Catch Me If You Can: A Practical Framework to Evade Censorship in Information-Centric Networks

Reza Tourani, Satyajayant (Jay) Misra, Joerg Kliewer[†], Scott Ortegel, Travis Mick

Computer Science Department
New Mexico State University

[†]Department of Electrical & Computer Engineering
New Jersey Institute of Technology

Outline

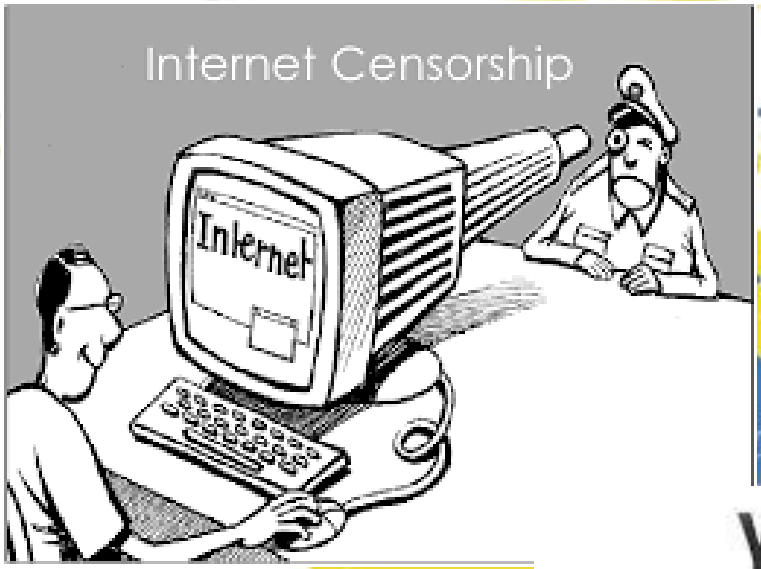
- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

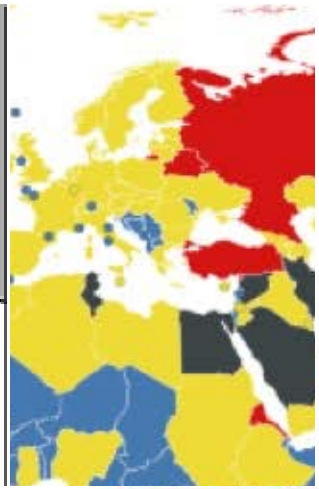
Outline

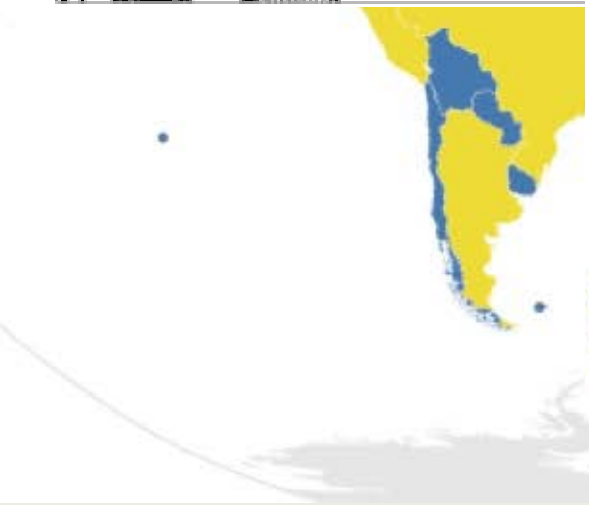
- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

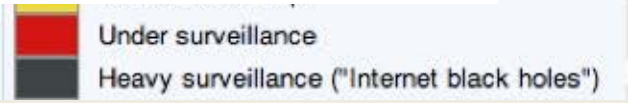
Censorship is common and widespread.

