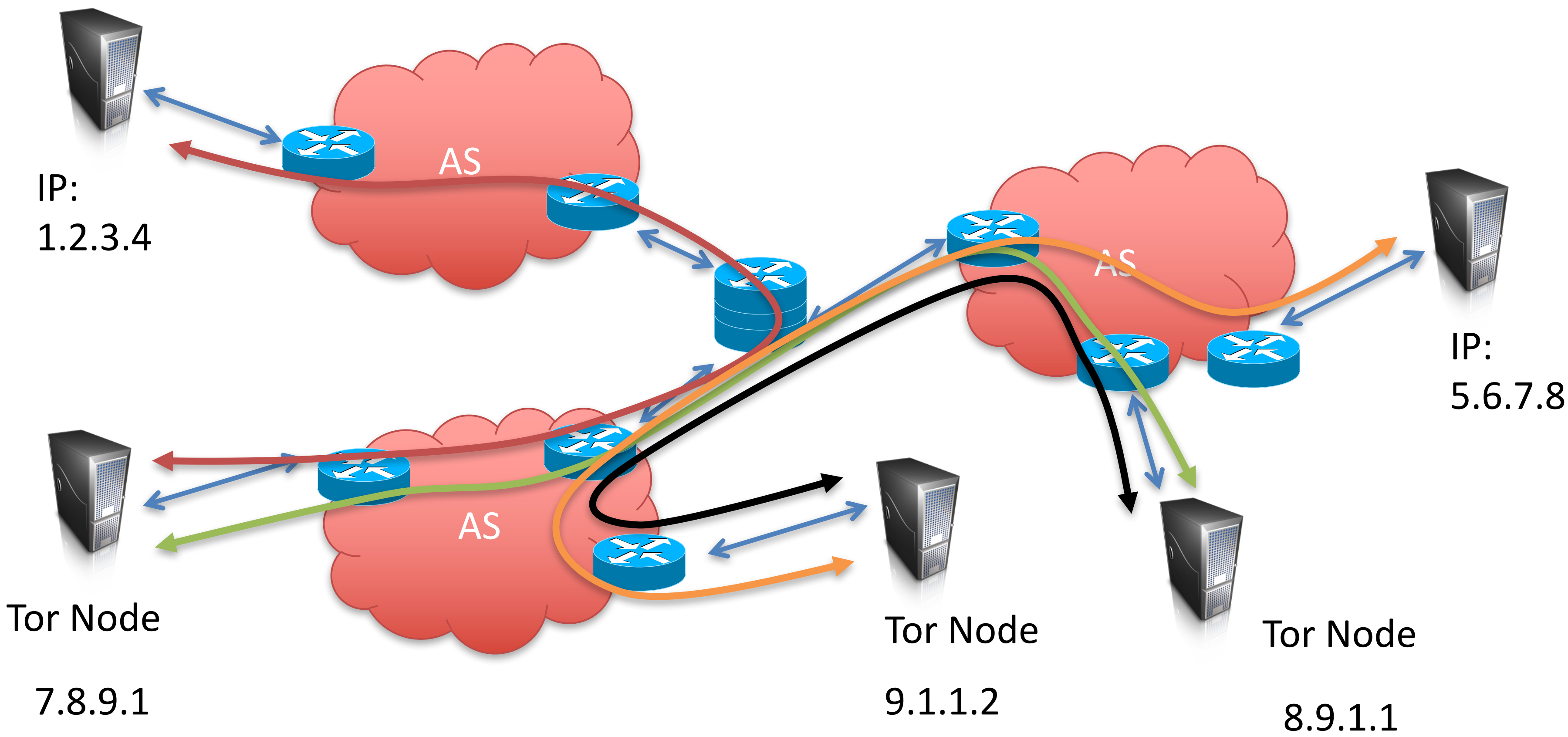
Traffic analysis resistance

Tom Ristenpart
CS 6431

Onion routing (low-latency)



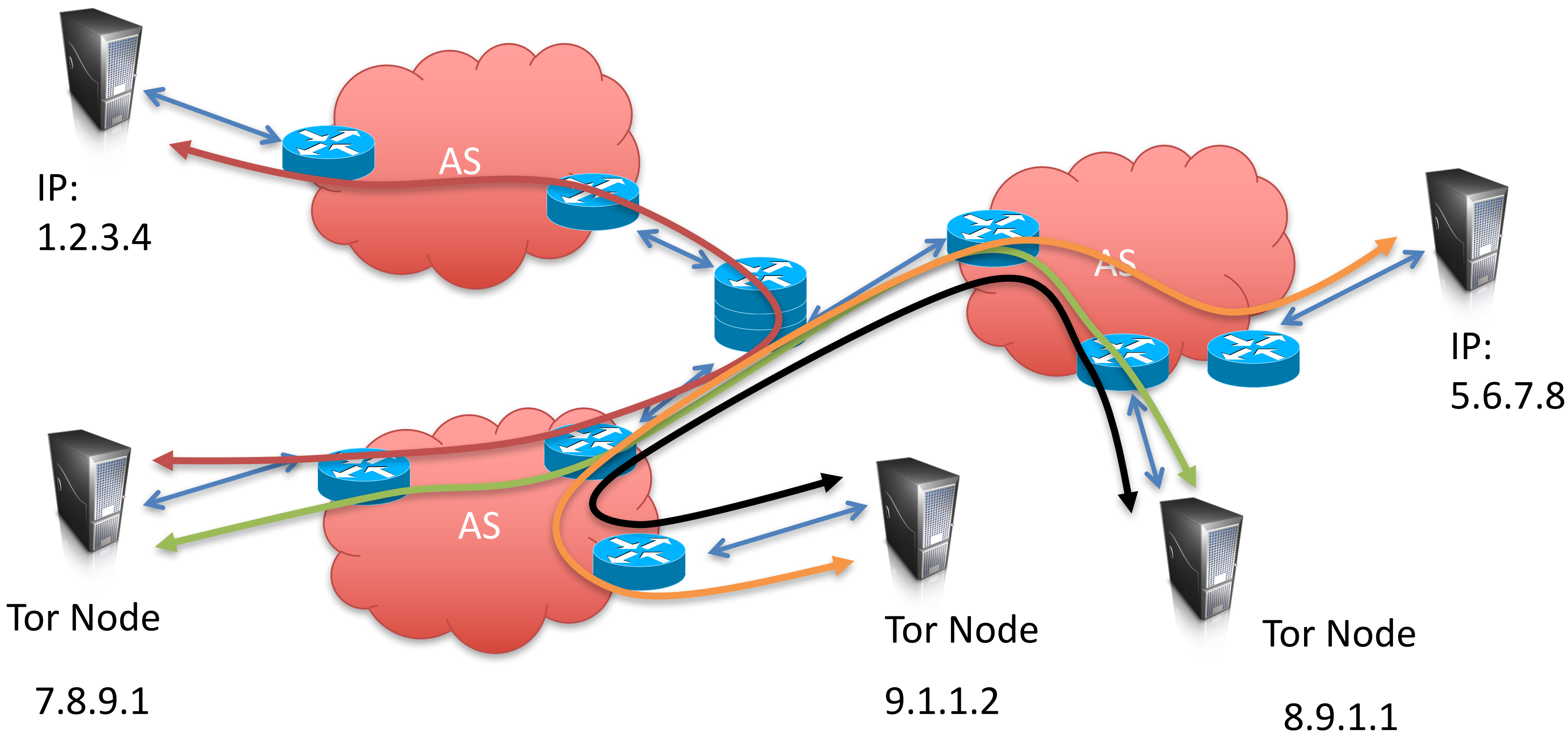
Onion routing (low-latency)



