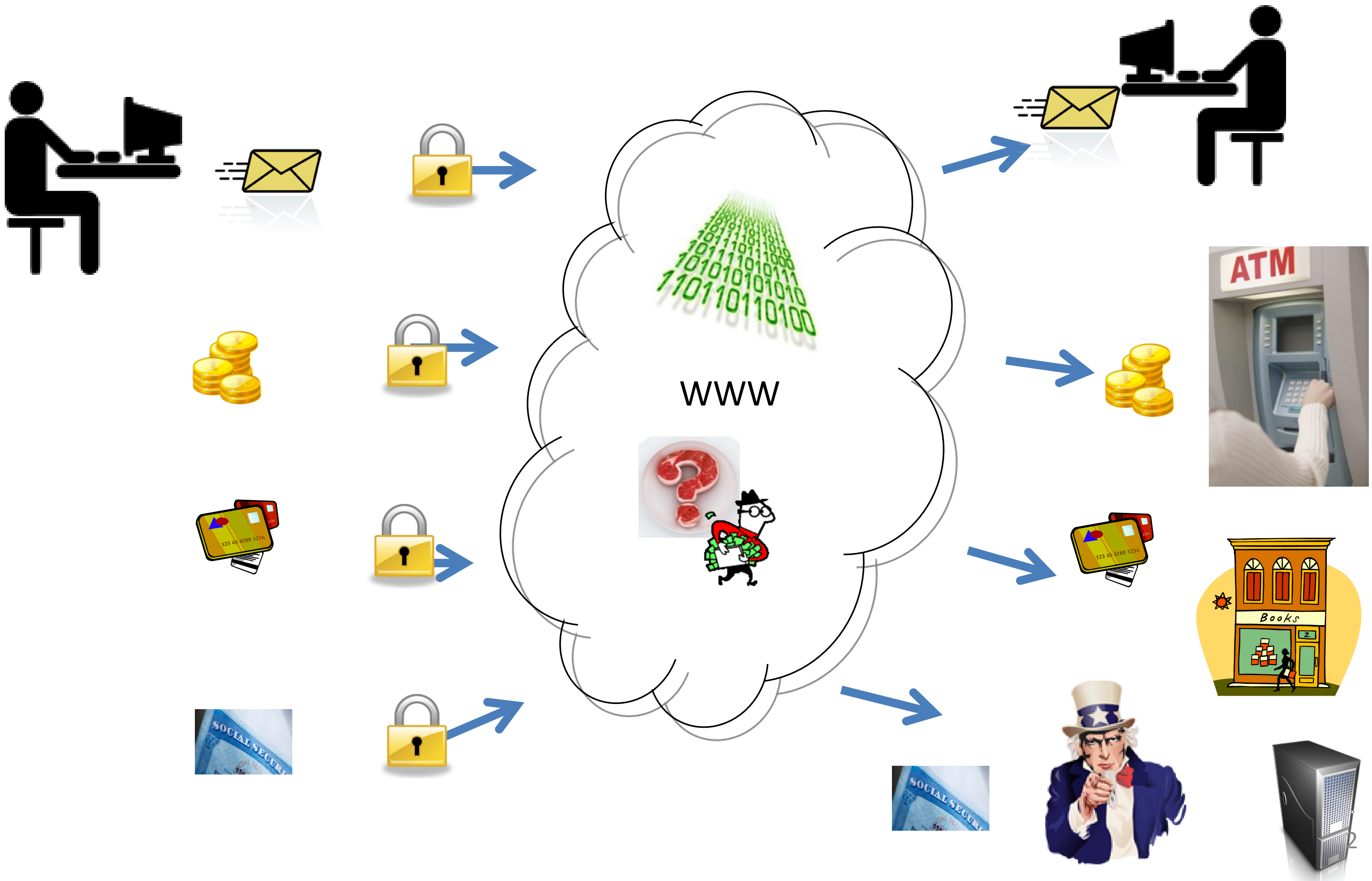
LibFTE: A User-Friendly Toolkit for Constructing Practical Format-Abiding Encryption Schemes

Daniel Luchaup, University of Wisconsin-Madison
Kevin Dyer, Portland State University
Somesh Jha, University of Wisconsin-Madison
Thomas Ristenpart, University of Wisconsin-Madison
Thomas Shrimpton, Portland State University

