Dissent in Numbers: Making Strong Anonymity Scale

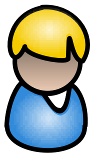
David Wolinsky[1], Henry Corrigan-Gibbs[1], Bryan Ford[1], and Aaron Johnson[2]

[1]Yale University, [2]US Naval Research Laboratory

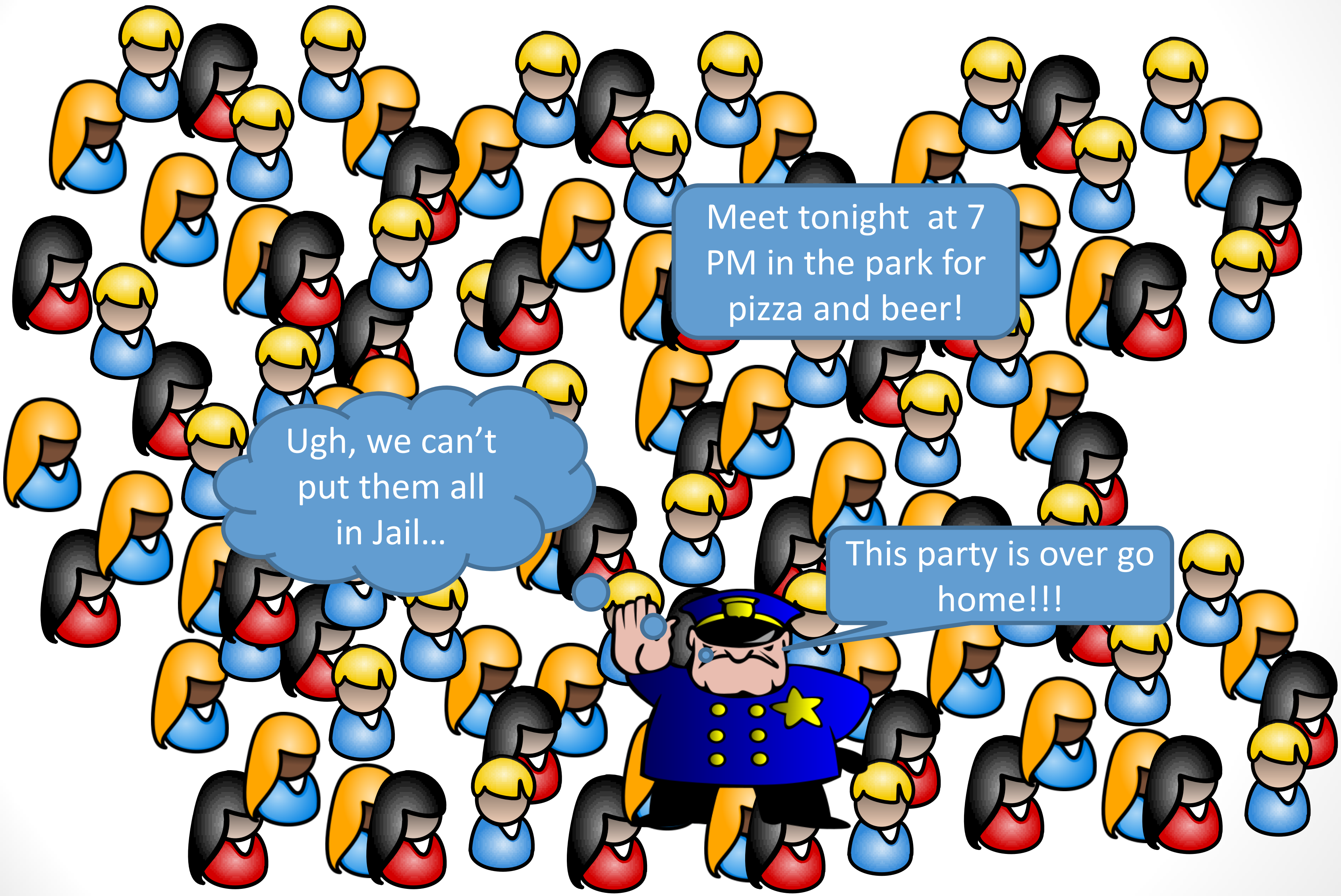
Motivation – Strength in Numbers

Motivation – Strength in Numbers

Motivation – Strength in Numbers



Motivation – Strength in Numbers



Making Strong Anonymity Scale?

- Challenge – tradeoff between scale and strength in anonymity systems favoring scale
- Goals
  - Strong anonymity (timing analysis resistant)
  - Scalability (100s to 1,000s of active participants)
  - Churn tolerant (unannounced member departures)
  - Accountability

Achieved in Dissent!

YALE

Organization

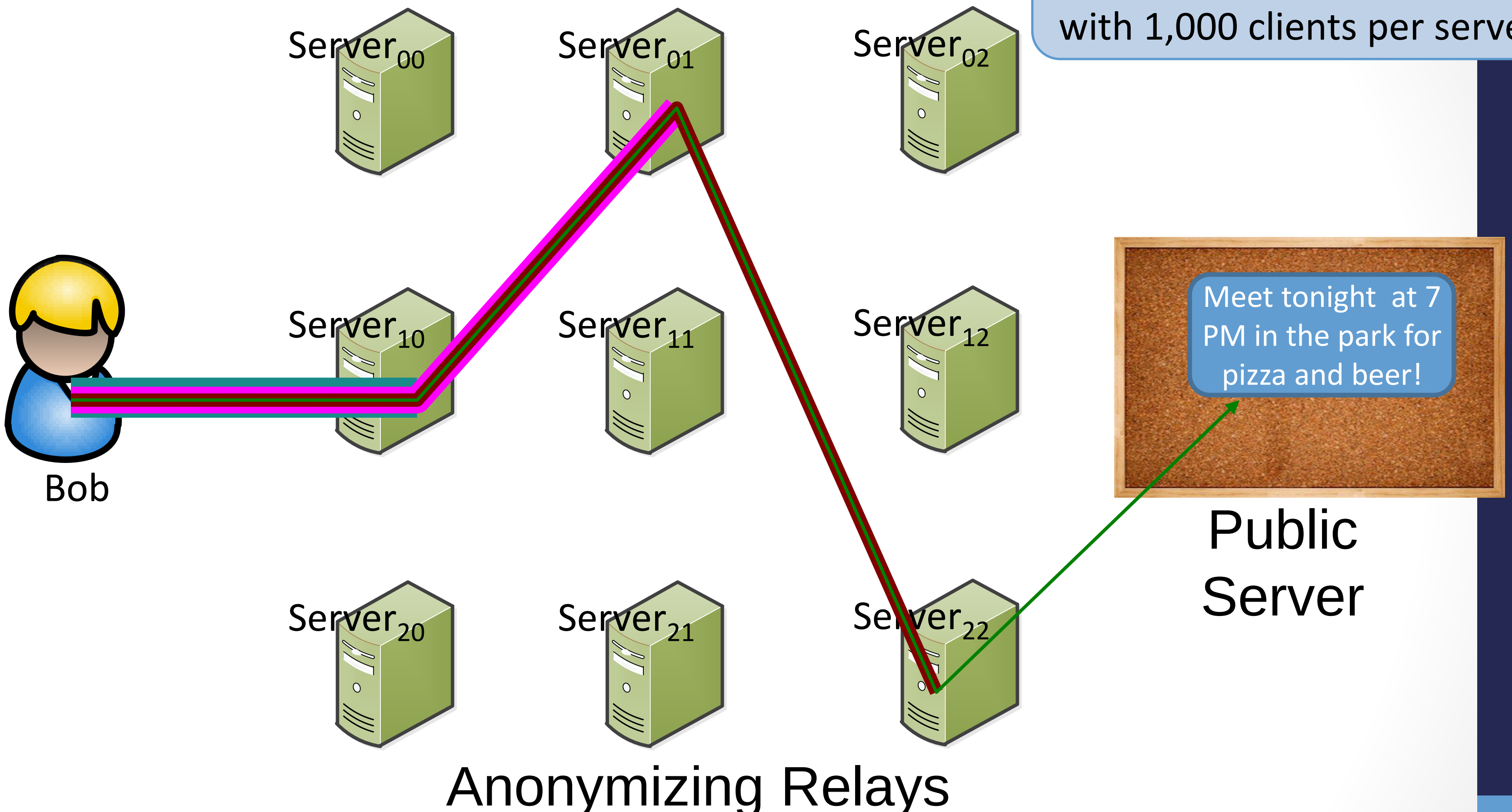
- Motivation
- Existing Approaches
- Dissent – Strong, Scalable Anonymity
  - Computational efficiency
  - Communication efficiency
  - Churn tolerant
  - Anonymity
  - Accountability
- Evaluation
- Conclusions

Organization

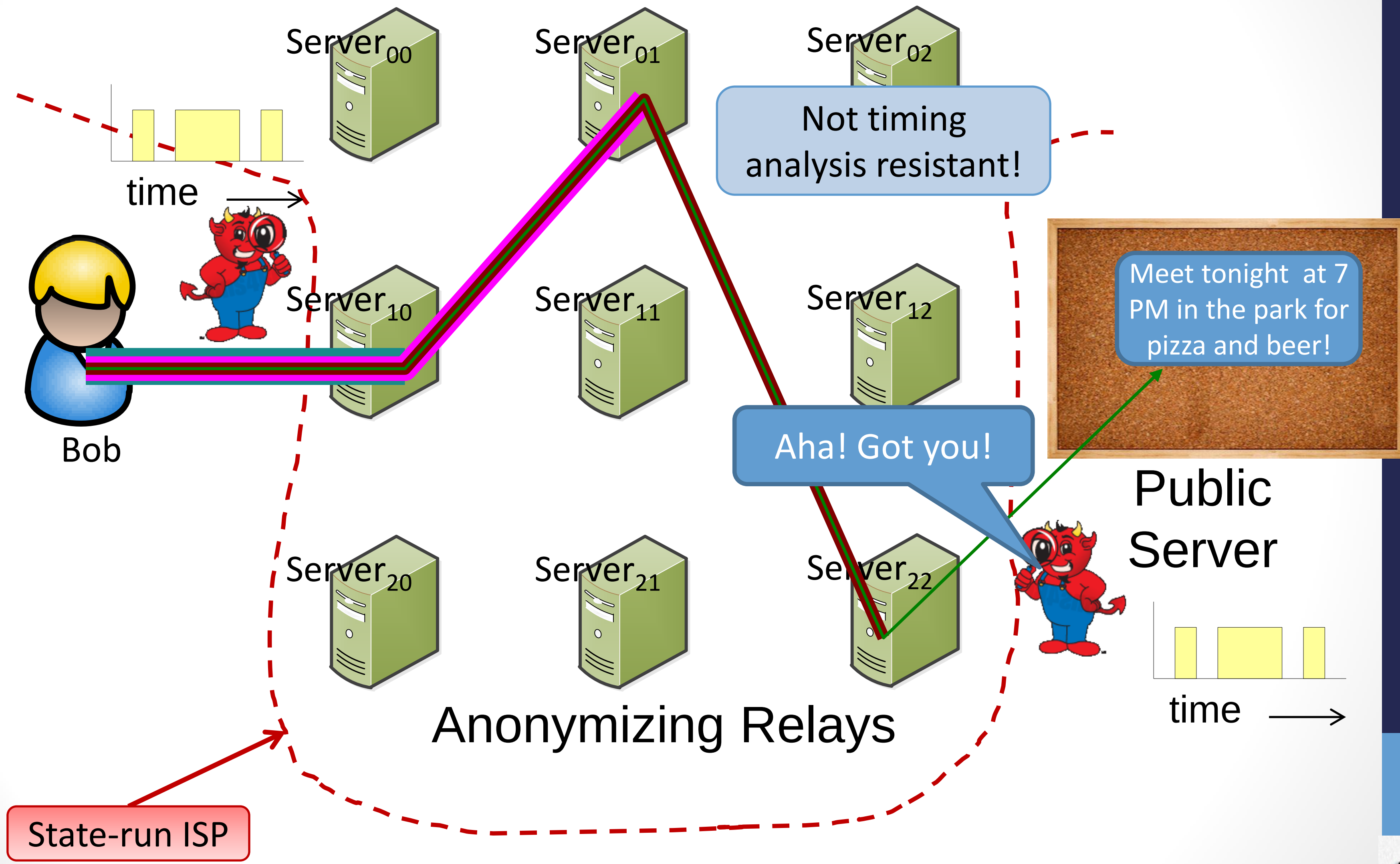
- Motivation
- **Existing Approaches**
- Dissent – Strong, Scalable Anonymity
  - Computational efficiency
  - Communication efficiency
  - Churn tolerant
  - Anonymity
  - Accountability
- Evaluation
- Conclusions

Tor – The Onion Router

Tor is scalable, supports more than 400,000 clients with 1,000 clients per server



Tor – The Onion Router

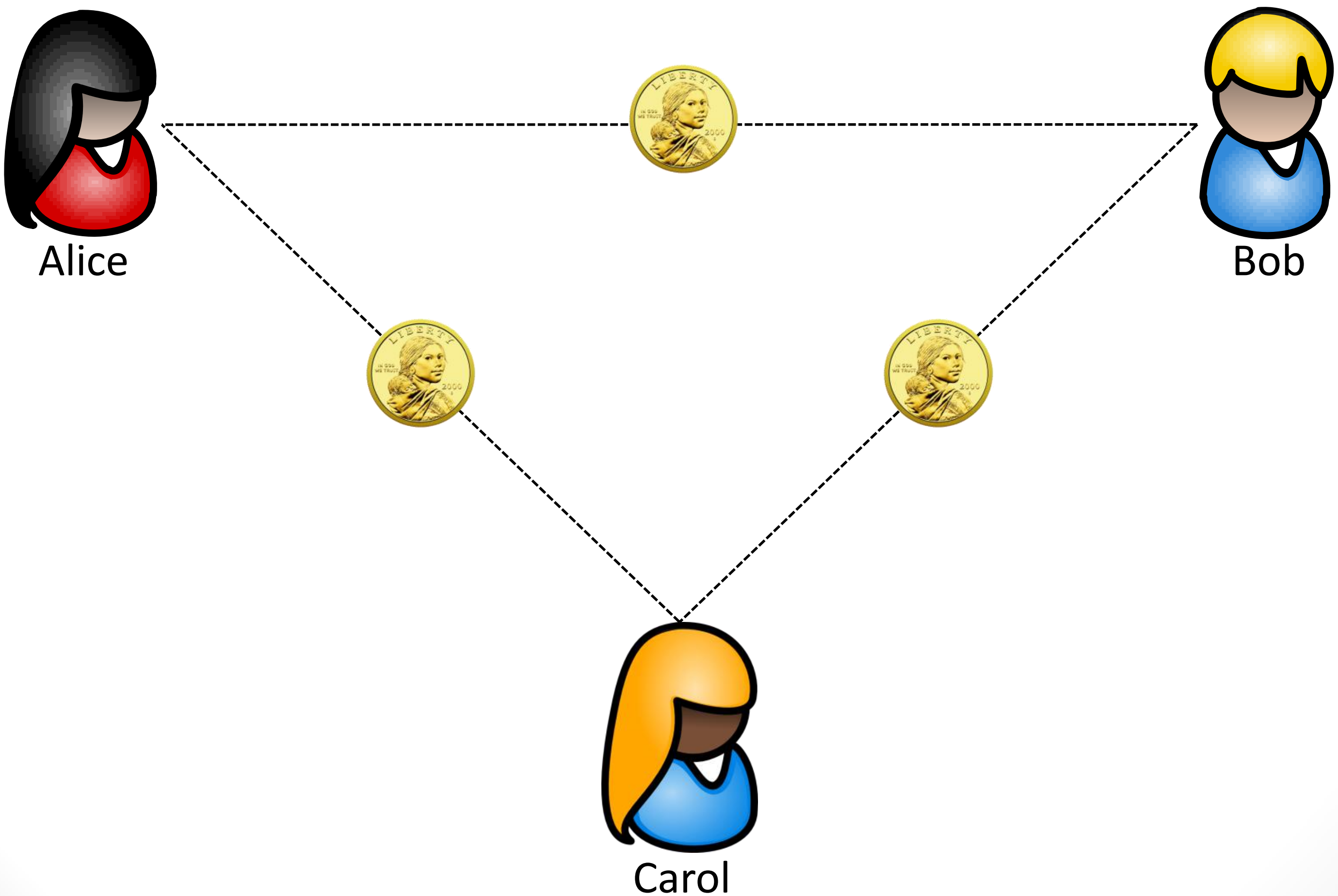


DC-net

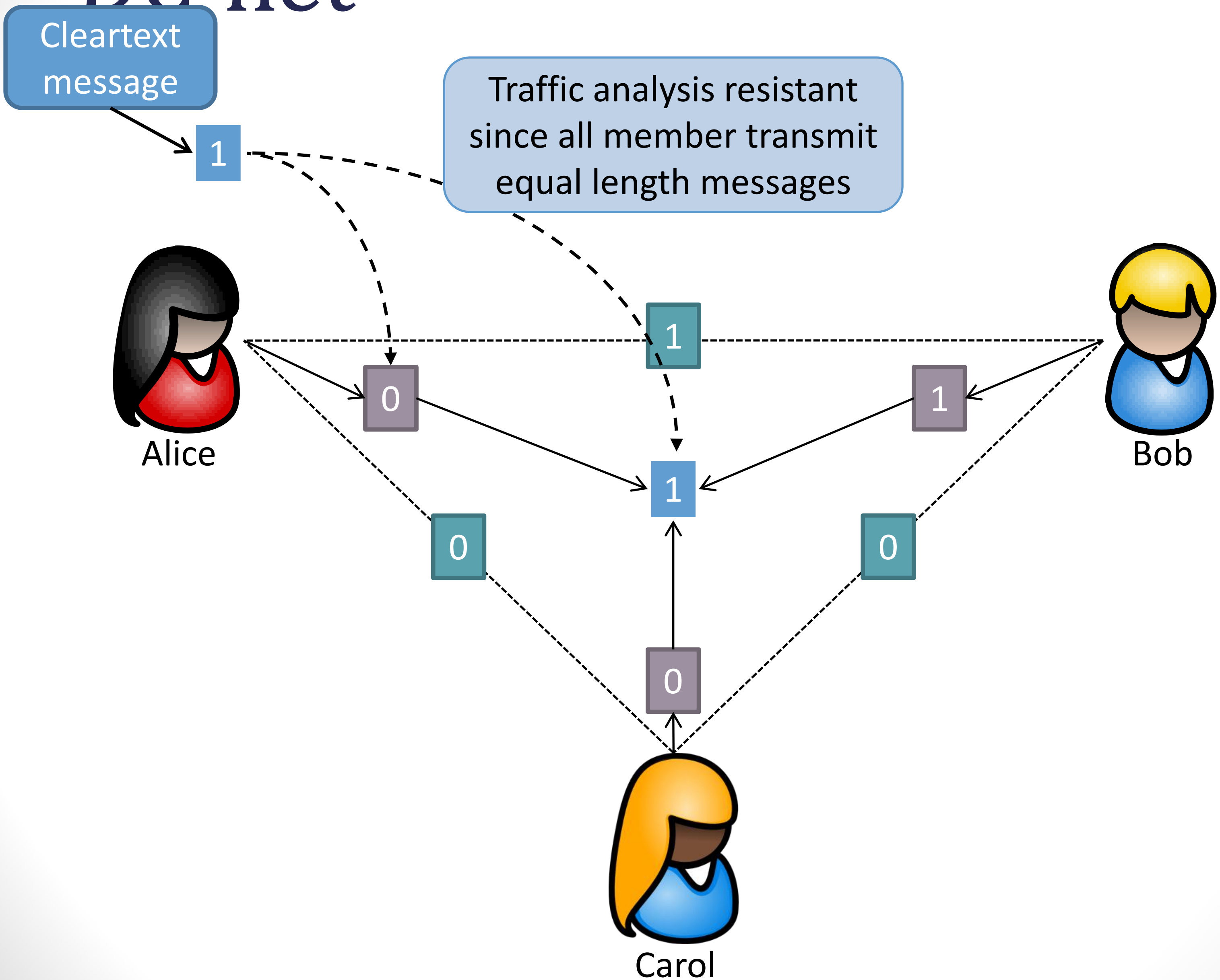
Cleartext message

1

Traffic analysis resistant since all member transmit equal length messages



DC-net



Cleartext message
1
Traffic analysis resistant since all member transmit equal length messages
Alice
Bob
Carol
1
0
1
1
0
0
0

Practical Considerations

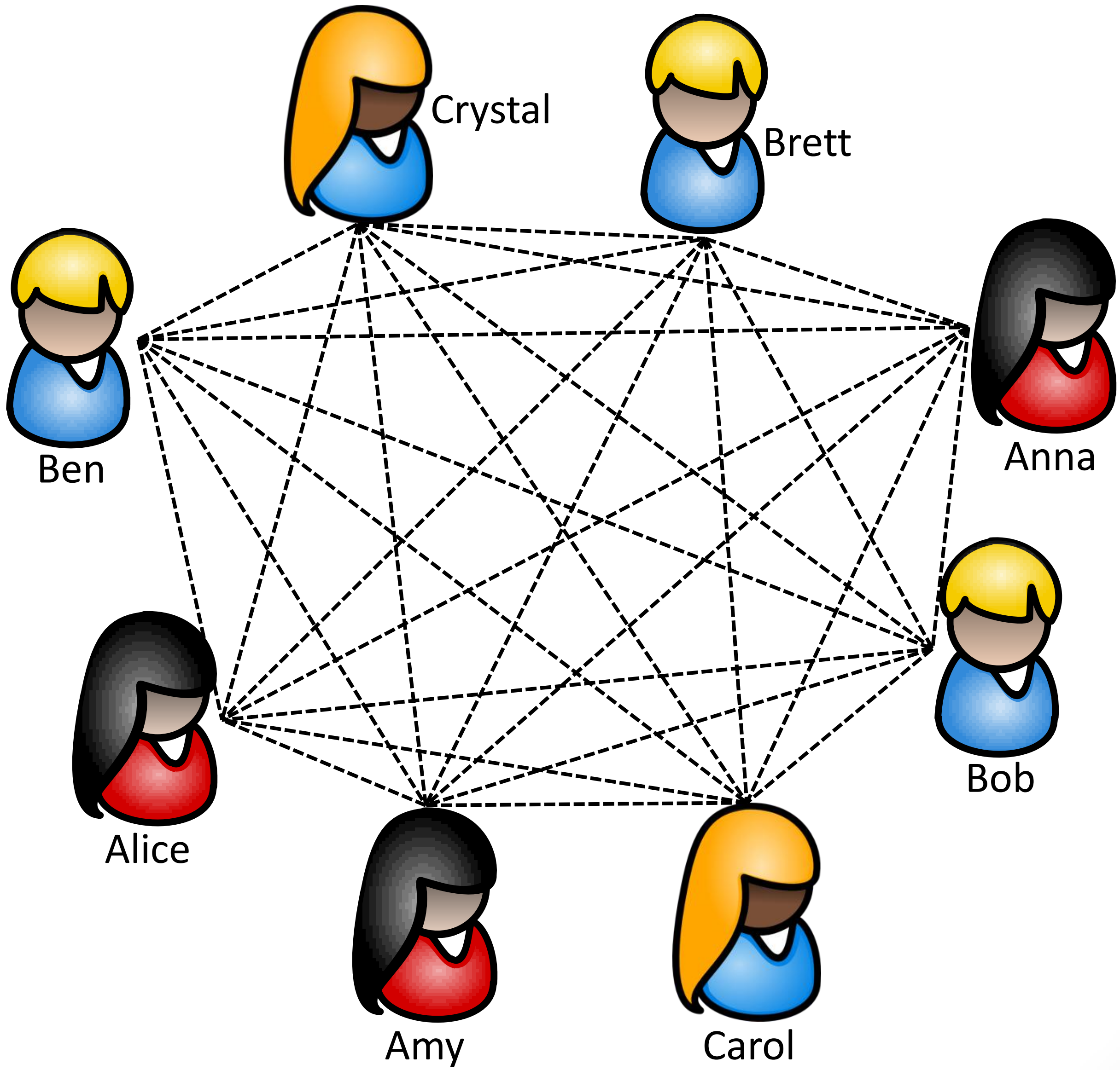
	Mix-nets	Tor	DC-nets
Strong anonymity	√		√
Scalability		√	√[1]
Churn tolerant	√	√	
Accountability			√[2]