At least: traffic correlation attacks

- Correlate timing of packets sent in from 1.2.3.4 and those received at 5.6.7.8

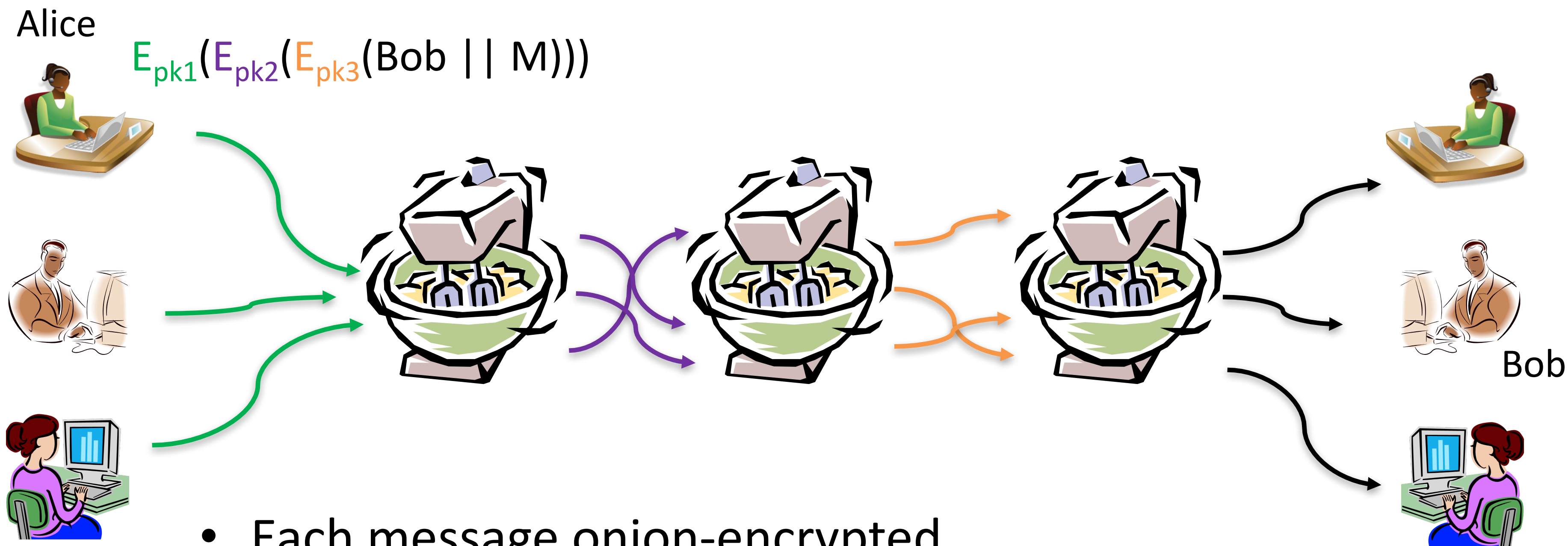
Onion routing (low-latency)



Many suggestions:

- adding noise (dummy requests/traffic) to obfuscate traffic patterns. Ad hoc suggests subsequently (academically) broken

Mixnets

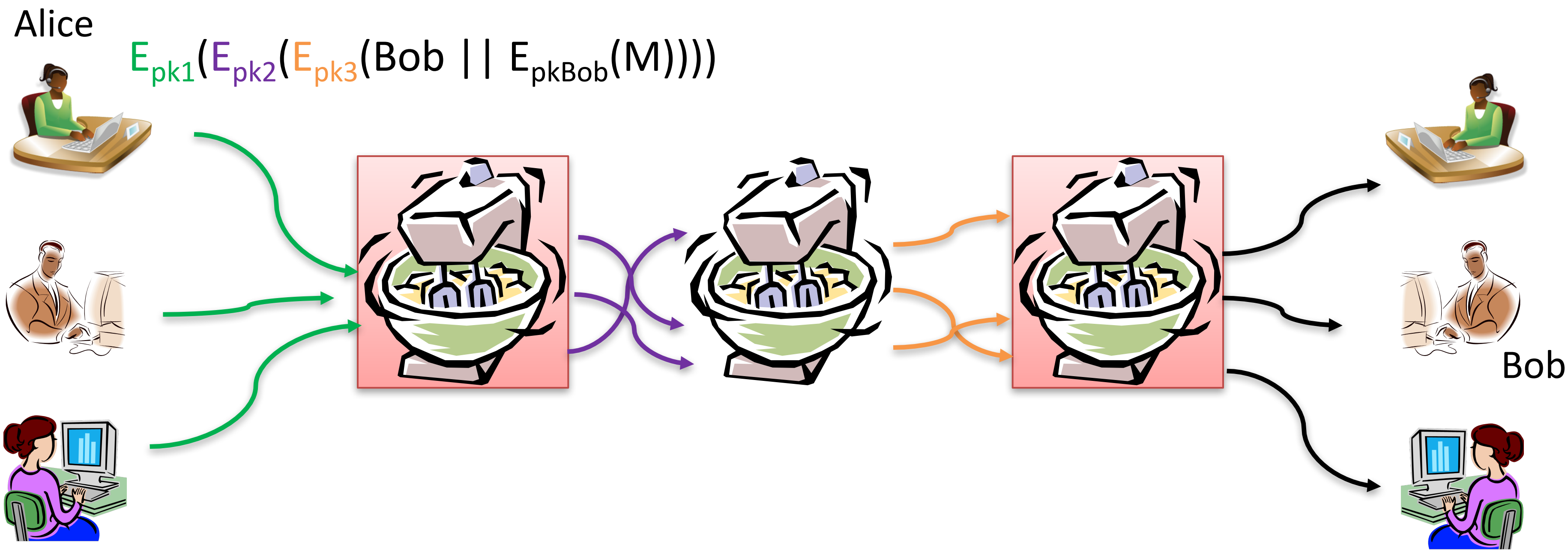


- Each message onion-encrypted
  - All messages must be padded to same length
- First mix node waits for lots of encrypted messages
  - Decrypts outer layer, shuffles, sends to next node
- Final node can send messages to destinations
- Security should hold if any single node trustworthy

All-but-one traffic analysis threat model

- Adversary controls all but one server
- Adversary can monitor, block, delay, inject traffic on any network link
  - Adversary knows all users that participate

Mixnets



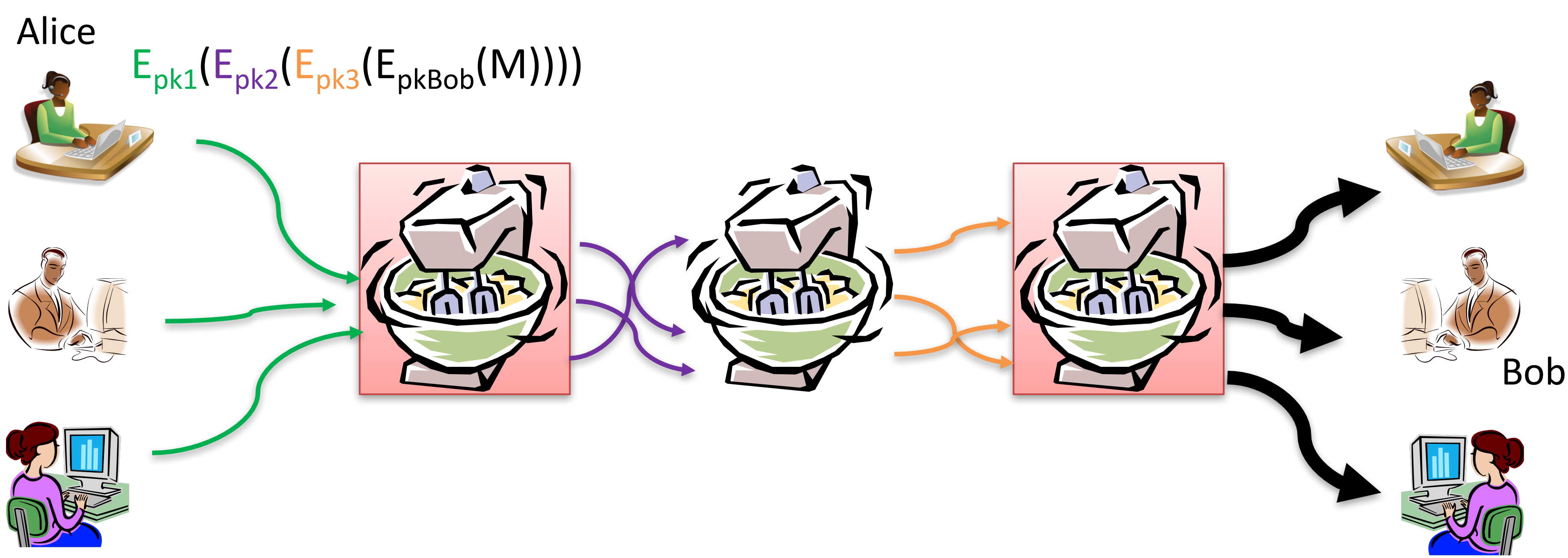
What is protected? What is leaked?

- Set of users who sent a message
- Set of users who received a message

Have all users always send a message (can be dummy)

Don't reveal recipient in final plaintext. All users download *all* final ciphertexts. Trial decrypt

Mixnets



What is protected? What is leaked?