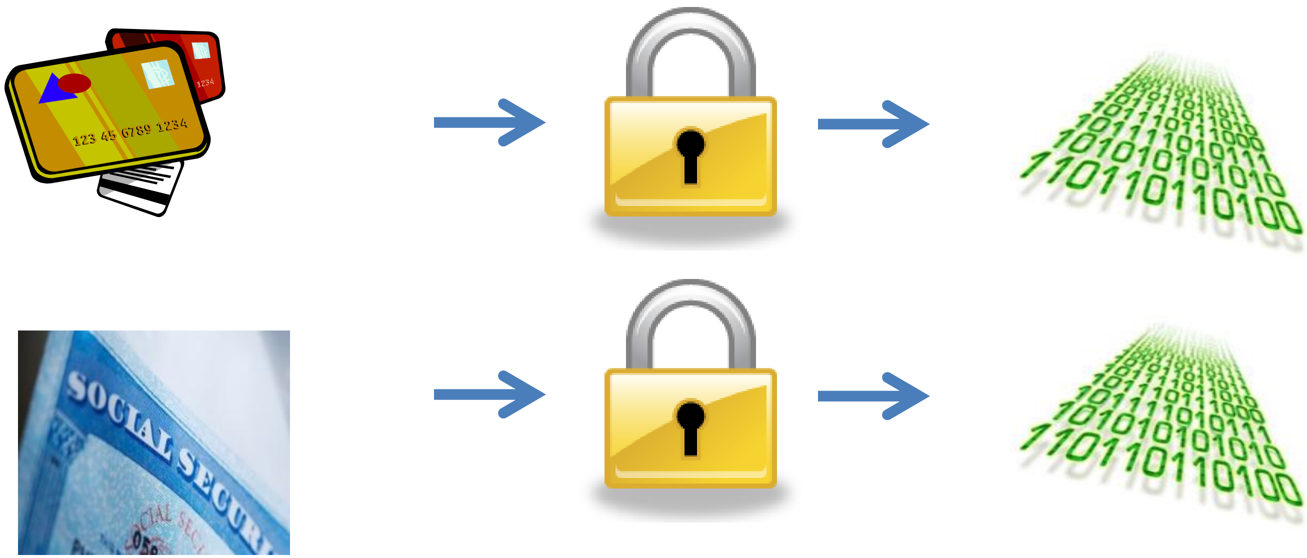
Encryption



Traditional Encryption

formatted data

unformatted sequence of bytes



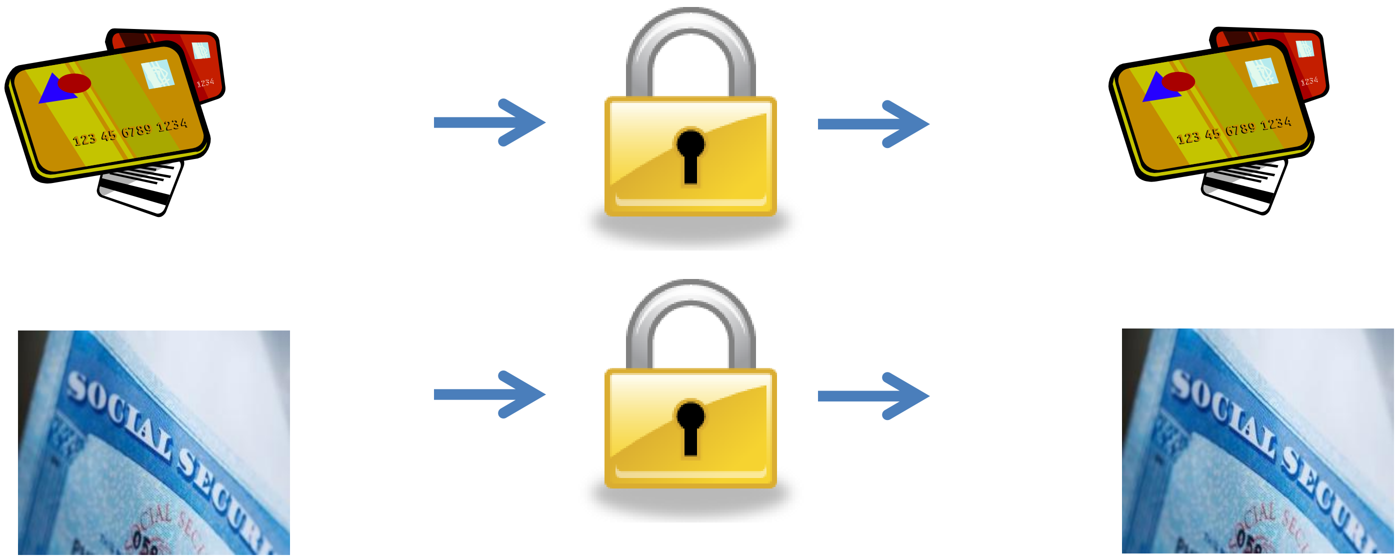
Drawback: traditional encryption does not preserve any format