- Mix-nets / Shuffles – Chaum, Neff, Wikstrom
- Onion Routing – Tor and I2P
- DC-nets – [1]Herbivore and [2]Dissent v1
  - Herbivore supported many concurrent users but distributed amongst many parallel DC-nets thus lacks the “Strength in Numbers” or large anonymity set sizes

Organization

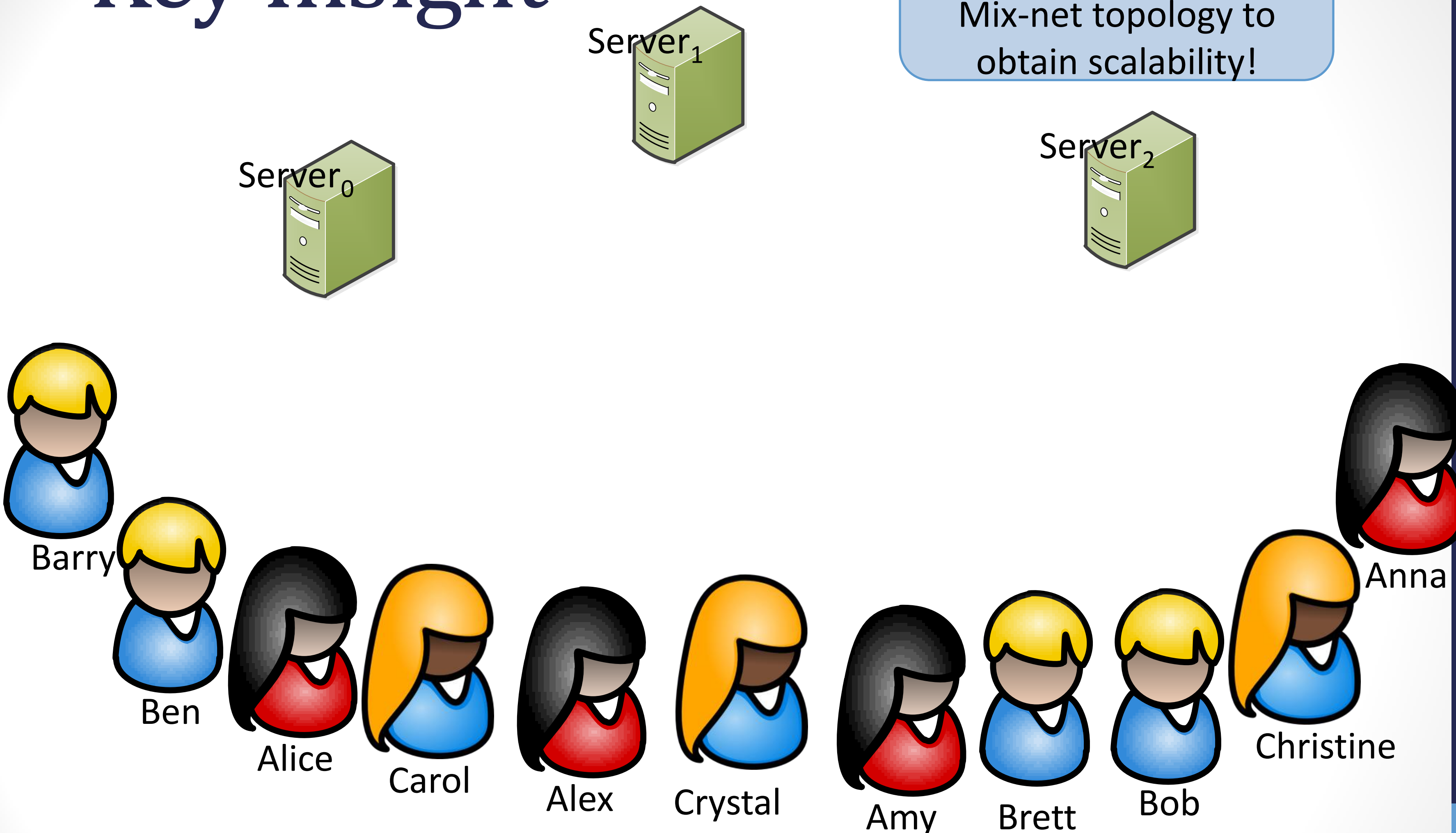
- Motivation
- Existing Approaches
- **Dissent – Strong, Scalable Anonymity**
  - Computational efficiency
  - Communication efficiency
  - Churn tolerant
  - Anonymity
  - Accountability
- Evaluation
- Conclusions

Key Insight



Key Insight

Use DC-net style anonymity within the Mix-net topology to obtain scalability!



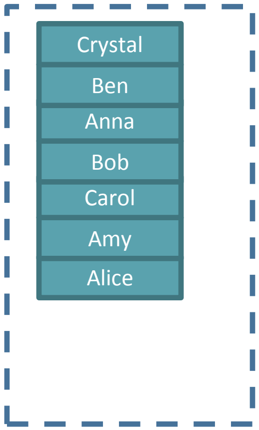
Making Strong Anonymity Scale!

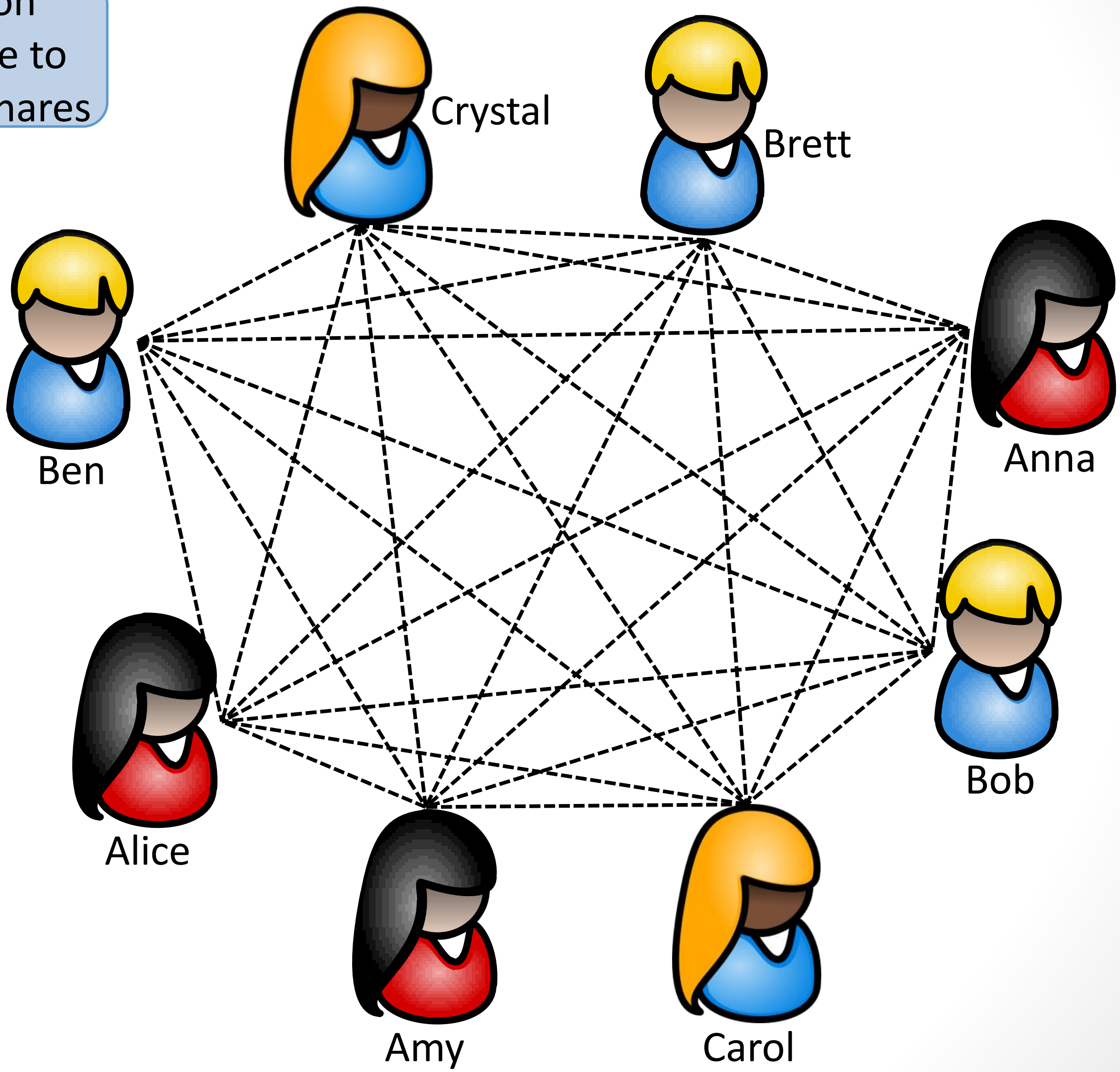
- Challenge – tradeoff between scale and strength in anonymity systems favoring scale
- Dissent's solution
  - Improving Computation Efficiency
  - Improving Communication Efficiency
  - Handling Churn
  - Identifying Disruptions
  - Maintaining Strong Anonymity

Improving Computational Efficiency

Computational Overhead

Computation overhead due to $O(N^2)$ secret shares

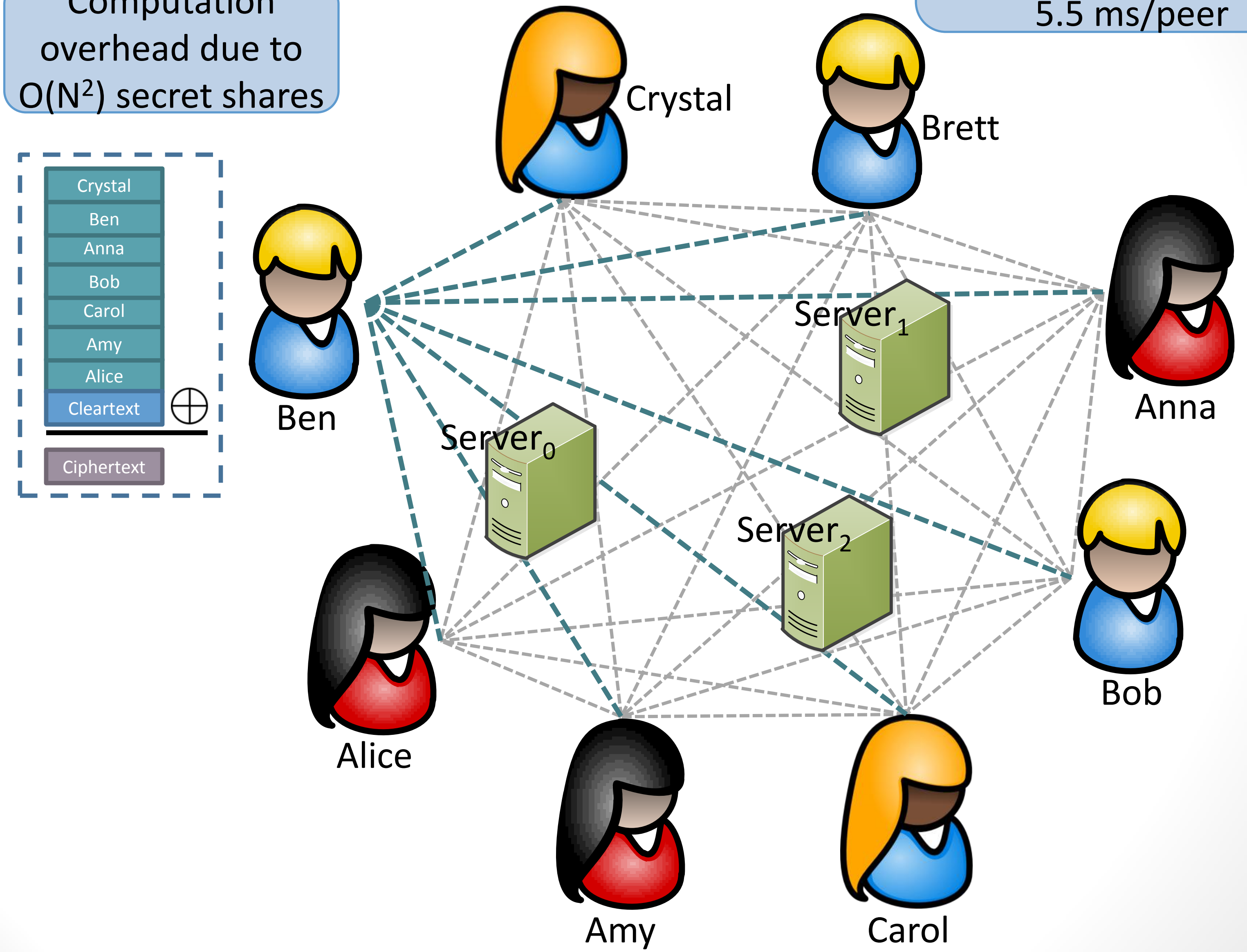




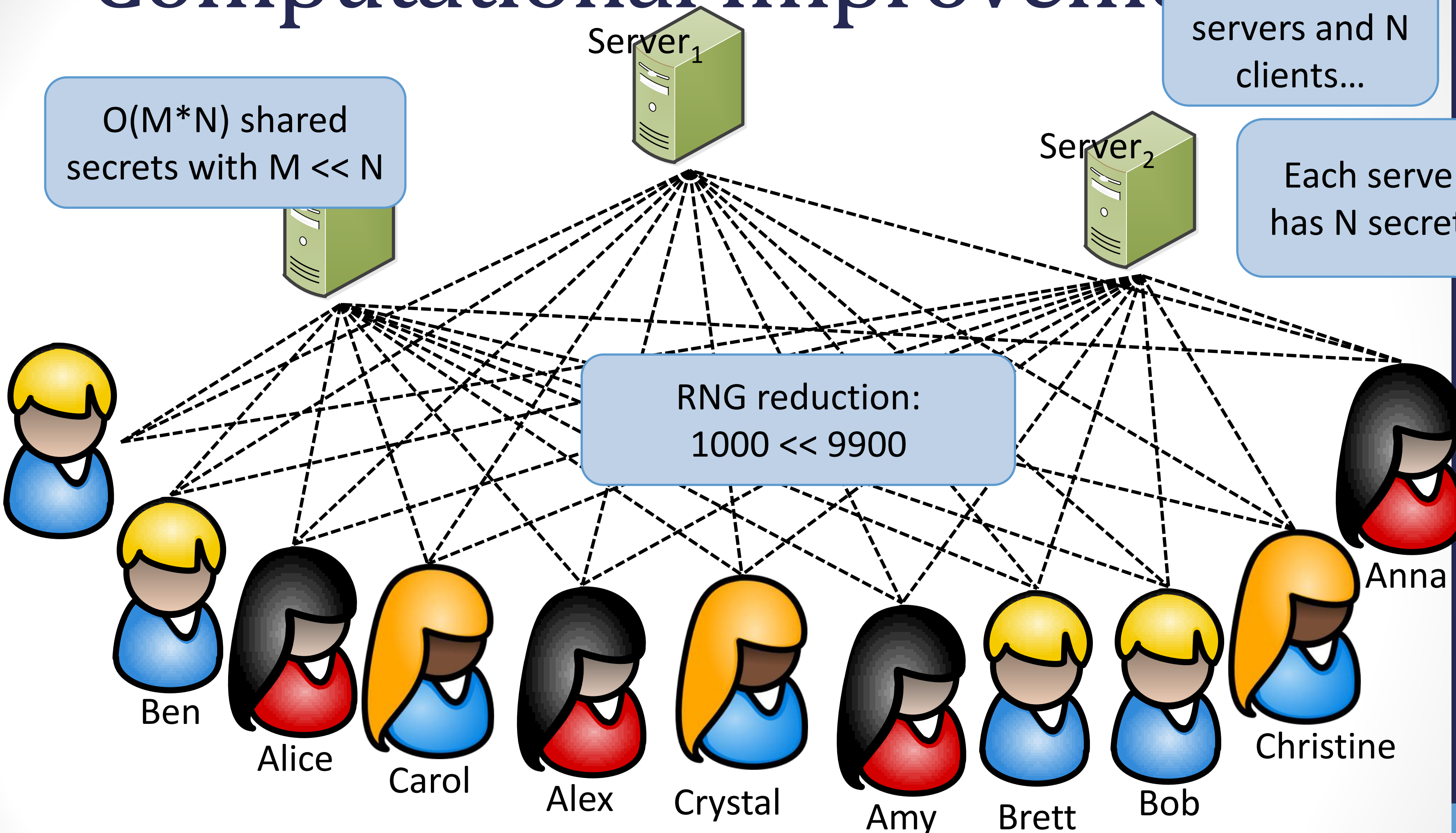
Computational Overhea

N = 100,
4950 shared secrets,
9900 RNG operations
5.5 ms/peer

Computation overhead due to $O(N^2)$ secret shares



Computational Improveme



With M servers and N clients…

O(M*N) shared secrets with M << N

Each server has N secrets

RNG reduction: 1000 << 9900

N = 100 and M = 5, 500 shared secrets, 1000 RNG operations

Each client has M secrets

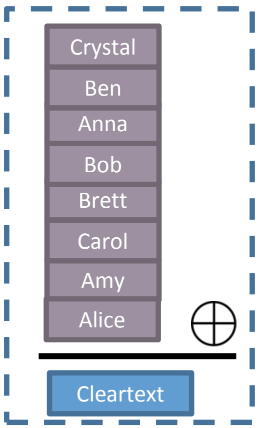
Improving Communication Efficiency

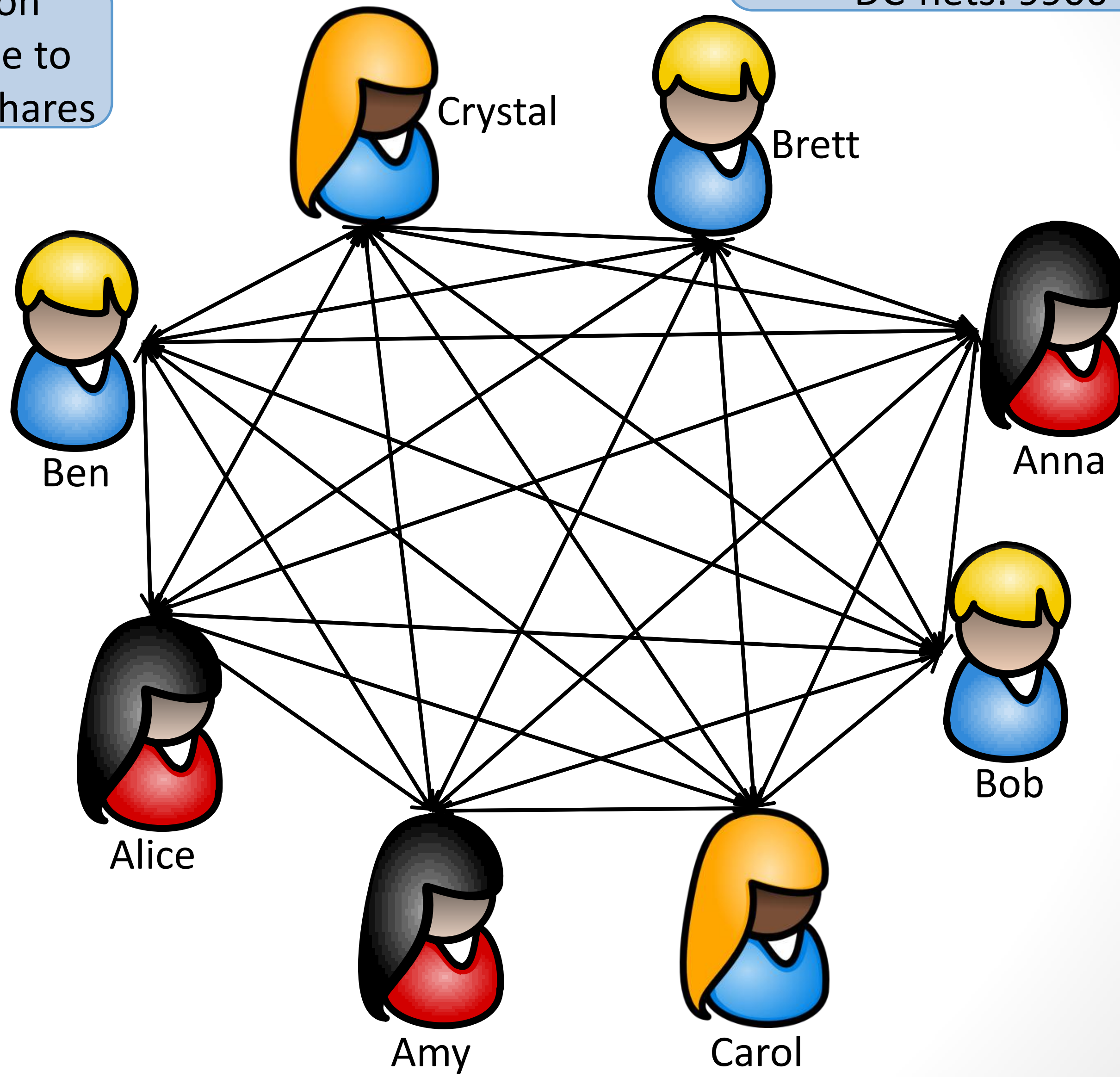
Bandwidth Overhead

N = 100,
Ciphertexts exchanged in DC-nets: 9900

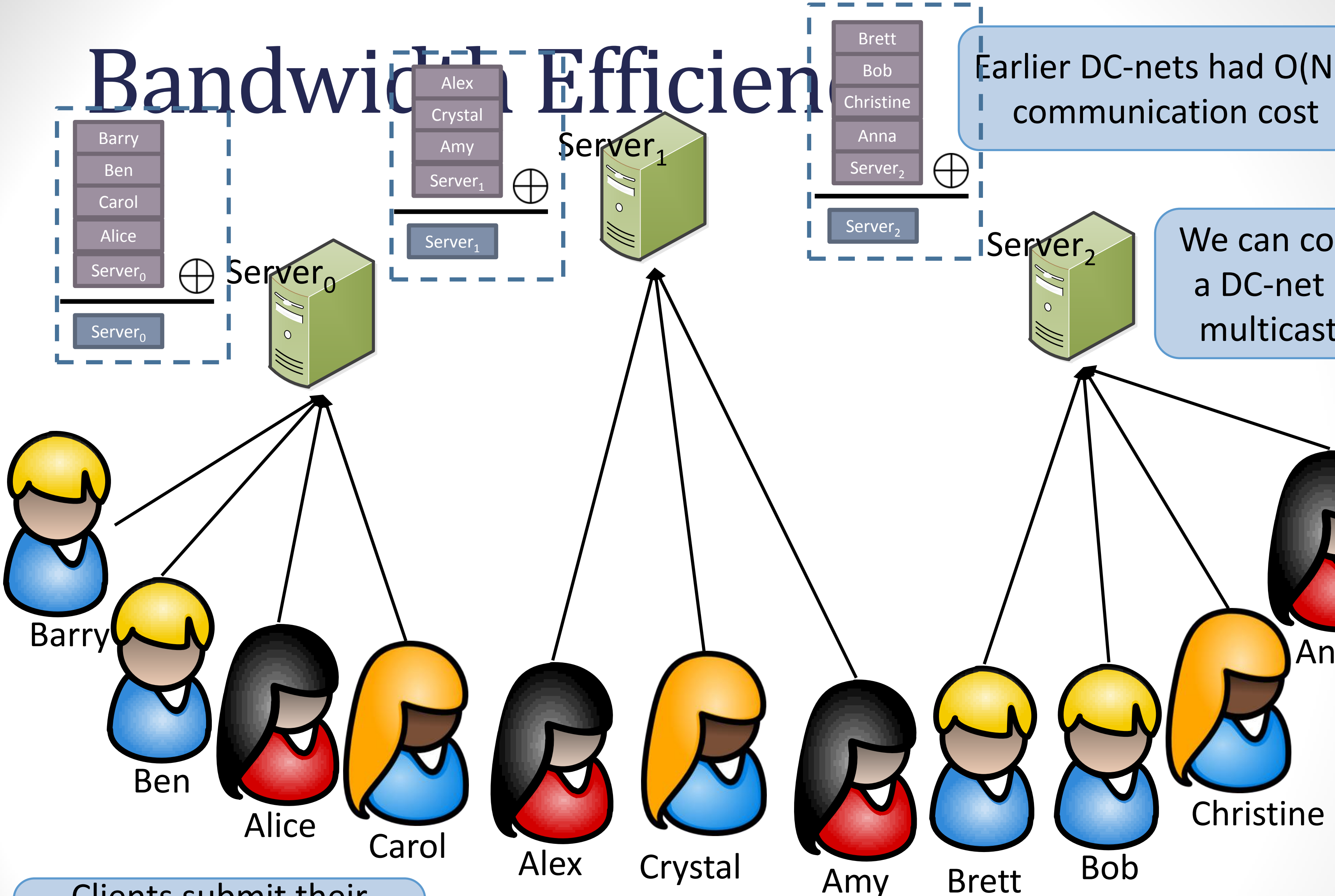
Computation overhead due to $O(N^2)$ secret shares