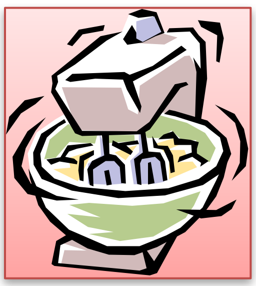
- Set of users who sent a message
- Set of users who received a message

Have all users always send a message (can be dummy)

Don't reveal recipient in final plaintext. All users download *all* final ciphertexts. Trial decrypt

Private information retrieval

Can Bob recover his ciphertext without downloading every ciphertext?
Chor, Goldreich, Kushilevitz and Sudan introduced PIR in 1995



$(1, C_1)$
$(2, C_2)$
…
(u, C_u)

Server learns nothing about i
Requires O(u) work on server

Use less than O(u) bandwidth?

i

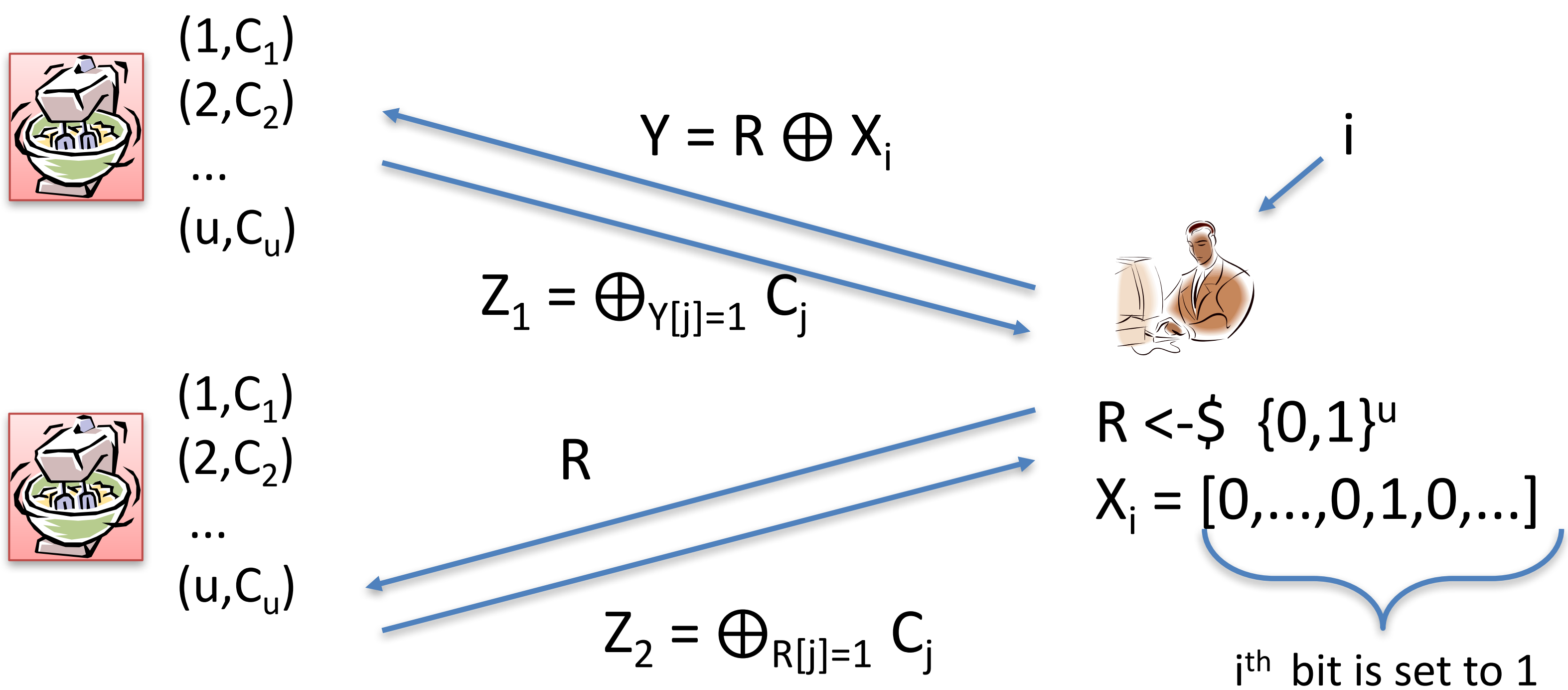
Bob learns C_i

Two variants:

- Information-theoretic (IT-PIR): Split database across k servers. As long as one is honest, adversary can't learn anything about i
- Computational (CPIR): Single computationally-bounded database can't learn anything about i

Basic IT-PIR scheme

Can Bob recover his ciphertext without downloading every ciphertext?

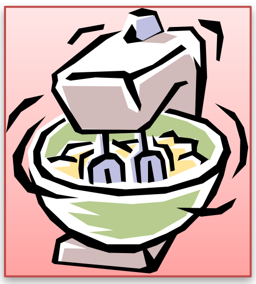


$C_i = Z_1 \oplus Z_2$

If servers don't collude, either learns nothing about i

Basic CPIR scheme

Uses homomorphic encryption: $Enc(k,m_1)*Enc(k,m_2) = Enc(k,m_1+m_2)$



(1,C_1)
(2,C_2)
…
(u,C_u)

$Q_1,\ldots,Q_u$

Z

i

$Q_j = Enc(k,0) \quad j \neq i$

$Q_j = Enc(k,1) \quad j = i$

$C_j = Dec(k,Z)$

$$Z = \prod_{j=1}^{u} Q_j^{C_j}$$

$j \neq i$: $Q_j^{C_j} = Enc(k,0)$

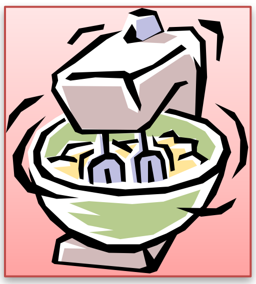
C_j times

$j = i$: $Q_j^{C_j} = Enc(k,1) * Enc(k,1) * \cdots * Enc(k,1)$

$= Enc(k,C_j)$

Basic CPIR scheme

Uses homomorphic encryption: $Enc(k,m_1)*Enc(k,m_2) = Enc(k,m_1+m_2)$



$(1,C_1)$
$(2,C_2)$
…
(u,C_u)

$Q_1,\ldots,Q_u$ ←

Z →

i

$Q_j = Enc(k,0) \quad j \neq i$

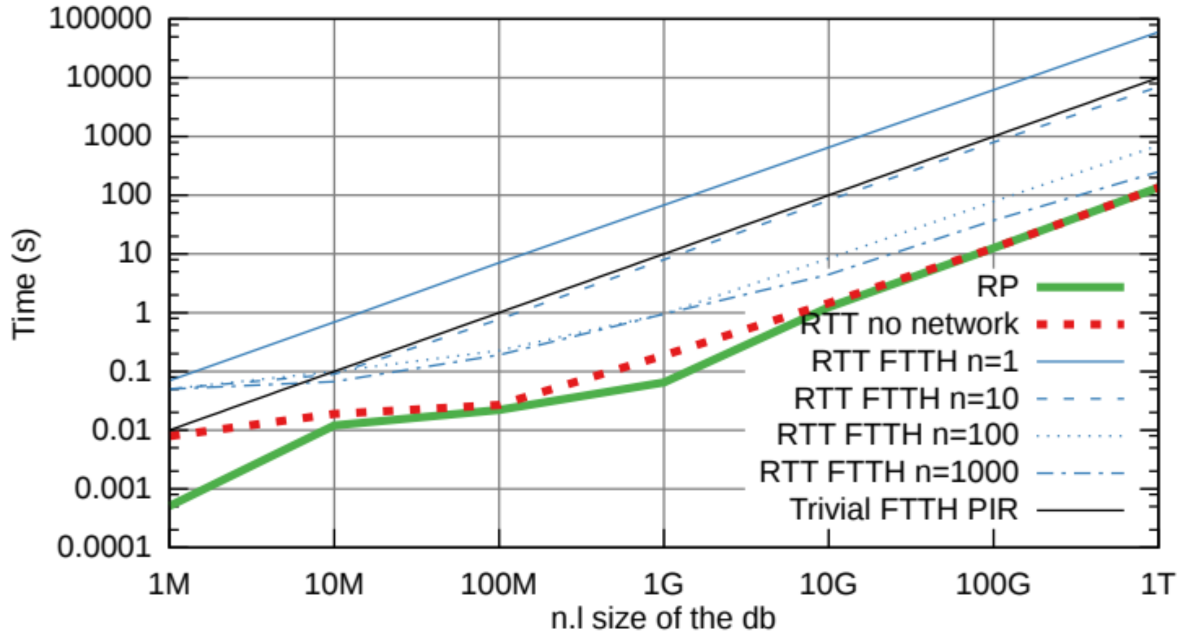
$Q_j = Enc(k,1) \quad j = i$

Security: as long as Enc is IND-CPA, no computationally bound adversary can determine i