http://commons.wikimedia.org/wiki/File:Internet_blackholes.svg#Legend





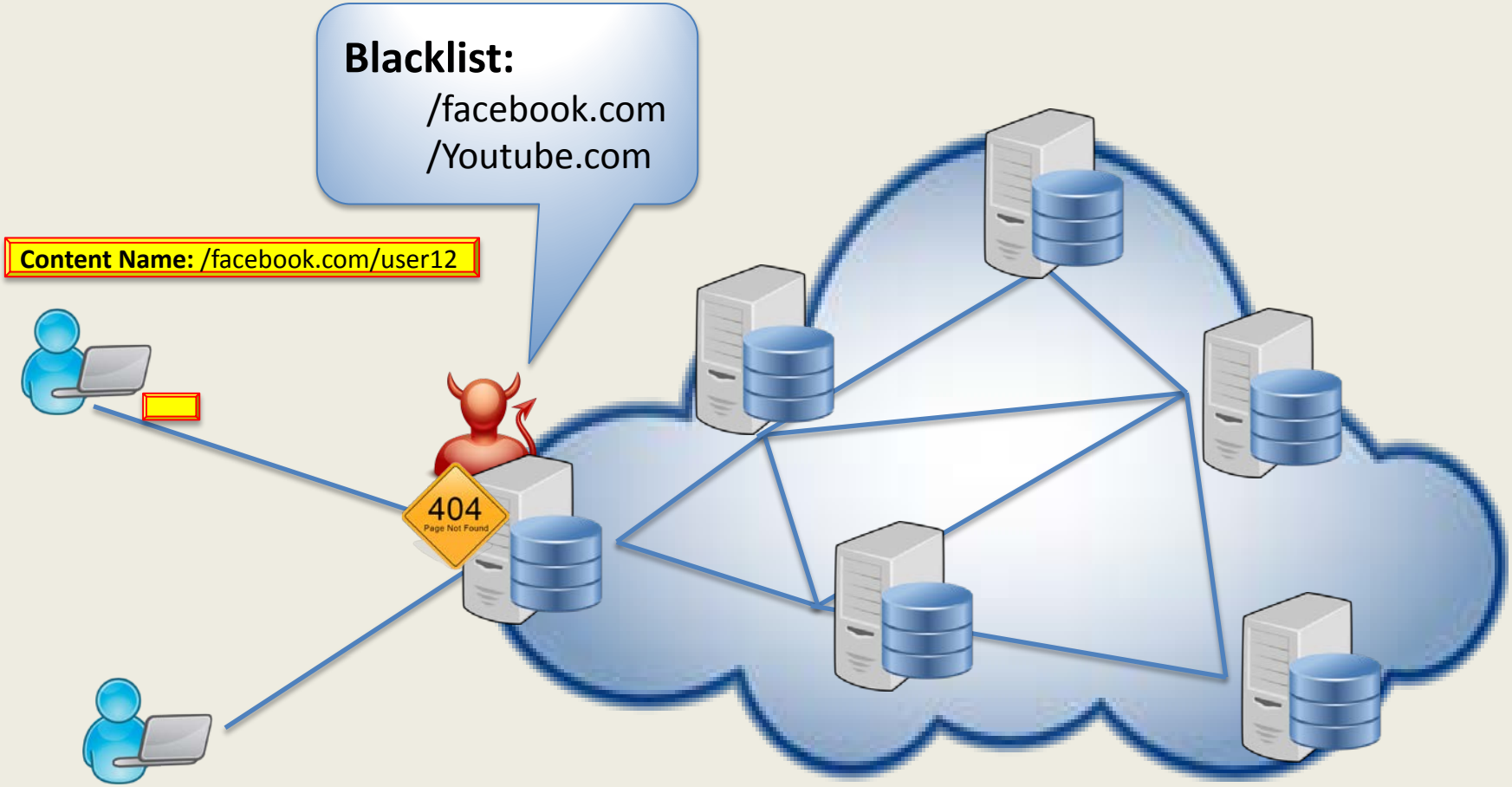




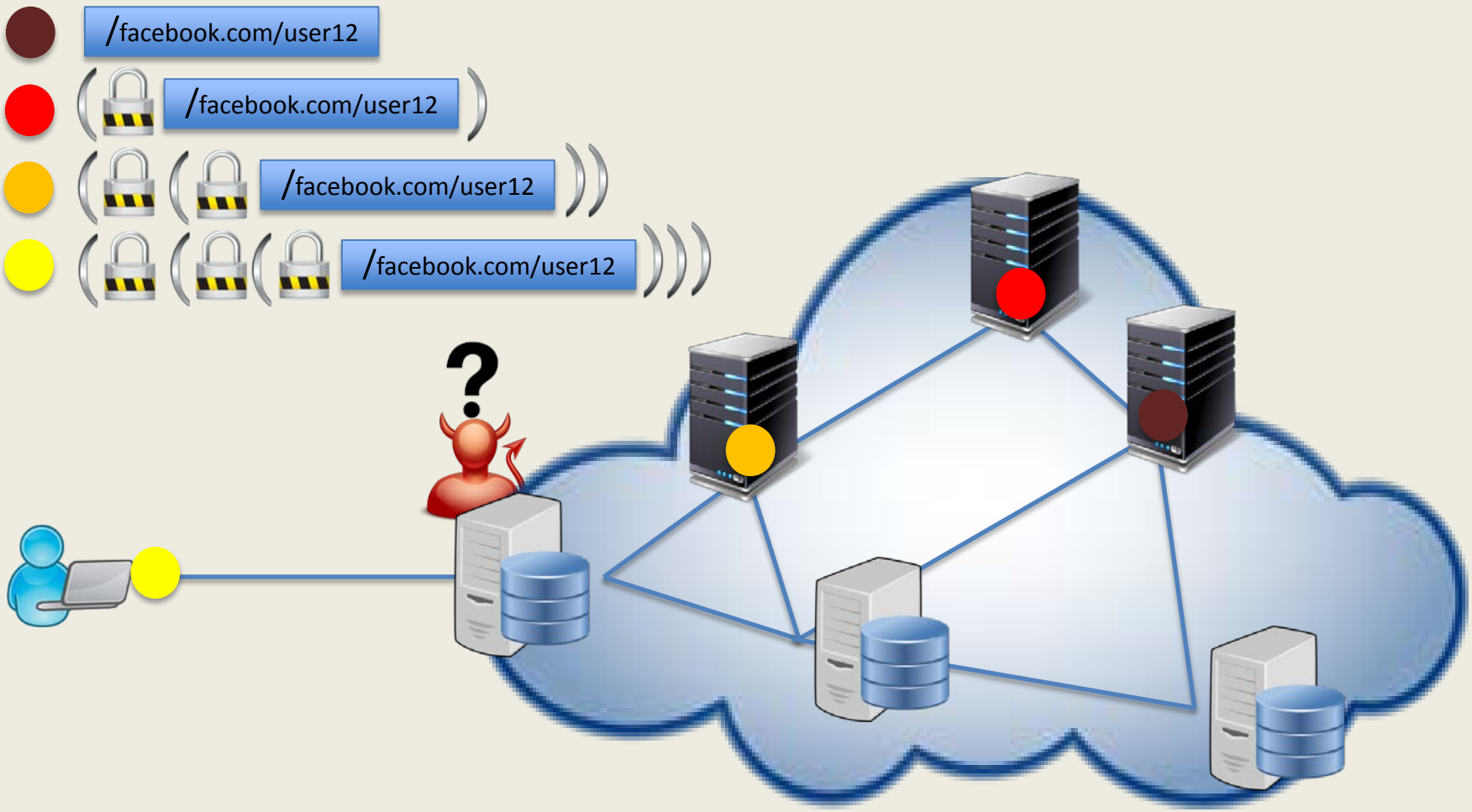
Outline

- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

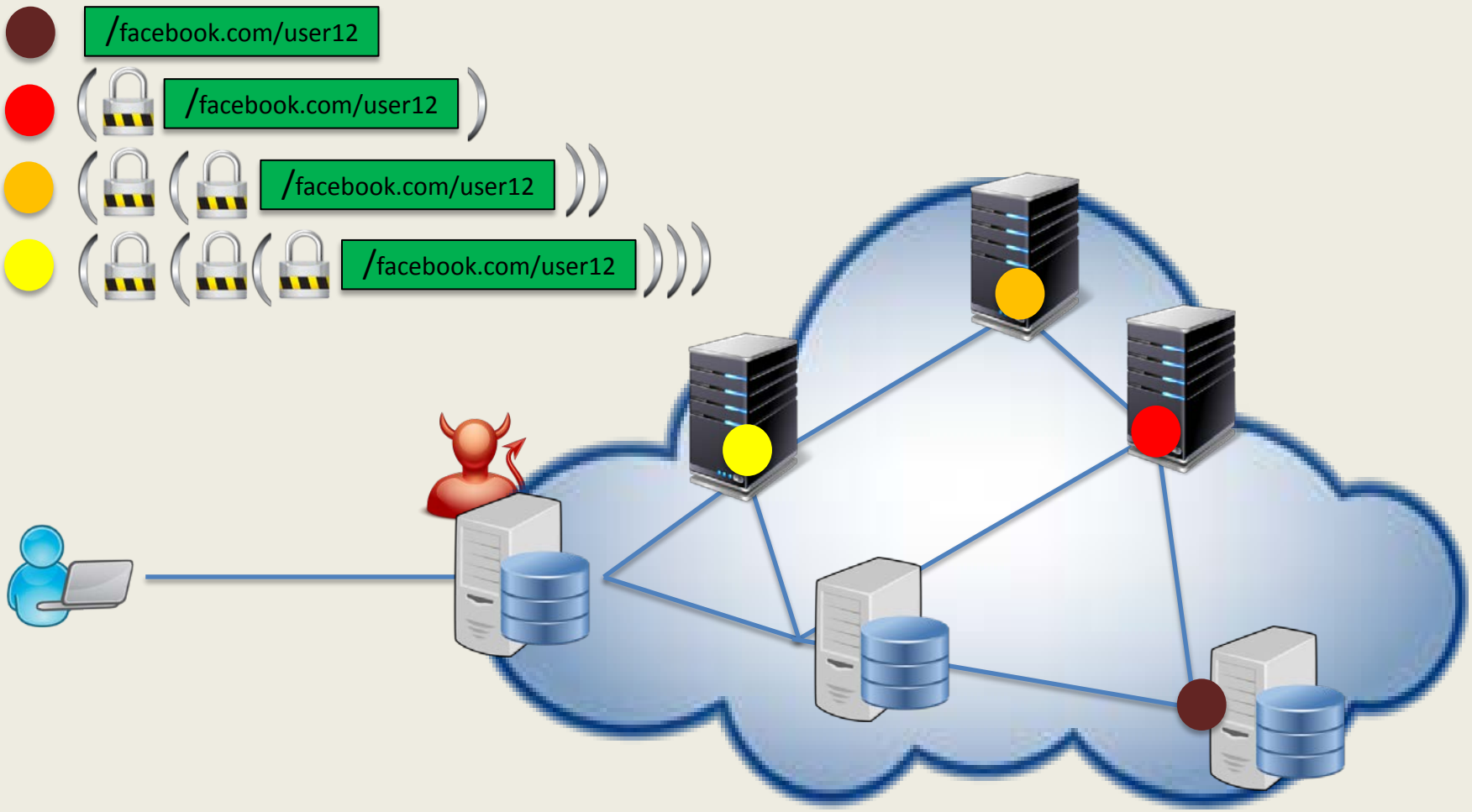
Censorship can be pervasive in ICNs.



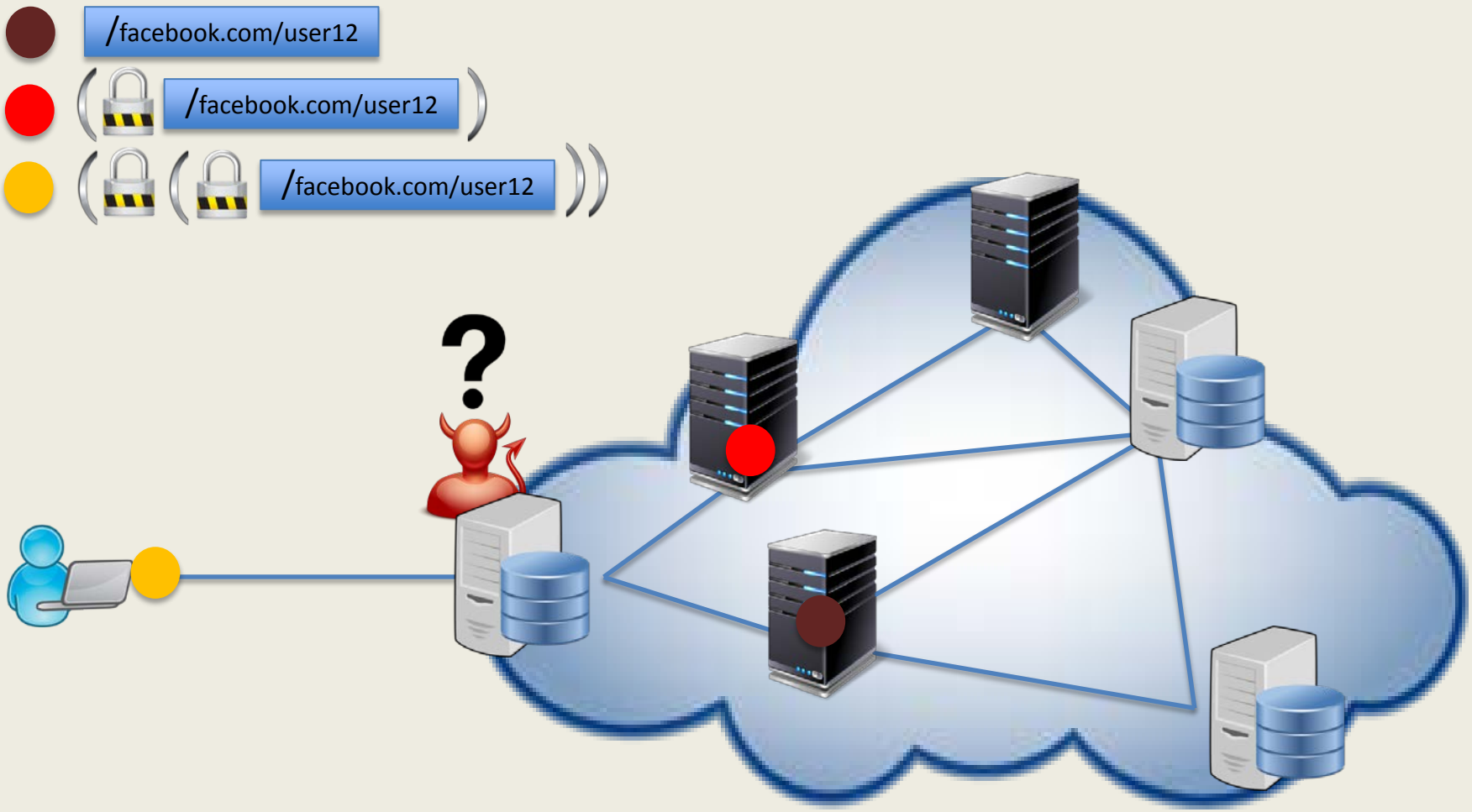
Tor: Using Onion Routing to evade censors.



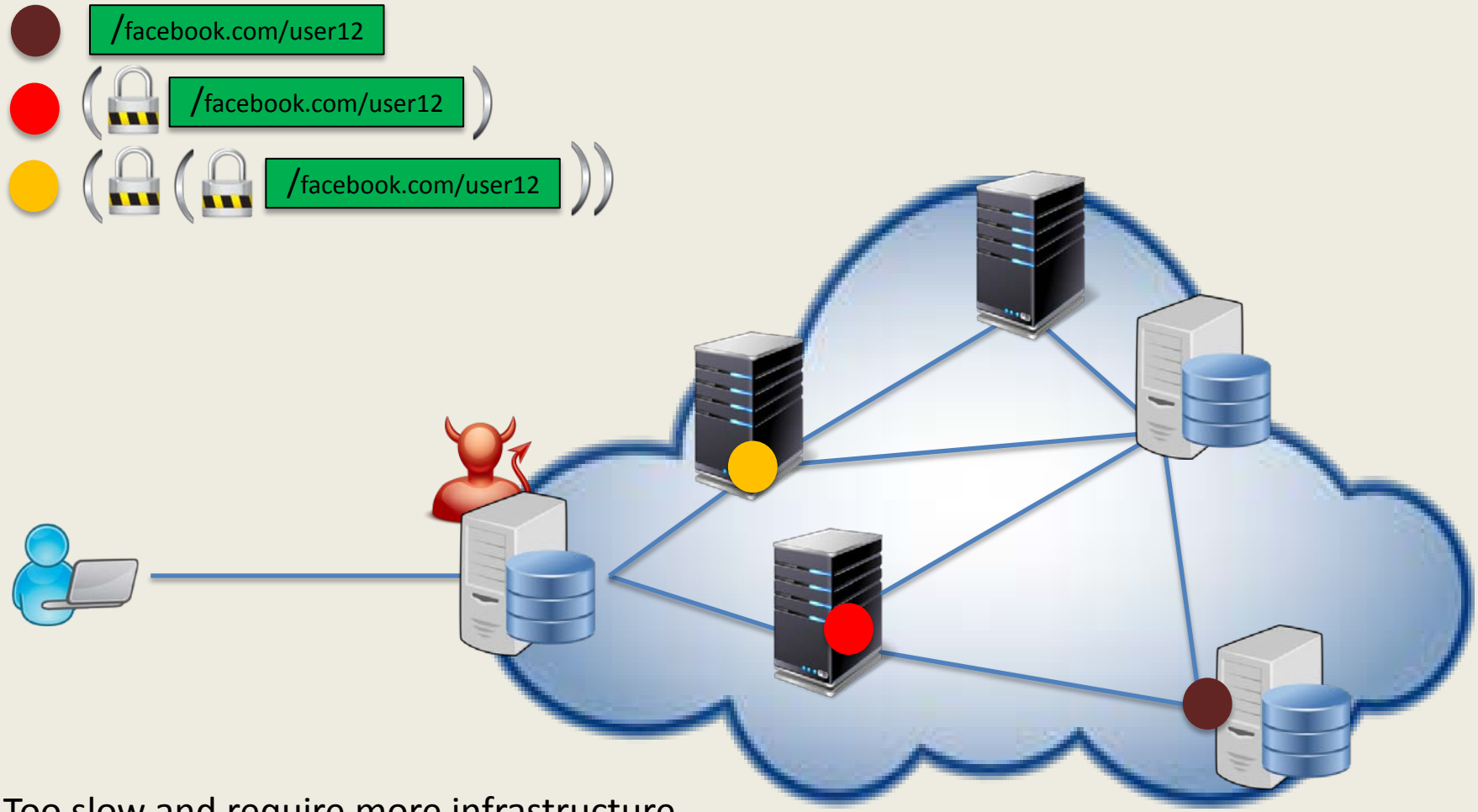
Tor: Using Onion Routing to evade censors.