Bandwidth overhead due to $O(N^2)$ communication





Bandwidth Efficiency



Earlier DC-nets had $O(N^2)$ communication cost

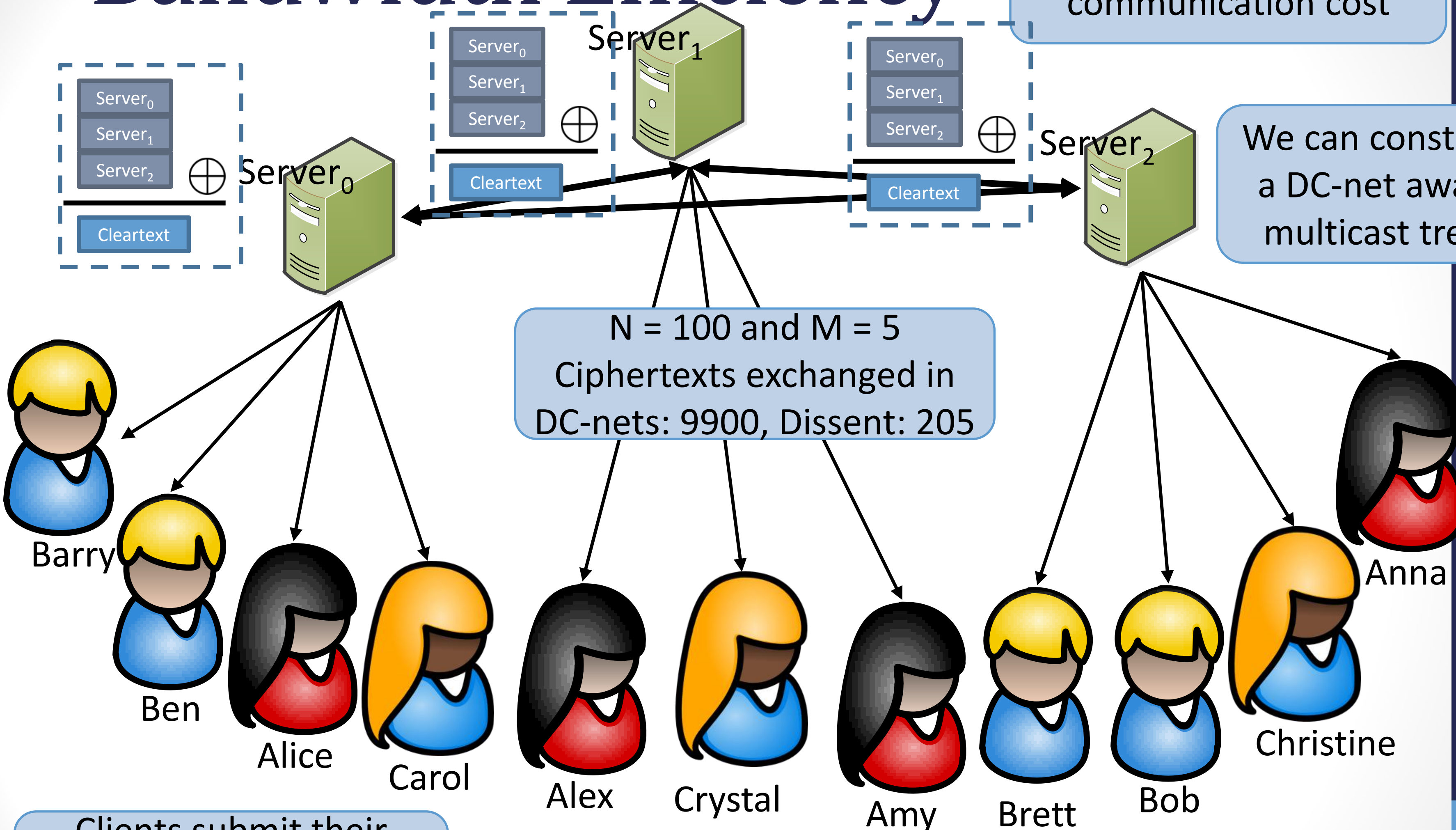
We can construct a DC-net aware multicast tree!

Clients submit their ciphertext upstream to one server

Bandwidth Efficiency

Earlier DC-nets had $O(N^2)$ communication cost

We can construct a DC-net aware multicast tree!



N = 100 and M = 5
Ciphertexts exchanged in DC-nets: 9900, Dissent: 205

Clients submit their ciphertext upstream to one server

Servers XOR these messages together and share with each other

Servers XOR these messages to compute the cleartext and distribute it to their downstream clients

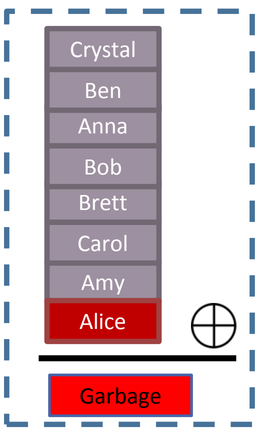
Creating Churn Tolerance

Churn Intolerance

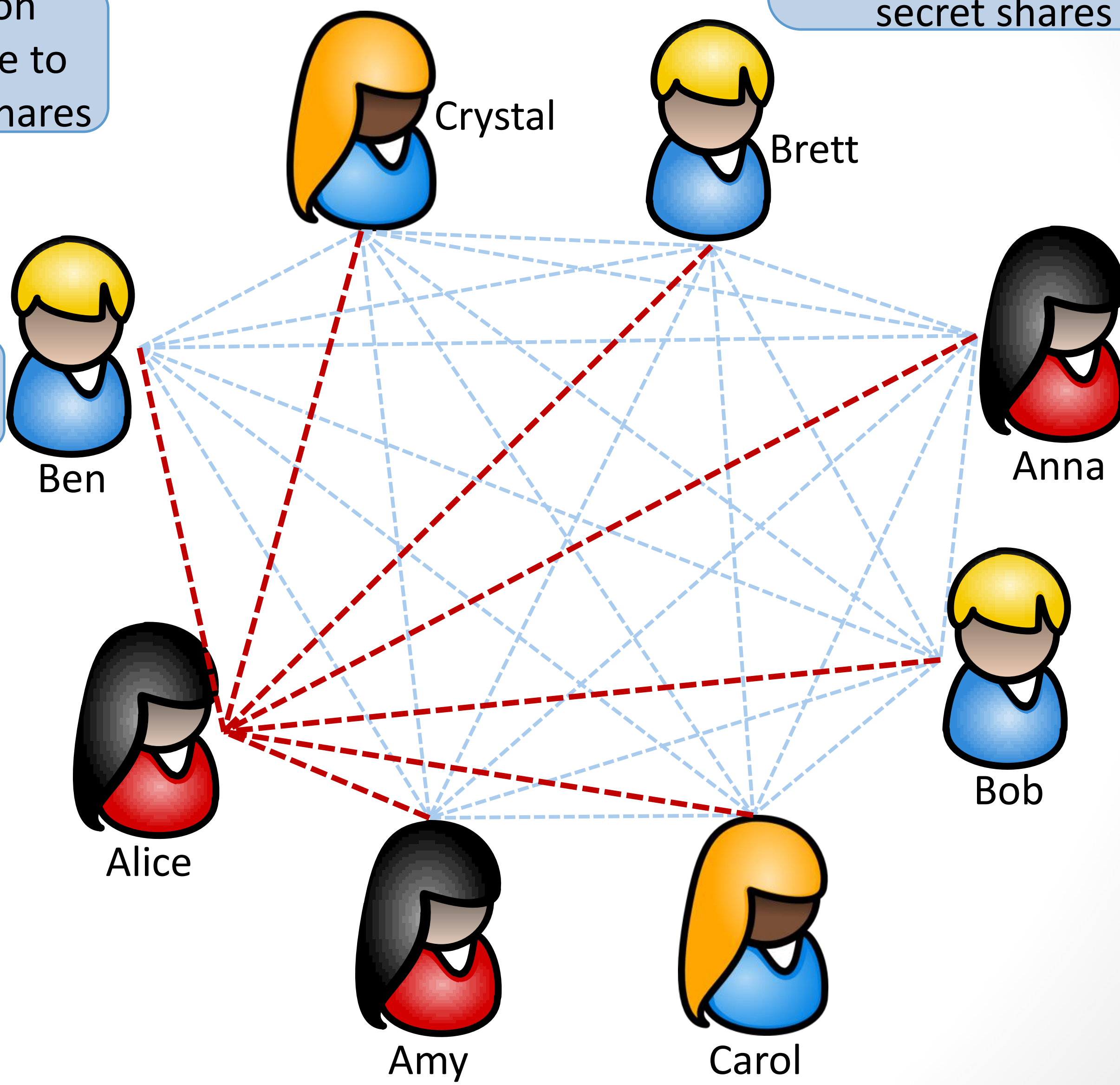
Computation overhead due to $O(N^2)$ secret shares

Bandwidth overhead due to $O(N^2)$ communication

What if Alice left without transferring?



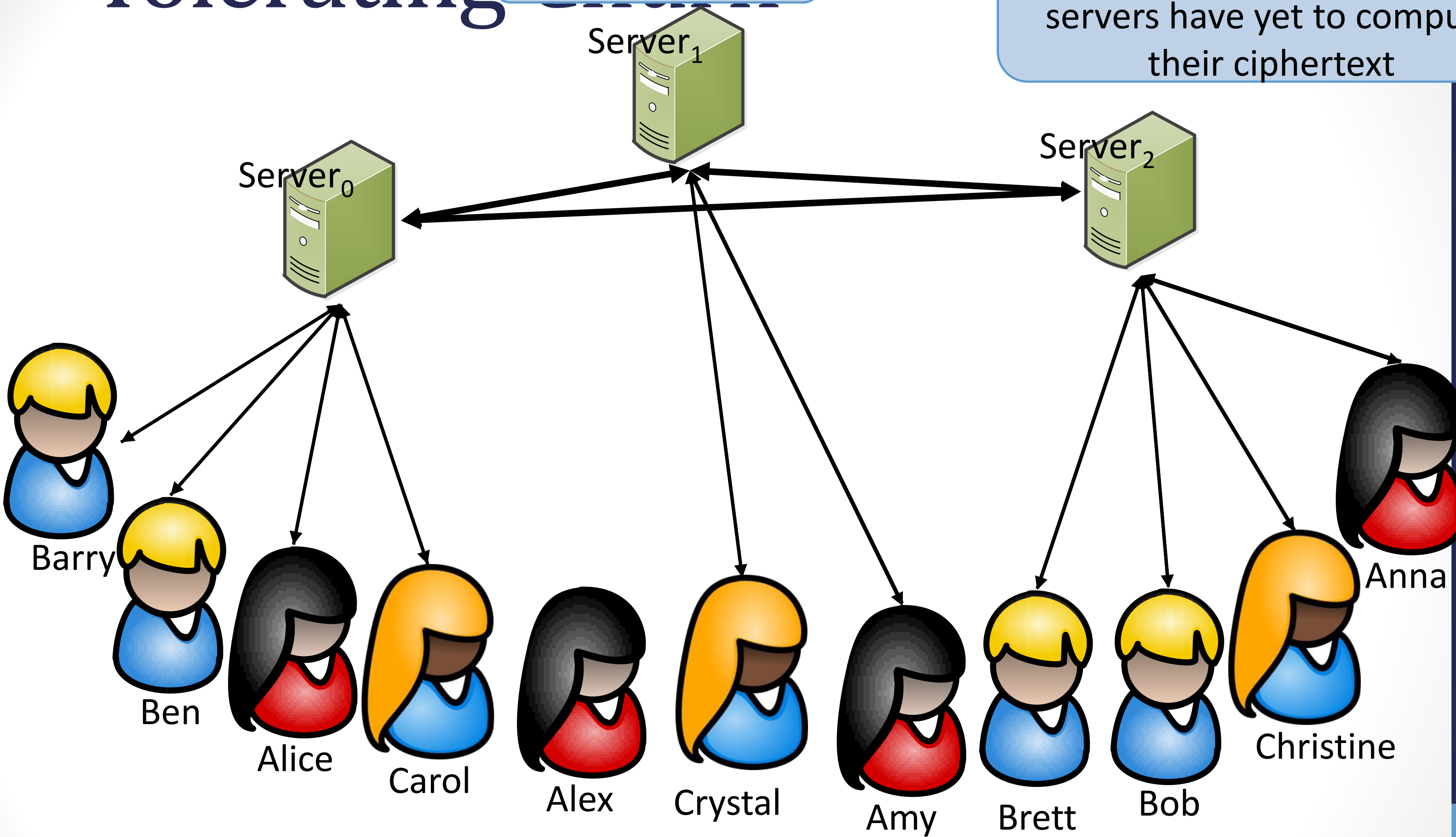
The resulting cleartext is garbage due to the dependency on Alice's secret shares



Tolerating

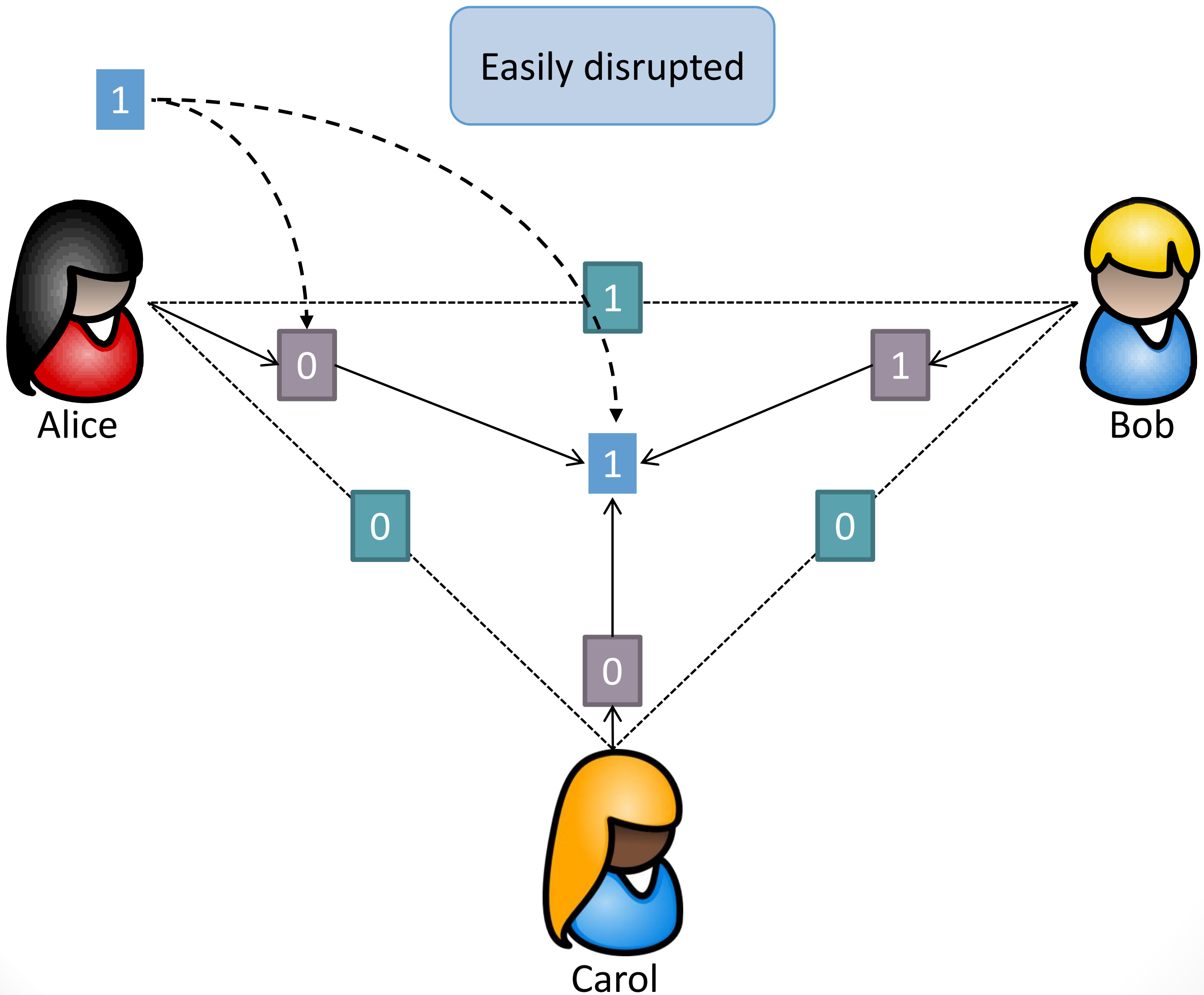
Server$_1$ will timeout on Alex

The protocol continues uninterrupted, since the servers have yet to compute their ciphertext



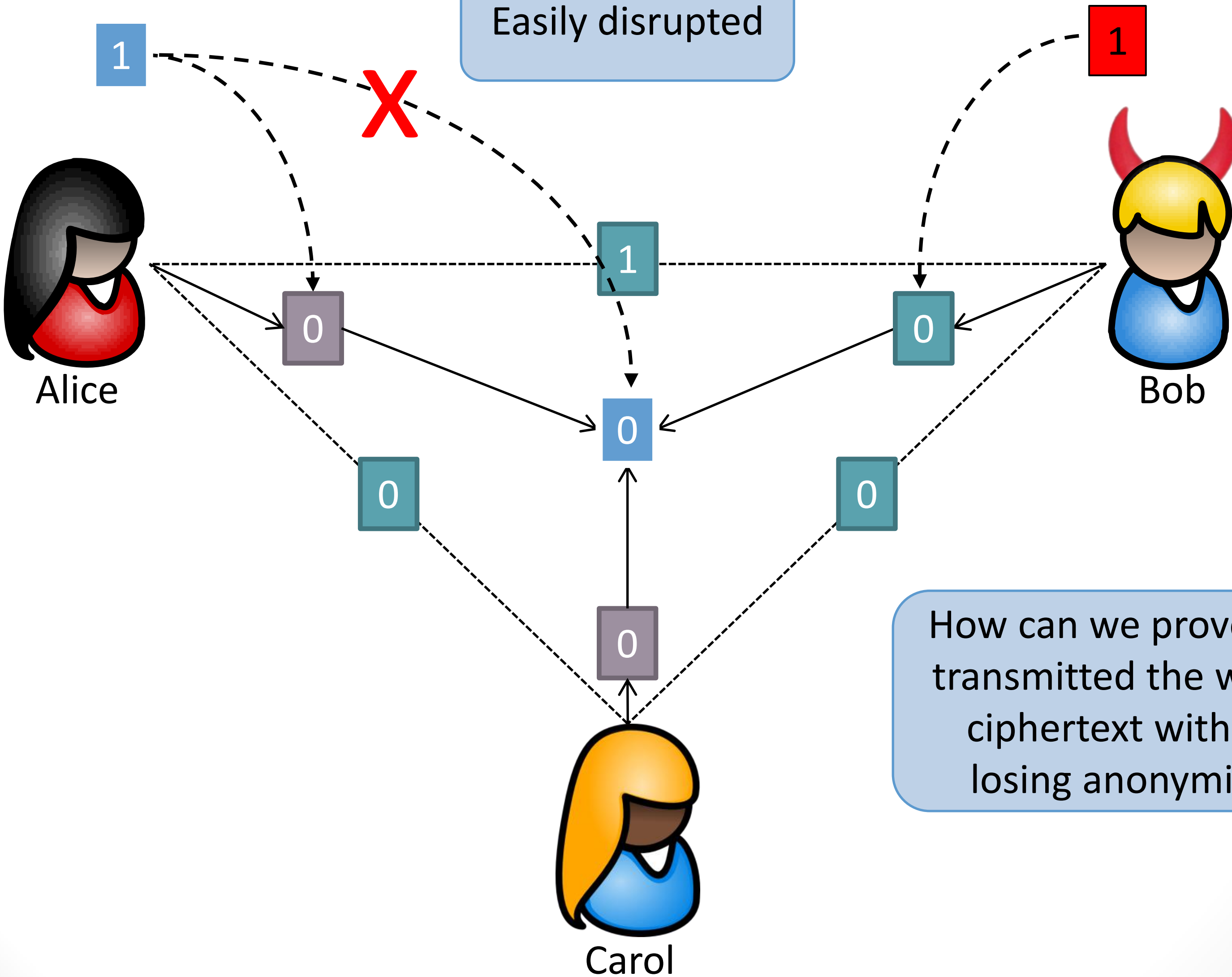
Handling Disruptions via Accountability…

DC-net



DC-net – Disruptions

Easily disrupted



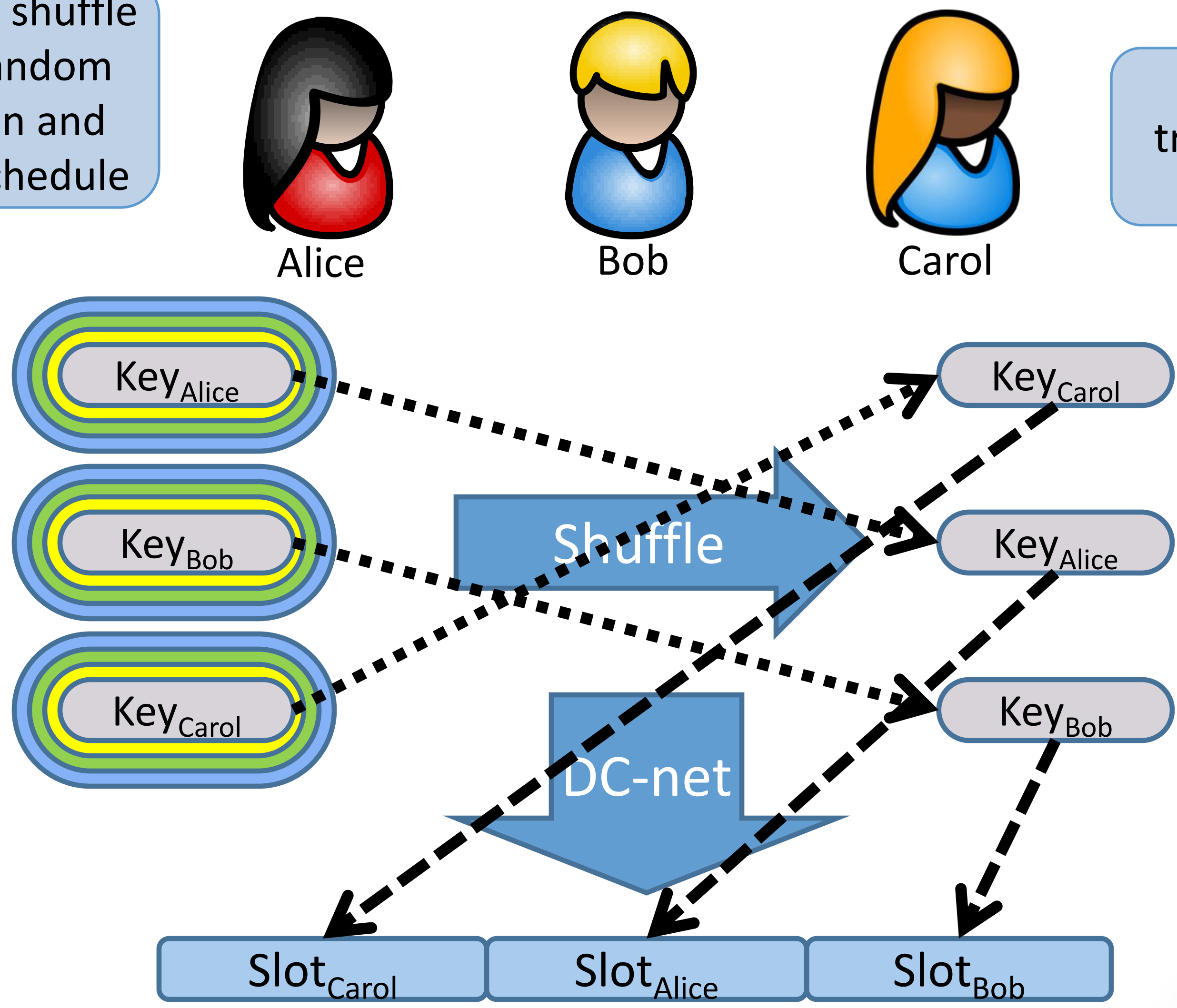
How can we prove Bob transmitted the wrong ciphertext without losing anonymity?