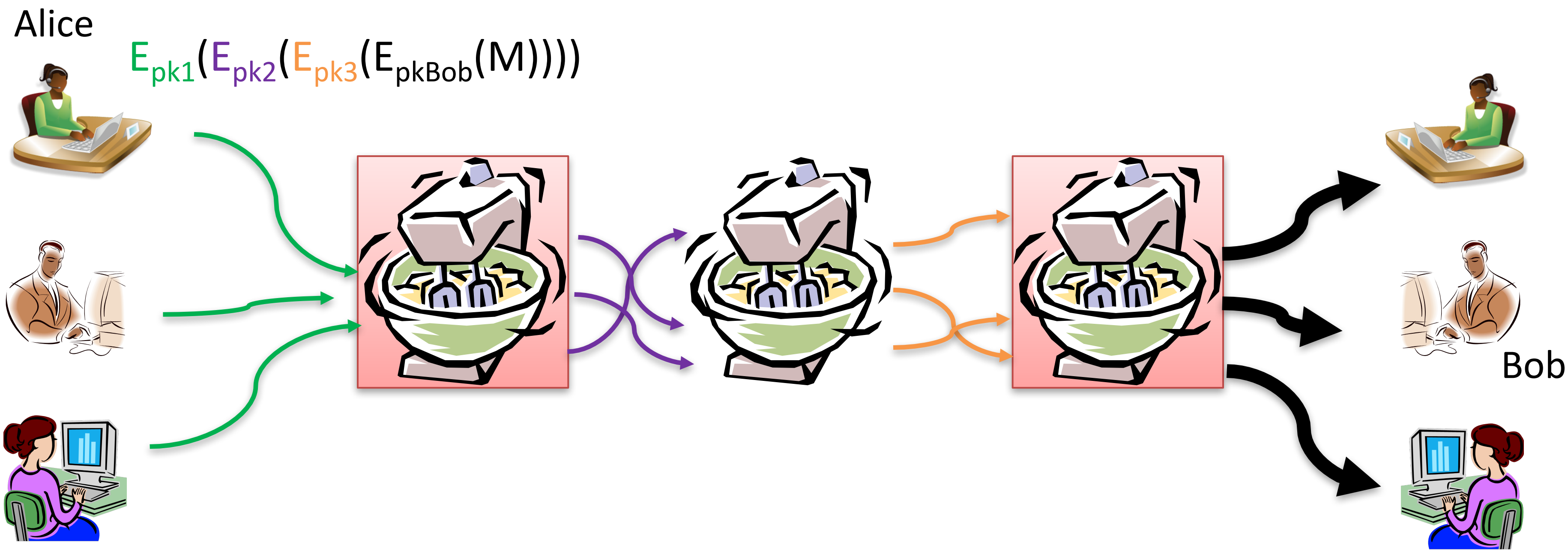
Fast CPIR Schemes

- XPIR scheme based on ring LWE (lattices)
  - Aguilar-Melchor et al. 2014

100 Gb database processed in a few seconds



Mixnets



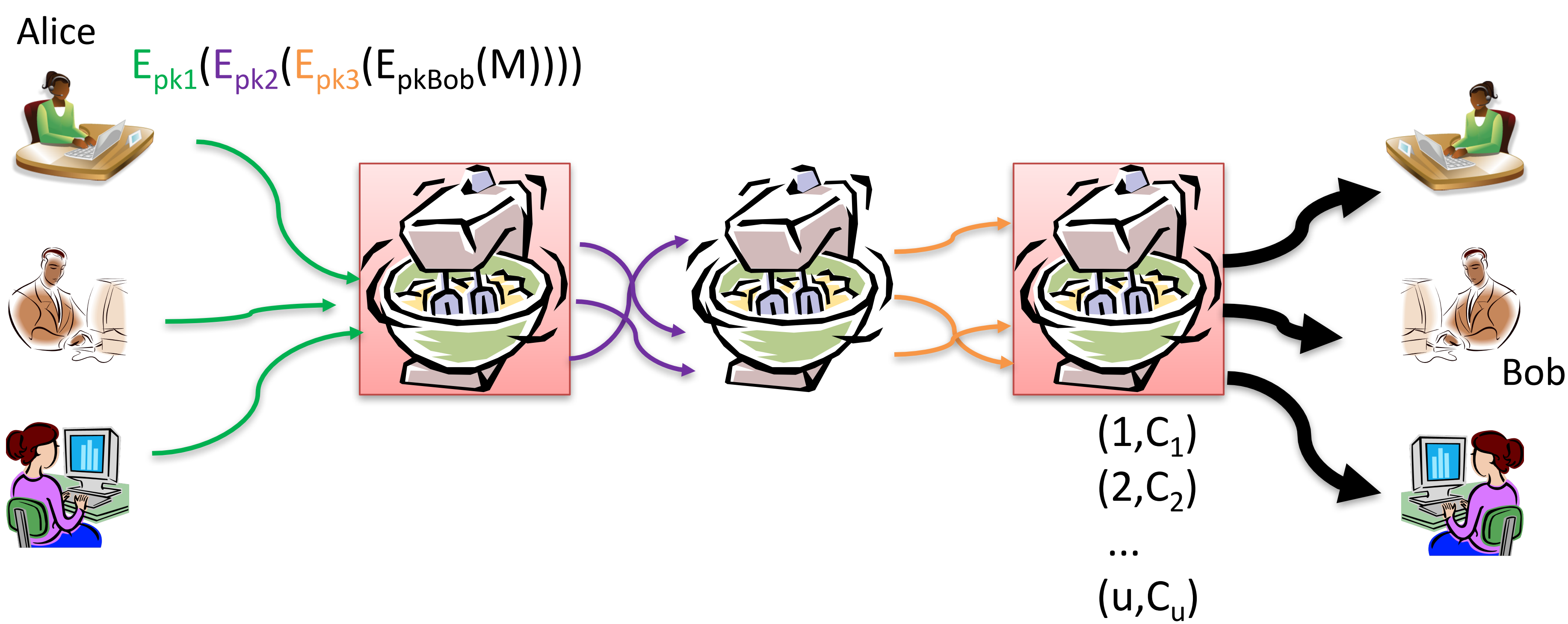
What is protected? What is leaked?

- Set of users who sent a message
- Set of users who received a message

Have all users always send a message (can be dummy)

Don't reveal recipient in final plaintext. All users download *all* final ciphertexts. Trial decrypt

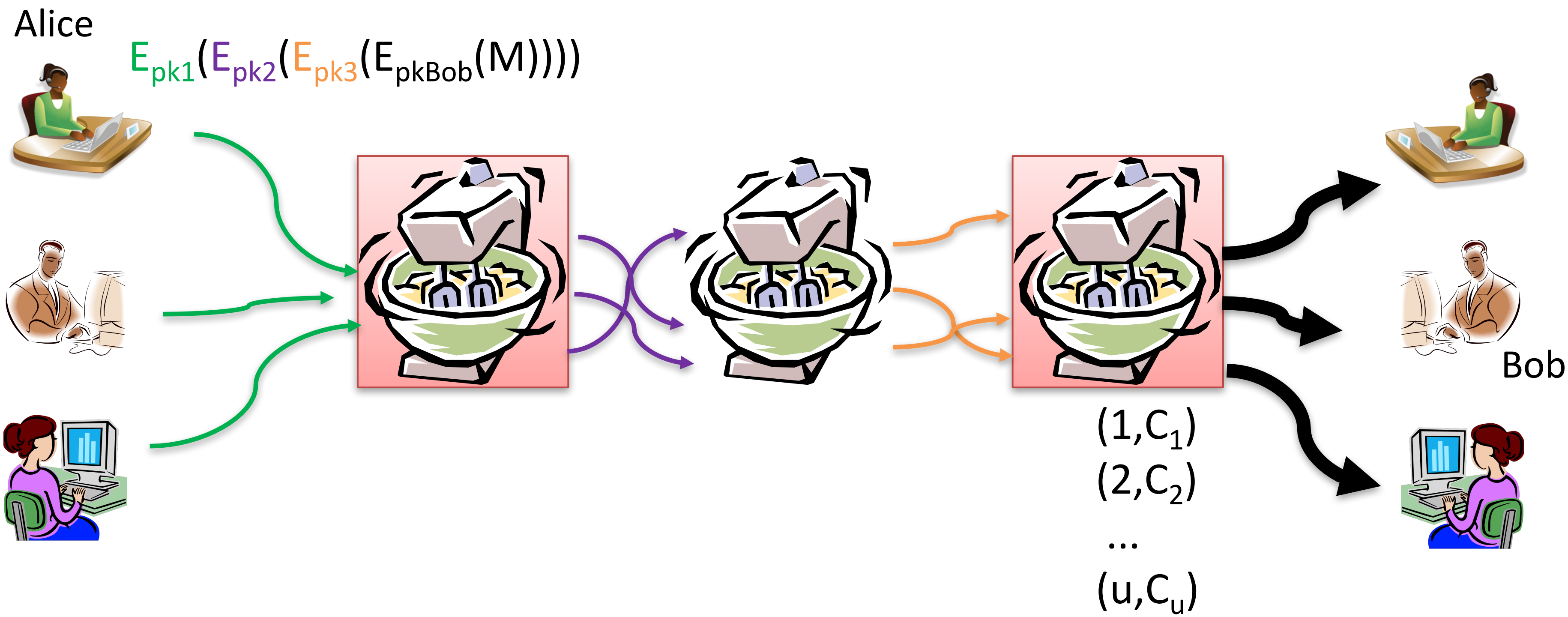
Mixnets



Recipients can each use CPIR to retrieve their ciphertext?