Format Preserving Encryption

(Bellare et. al., 2009)

formatted data

encrypted & formatted data



FPE: plain text and cipher text have similar format

Format Preserving Encryption

(Bellare et. al., 2009)

formatted data

encrypted & formatted data

John Doe 1234567890
Jane Doe 2345678909

Abcd Efg 7865409889
Hlmn Opl 8099087217

John Doe 987-65-4321
Jane Doe 876-54-3210

Mike Kay 900-88-7777
Paul Kim 800-77-5555

FPE: plain text and cipher text have similar format

Format Transforming Encryption

(Dyer et. al., 2013)

formatted data

encrypted & formatted data

John Doe 1234567890

Jane Doe 2345678909

Abcd Efg 7865409889

Hlmn Opl 8099087217

Can we change the format?

John Doe 987-65-4321

Jane Doe 876-54-3210

Mike Kay 900-88-7777

Paul Kim 800-77-5555

Format Transforming Encryption

(Dyer et. al., 2013)

formatted data

encrypted & formatted data

John Doe 1234567890
Jane Doe 2345678909

Mike Kay 900-88-7777
Paul Kim 800-77-5555

John Doe 987-65-4321
Jane Doe 876-54-3210

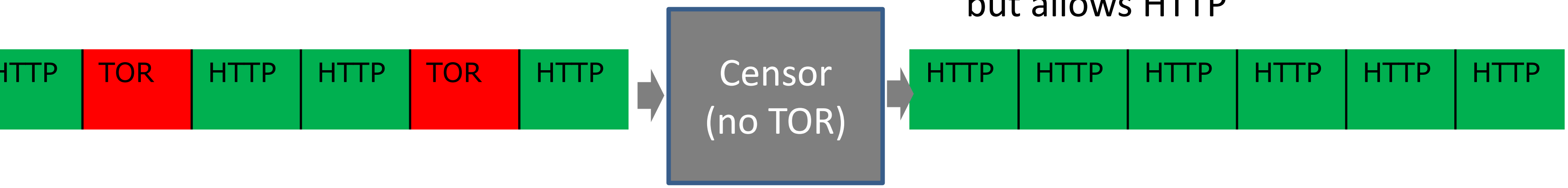
Abcd Efg 7865409889
Hlmn Opl 8099087217

FTE: plain text and cipher text have different formats

FTE: Censorship Circumvention

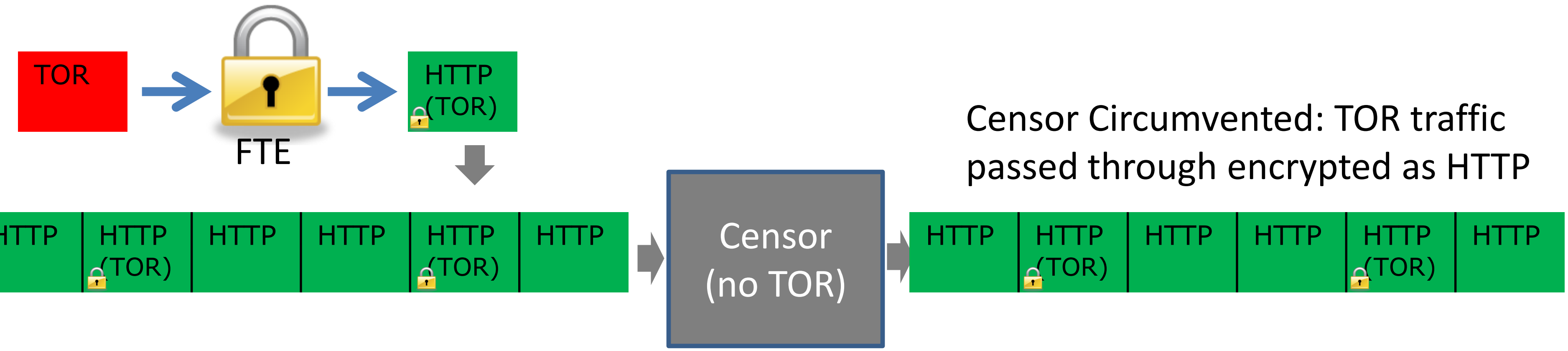
(Dyer et. al., 2013)