ANDaNA


/facebook.com/user12
/facebook.com/user12
/facebook.com/user12
?

ANDaNA



Too slow and require more infrastructure

Can we find something better?!?

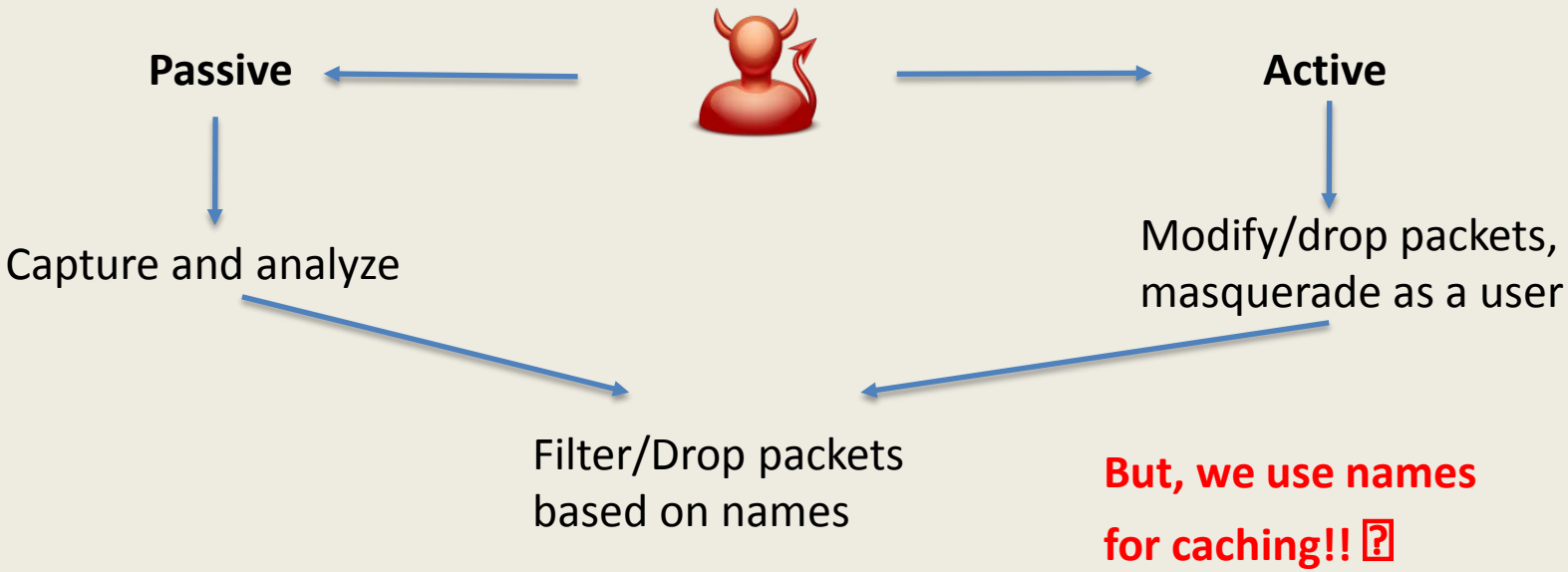
Outline

- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

System Model

- Set of users (U), Set of routers (R), Set of providers (P), Set of anonymizers (A), Filtering router (R_f)
- Each u ∈ U is connected to an $R_i \in R$ (R_i can be filtering/not)
- Users can retrieve the set A securely and privately.
- Content names follow a conventional (ICN) hierarchical naming scheme (E.g.: */www.facebook.com/user12/frontpage.html*).
- M^k: Name of k-bits; $\underline{Z}$: Encrypted message; N = Alphabet Size.

Attack Model



❖ **Privacy-Caching Trade-off:** Privacy-preservation is more important than caching resultant efficiency.

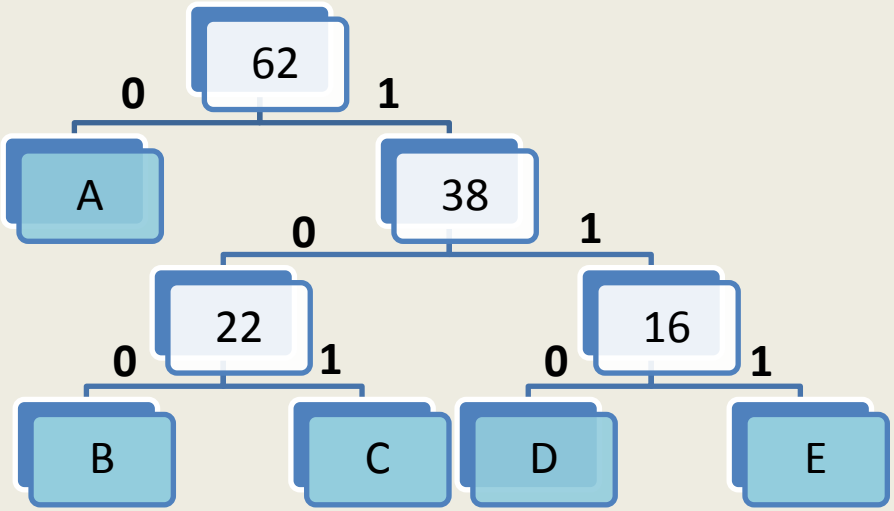
Outline

- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Numerical Results
- Conclusions and Future Work

Preliminaries (Huffman Coding)

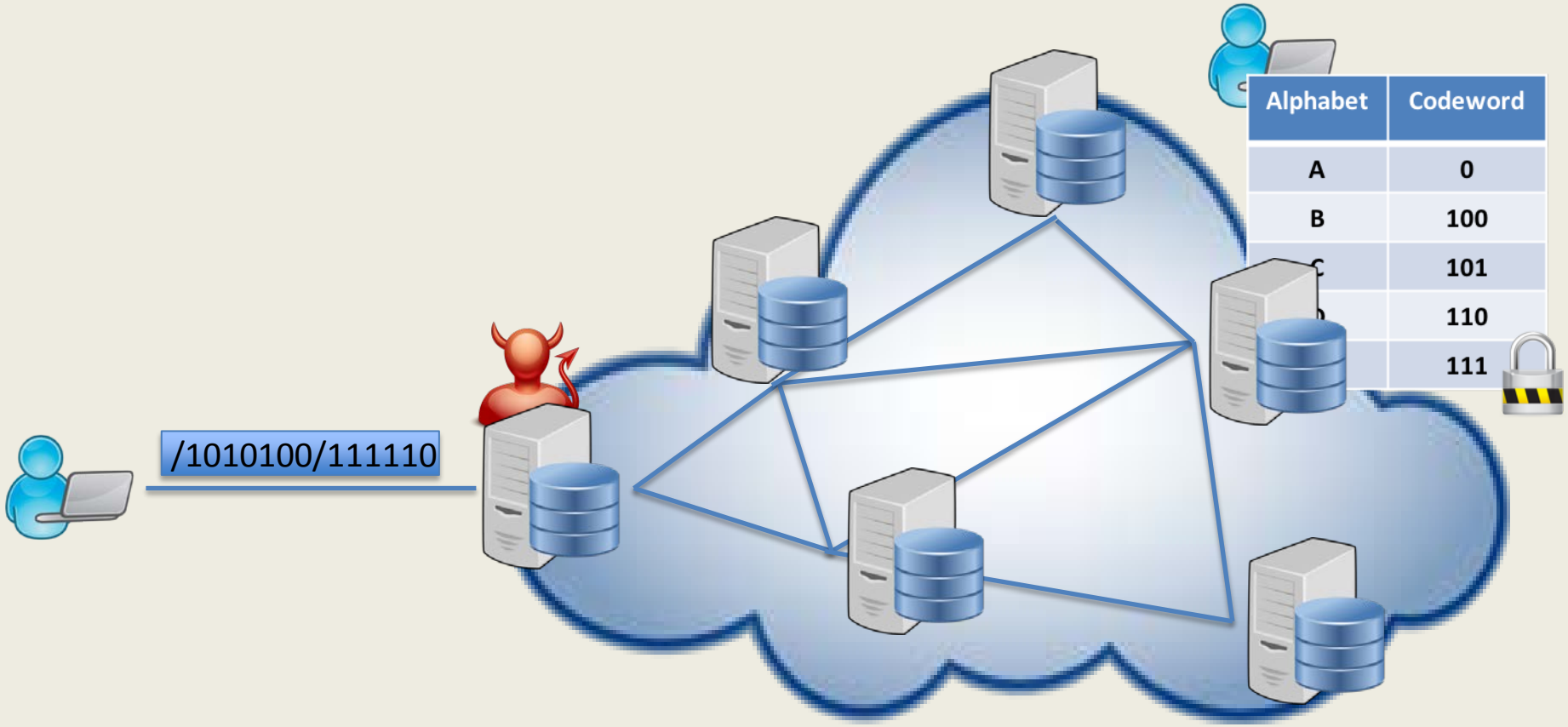
- Huffman coding[†] leverages the frequency of the source message symbols for data compression.
- Example: The frequency of the alphabet in the source message and the corresponding tree.

Alphabet	Frequency	Codeword
A	24	0
B	12	100
C	10	101
D	8	110
E	8	111



[†]D. Huffman et al. A method for the construction of minimum redundancy codes. Proc. IRE, 40(9):1098–1101, 1952.

Vanilla Huffman Coding is not Secure!