How does Bob know what index i his ciphertext is in?

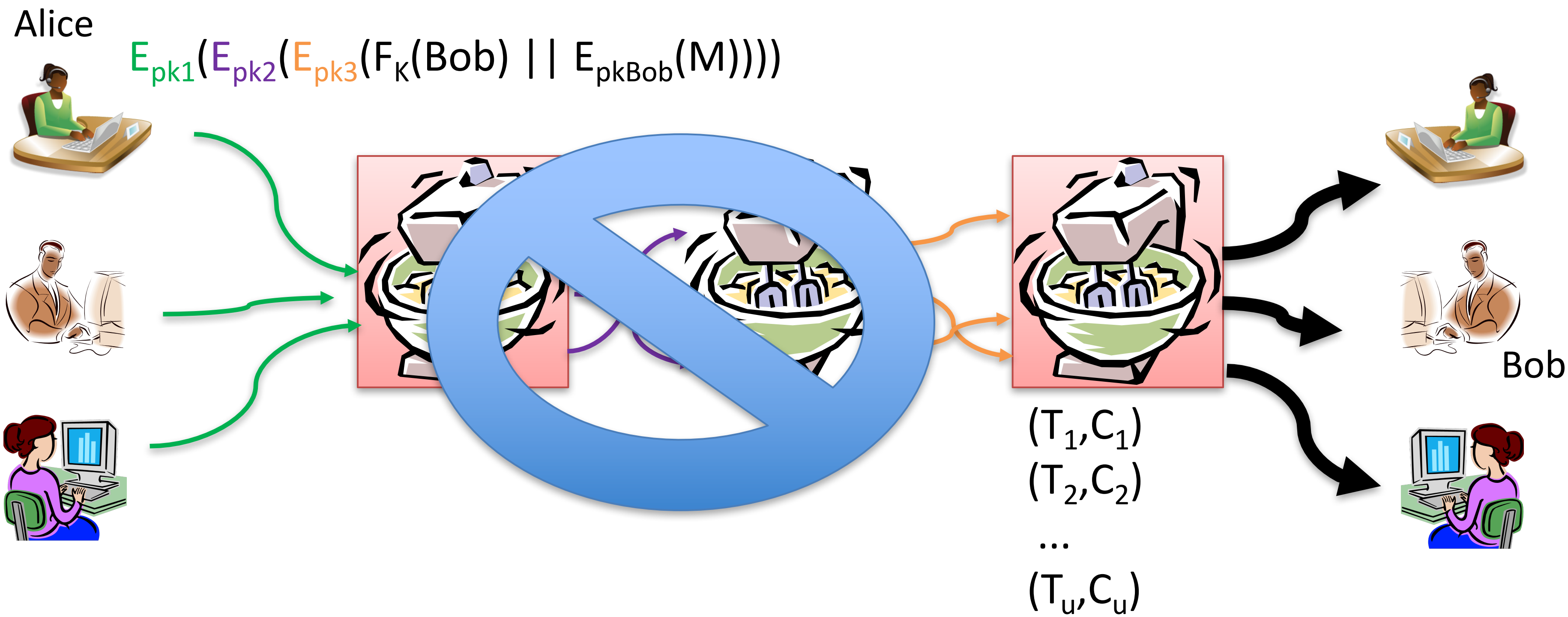
Mixnets



Recipients can each use CPIR to retrieve their ciphertext?

How does Bob know what index i his ciphertext is in?

Mixnets



Replace indices i with $T_i = F_K(Bob)$ for PRF F and key K known to Alice and Bob.

Do PIR over compact data structure representation of this table

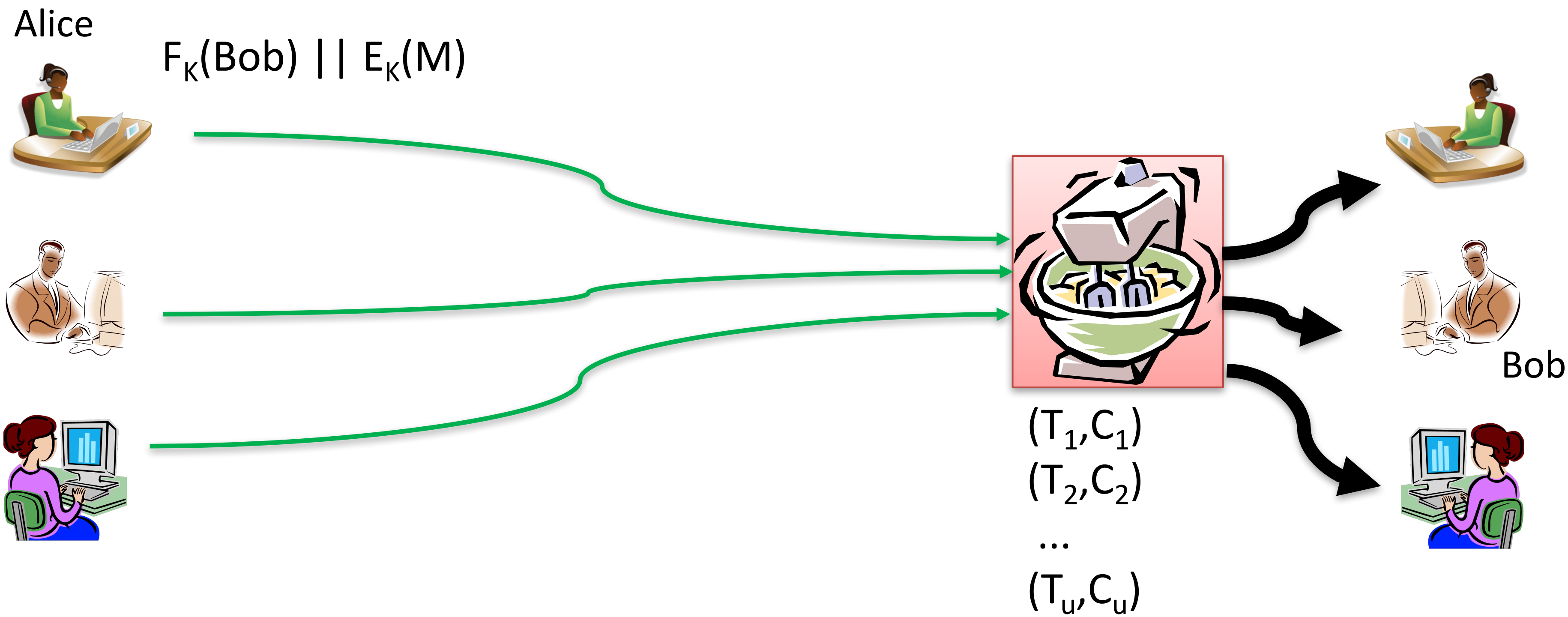
- Pung [Angel, Setty 2016] uses binary search trees

No-trust* traffic analysis threat model

- Adversary controls ***all*** servers
- Adversary can monitor, block, delay, inject traffic on any network link
  – Adversary knows all users that participate

* Still need to trust developers, other communication partners, end-point security, etc.