Censor filters out TOR traffic
but allows HTTP



FTE: Censorship Circumvention

(Dyer et. al., 2013)

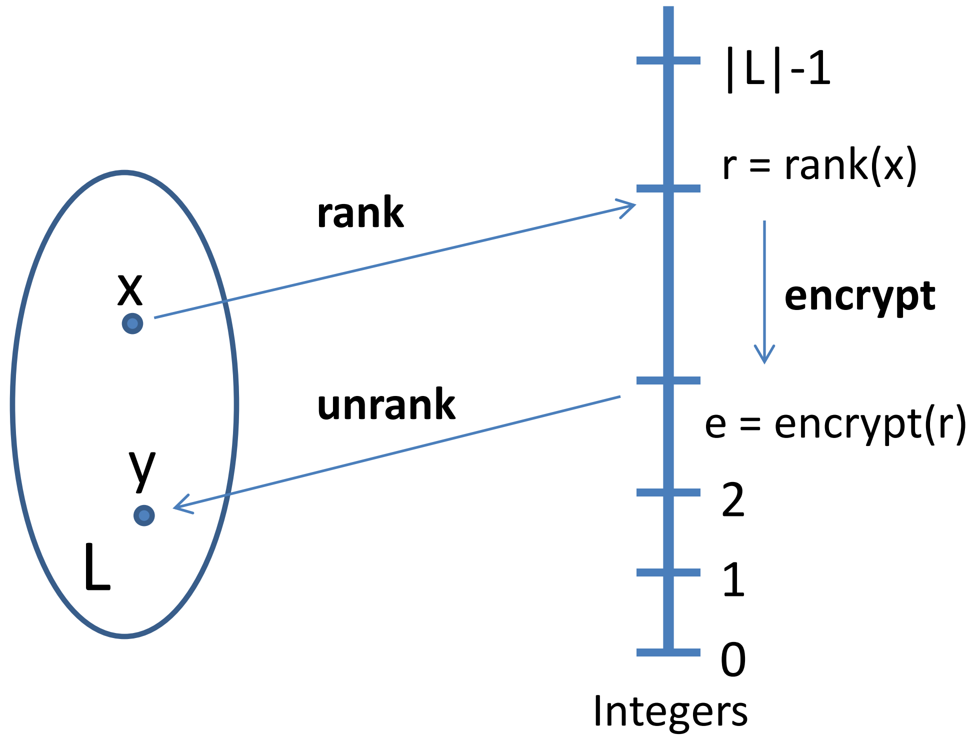


Censor Circumvented: TOR traffic passed through encrypted as HTTP

Format

- A format is simply a language L
  - regular languages L defined by regular expressions
    - [a-zA-Z]+
    - [a-zA-Z]+\ [a-zA-Z]+\ [0-9]{9}
  - Finite limits specified by the problem
    - $w \in L([a\text{-}zA\text{-}Z]^*)$, $|w| < 20$