Scheduling

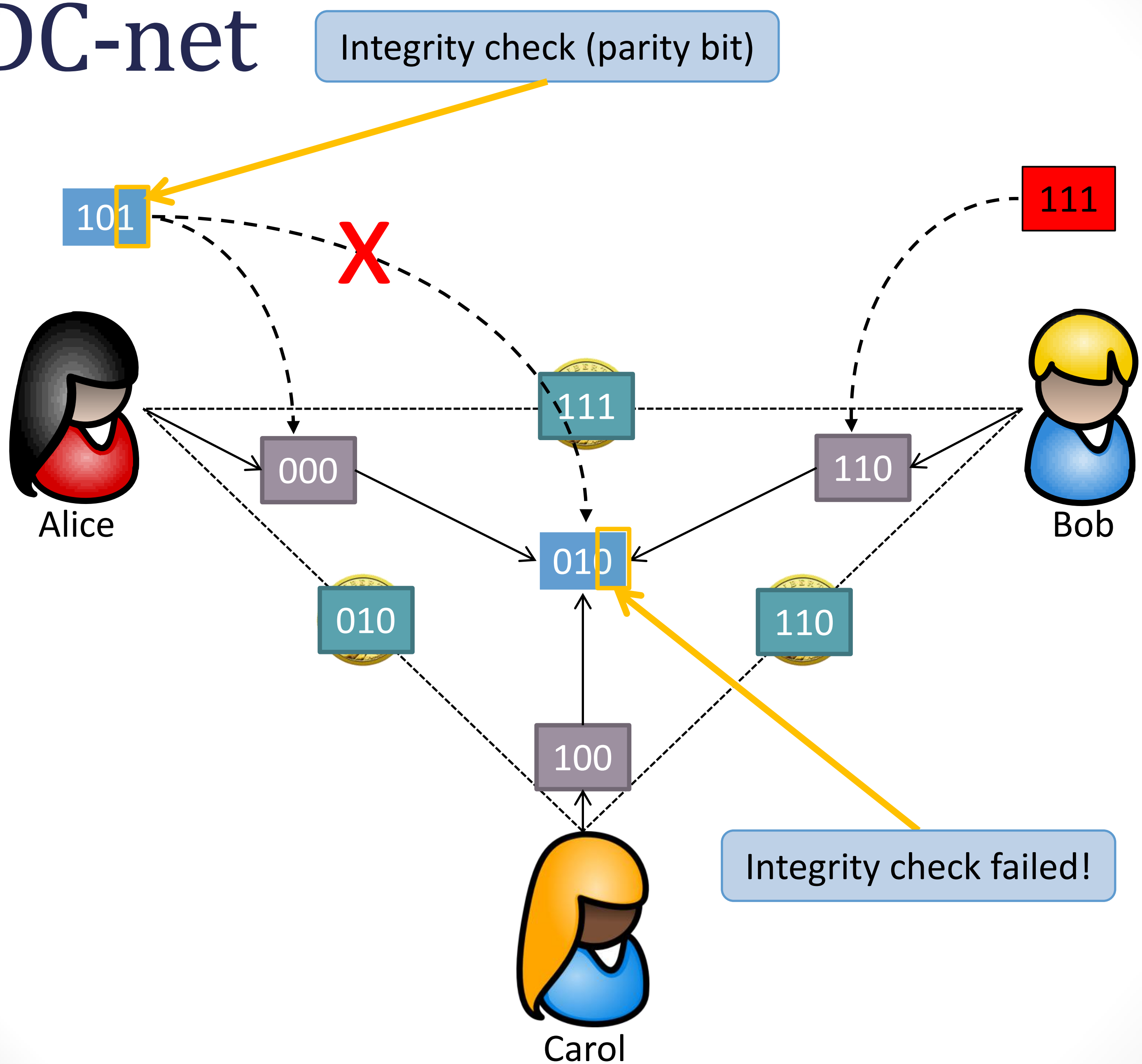
How do many members share the DC-net without disrupting each other?

Anonymizing shuffle produces random permutation and hence the schedule

Create a transmission schedule!

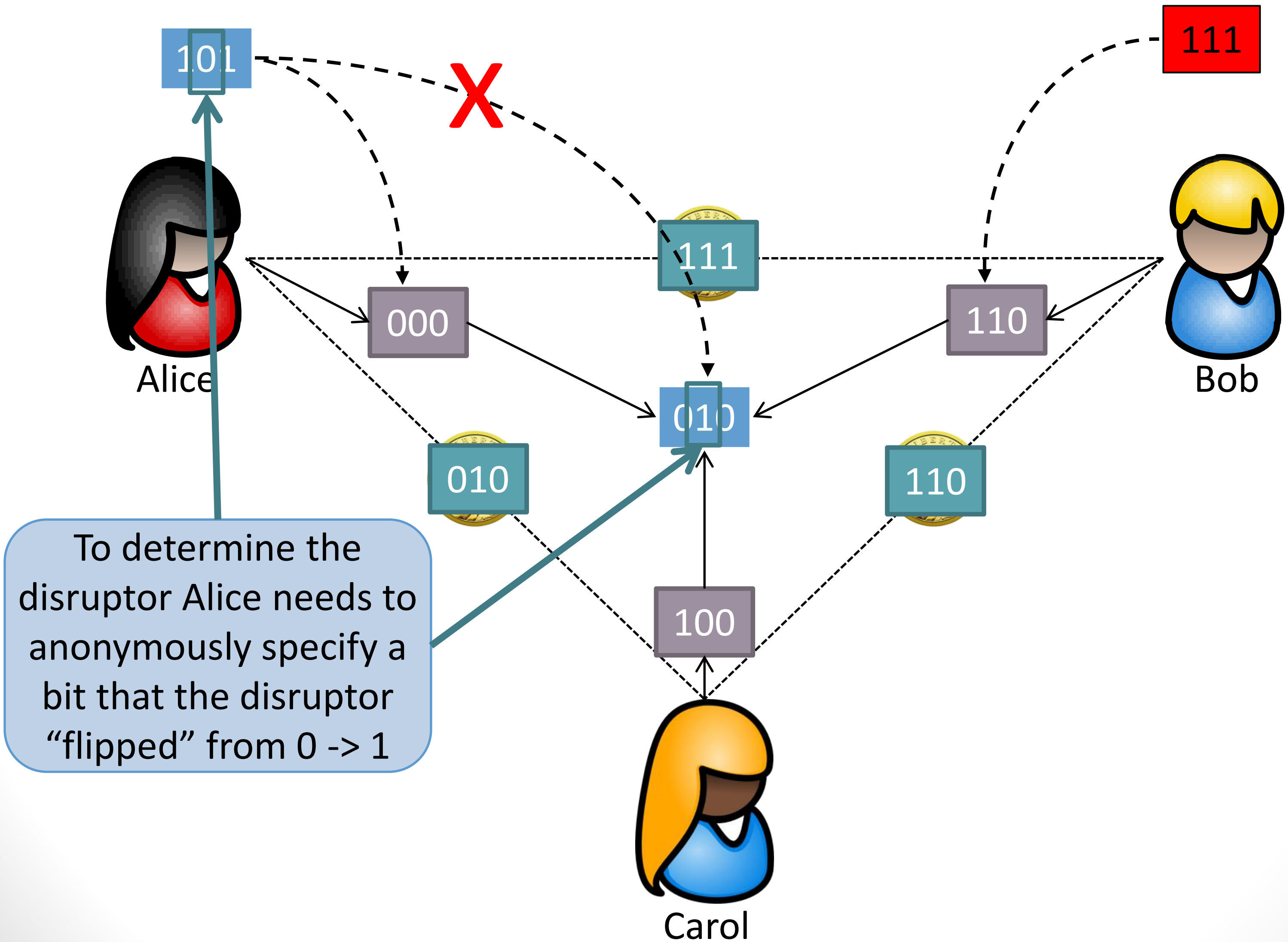


DC-net



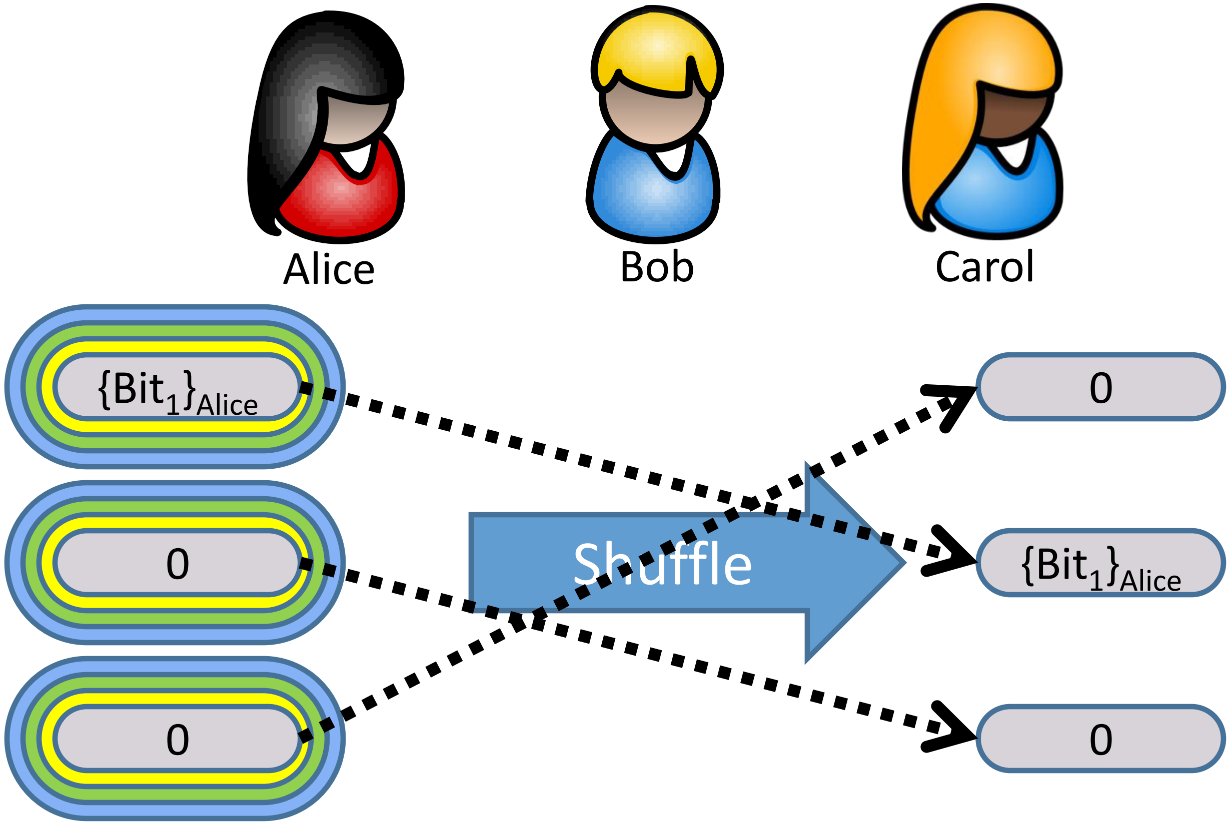
Integrity check (parity bit)
101
111
X
Alice
Bob
Carol
000
111
110
010
010
110
100
Integrity check failed!

DC-net



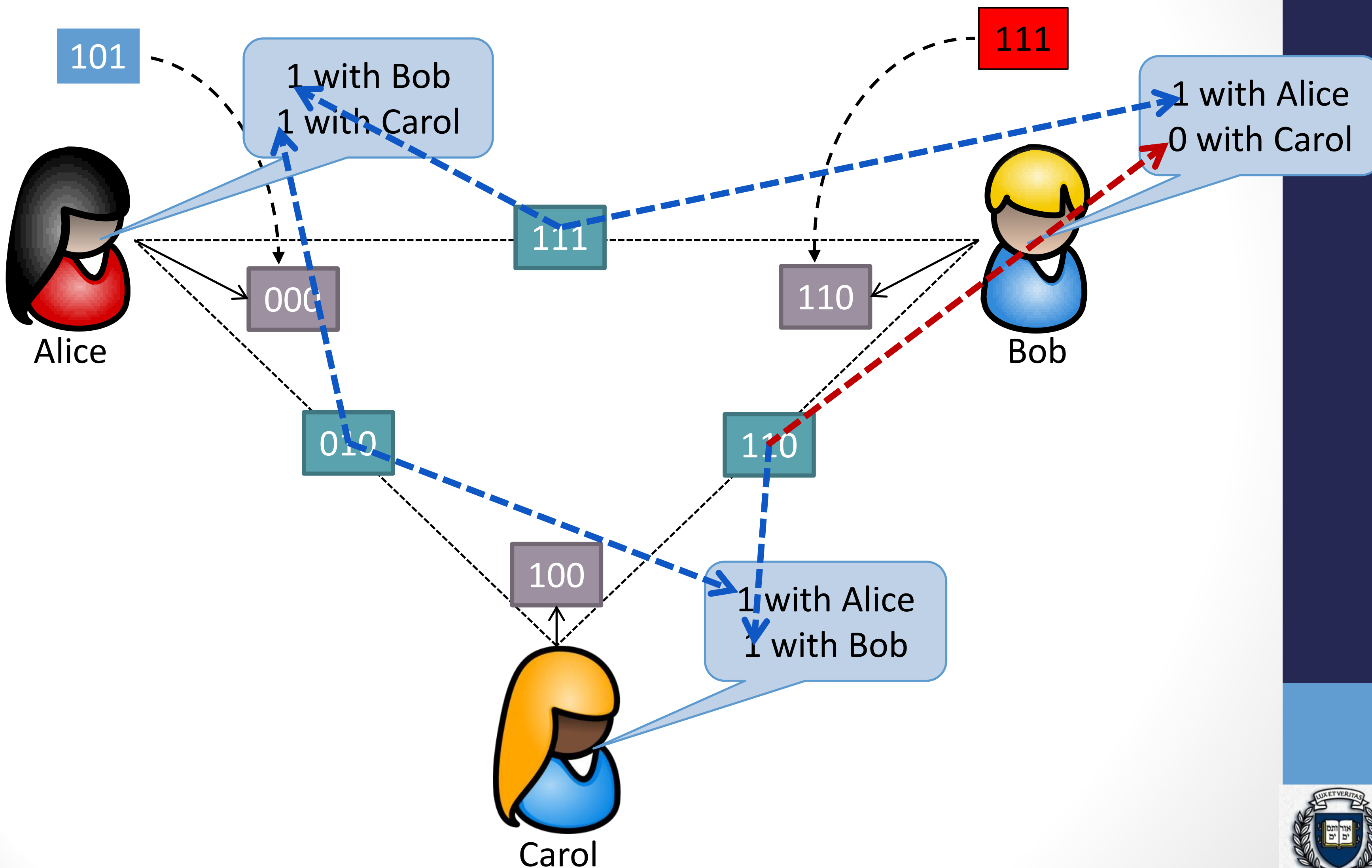
101
111
X
Alice
Bob
000
111
110
010
010
110
100
Carol
To determine the disruptor Alice needs to anonymously specify a bit that the disruptor “flipped” from 0 -> 1

Safely Deanonymize a Bit



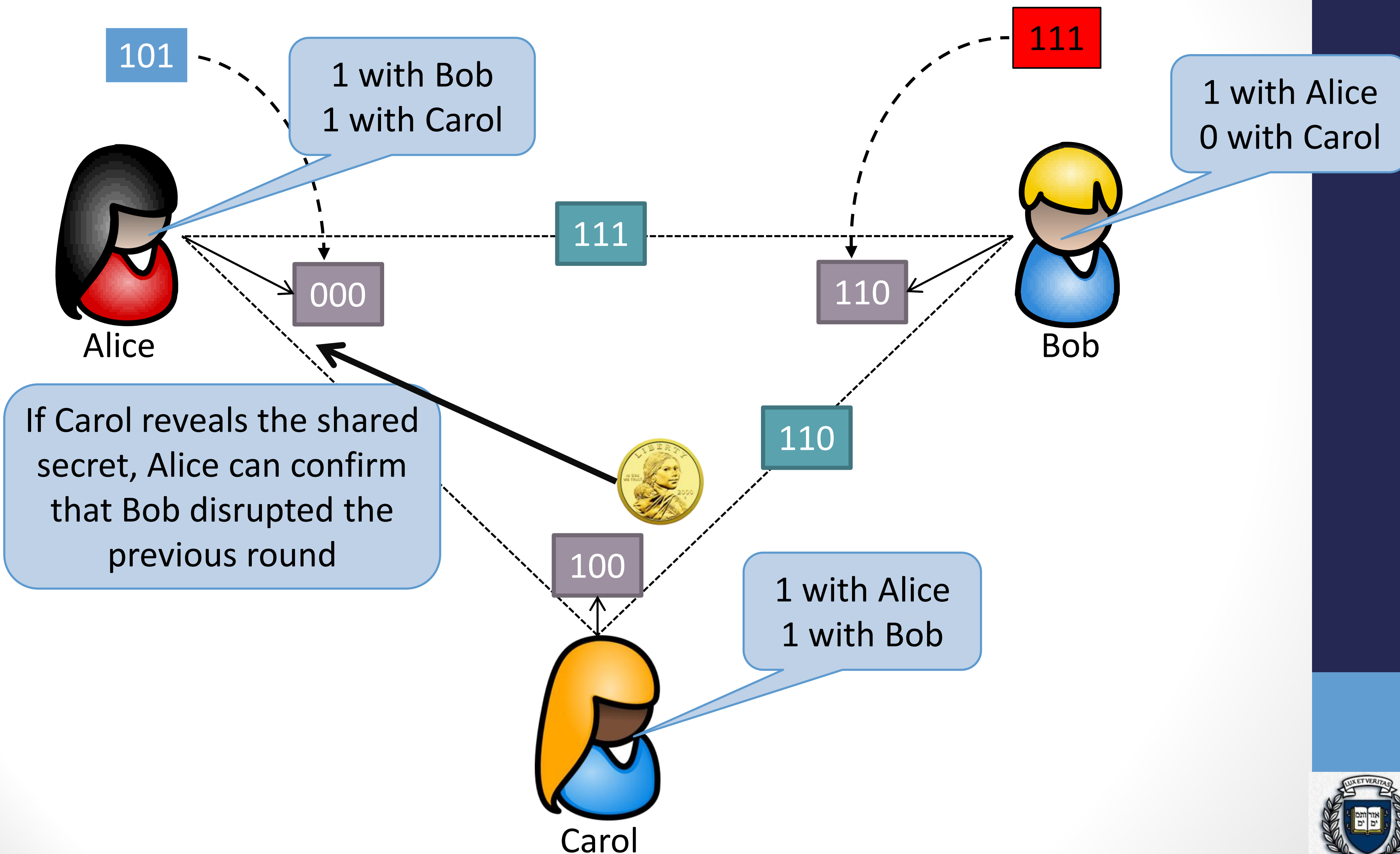
DC-net

In practice, this is a bit more complicated though the details are in the paper.



DC-net

In practice, this is a bit more complicated though the details are in the paper.

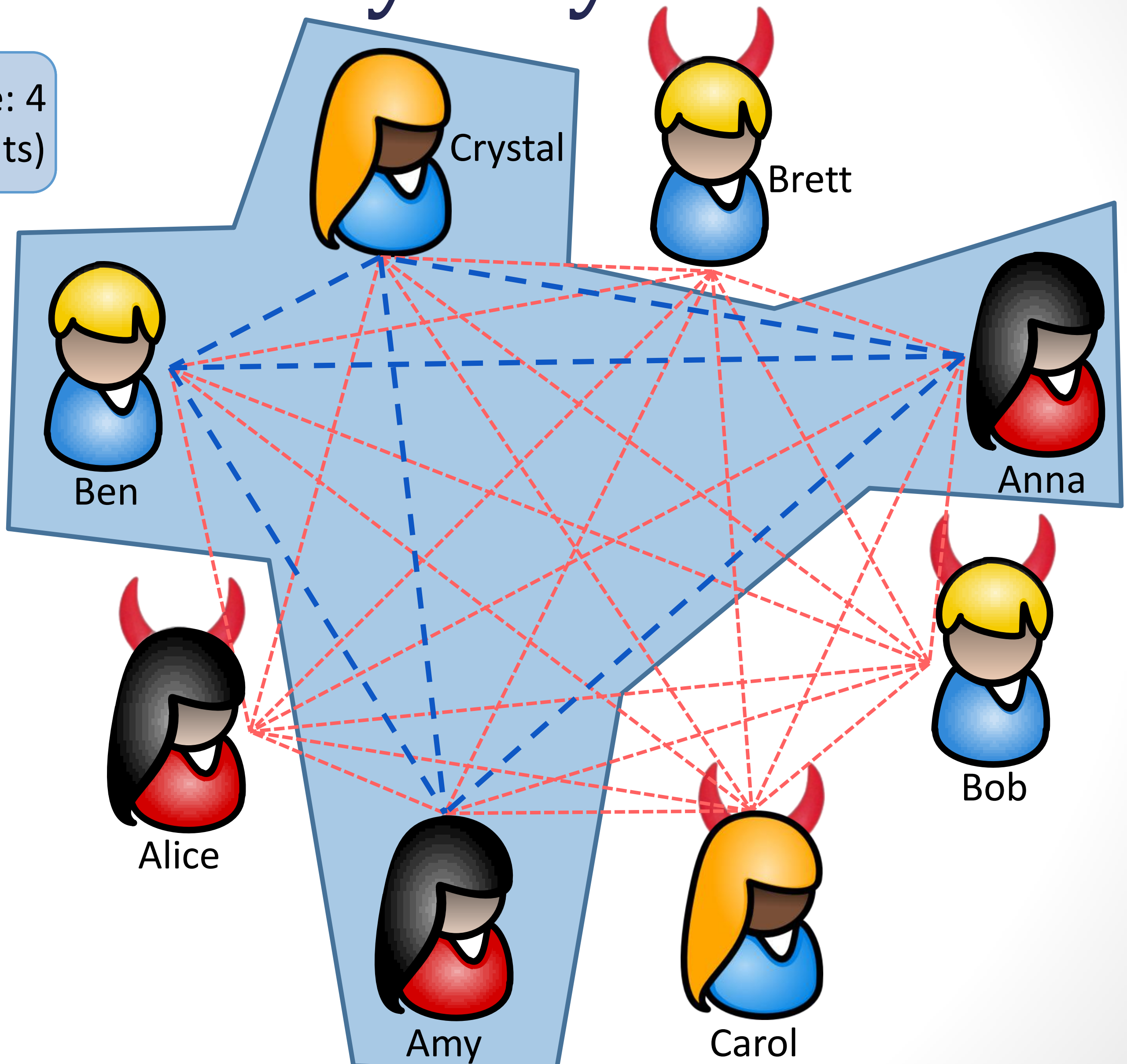


Progress!

- We have gained
  - Improvements in computation and communication
  - Ability to tolerate churn
  - Identify disruptors
- How does this impact strong anonymity?