PIR-based schemes



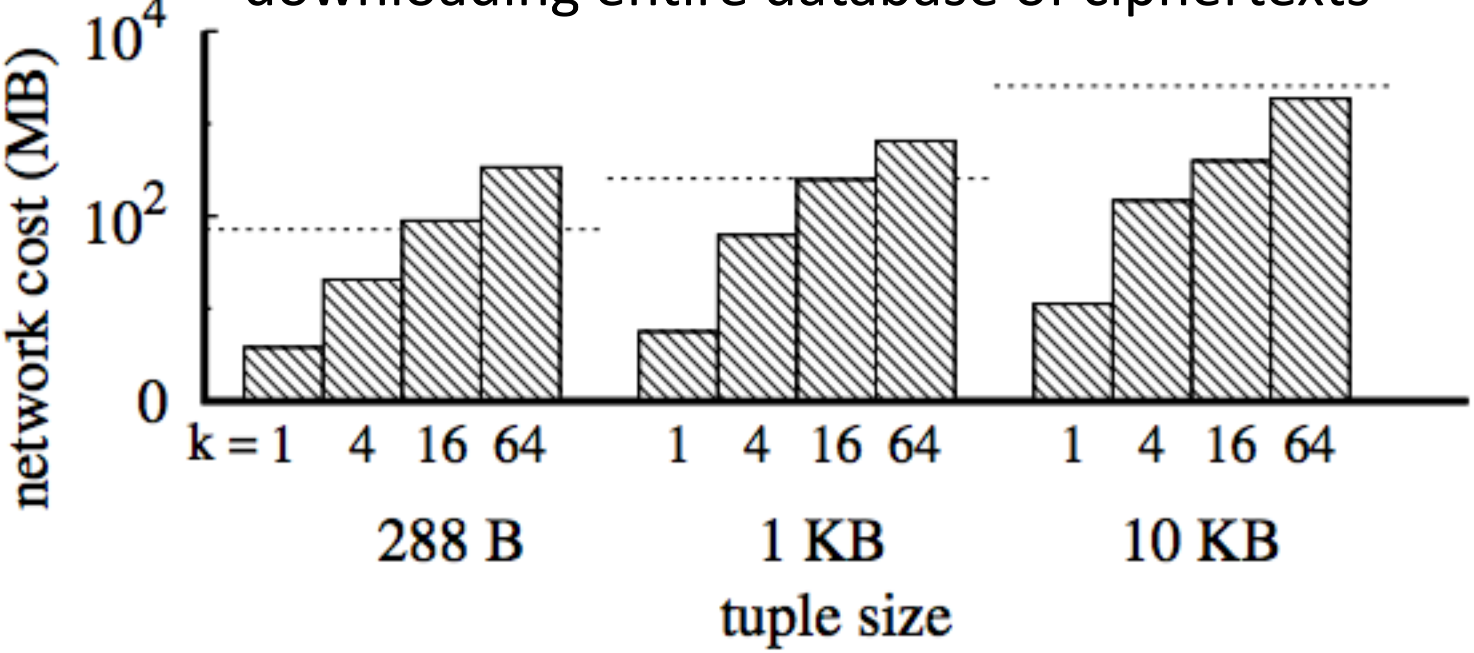
Get rid of mixnets entirely

Security holds even if server is adversarial

- Still need every client to always send messages
- Need to never reuse tags (add counters)

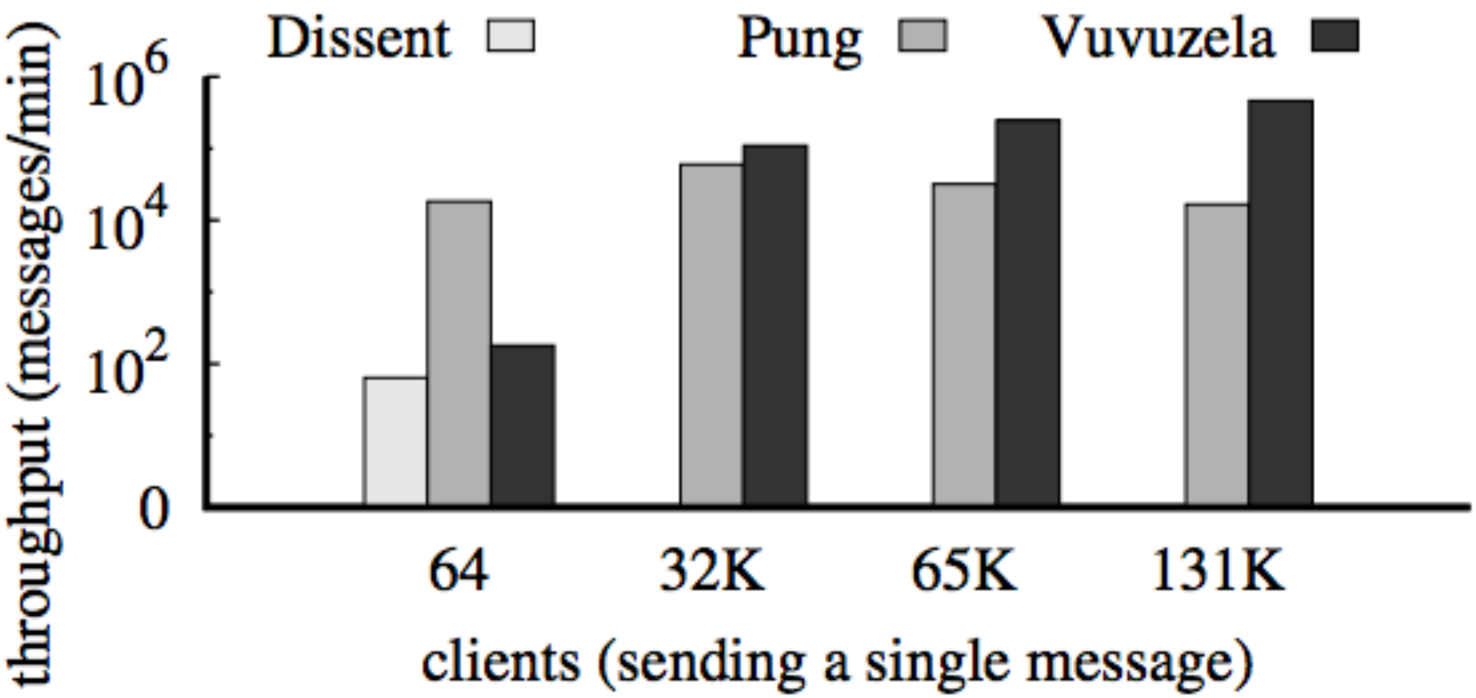
Pung network bandwidth costs

Dashed line indicates trivial PIR of
downloading entire database of ciphertexts



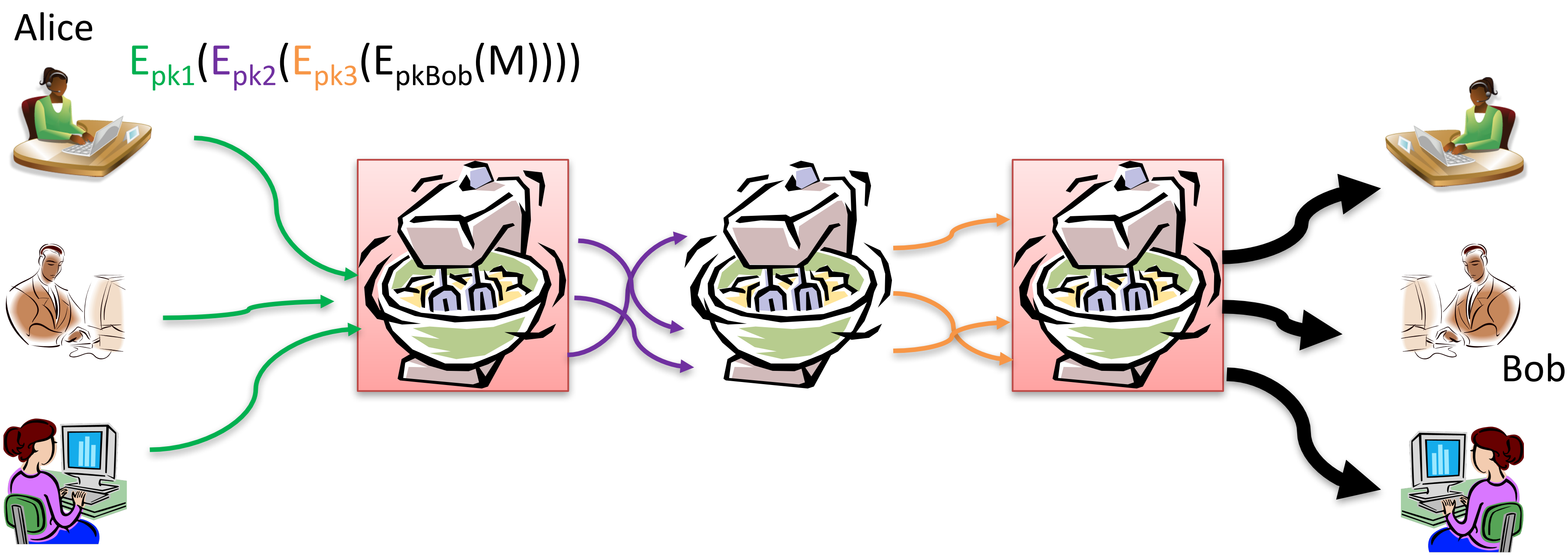
"Perhaps surprisingly, we find that under certain regimes (e.g., small tuple sizes, high k), it is beneficial for clients to simply download the entire collection instead of using Pung's multi-retrieval."