Rank And Encipher: FPE



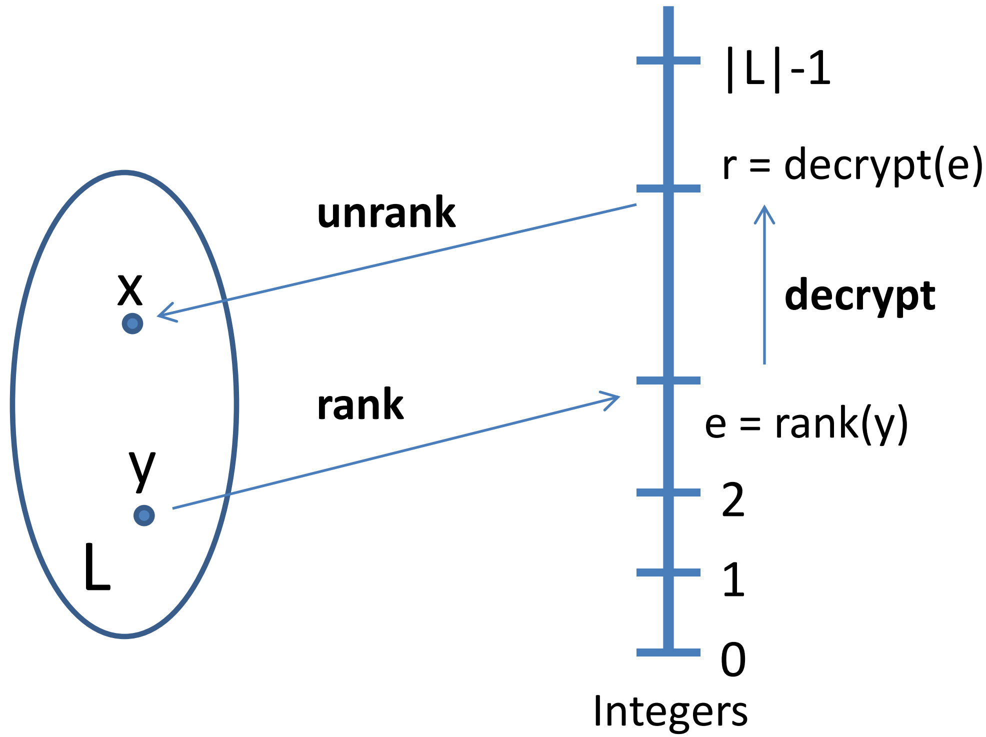
- Ranking Scheme:
  - rank: L → $Z_{|L|}$
  - unrank: $Z_{|L|}$ → L
- Integer encryption:
  - encrypt E: $Z_{|L|} \rightarrow Z_{|L|}$
  - decrypt D: $Z_{|L|} \rightarrow Z_{|L|}$

- FPE
  - encryption of x:

  y = unrank(encrypt(rank(x)))
  - decryption of y:

  x = rank(decrypt(unrank(y)))