DC-net – Anonymity Set

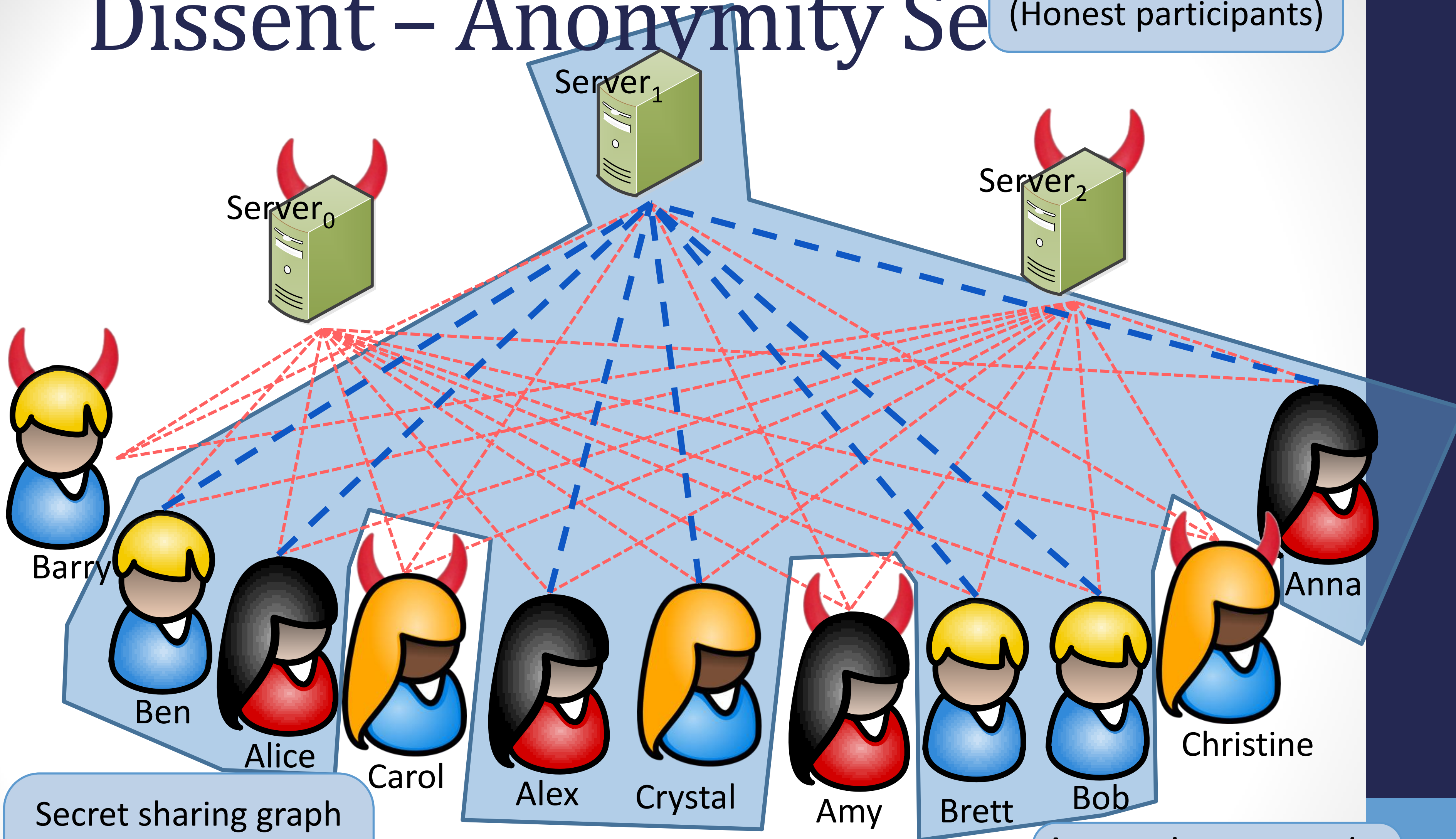
Anonymity set size: 4
(Honest participants)



Dissent retains this feature…

Dissent – Anonymity Se

Anonymity set size: 7
(Honest participants)



Secret sharing graph prevents the clients upstream server from deanonymizing it

Anonymity set remains equal as long as there is 1 honest server

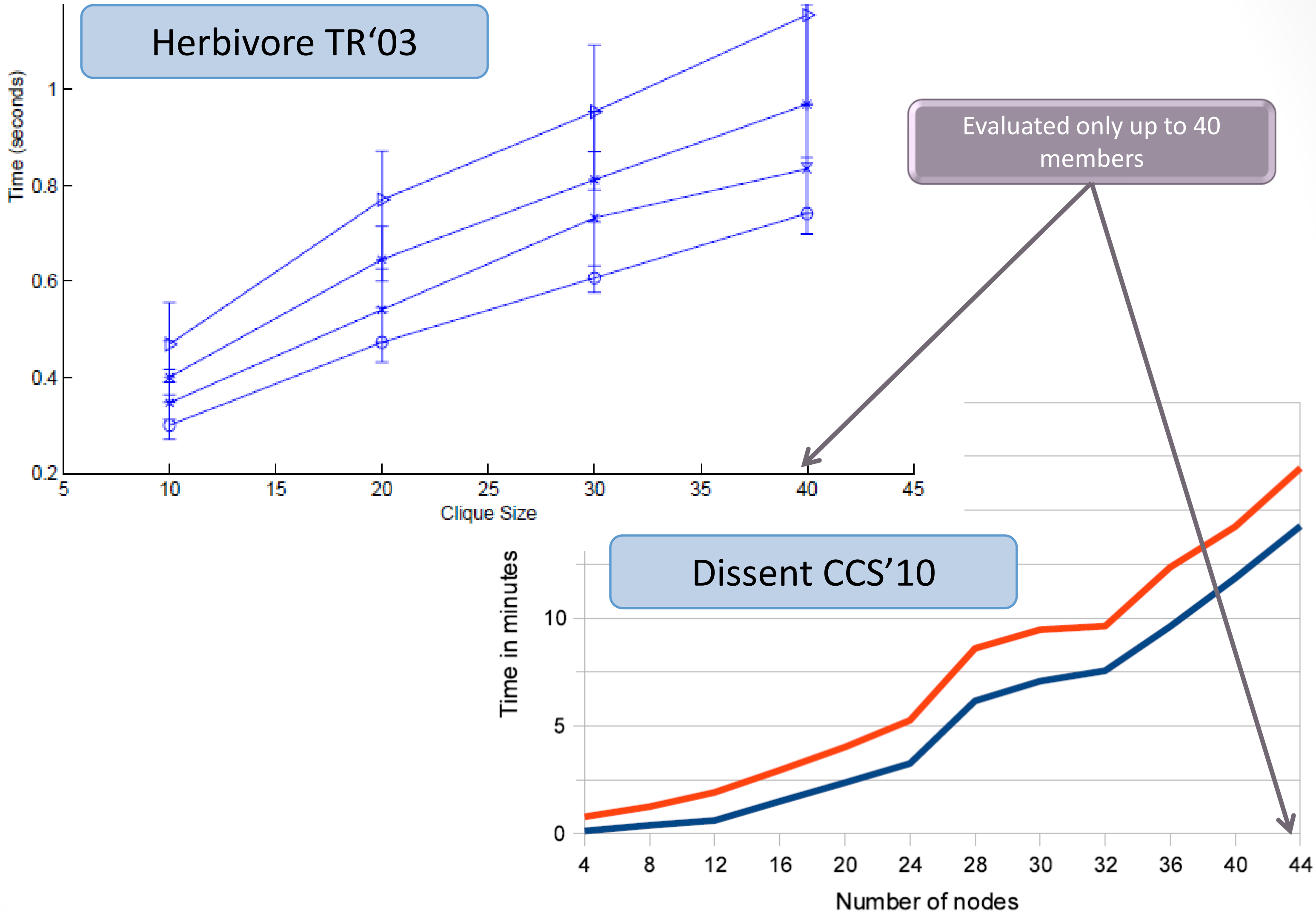
Organization

- Motivation
- Existing Approaches
- Dissent – Strong, Scalable Anonymity
  - Computational efficiency
  - Communication efficiency
  - Churn resistance
  - Anonymity
  - Accountability
- **Evaluation**
- Conclusions

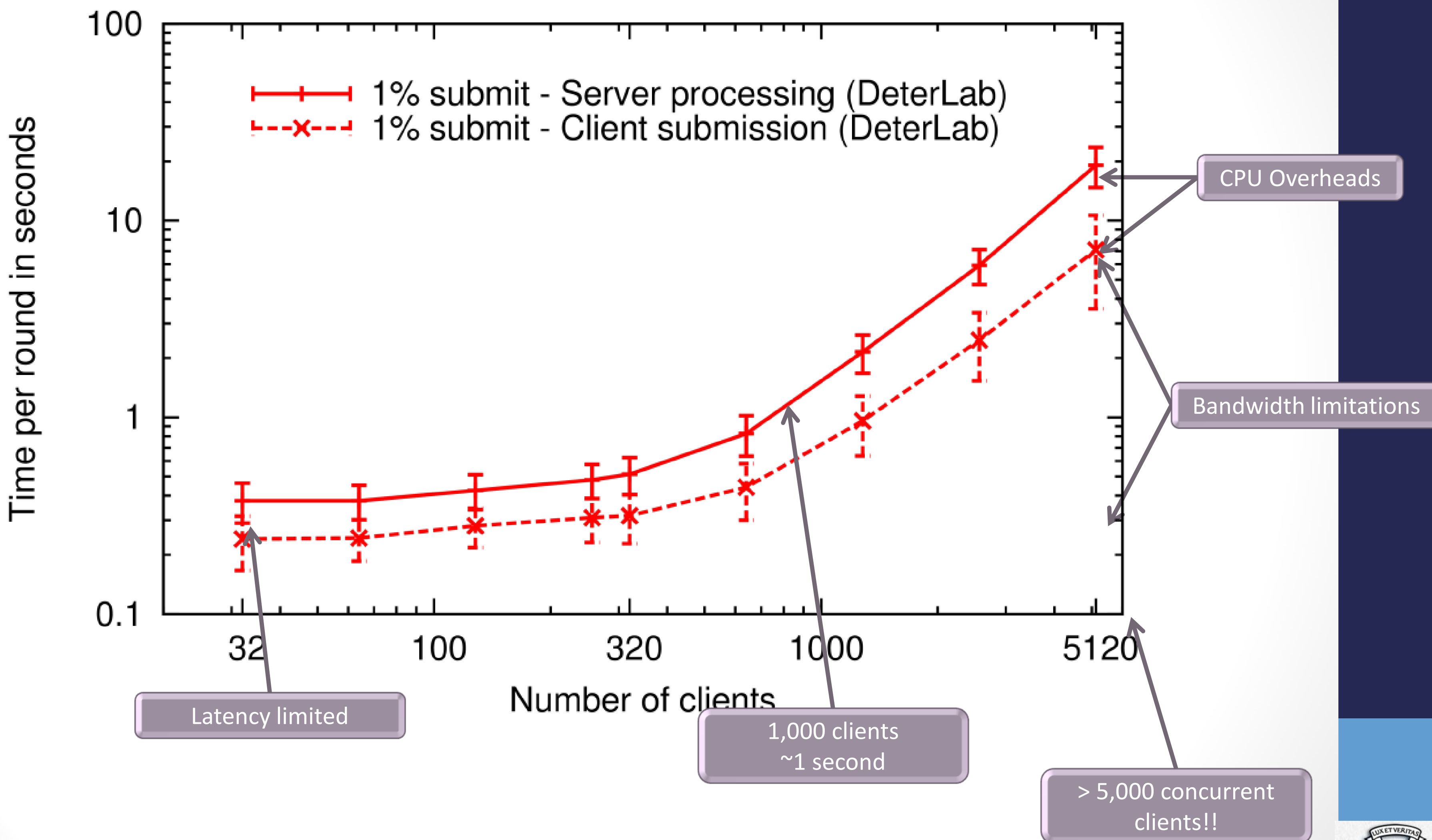
Dissent – Prototype

- Written in C++
  - Qt from networking, serialization, and events processing
  - Crypto++ as the crypto library

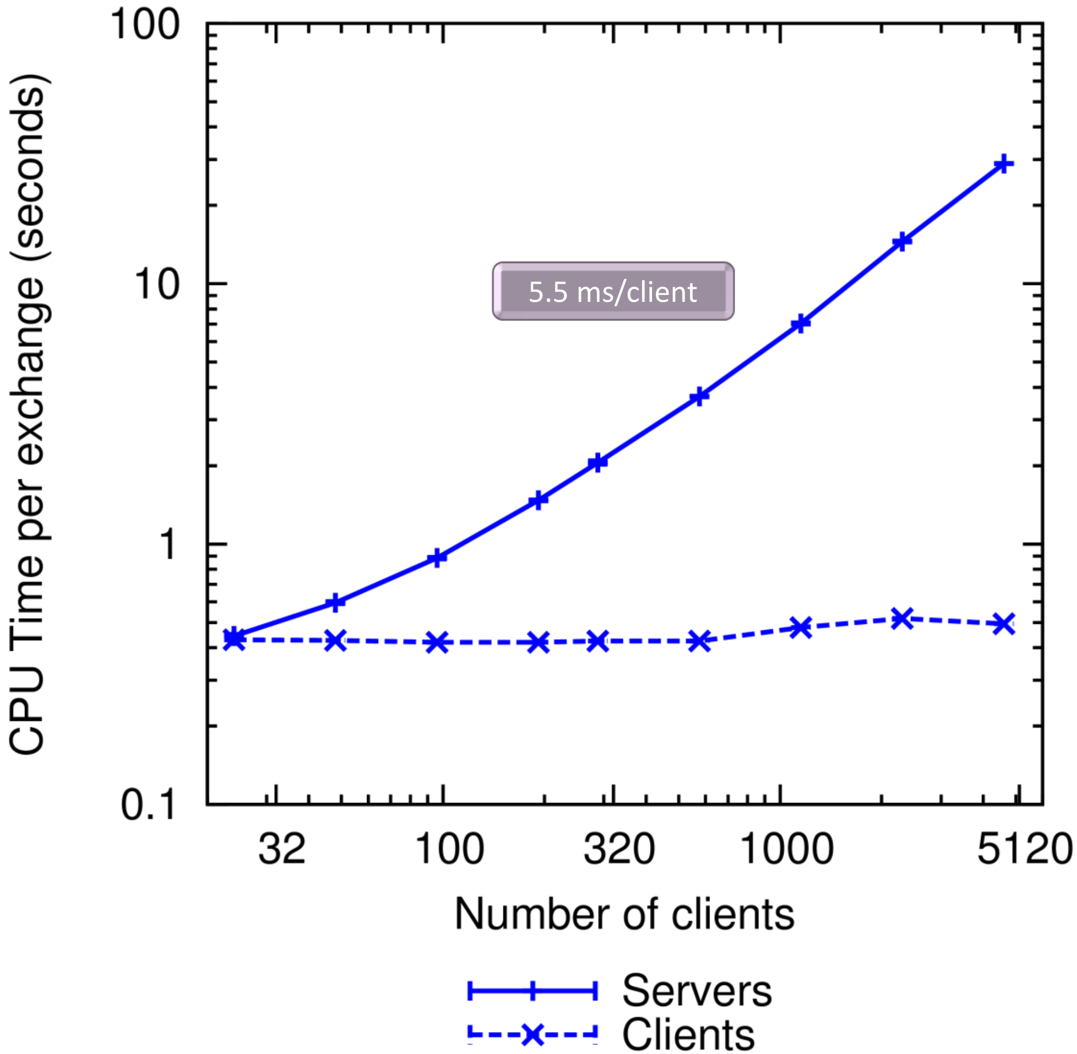
Related Work



Scaling to Thousands of Clients

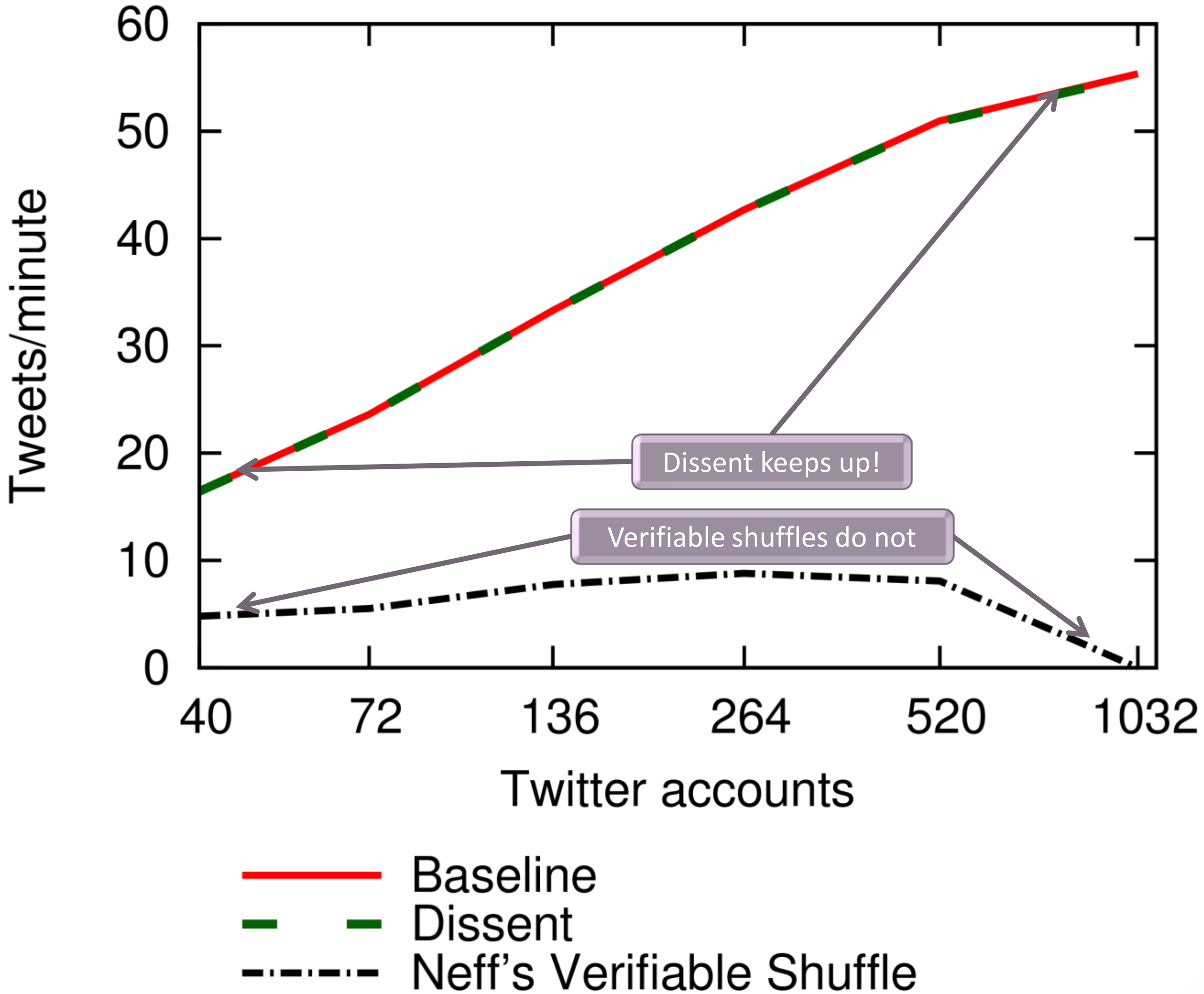


CPU Time

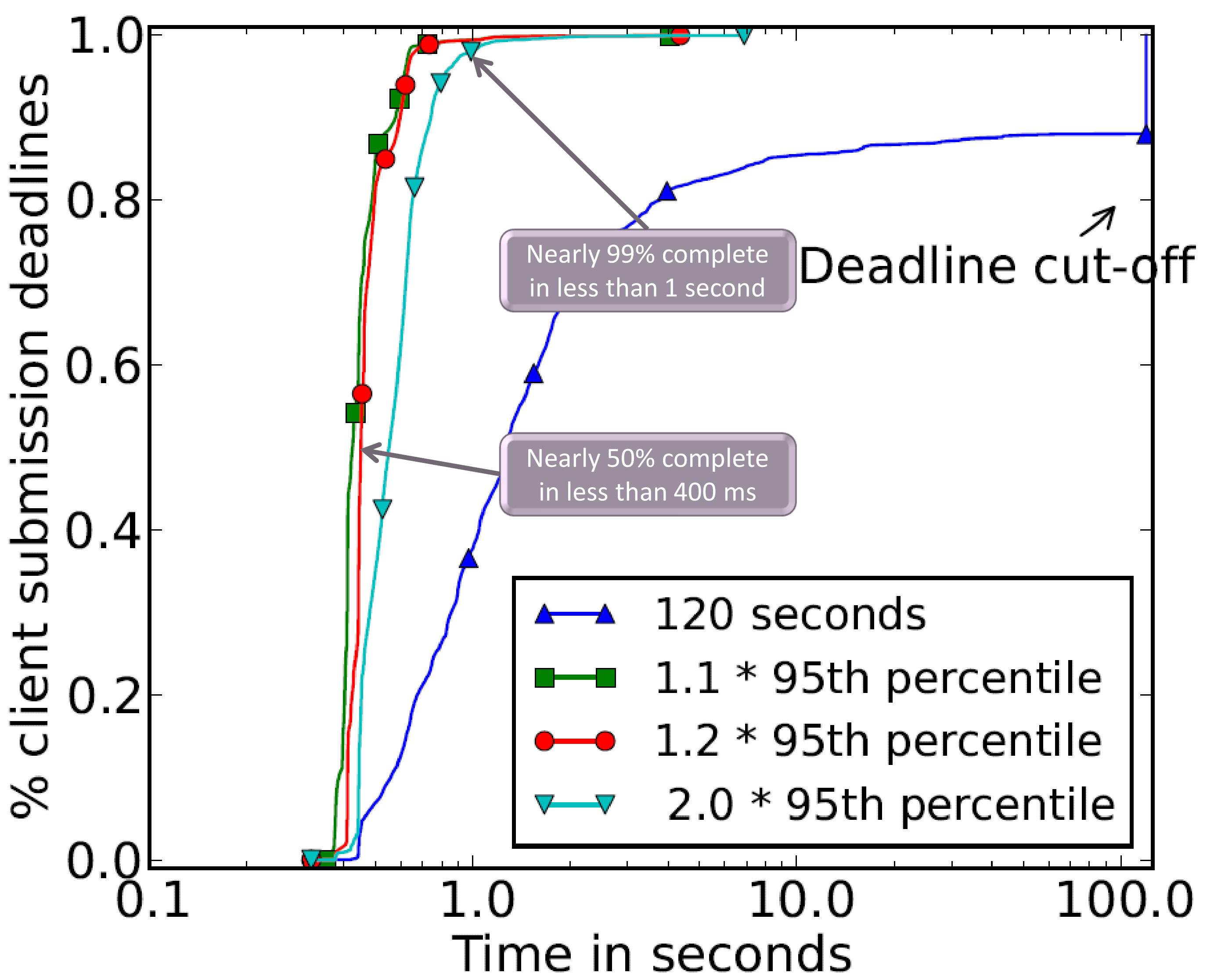


100
10
1
0.1
CPU Time per exchange (seconds)
5.5 ms/client
32
100
320
1000
5120
Number of clients
Servers
Clients

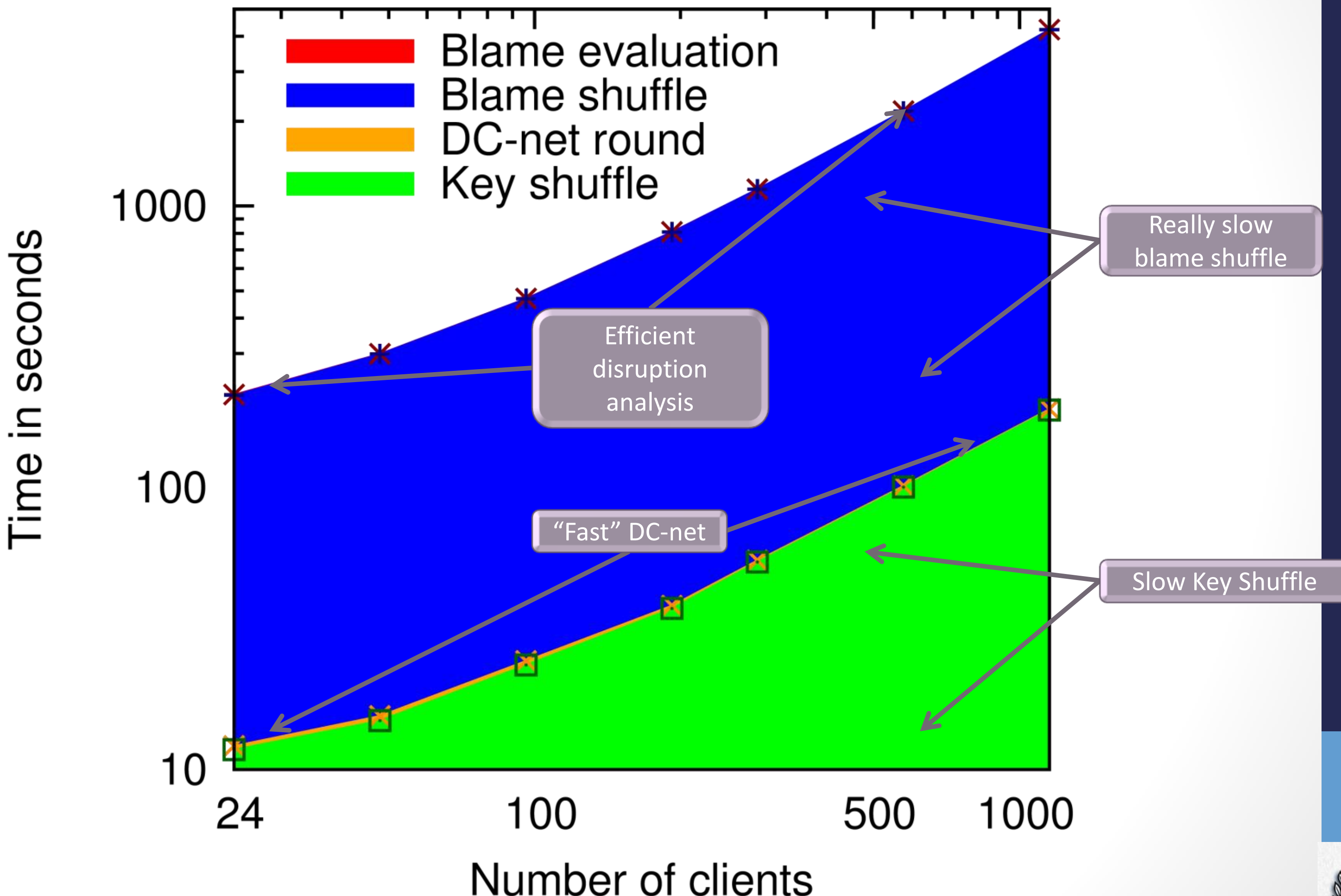
Comparison to Shuffles



Churn Resilience



Protocol Breakdown



Organization

- Motivation
- Existing Approaches
- Dissent – Strong, Scalable Anonymity
  - Computational efficiency
  - Communication efficiency
  - Churn resistance
  - Anonymity
  - Accountability
- Evaluation
- **Conclusions**

Key Take Aways

- We can construct strong **and** scalable anonymous communication systems
  - $O(N^2)$ communication cost to $O(N)$
  - Churn tolerance
  - Provides an effective means to identify disruptors
- Two orders of magnitude larger anonymity sets than previous DC-net approaches
- Maintains strong anonymity properties from DC-nets

Future Work

- Further bandwidth and computation optimizations
- Slot length scheduling policies
- Better ways to anonymously distribute blame
- Handling long term intersection attacks
- Formal security analysis
- Making available for real applications and real users

Finished!