Pung throughput



256 byte message

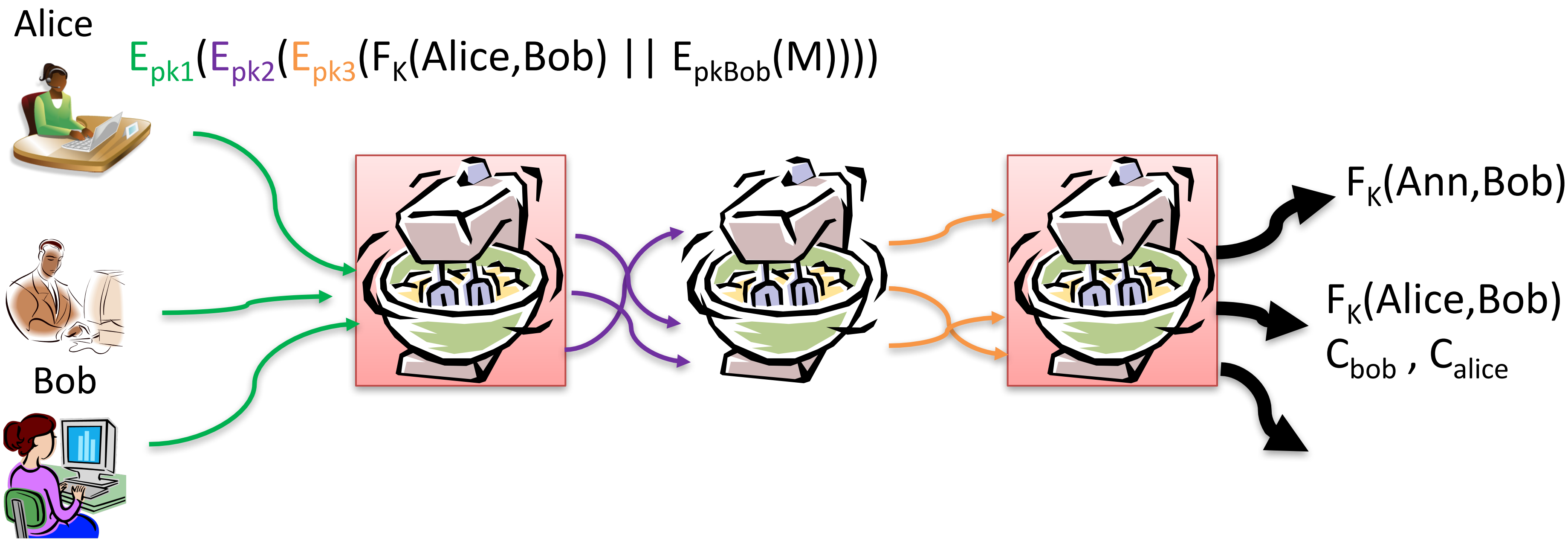
Towards Vuvuzela



What is protected? What is leaked?

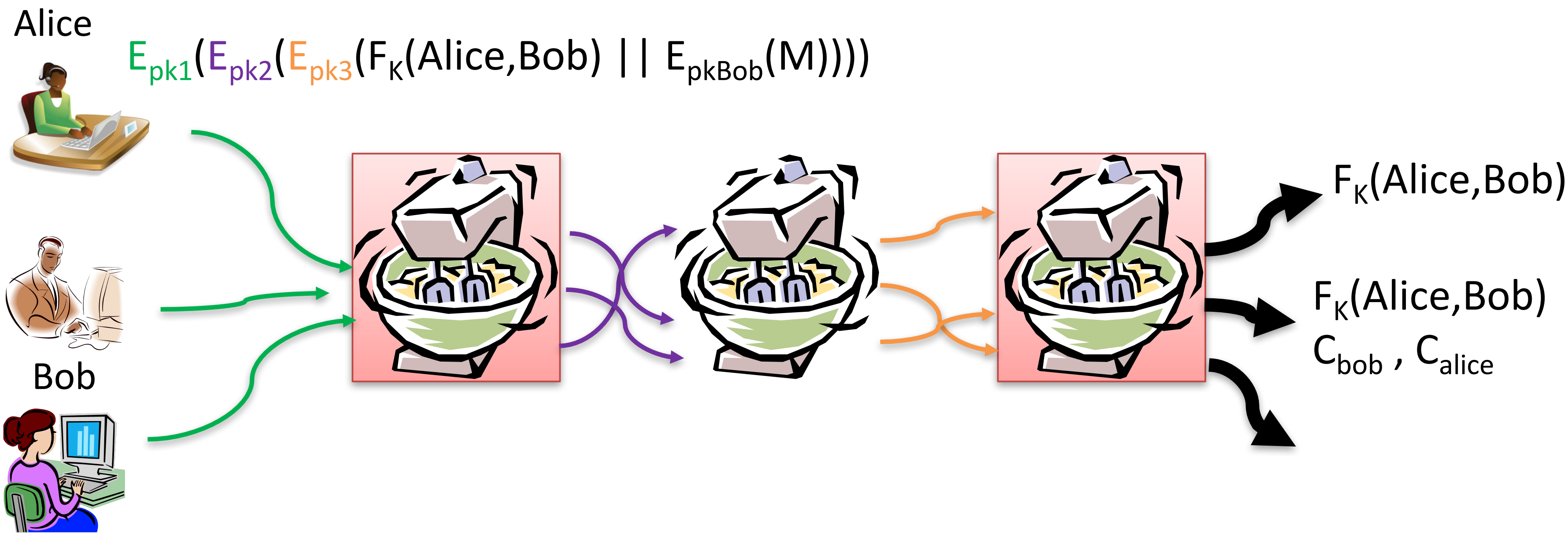
- Set of users who sent a message
- Set of users who received a message

Vuvuzela



- Weaken security goal to differential privacy (more in a second)
- Have final node store message at deaddrop F_K(Alice,Bob)
- Every round is both read and write through mixnet for all users
  - No communication, send dummy message to random tag
  - Messages sent to same deaddrop sent back through via reverse onion encryption
- Must use counters to avoid repeat use of tag F_K(Alice,Bob)

Vuvuzela

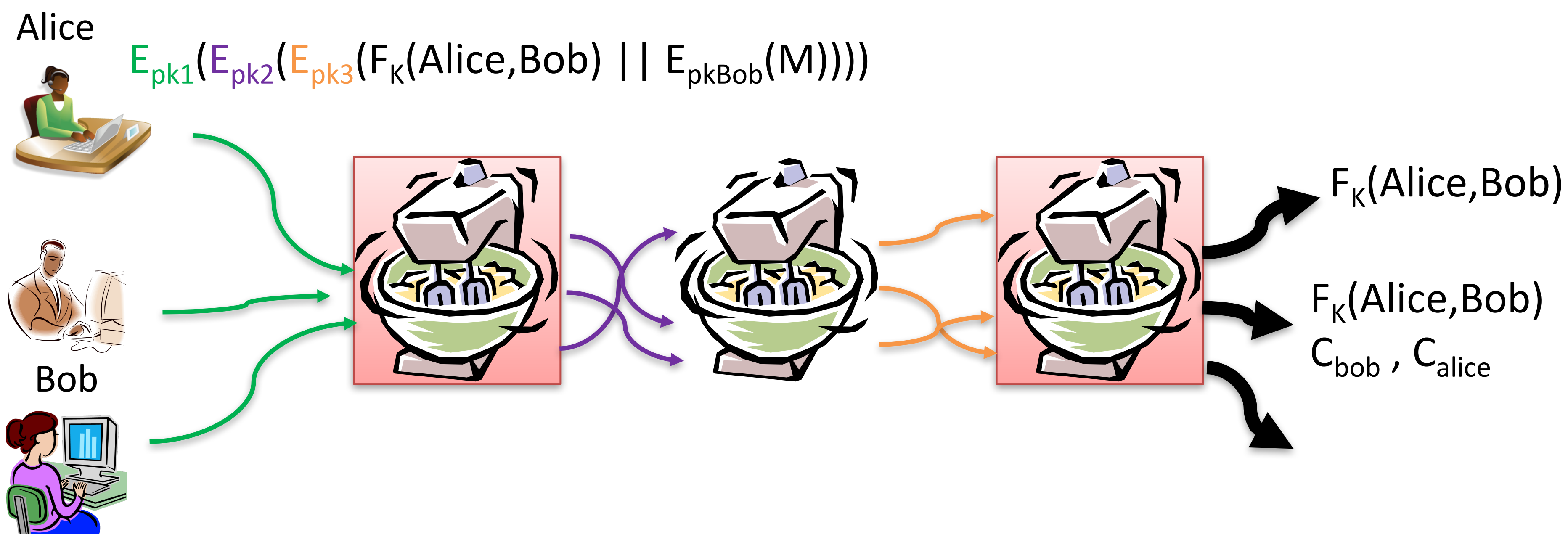


What still leaks if we stop here?