Rank And Encipher: FPE



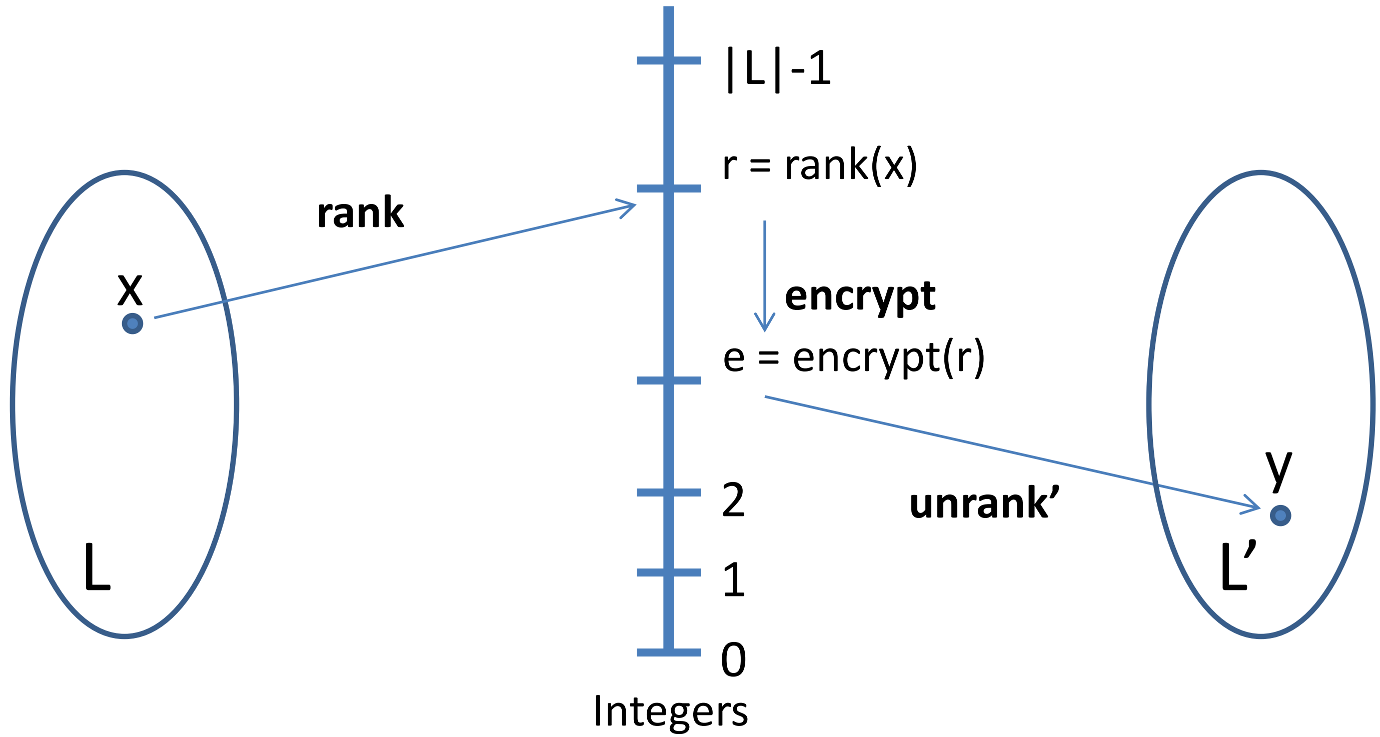
- Ranking Scheme:
  - rank: L → $Z_{|L|}$
  - unrank: $Z_{|L|}$ → L
- Integer encryption:
  - encrypt E: $Z_{|L|} \rightarrow Z_{|L|}$
  - decrypt D: $Z_{|L|} \rightarrow Z_{|L|}$

- FPE
  - encryption of x:

    y = unrank(encrypt(rank(x)))
  - decryption of y:

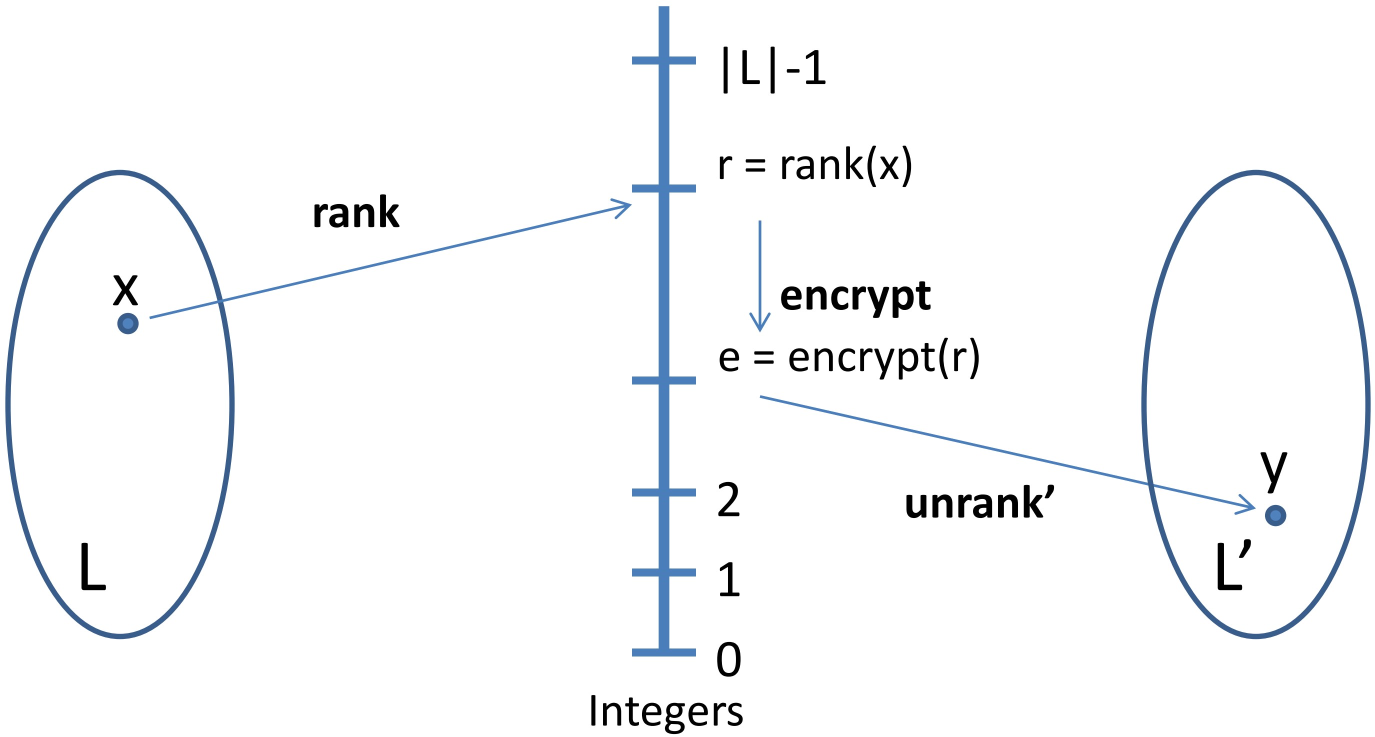
    x = rank(decrypt(unrank(y)))