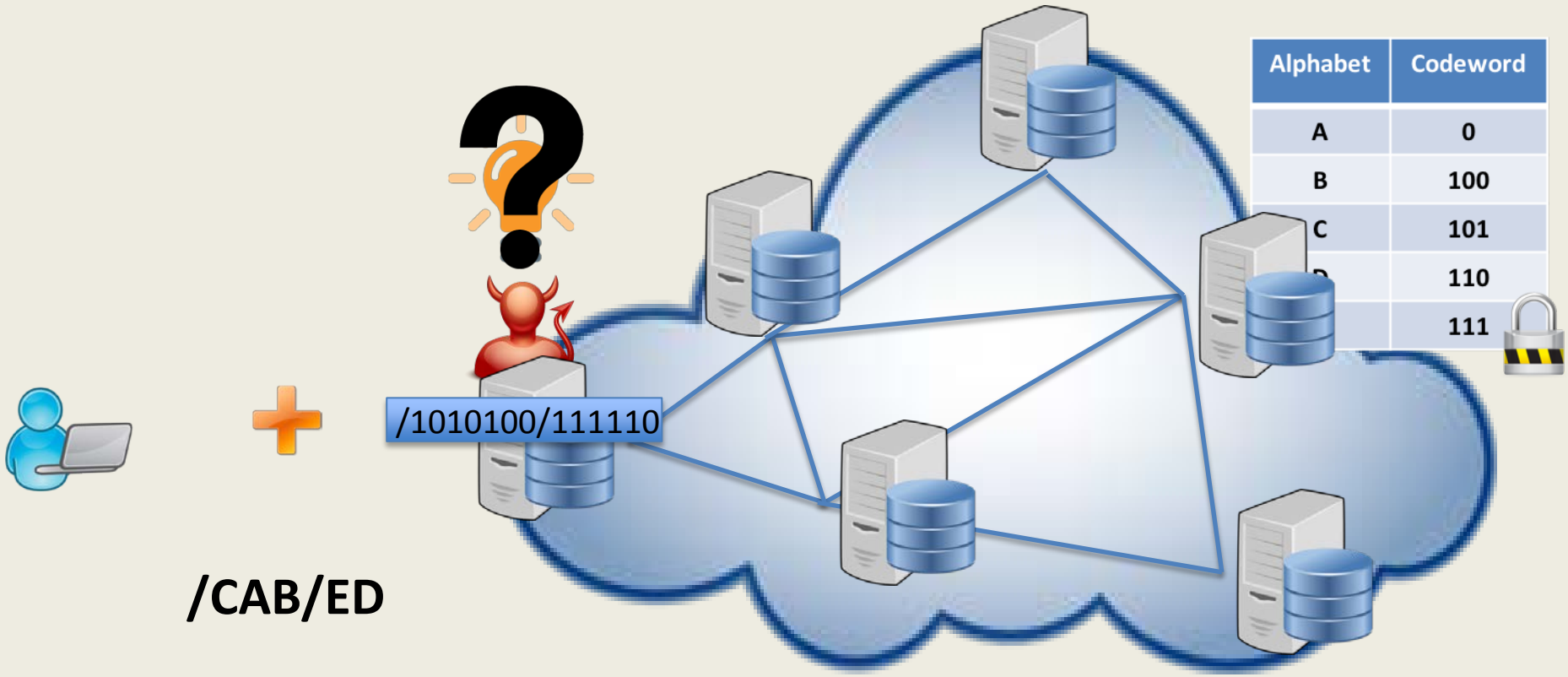


Alphabet	Codeword
A	0
B	100
C	101
D	110
E	111

Plaintext Interest
/CAB/ED

Encoded Interest
/1010100/111110

Vanilla Huffman Coding is not Secure!



Alphabet	Codeword
A	0
B	100
C	101
D	110
E	111

How to Augment Vanilla Approach?

- Increasing the number of unique coding tables.

- Assigning each client a unique coding table that can be changed at certain frequency (as needed).

- Sources of randomness:
  - **The Huffman tree structure.**
  - **The conventional key.**
  - The alphabet placement on leaf nodes.

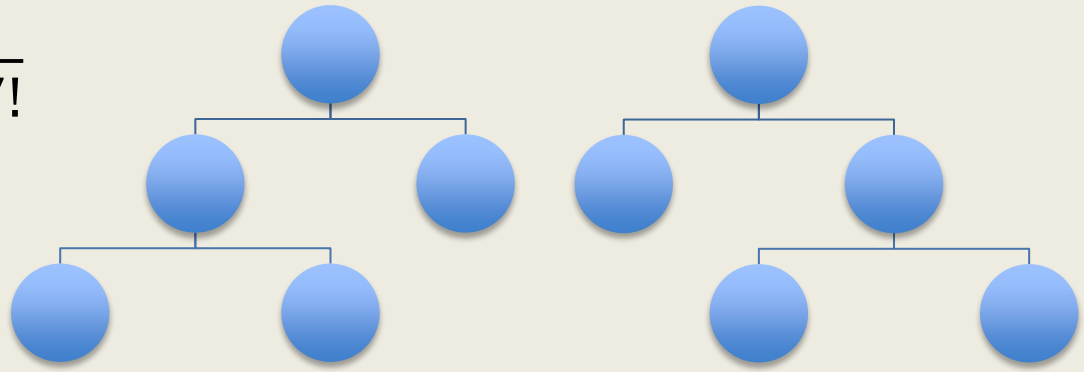
Preliminaries (Tree Structure)

Number of mutually independent full binary trees with N leaves (N is the alphabet size) is the *(N-1)*th Catalan number.

$$C_{N-1} = \frac{\big(2 \times (N-1)\big)!}{N! \times (N-1)!} \approx \Omega\left(\frac{4^N}{N^{3/2}}\right)$$

$$For\ N = 3:\ C_{N-1} = C_2 = \frac{4!}{3! \times 2!} = 2$$

$$For\ N = 128: \frac{254!}{128! \times 127!}$$

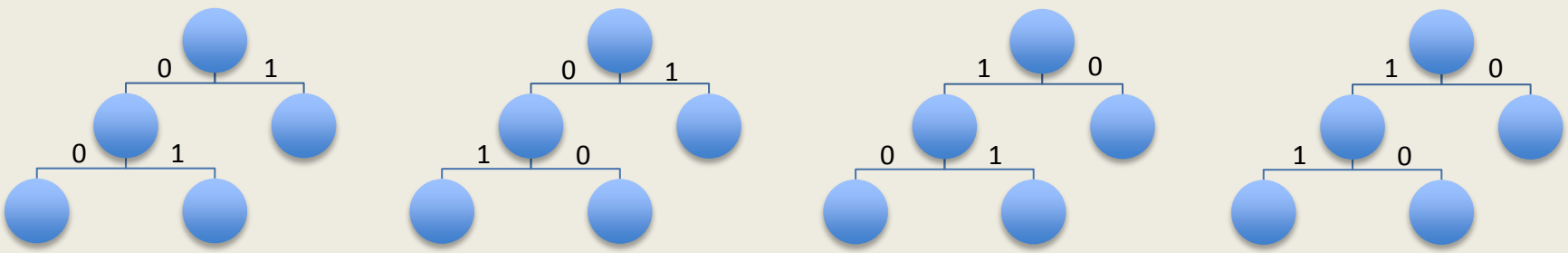


Preliminaries (Conventional Key)

Number of mutation trees for a binary tree with N leaves and N-1 internal nodes is $2^{(N-1)}$ (each mutation tree is equivalent to a key). The key is the BFS traversal of the tree.

$$For\ N = 3:\ 2^{N-1} = 2^{3-1} = 4$$

$$For\ N = 128:\ 2^{127}$$



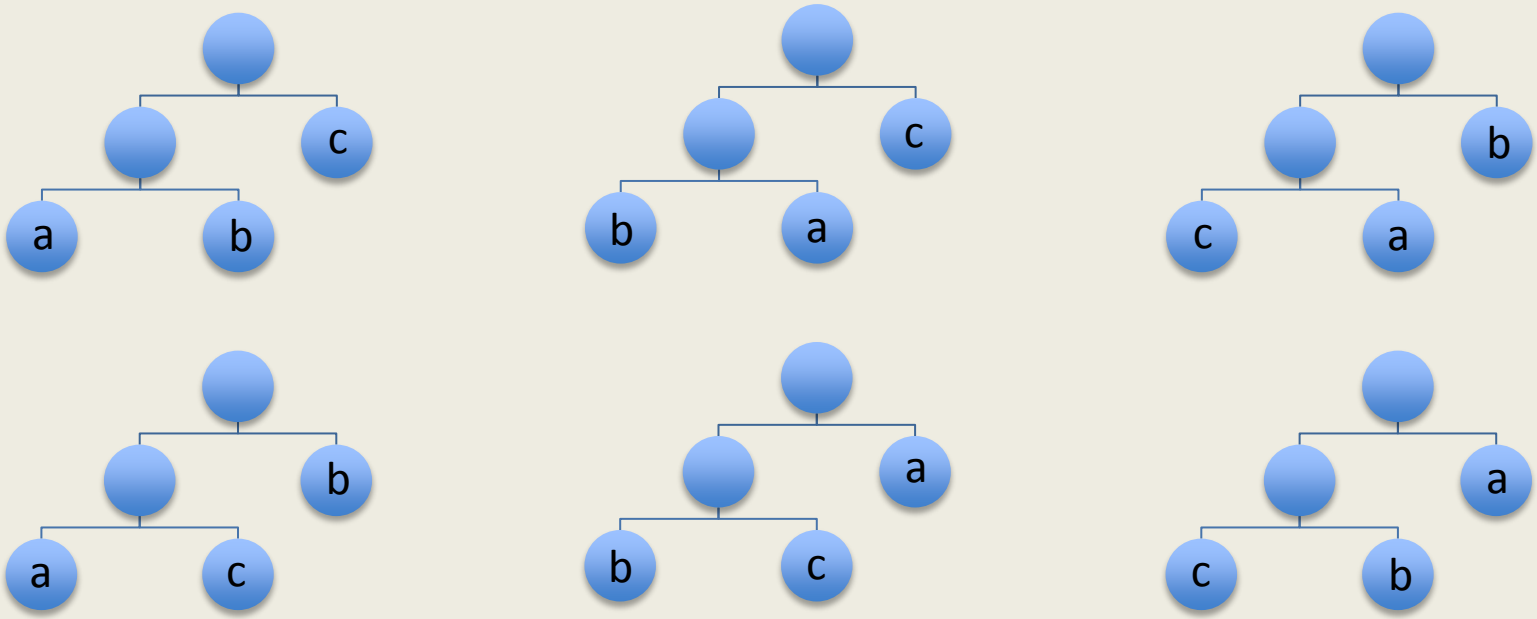
0101 **0110** **1001** **1010**

Preliminaries (Alphabet Placement)

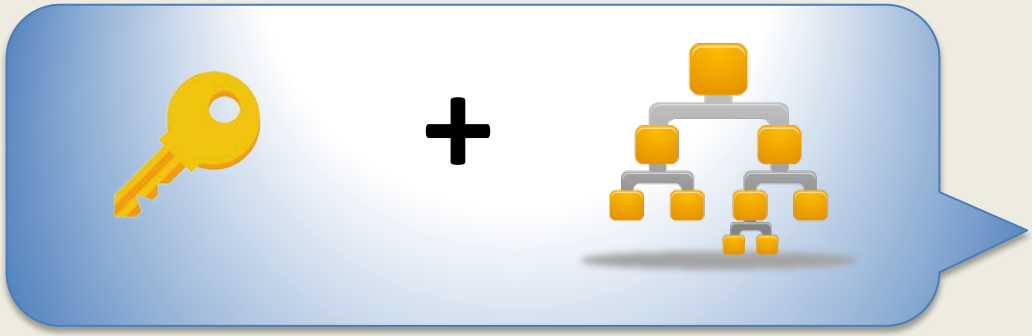
Number of different alphabet placements on a tree with N leaves is equal to $N!$.