Thanks, questions?

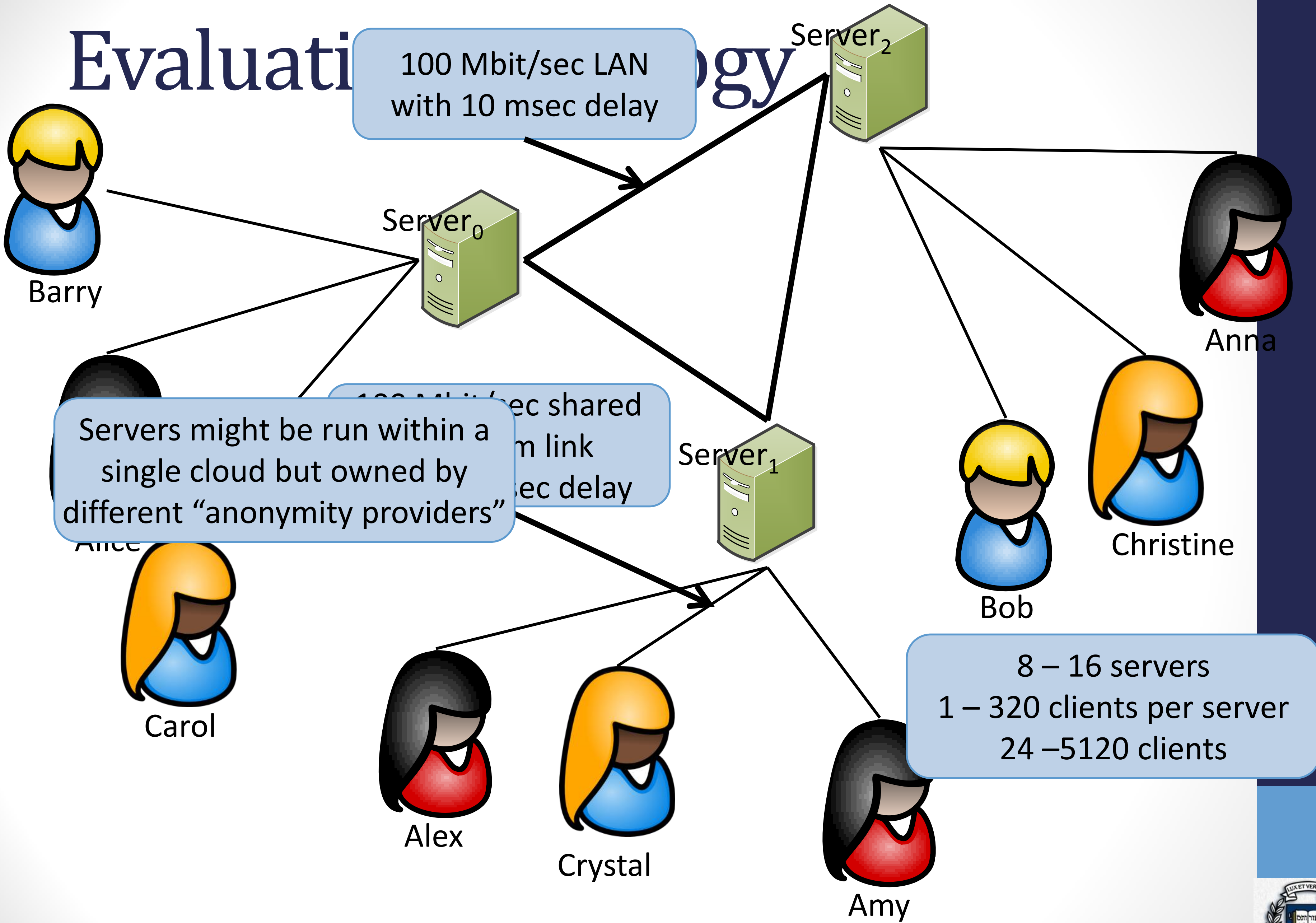
Dissent – Strong, scalable accountable anonymity

Find out more at
http://dedis.cs.yale.edu/2010/anon/

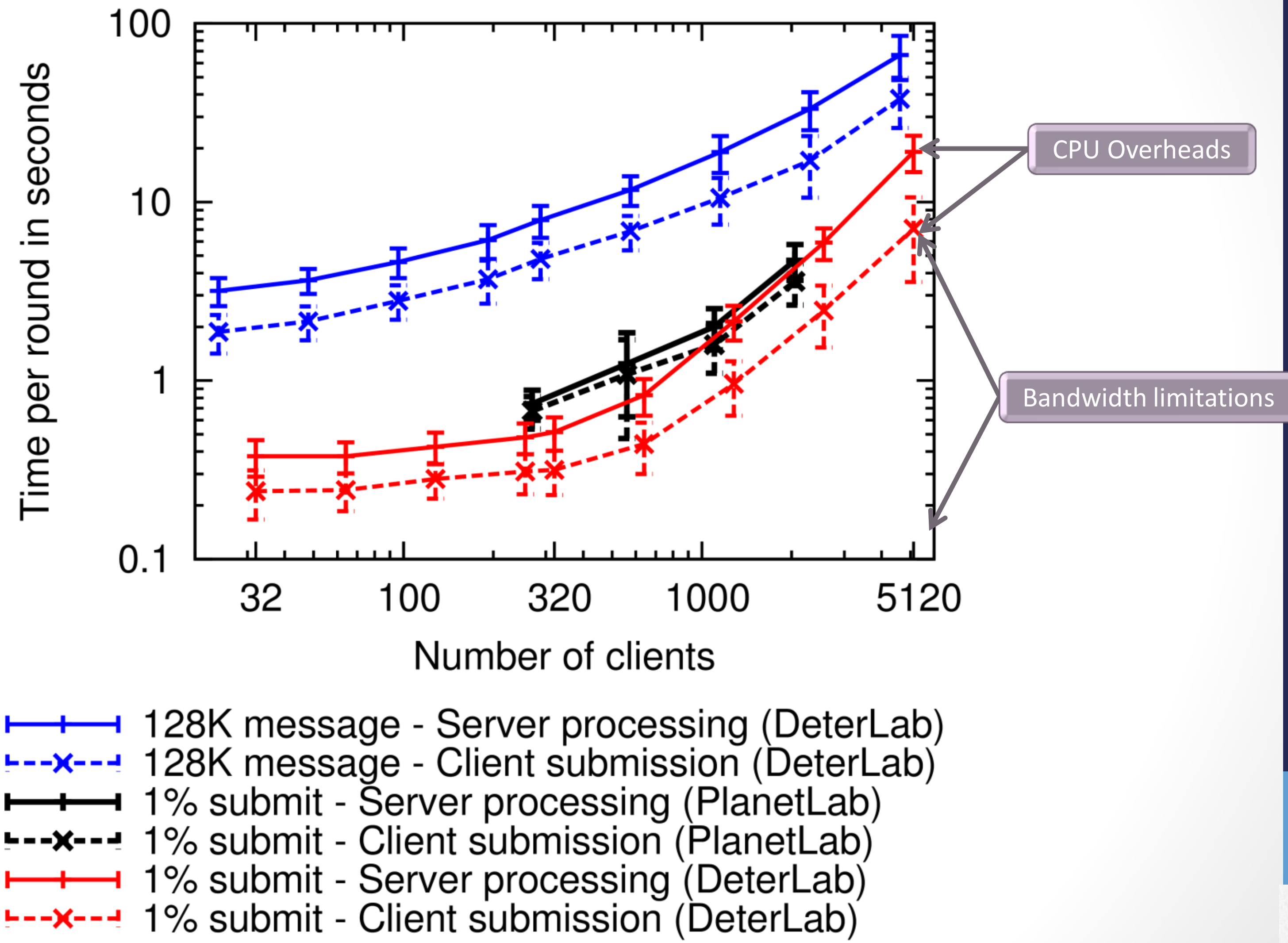
We’ll be at the poster session tonight!

Extra slides

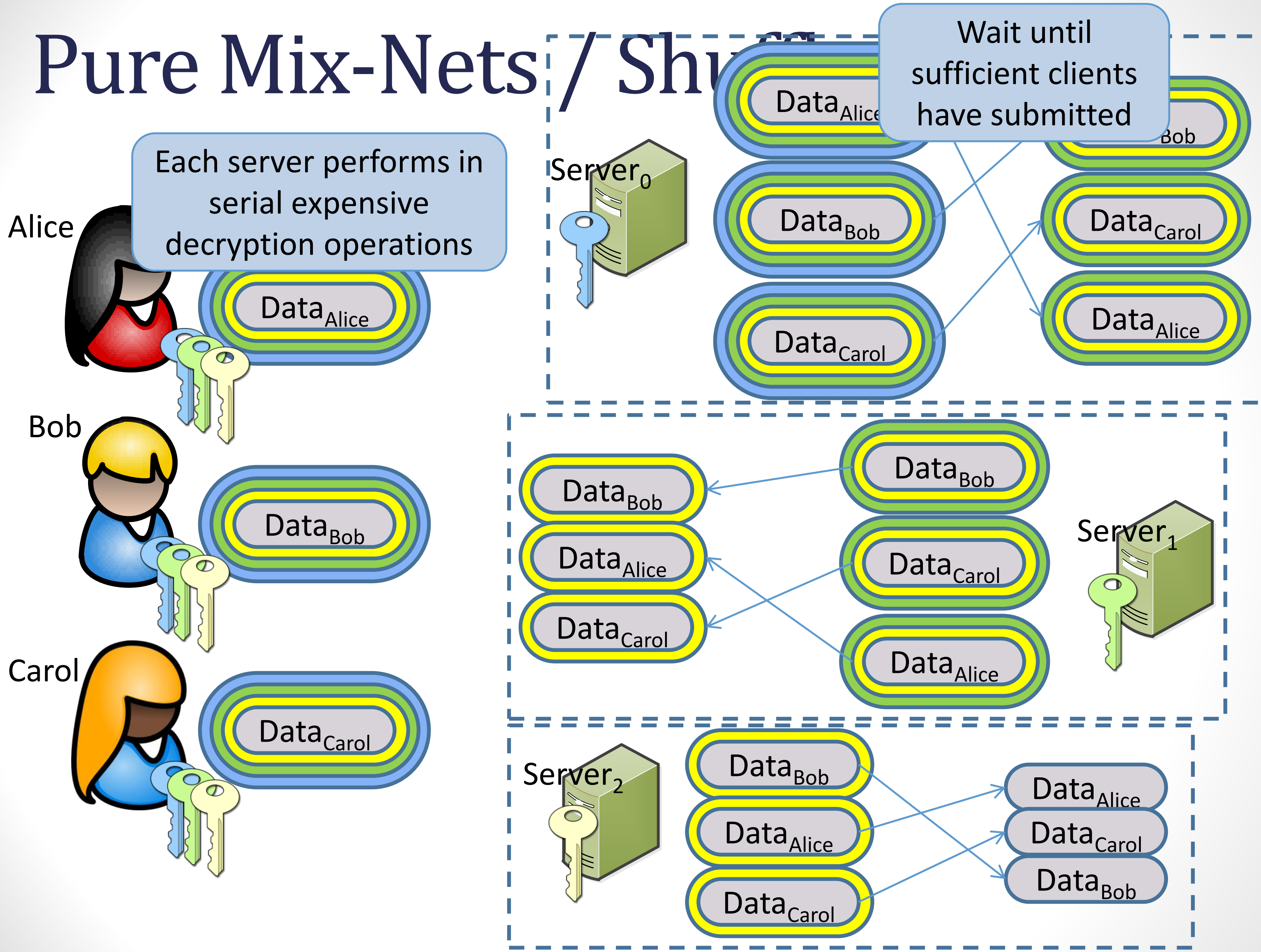
Evaluati ogy



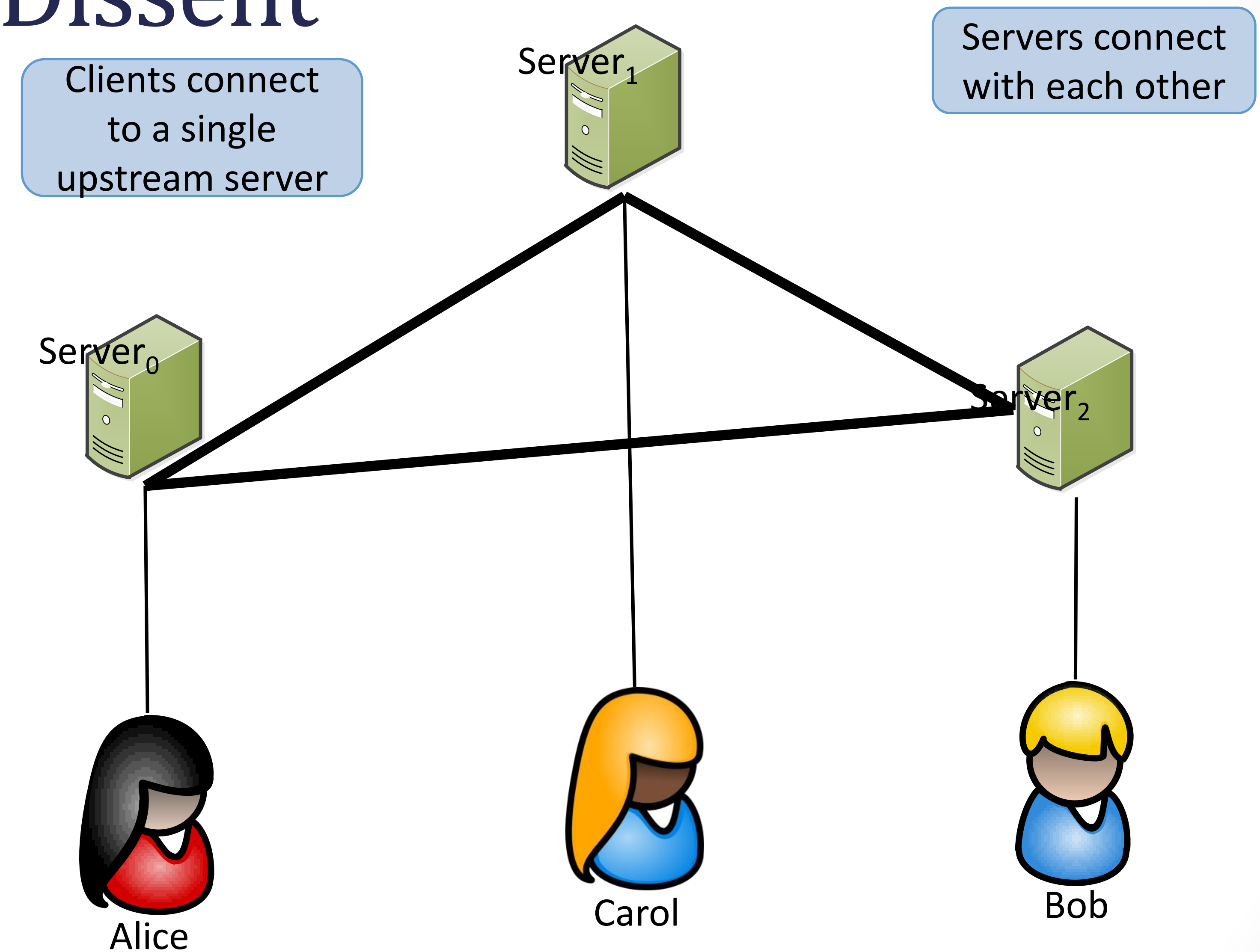
Scaling to Thousands of Clients



Pure Mix-Nets / Shu



Dissent

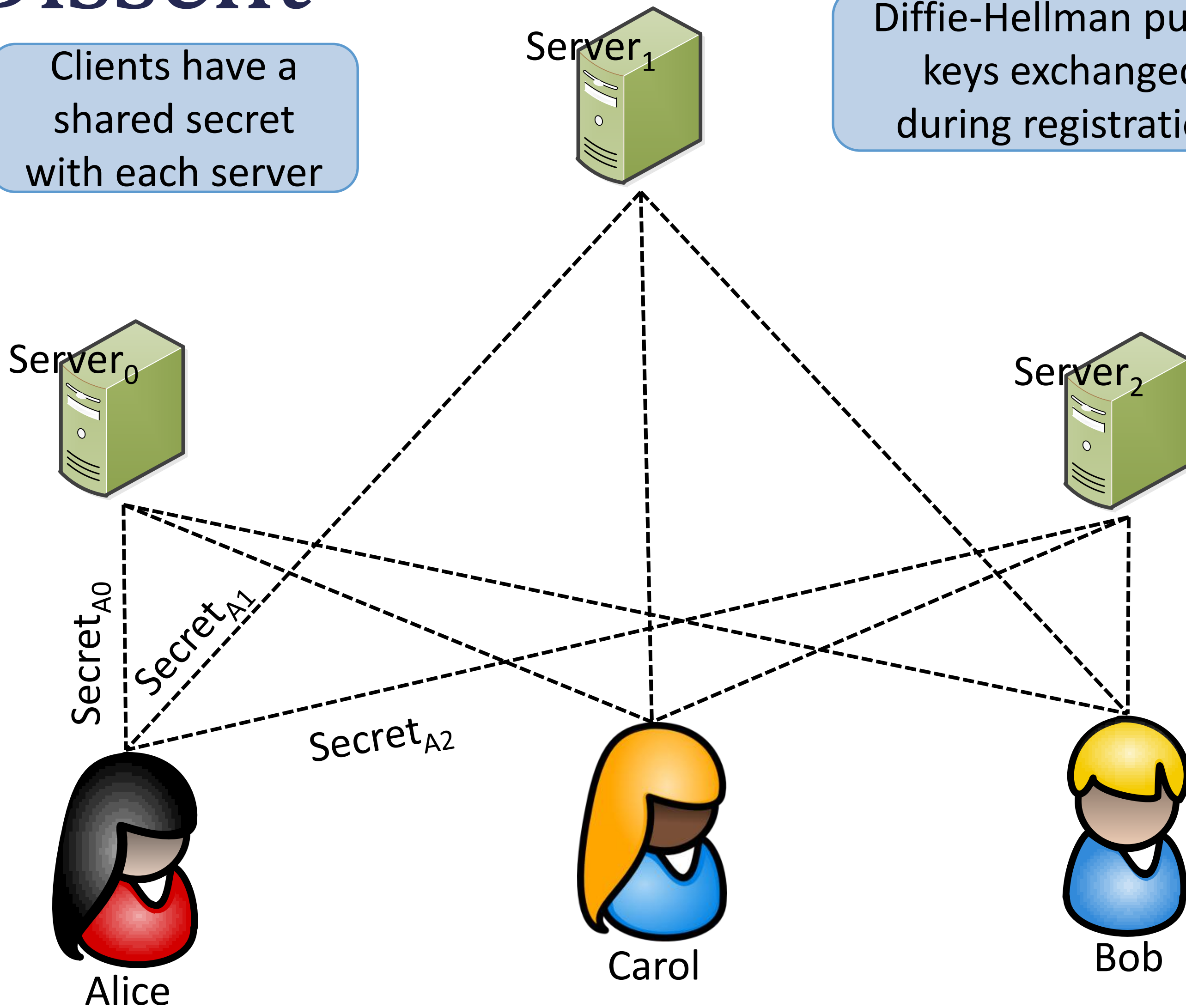


Clients connect to a single upstream server
Servers connect with each other
Server$_1$
Server$_0$
Server$_2$
Alice
Carol
Bob