If Alice, Bob communicating, must be double access to a tag

Adversary drops all others’ communications and sees if there’s a double access to any tag... Confirms Alice, Bob communicating

Vuvuzela



Servers carefully add dummy messages to get *differential privacy* for leakage (# of double accesses and # of single accesses)

Whether Alice, Bob communicating or not gives rise to approximately same distribution of double deaddrop accesses

Vuvuzela DP goal

Let M be algorithm that adds noise to # of single accesses and # of double accesses. Then M is (ε,δ)-DP if

$$\Pr[M(x) \in S] \leq e^{\varepsilon} \Pr[M(y) \in S] + \delta$$

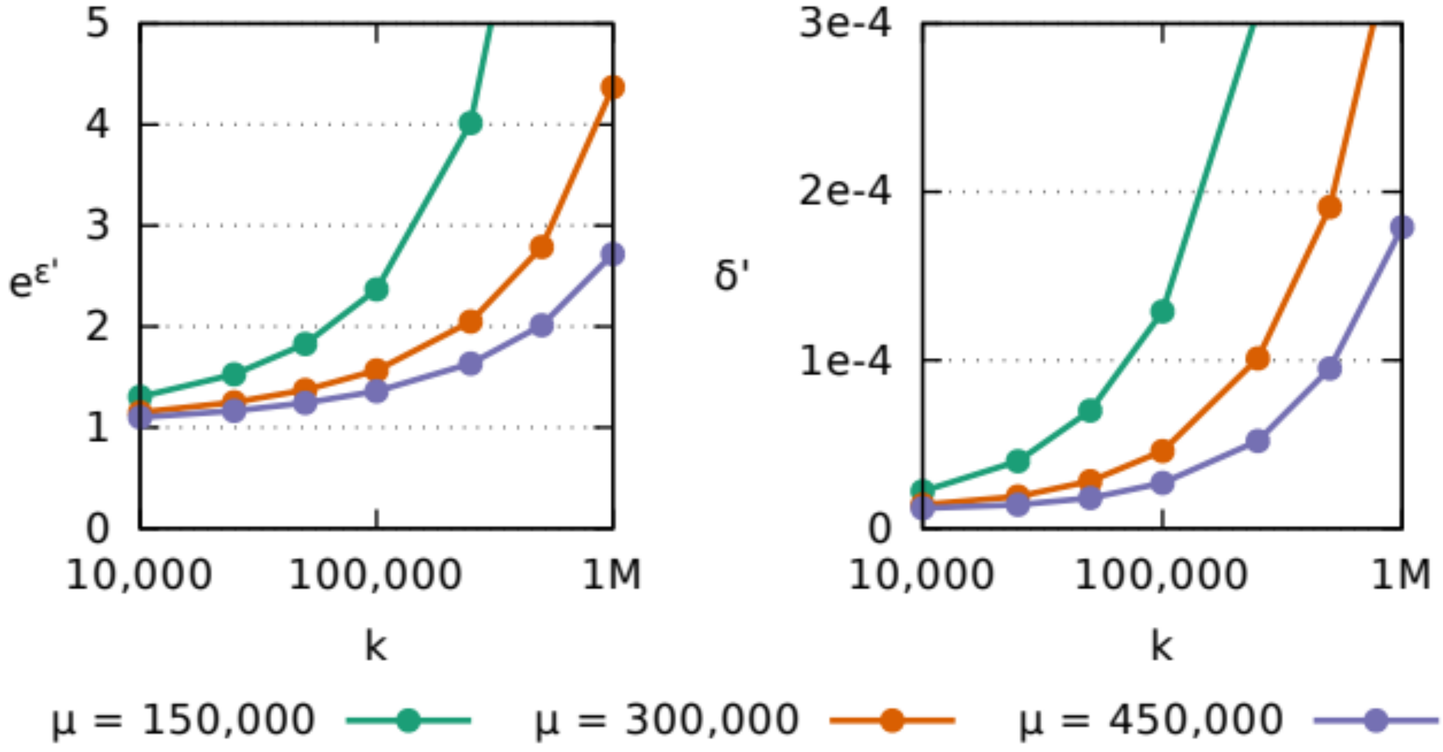
Thm: Amount of noise scales with sqrt(k) for k = # of rounds, independent of number of users

They target ($\ln 2, 10^{-4}$)-DP.
Fixes the number of rounds one can get this level of DP for
Bounds degrade if one goes beyond this number of rounds

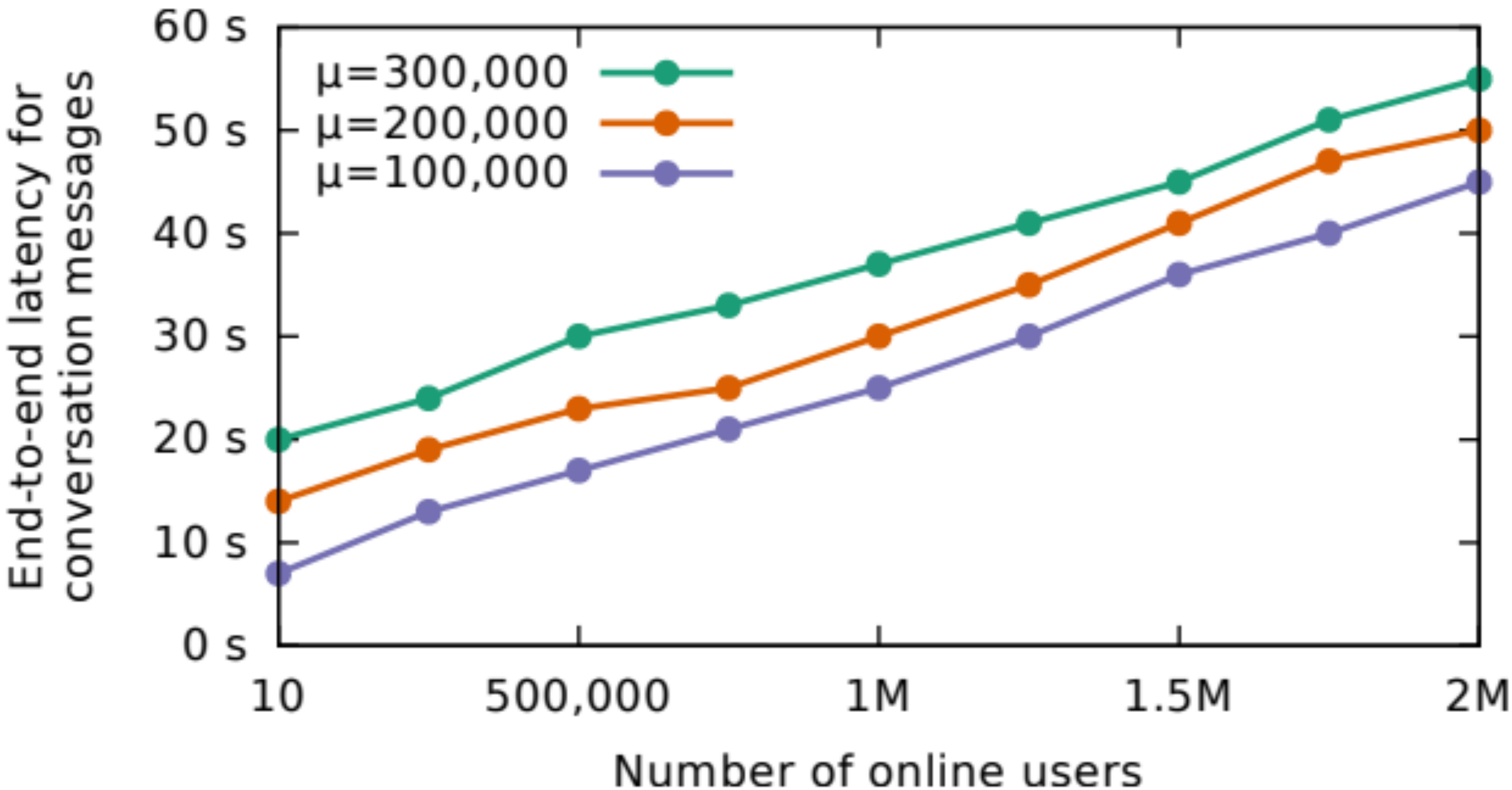
Vuvuzela DP goal

Let M be algorithm that adds noise to # of single accesses and # of double accesses. Then M is (ε,δ)-DP if

$$\Pr[M(x) \in S] \leq e^{\varepsilon} \Pr[M(y) \in S] + \delta$$



Performance



Doesn’t count “dialing protocol” for telling someone that you want to talk. Double all latencies. Also does not include ***variance*** due to noise (!!)

Is it practical?

- What does one do as # of rounds increases?
  - Nothing... system doesn’t provide meaningful formal security guarantees
- Latency fundamentally slow (must wait for all messages from all participants). See follow-up work [Tyagi et al. 2017]
- Expensive: $30k a month to run 3 servers

Broader issues of all recent systems:

- Users must get other’s keys out-of-band
- Clients must always participate when online
  - Huge waste of bandwidth!

Security levels

- No-trust model
- All-but-one model
- "Plausible deniability" model (differential privacy)

None prevent ***intersection attacks***:
can't prevent leakage when whether client is online or not is correlated with who they are talking to