$For\ N = 3:\ N! = 3! = 6$

$For\ N = 128:\ N! = 128!$



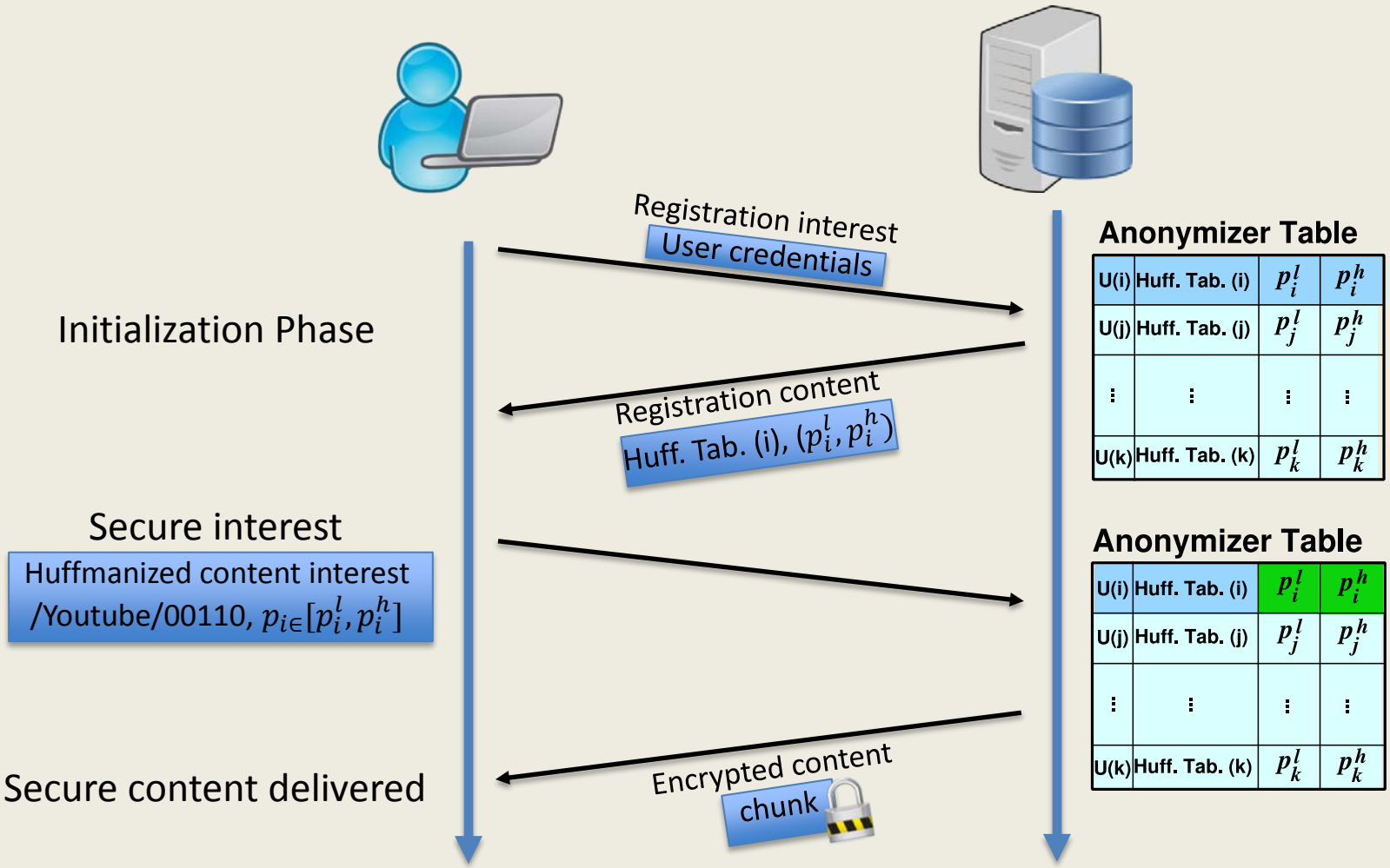
A combination of these results in a table.



Combine tree structure and the key perturbation to create different Huffman encoding tables (this study).

Assign one each to each client.

Communication Flow in our framework



Communication Flow in words

- Initialization Phase
  - Client interest with credentials.
  - Coding table generation and pseudonym assignment by the anonymizer.
  - Sharing the coding table and the pseudonym range with the client.
- Secure content request
  - Interest creation with encoded name and an in-range random pseudonym.
- Secure content response
  - Client lookup by the anonymizer through pseudonym.
  - Interest decoding by the corresponding table.
  - Extended PIT entry creation for the decoded interest.
  - Content retrieval from the network and forwarding to the client.

Privacy Evaluation of the Framework.

- *Information-theoretic* secrecy
- *Guessing-entropy* based secrecy
- *Breakability* due to brute force

Our Information-Theoretic Secrecy

- The per symbol entropy for the alphabet size of N is:

$$H(X) = -\sum_{k=1}^{N} p(x_k)\log\big(p(x_k)\big) = \log(N).$$

- The selection of a mutation tree uniformly at random results in the key entropy as:

$$H(K) = -\sum_{i=1}^{2^{N-1}} p(i)\log\big(p(i)\big) = N - 1.$$

- The entropy of a random tree structure selection is:

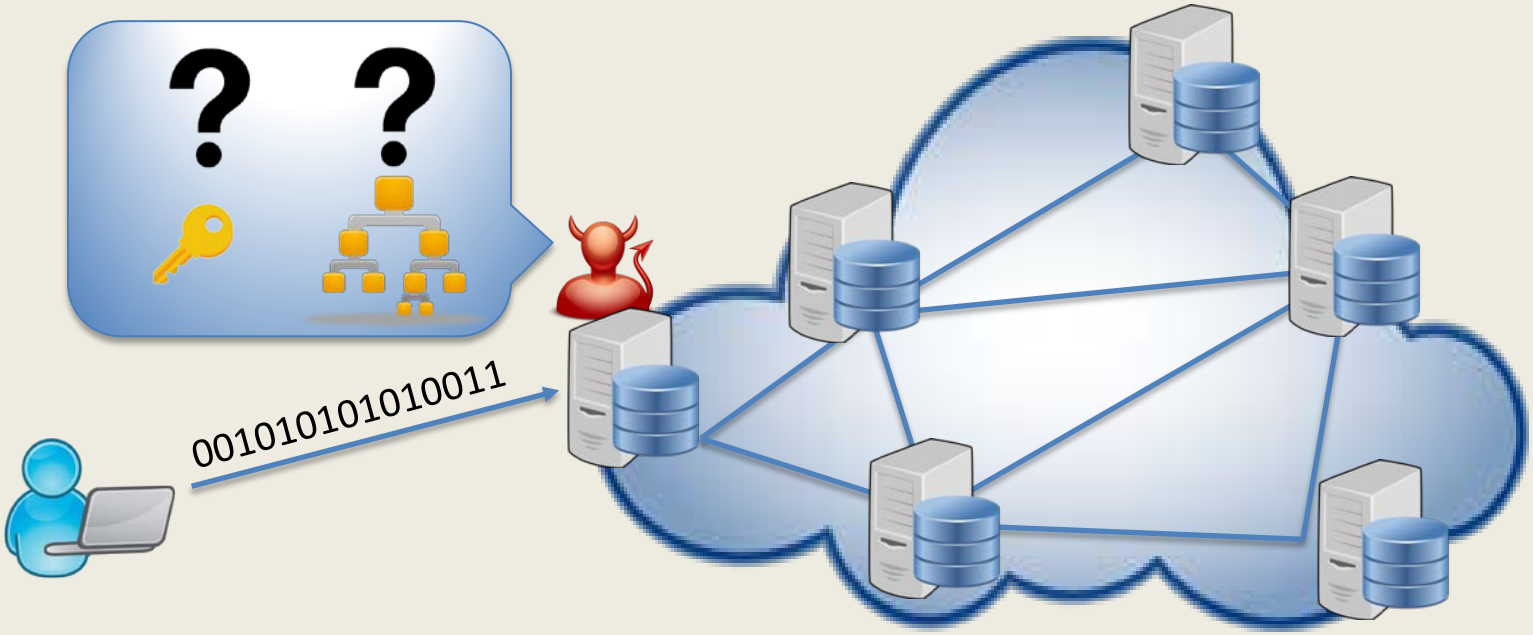
$$H(T_r) = -\sum_{j=1}^{\left(\frac{4^N}{N^{3/2}}\right)} p(j)\log\big(p(j)\big) = 2N - {}^{3}/_{2}\log(N).$$

Attack Scenario 1 and information leakage

- **Scenario 1 (TKU).** The tree structure and the key are unknown:

$$I(M^k; \underline{Z}) = H(M^k) - H(M^k|\underline{Z}) = k \log N - 3N + {}^{3}/_{2} \log N + 1.$$

$$For\ N = 256 \Rightarrow k \leq 94.3\ (largest\ name)$$

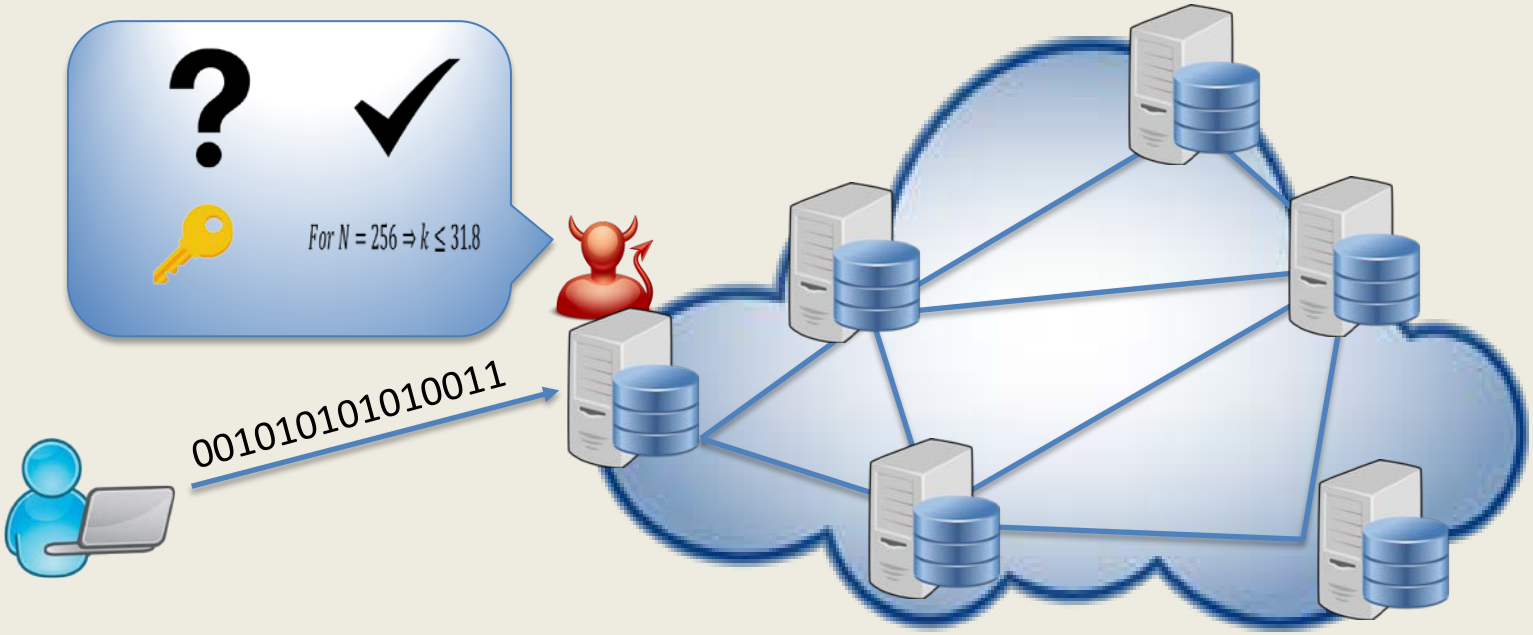


Attack Scenario 2 and information leakage

- **Scenario 2 (TK-KU).** Tree structure known, but the key is unknown:

$$I(M^k; \underline{Z}) = H(M^k) - H(M^k|\underline{Z}) = k \log N - N + 1.$$

$$For\ N = 256 \Rightarrow k \leq 31.8$$

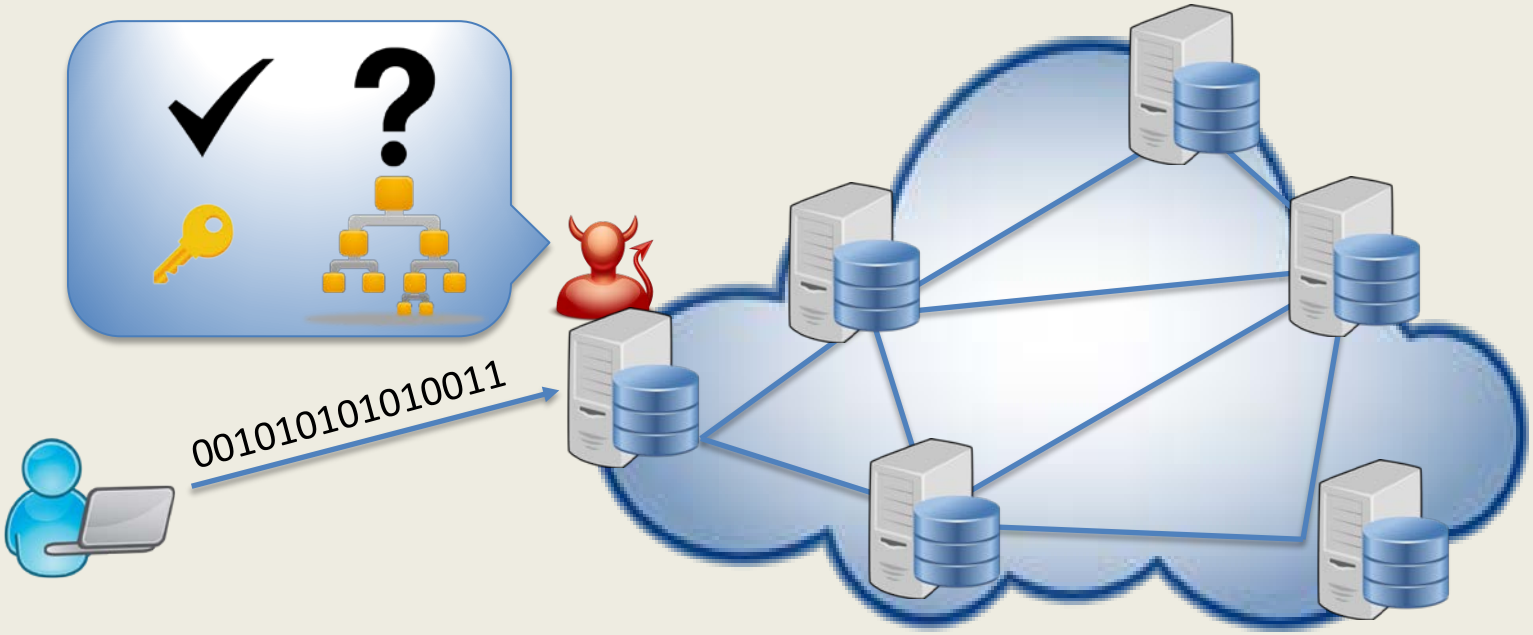


Attack Scenario 3 and information leakage

❖ **Scenario 3 (TU-KK).** Tree structure unknown but key known:

$$I(M^k; \underline{Z}) = H(M^k) - H(M^k|\underline{Z}) = (k + {}^3/_2)\log N - 2N.$$

$$For\ N = 256 \Rightarrow k \leq 62.5$$



Information Leakage Threshold

Scenario	N=32	N=64	N=128	N=256	N=512
TKU	**17.5**	**30.3**	**53.2**	**94.3**	**169.1**
TK-KU	**6.2**	**10.5**	**18.1**	**31.8**	**56.7**
TU-KK	**11.3**	**19.8**	**35.07**	**62.5**	**112.2**

Maximum possible source message length *k* (in symbols) for perfect secrecy in i.i.d. messages.

Leakage $\not\Rightarrow$ privacy breach

AES-128 leaks after 128 bits of encrypted message!!

What is the chance attacker can get lucky?

Guessing entropy: The expected number of guesses required by the attacker to ascertain the correct source message from an encoded message.

$$E[G(M^k|\underline{Z})] \geq 2^{H(M^k|\underline{Z})-2} + 1.^{\dagger}$$

Hence,

- **Scenario 1 (TKU):**

$$E[G(M^k|\underline{Z})] \geq 2^{(3N-{}^{3}/_{2}\log(N)-3)} + 1.$$

- **Scenario 2 (TK-KU)**:

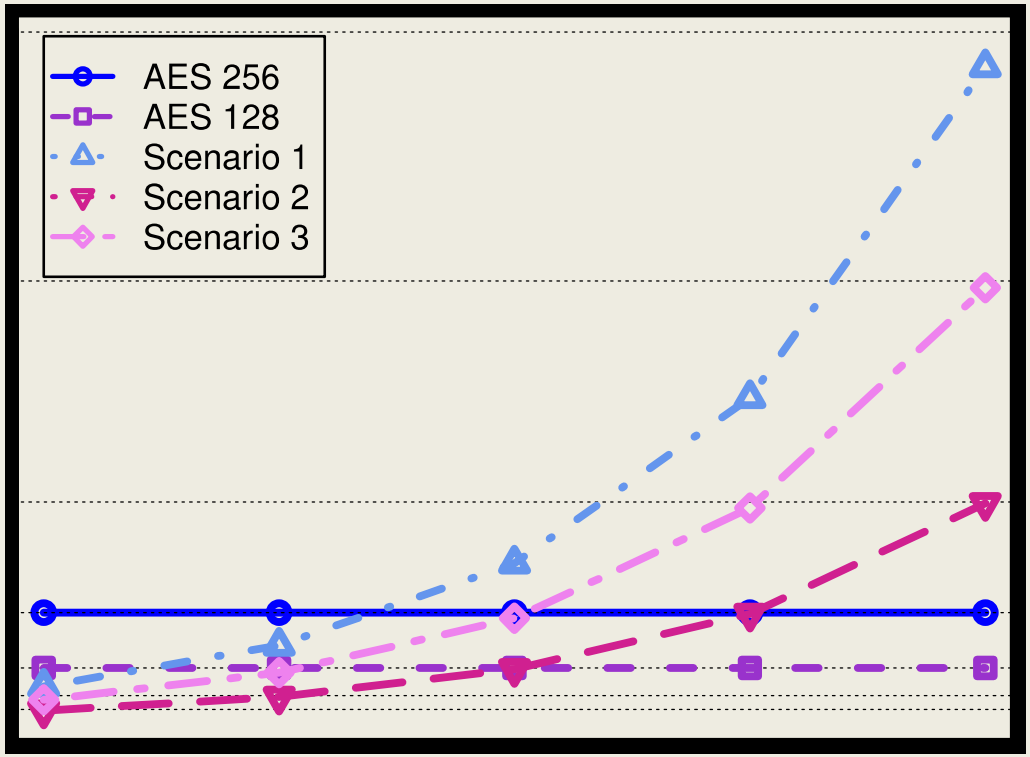
$$E[G(M^k|\underline{Z})] \geq 2^{(2N-{}^{3}/_{2}\log(N))-2} + 1.$$

- **Scenario 3 (TU-KK)**:

$$E[G(M^k|\underline{Z})] \geq 2^{(N-1)-2} + 1.$$

[†]G. Smith. On the foundations of quantitative information flow. In Foundations of Software Science and Computational Structures, pages 288–302.

Guessing Entropy Comparison.



The lower bound on the guessing entropy.

Computation Secrecy and Breakability

Using a brute-force approach for identifying the key and the tree structure, on average, the attacker has to verify half of all the possible coding tables.

$$Total\ \#\ of\ coding\ tables\ \approx \underbrace{\frac{\big(2 \times (N-1)\big)!}{N! \times (N-1)!}}_{\text{Number of structures}} \times \underbrace{2^{N-1}}_{\text{Number of mutation trees}}$$

Outline

- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

Huffman Encoding is much quicker!

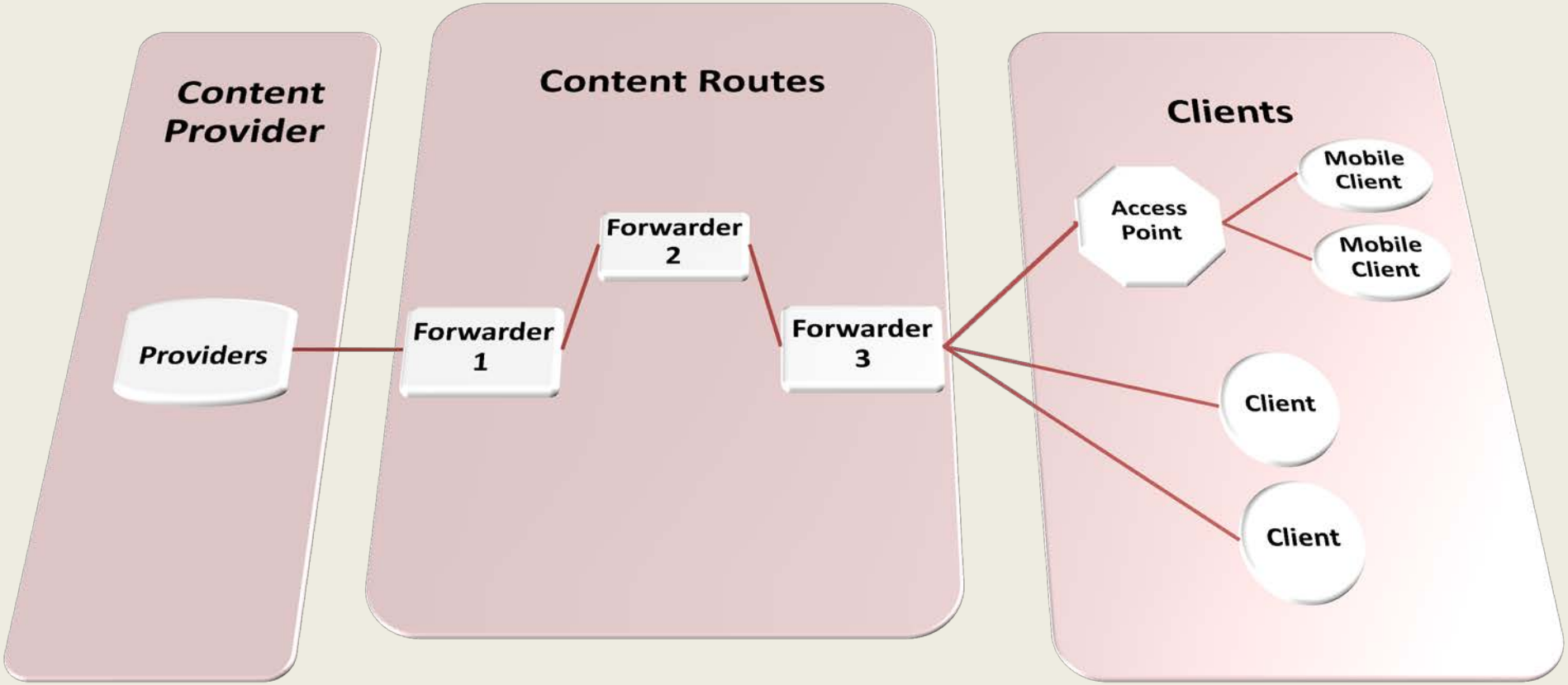
Encoding Scheme	Encoding (Sec)	Decoding (Sec)
Unix aescrypt (L)	0.050	0.021
AES openssl (L)	0.010	0.008
Huffman Coding (L)	0.004	0.004
Huffman* (L)	0.000034	0.000027
AES openssl (M)	0.041	0.023
Huffman Coding (M)	0.006	0.005
SHA-1 (L)	0.000093	0.000093

(L): AMD Turion, 2.4 GHz, dual core laptop.
(M): Nexus 5 smartphone.