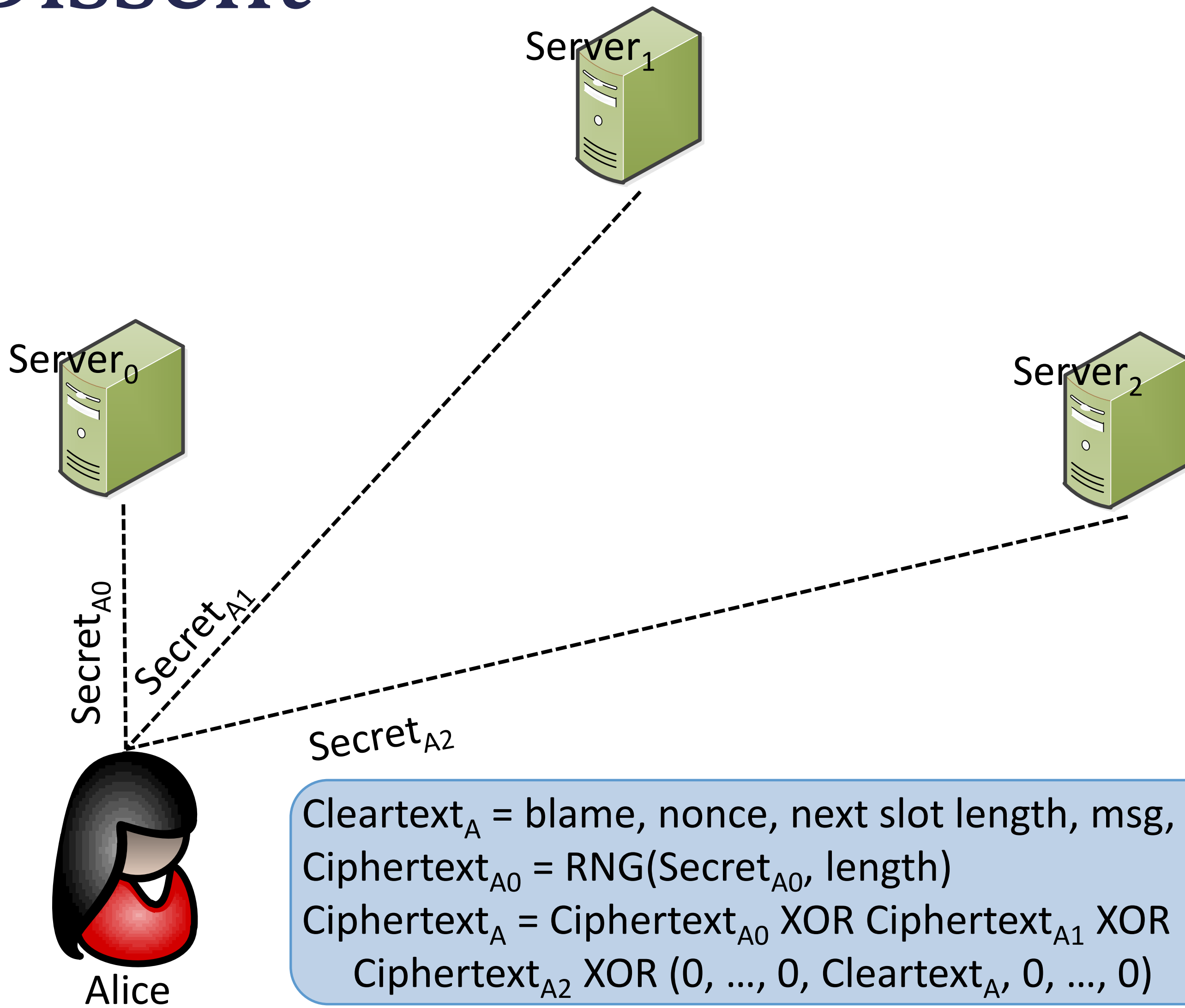
Dissent

Clients have a shared secret with each server

Diffie-Hellman public keys exchanged during registration

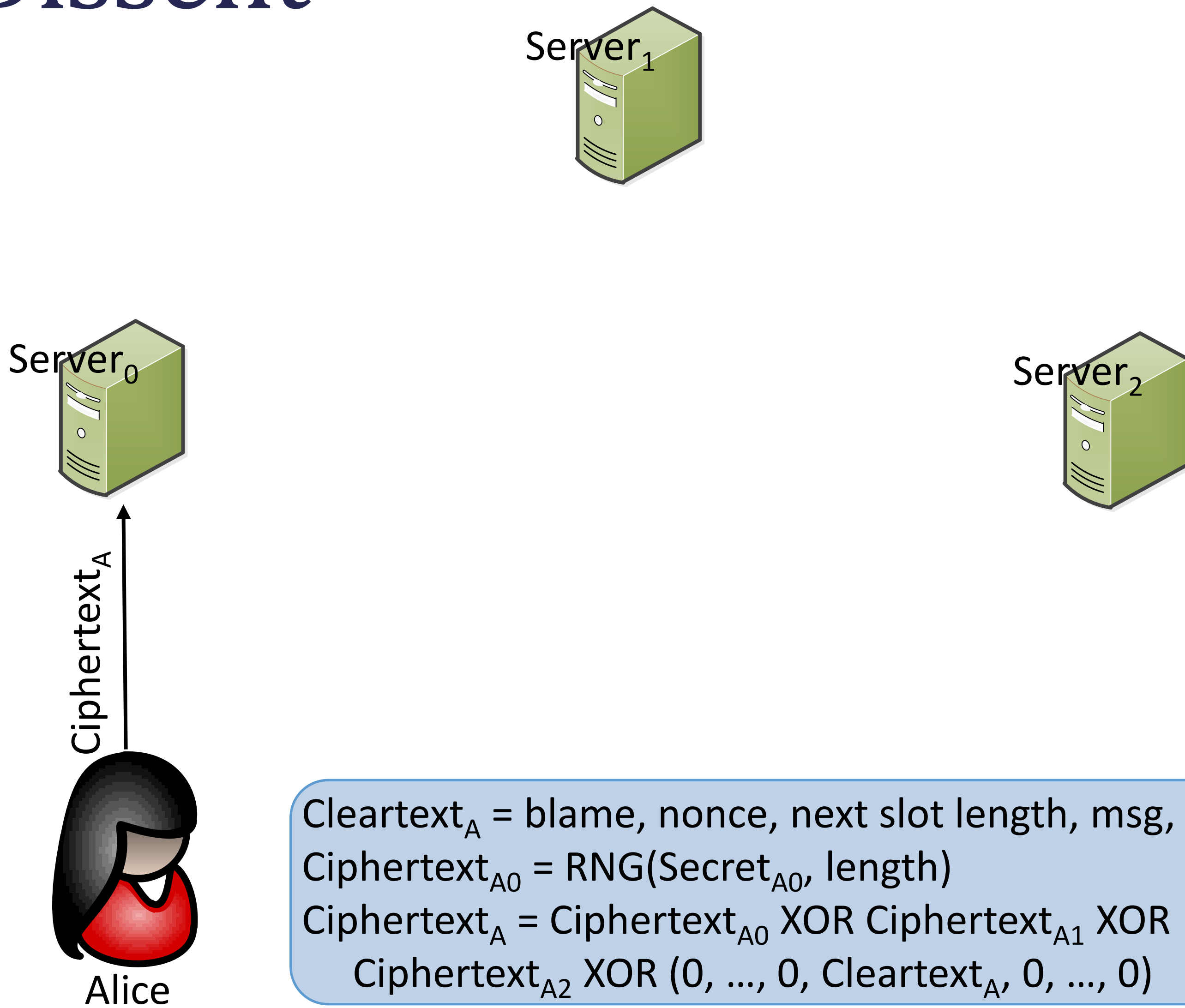


Dissent



Cleartext$_A$ = blame, nonce, next slot length, msg, hash
Ciphertext$_{A0}$ = RNG(Secret$_{A0}$, length)
Ciphertext$_A$ = Ciphertext$_{A0}$ XOR Ciphertext$_{A1}$ XOR
Ciphertext$_{A2}$ XOR (0, …, 0, Cleartext$_A$, 0, …, 0)

Dissent

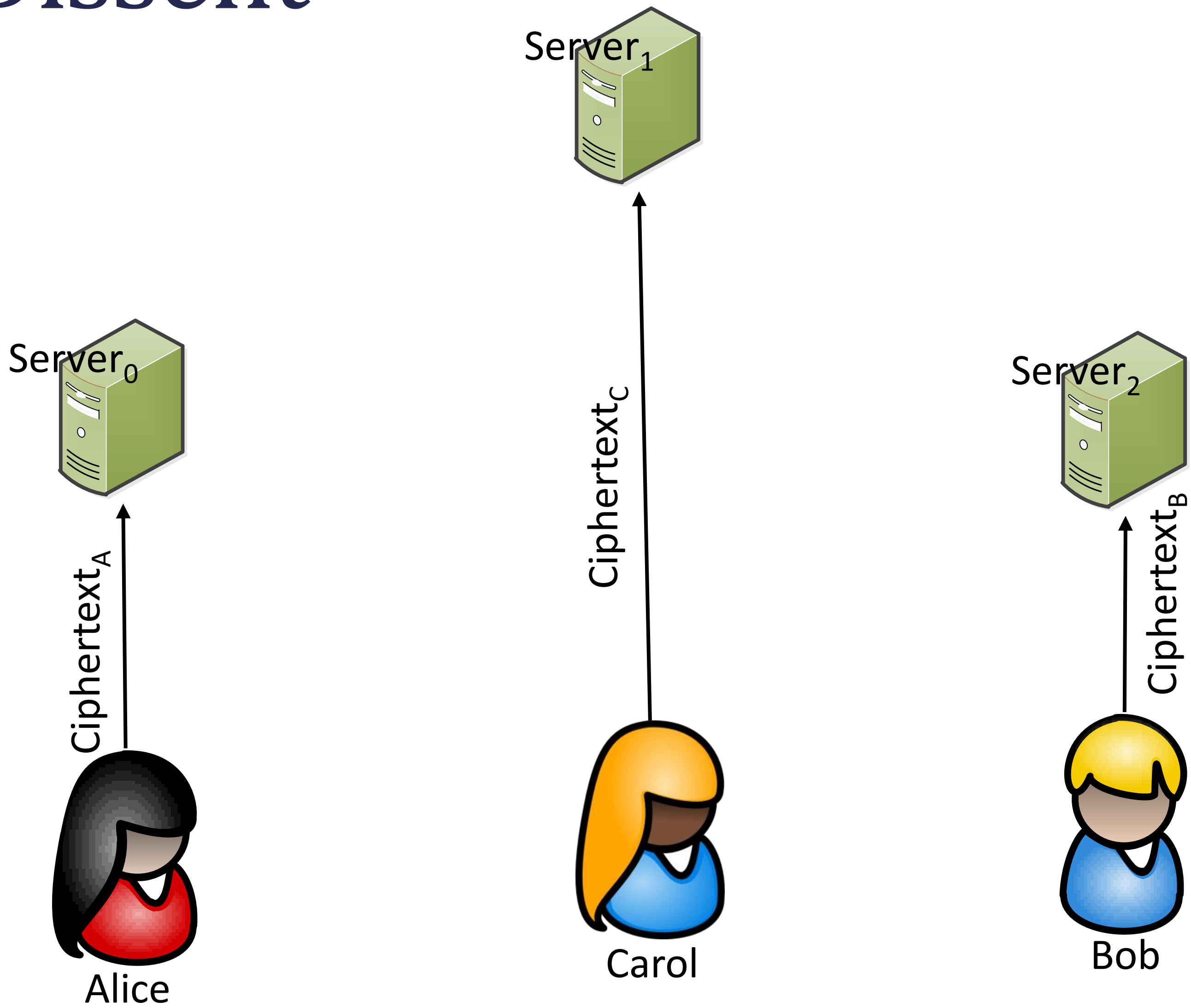


Cleartext$_A$ = blame, nonce, next slot length, msg, hash

Ciphertext$_{A0}$ = RNG(Secret$_{A0}$, length)

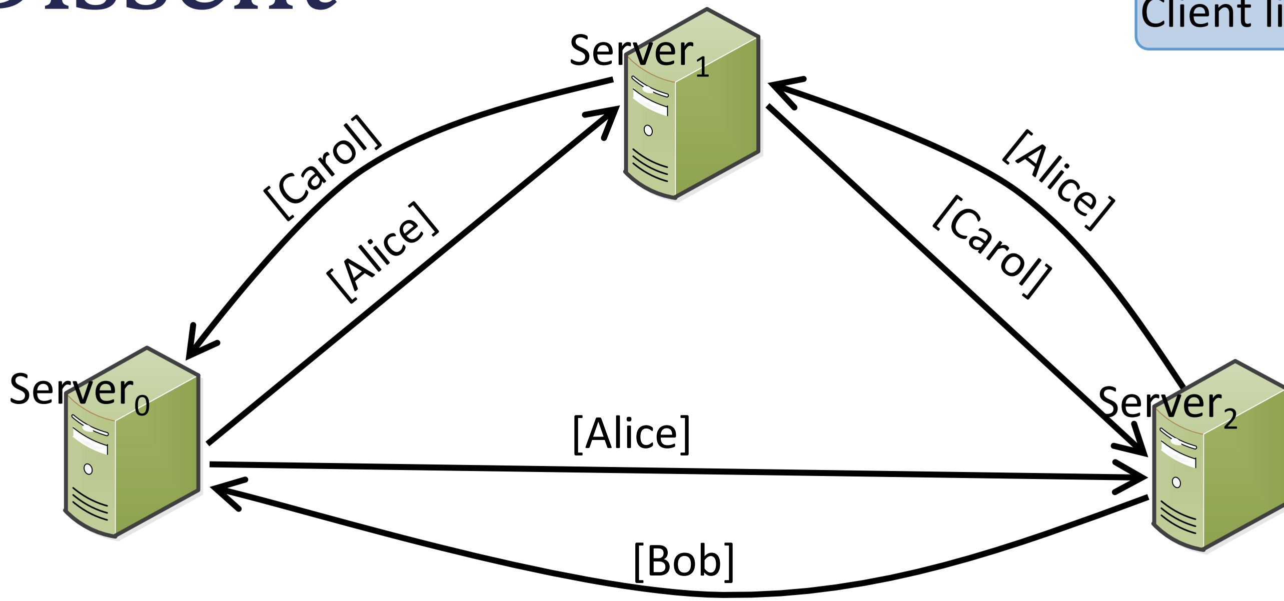
Ciphertext$_A$ = Ciphertext$_{A0}$ XOR Ciphertext$_{A1}$ XOR
Ciphertext$_{A2}$ XOR (0, …, 0, Cleartext$_A$, 0, …, 0)

Dissent

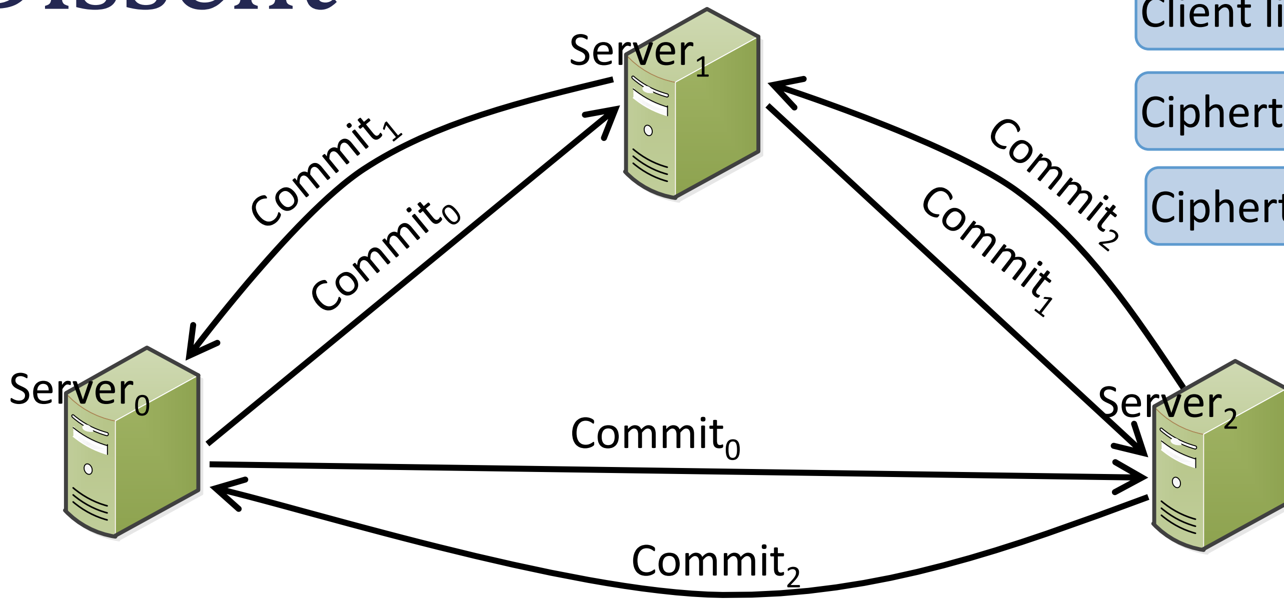

Server$_1$
Server$_0$
Server$_2$
Ciphertext$_A$
Ciphertext$_C$
Ciphertext$_B$
Alice
Carol
Bob

Dissent

Client list exchange



Dissent



Client list exchange

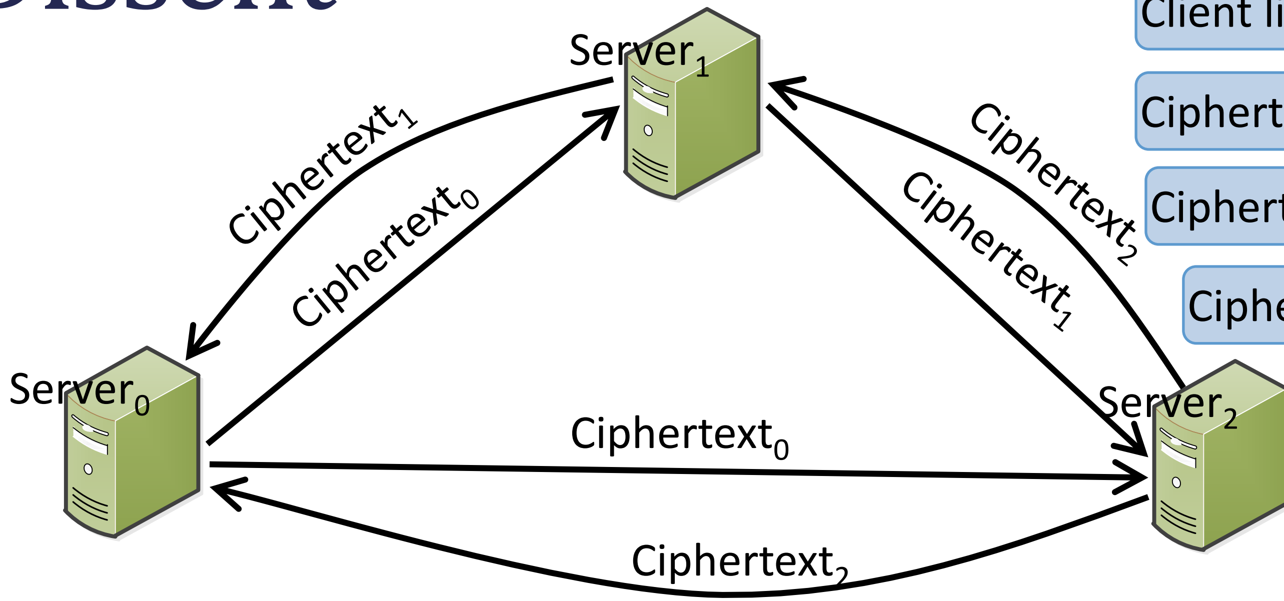
Ciphertext evaluation

Ciphertext commit

Server_0 knows that Alice, Bob, and Carol submitted:
$\text{Ciphertext}_0 = \text{Ciphertext}_A \text{ XOR } \text{Ciphertext}_{A0} \text{ XOR } \text{Ciphertext}_{B0} \text{ XOR } \text{Ciphertext}_{C0}$

$\text{Commit}_0 = \text{Hash}(\text{Ciphertext}_0)$

Dissent



Client list exchange

Ciphertext evaluation

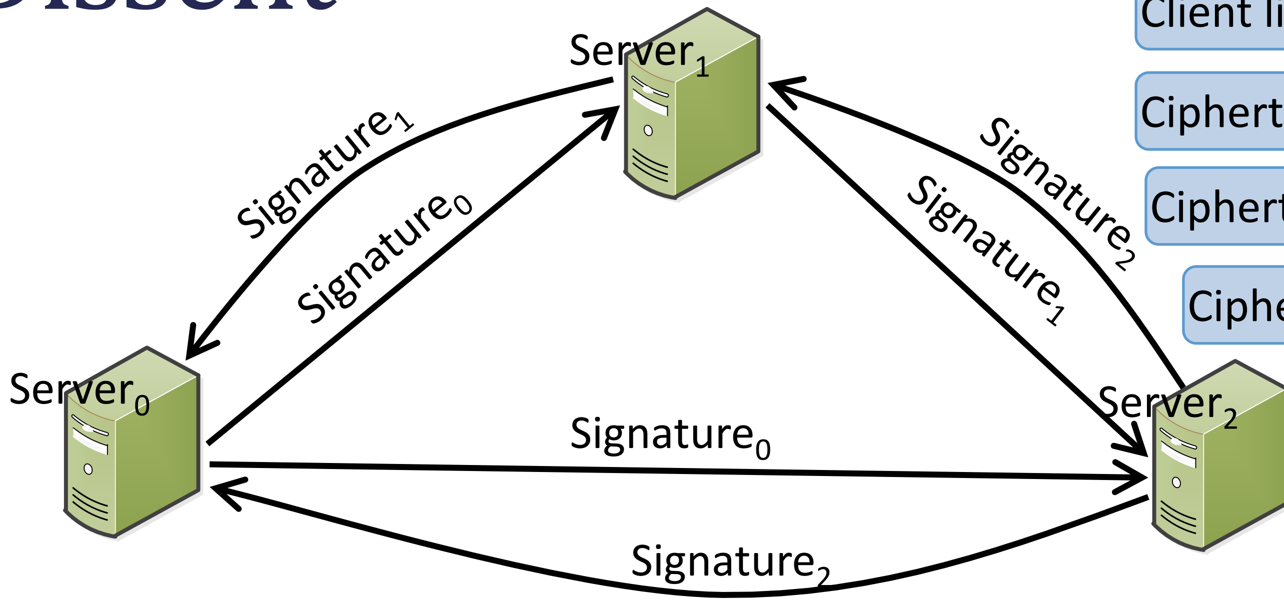
Ciphertext commit

Ciphertext exchange

Server$_0$ knows that Alice, Bob, and Carol submitted:
Ciphertext$_0$ = Ciphertext$_A$ XOR Ciphertext$_{A0}$ XOR
Ciphertext$_{B0}$ XOR Ciphertext$_{C0}$

Commit$_0$ = Hash(Ciphertext$_0$)

Dissent



Client list exchange

Ciphertext evaluation

Ciphertext commit

Ciphertext exchange

Cleartext evaluation

Cleartext commit

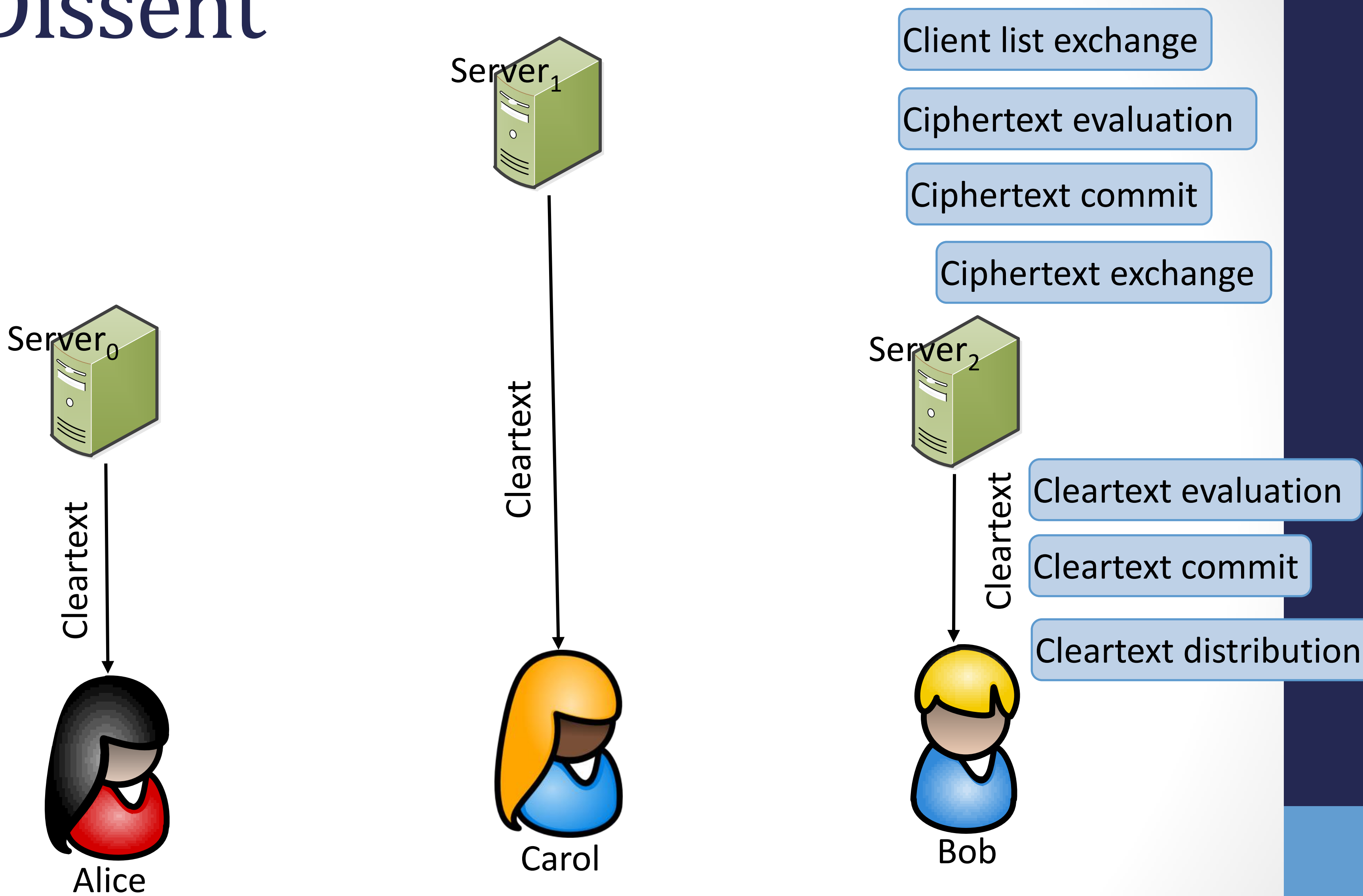
Server$_0$ knows that Alice, Bob, and Carol submitted:
Ciphertext$_0$ = Ciphertext$_A$ XOR Ciphertext$_{A0}$ XOR Ciphertext$_{B0}$ XOR Ciphertext$_{C0}$

Commit$_0$ = Hash(Ciphertext$_0$)

Cleartext = Ciphertext$_0$ XOR Ciphertext$_1$ XOR Ciphertext$_2$

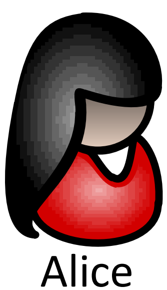
Signature$_0$ = {Cleartext}$_{Key0}$

Dissent



Client list exchange
Ciphertext evaluation
Ciphertext commit
Ciphertext exchange
Server$_1$
Server$_0$
Server$_2$
Cleartext
Cleartext
Cleartext
Cleartext evaluation
Cleartext commit
Cleartext distribution
Alice
Carol
Bob

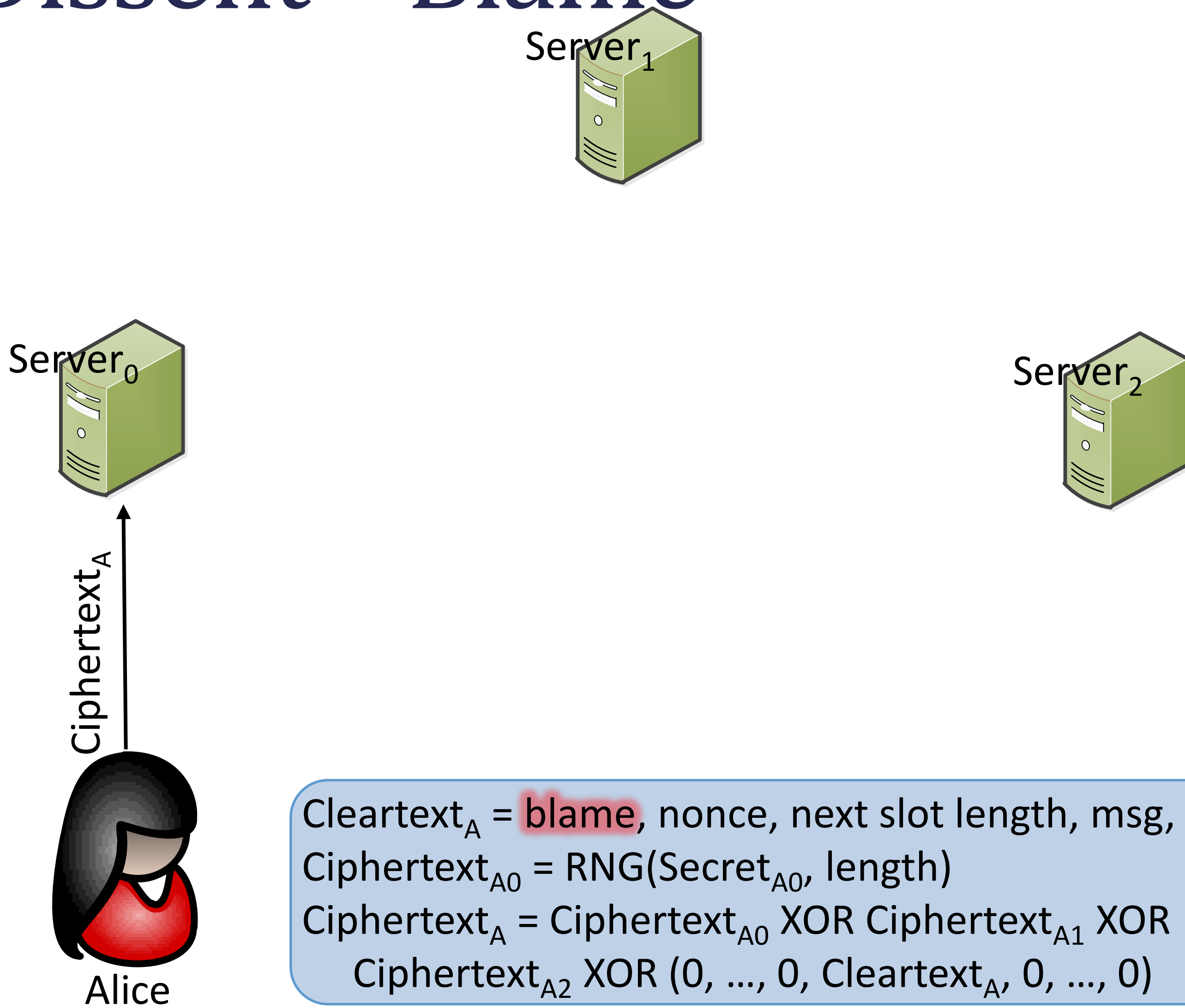
Dissent – Blame



Cleartext_A = blame, nonce, next slot length, msg, hash
$\text{Ciphertext}_{A0} = \text{RNG}(\text{Secret}_{A0}, \text{length})$
$\text{Ciphertext}_A = \text{Ciphertext}_{A0} \text{ XOR } \text{Ciphertext}_{A1} \text{ XOR }$
$\quad \text{Ciphertext}_{A2} \text{ XOR } (0, \ldots, 0, \text{Cleartext}_A, 0, \ldots, 0)$

Identifying Disruptors

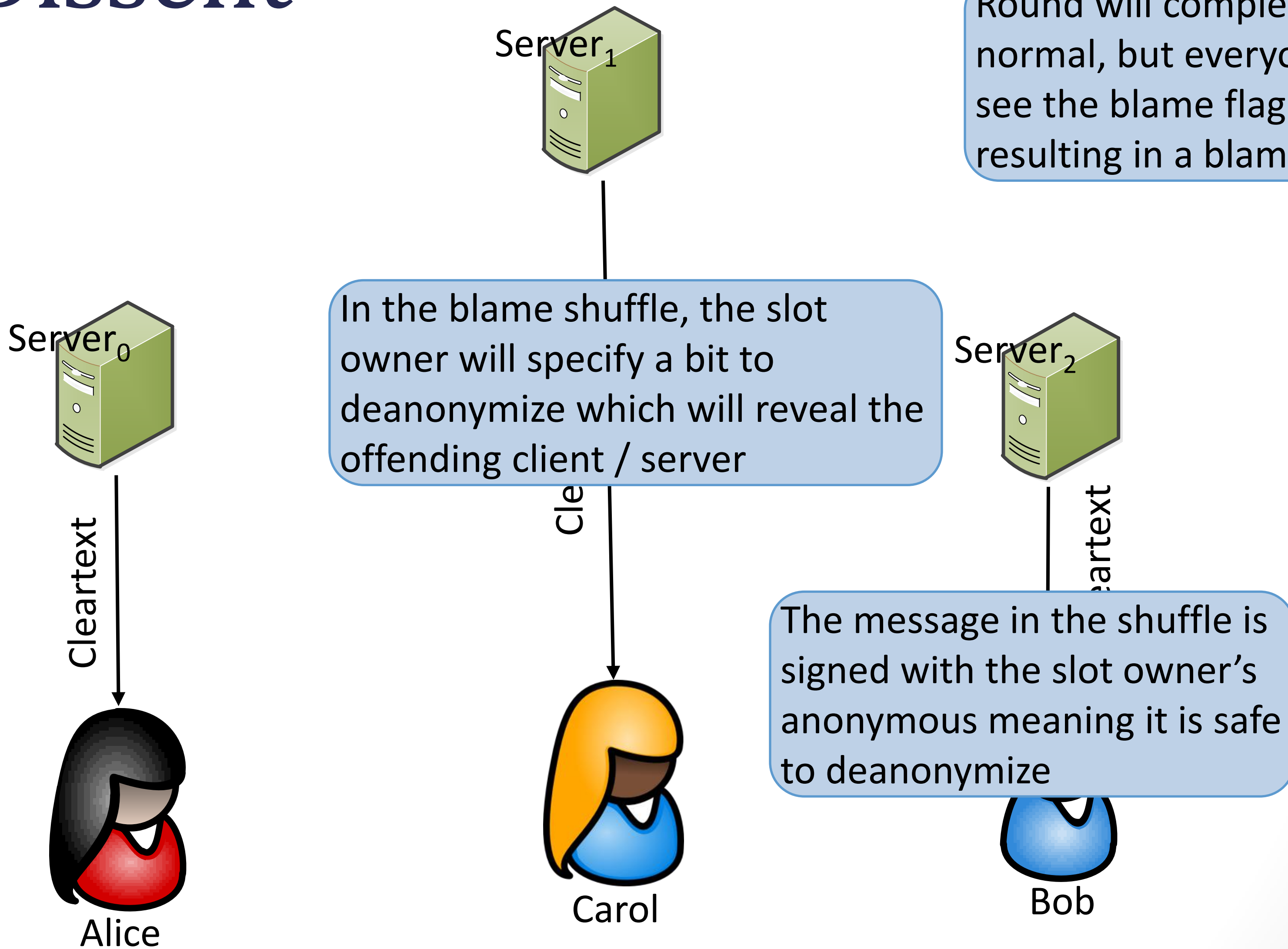
Dissent – Blame



Cleartext$_A$ = blame, nonce, next slot length, msg, hash
Ciphertext$_{A0}$ = RNG(Secret$_{A0}$, length)
Ciphertext$_A$ = Ciphertext$_{A0}$ XOR Ciphertext$_{A1}$ XOR
Ciphertext$_{A2}$ XOR (0, …, 0, Cleartext$_A$, 0, …, 0)

Dissent

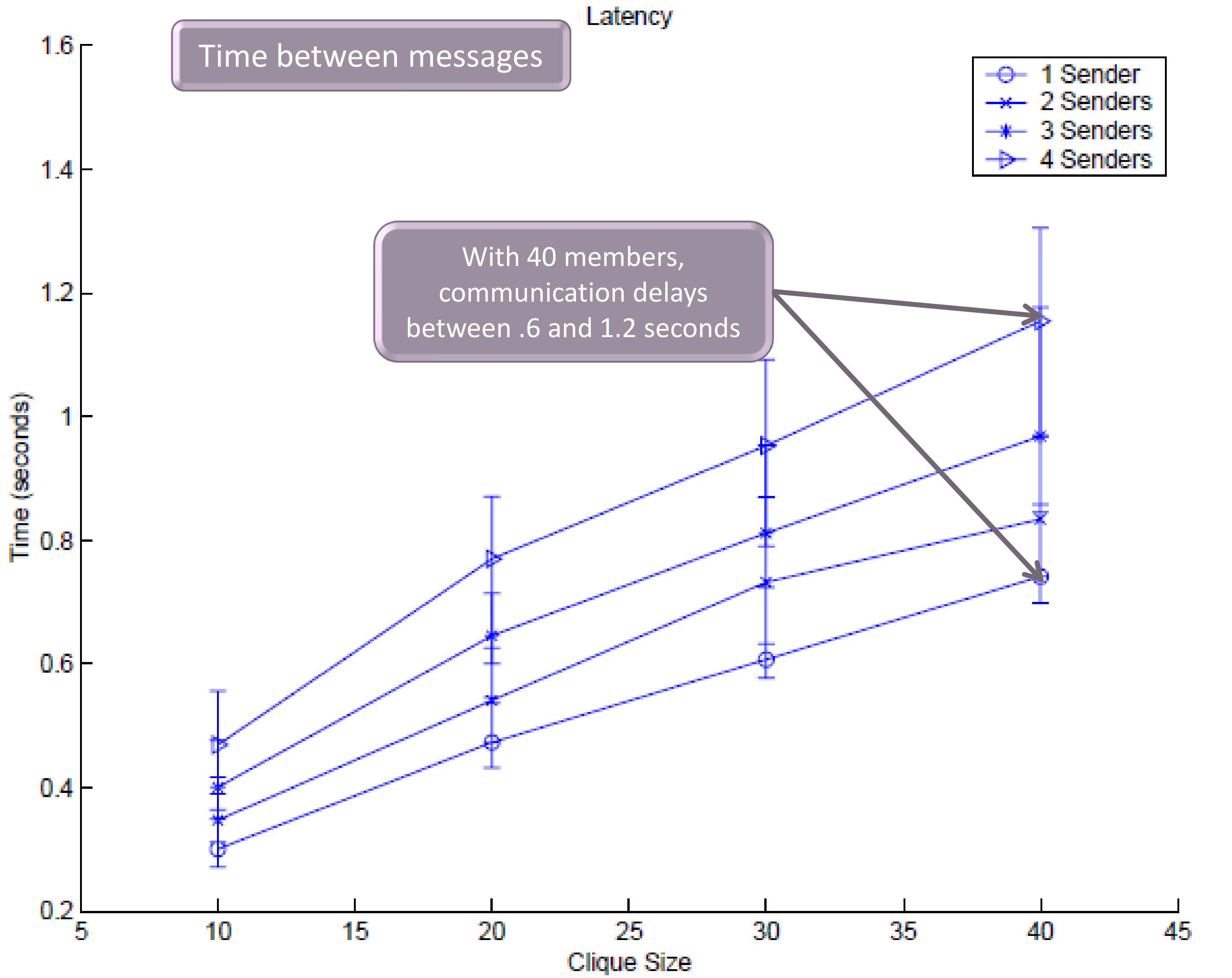
Round will complete as normal, but everyone will see the blame flag set, resulting in a blame shuffle



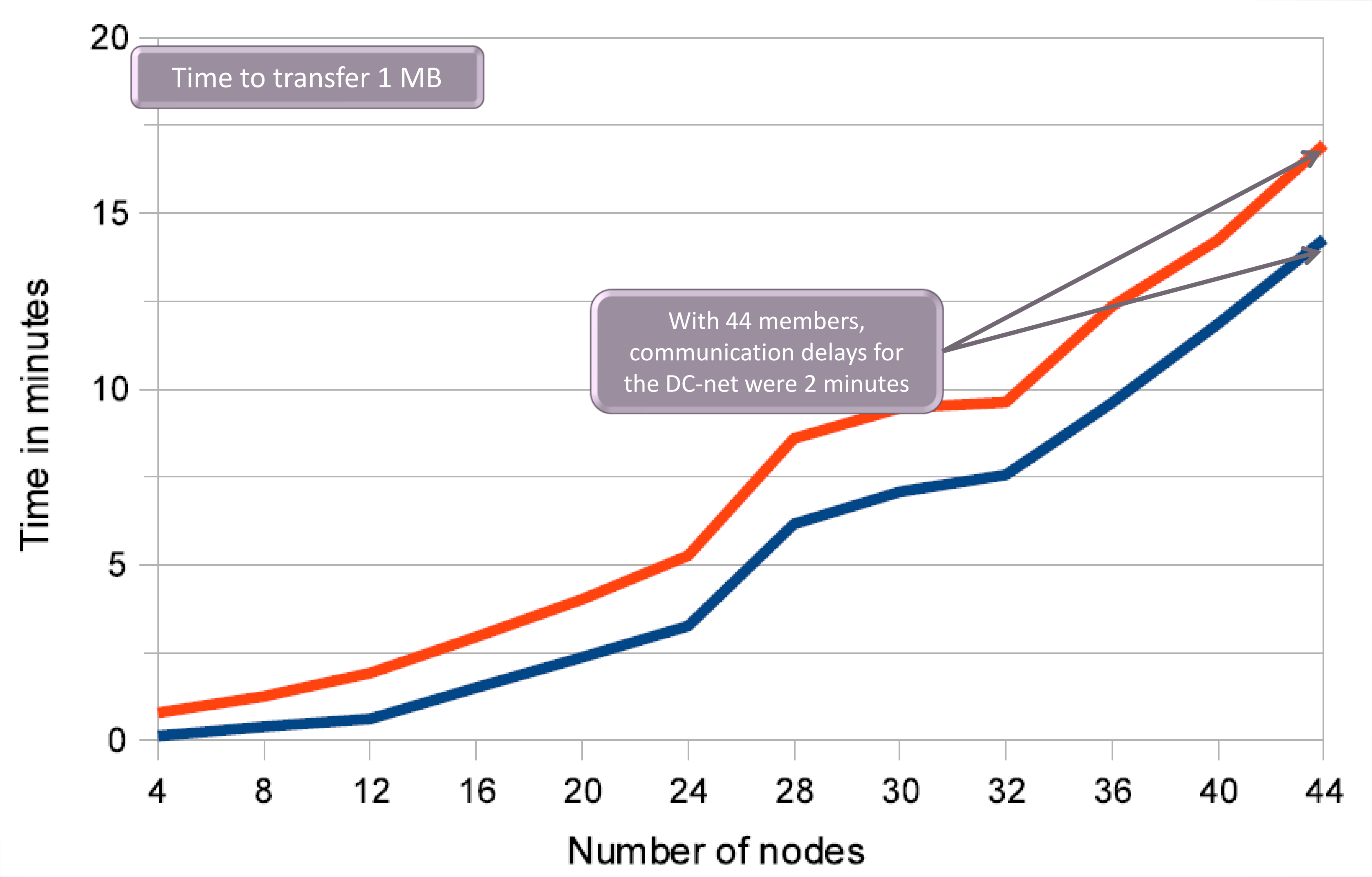
In the blame shuffle, the slot owner will specify a bit to deanonymize which will reveal the offending client / server

The message in the shuffle is signed with the slot owner's anonymous meaning it is safe to deanonymize

Related Work – Herbivore



Related Work – Earlier Dissent



Future Work in Dissent

- Disruption resistance is online, requires additional steps after the protocol has completed
- Practical use in real environments – Such as using WiFi enabled smart phones
- Anonymity boxes – isolated environments running within a virtual machine isolating the user’s private information from the anonymity network
- Participation limits to prevent Sybil attacks

Dissent Disruption Resistance

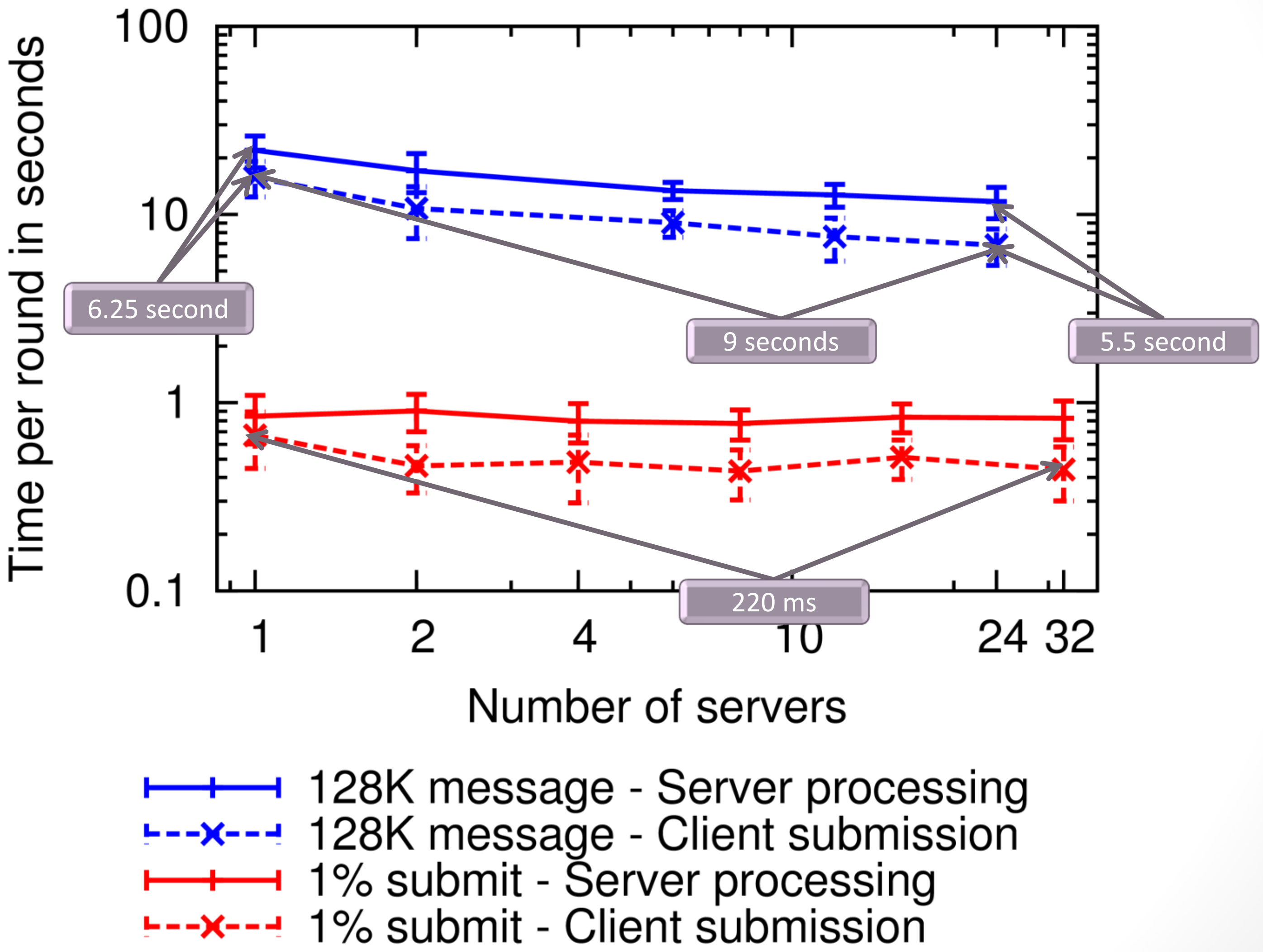
- A malicious bit flip resulting from a 0 -> 1 in the cleartext can be used to generate an accusation
  - In a DC-net, client requests accusation shuffle
  - In shuffle, client specifies the flipped bit
- Servers share bits for this bit index, finding either
  - A server sent bits that do not match his ciphertext – thus he is guilty of the disruption
  - A client's ciphertext does not match the accumulation of the server's bits
- Clients rebut by sharing with servers the shared secret of the offending server, accepting blame, or remaining suspect

Analytical Comparison

Feature	DC-Nets	Herbivore	Dissent
Messages	$O(N^2)$	$O(N)$	$O(N)$
Secrets	$O(N^2)$	$O(N^2)$	$O(N*M)$
Anon	$O(K)$	$O(K)$	$O(K)$, assuming 1 honest server

N = Members (clients)
M = Servers
K = honest members

Server Count Effects



Analytical Comparison

	Feature	Dissent	D3
Shuffle	Comm	O(N) serial steps	O(1)
	Anon	O(K), K = honest members	O(K), K = honest members, assuming 1 honest server
DC-net	Comm	$O(N^2)$ messages $O(N^2)$ shared secrets	O(N) messages O(N) shared secrets
	Anon	O(K), K = honest members	O(K), K = honest members, assuming 1 honest server

Client/Server Trust Models

- Trust all servers
  - Unrealistic in the real world
- Trust no servers – SUNDR
  - Ideal but complicated due to lack of knowledge and message time constraints
- Trust at least one server – Anytrust
  - With one honest server, anonymity set is equal to the set of all honest members (clients)
  - No need to know which server to trust
  - (Used in Mix-nets)

DC-Nets Generalized

- Members share secrets with each other
  - Such as Diffie-Hellman exchanges
  - Can be used to generate variable length string
- Each member constructs a ciphertext
  - XOR in the string generated by each shared secret
  - Optionally, XOR secret message
- Positions inside a DC-net can be assigned via randomness (Ethernet style backoff) or a Mix-Net
- After obtaining a copy of each ciphertext
  - XOR each ciphertext together
  - Effectively, cancelling out generated strings
  - Revealing secret messages

Existing Approaches

Method	Weakness
Mix-Nets, Tor	Traffic analysis attacks
Group / Ring Signatures	Traffic analysis attacks
Voting Protocols	Fixed-length messages
DC Nets	Anonymous DoS attacks
Dissent	Intolerant to churn / long delays between msgs
Herbivore	Small anonymity set