Testbed Setup

- **Content providers**
  - *Three* 2.4 GHz Intel Core i7, 8 GB RAM nodes.
- **Content forwarder**
  - *Four* 2.5 GHz Intel Core 2 Quad, 3.8 GB RAM nodes.
- **Clients**
  - *Six* 1.66 GHz Intel Centrino Duo, 2.5 GB RAM nodes (Stationary)
  - *One* 3 GHz Intel Xeon Quad Core, 2 GB RAM nodes (Stationary)
  - *Three* Nexus 4 mobile phones (1.5 GHz Quad core, 2GB RAM)
  - *One* Nexus 5 mobile phone (2.3 GHz Quad core, 2GB RAM)
- **Access point**
  - 802.11 n
- **Switches**
  - 100 Mb/s switches

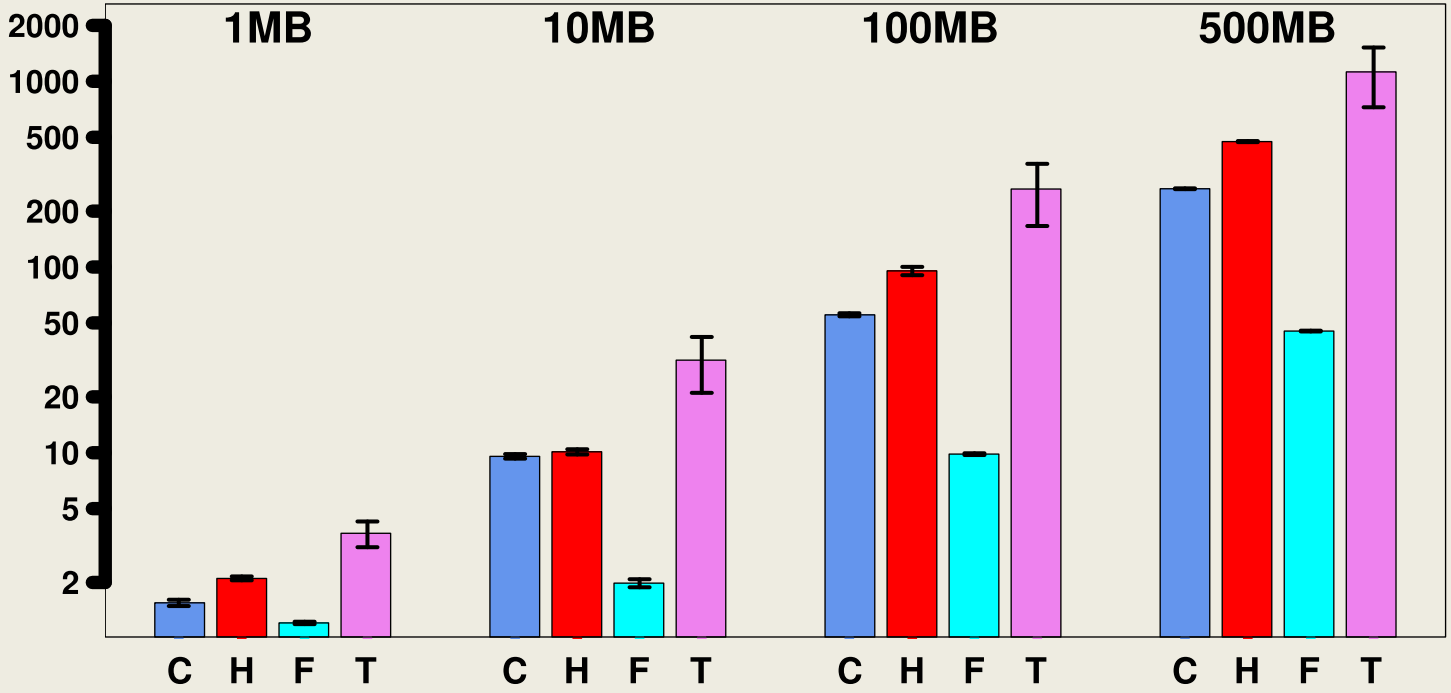
Testbed Setup



Test Setup

- Using the **CCNx-0.7** *ccnputfile* and *ccngetfile* tools to store/retrieve contents to/from the content provider.
- One client requests the content from the provider.
- Caching was disabled on all the routers for the sake of fair comparison.
- Various content object sizes: {1 MB, 10 MB, 100 MB, and 500MB}.
- We compare latency and protocol overhead over baseline CCN, our anti-censorship framework (CCN+Huffman), FTP, and Tor (The Onion Routing).
- Tor includes three layers of encryption at the forwarders.
- The results were averaged over 100 runs.

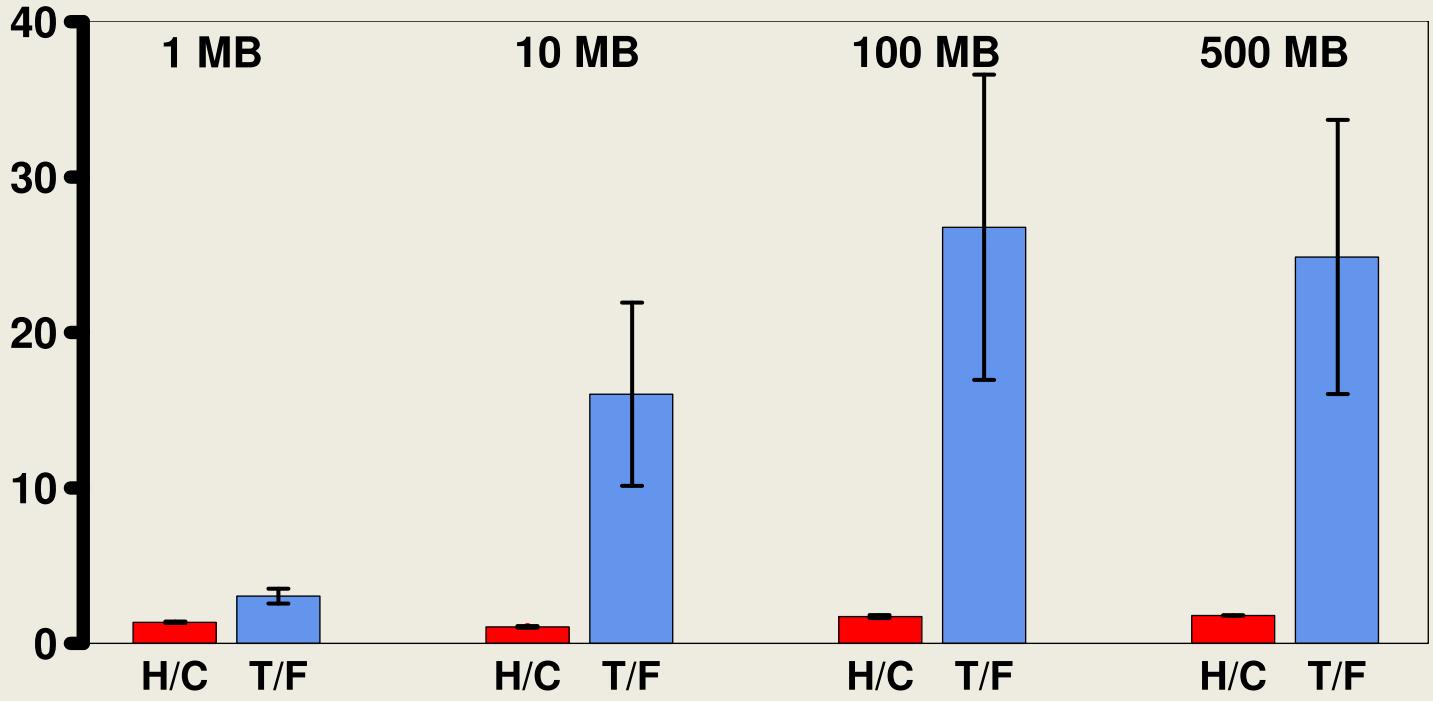
Comparable latency between Huffman and CCN (log-scale graph).



Average download time comparison on the laptop clients.

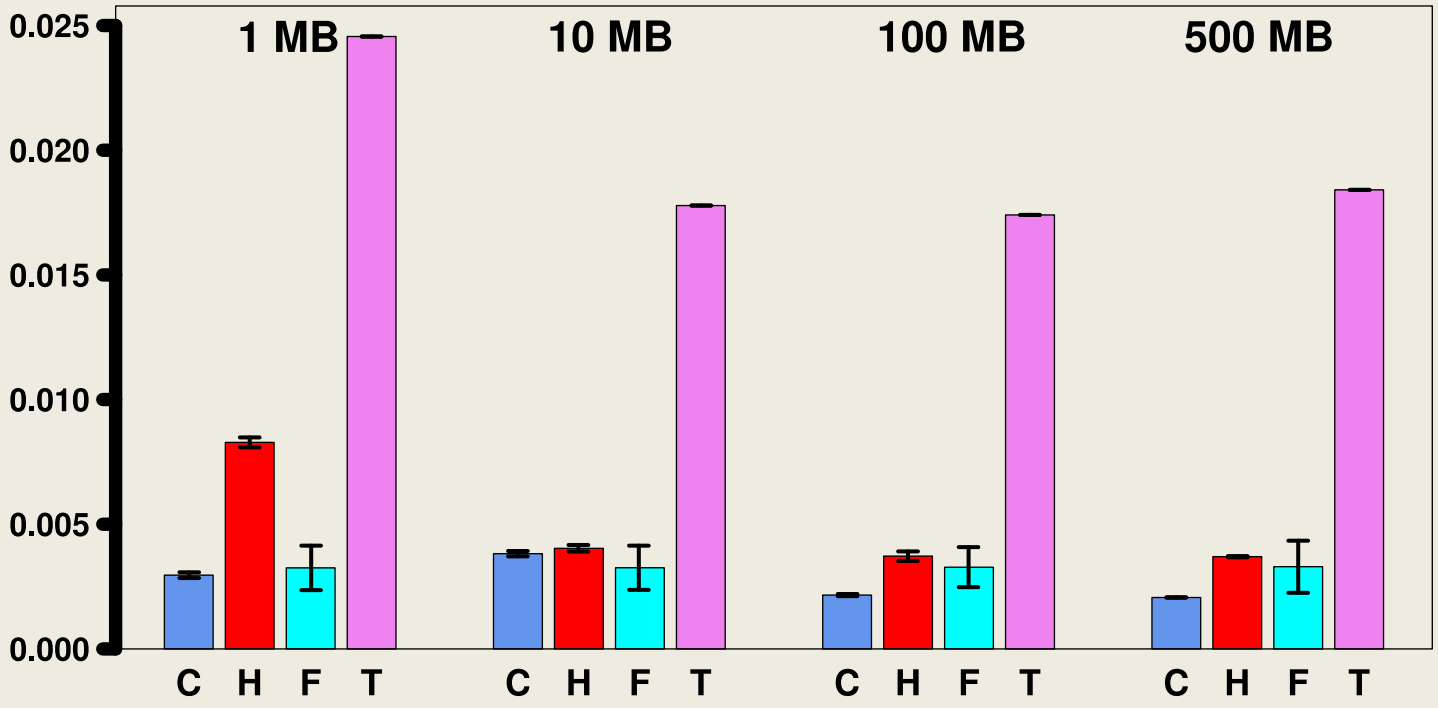
C: Baseline CCN
H: CCN+Huffman
F: FTP
T: Tor

Latency overhead ratio of Tor is dramatically higher for larger content.



Latency overhead comparison between Huffman over CCN (H/C) and Tor over FTP (T/F) on laptops.

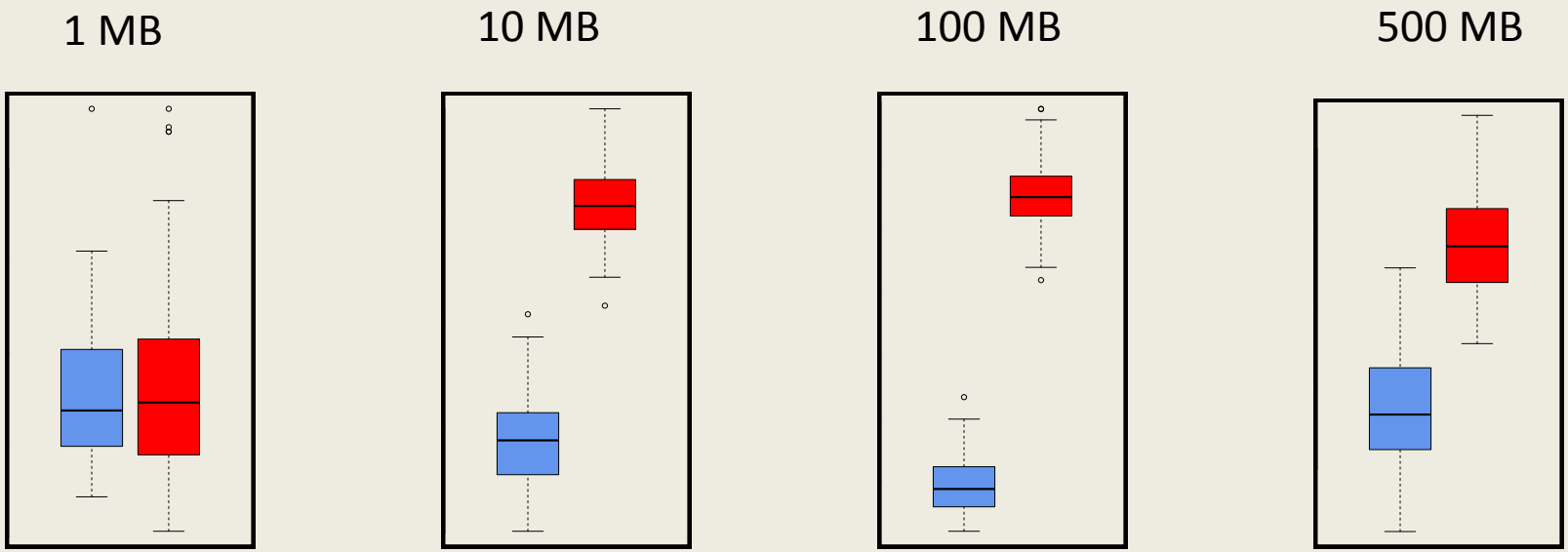
Layers of encryption increase Tor's cost in comparison to Huffman.



Estimated average round trip time on the laptop clients.

C: Baseline CCN
H: CCN+Huffman
F: FTP
T: Tor

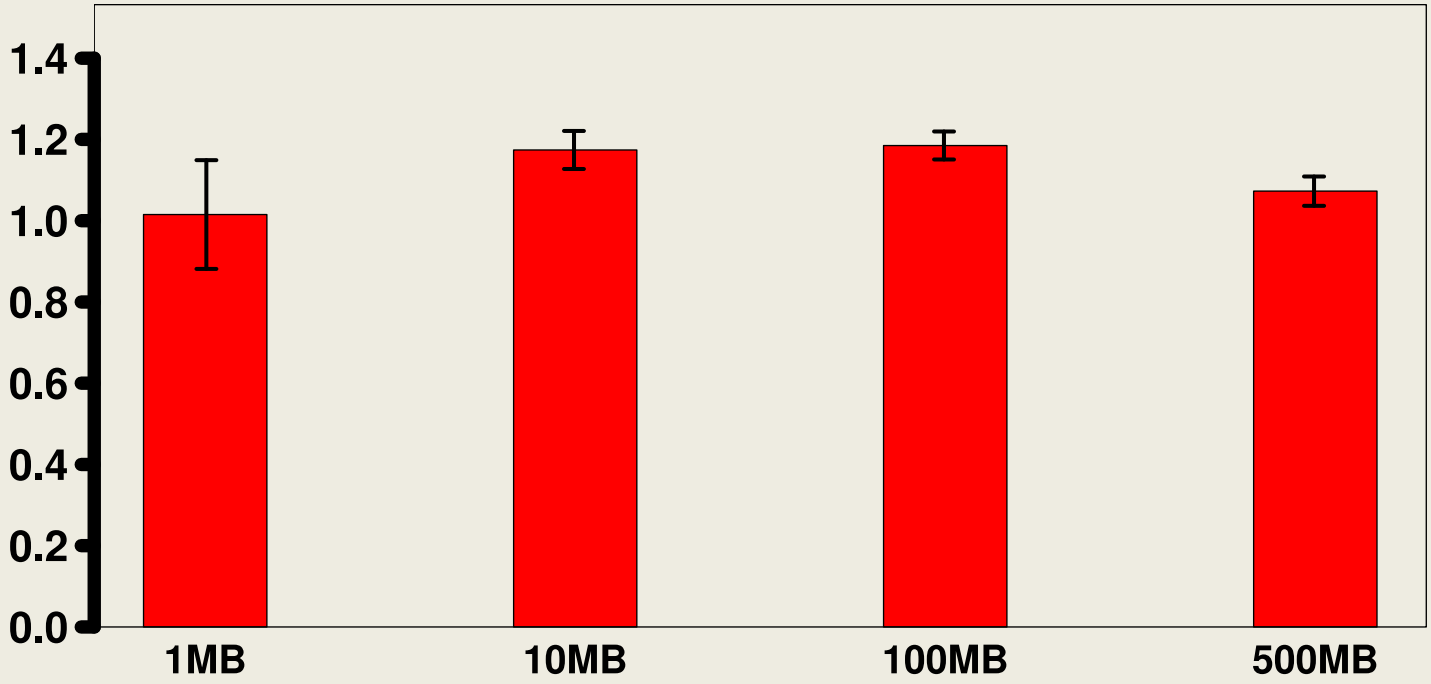
Smartphone clients experience higher latency due to the lossy channel.



Average download time comparison on the smartphone client.

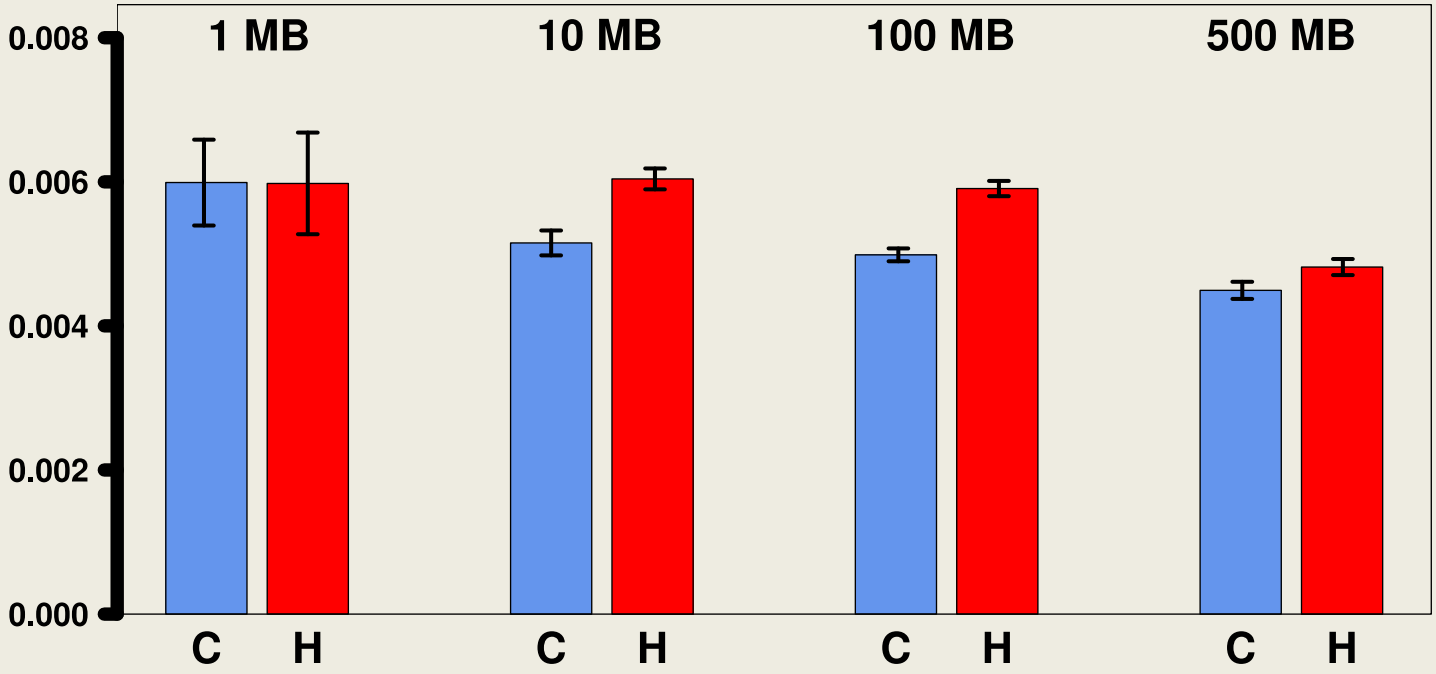
C: Baseline CCN
H: CCN+Huffman

Modest Huffman overhead ratio (1.2) for smartphone clients.



Latency overhead of Huffman (H/C) in smartphone client.

Lightweight Huffman encoding and decoding maintain comparable RTT.



Estimated average round trip time on the smartphone client.

C: Baseline CCN
H: CCN+Huffman

Outline

- Introduction and Motivation
- Problem Definition
- Models and Assumptions
- Framework Design
- Experimental Results
- Conclusions and Future Work

Conclusions and Future Work

- In this article, we present a lightweight anti-censorship framework for ICN clients, applicable to mobile users.
- We proved the conditions and thresholds for perfect secrecy as well as breakability analysis of the proposed framework over AES.
- For future, we will analyze the trade-off between the privacy and caching by decoupling the anonymizer from the provider.
- We will also investigate the design of an algorithm for a seamless dynamic coding table updates.

- Take-away: May be we do not need to use Tor. We propose something faster for smartphones, IoTs, mobile nodes, etc.

Thank you!

Email:misra@cs.nmsu.edu

Research funded by the US National Science Foundation and the US Dept. of Defense.