Rank And Encipher: FTE



- What about |L| versus |L'| ?
  - |L| > |L'| cannot encrypt
  - |L| ≤ |L'| ok

Formatted Encryption



- Specification of language/formats L, L’
  - finite vs. infinite
- Aware of sizes |L| , |L’|
- Efficient rank/unrank
- Efficient integer encryption

Limitations of prior work

- Limited Ranking/Unranking
  - DFA based: only work with simple regex
    NFA ranking thought impossible
- No public implementation
- Awkward format specification
  - theoretically specified for fixed slices of regular lang.
- No performance analysis
- No configuration
  - Input/Output language selection
  - Reasoning about language sizes.
- Need: generic framework, simple specification, yet fast

New Work: LibFTE *(Luchaup et. al., 2013)*

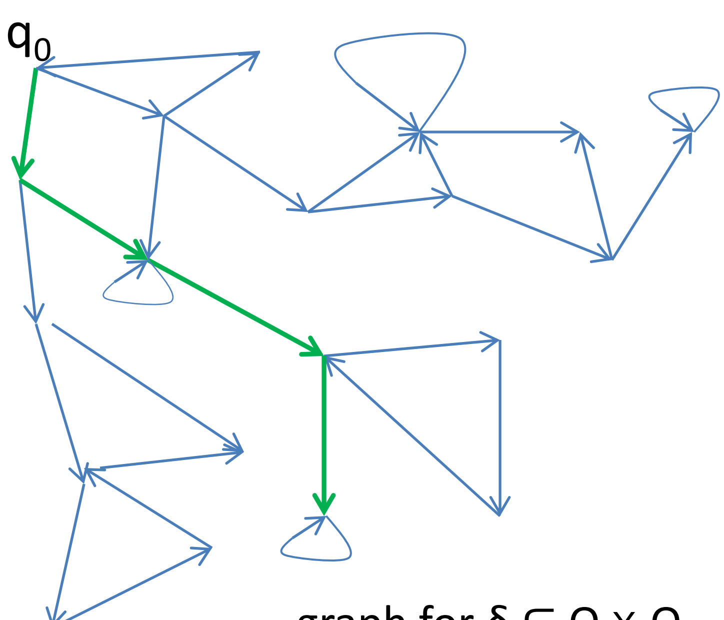
- Public implementation
- Generic framework, simple specification
  - regular expression, size ranges
- Fast
  - Improved DFA ranking
  - NFA ranking
  - Choice of DFA/NFA ranking transparent to user
- Configuration
  - Input/Output language selection
  - Tool to help user reasoning about configuration choices
- Performance analysis
- Applications:
  - In browser encryption
  - DB encryption and compression

LibFTE: NFA-based ranking

DFA:

- D = (Q, ∑, δ, q_0, F)
  - δ is deterministic
  - count accepting paths
  - unique accepting paths

  large

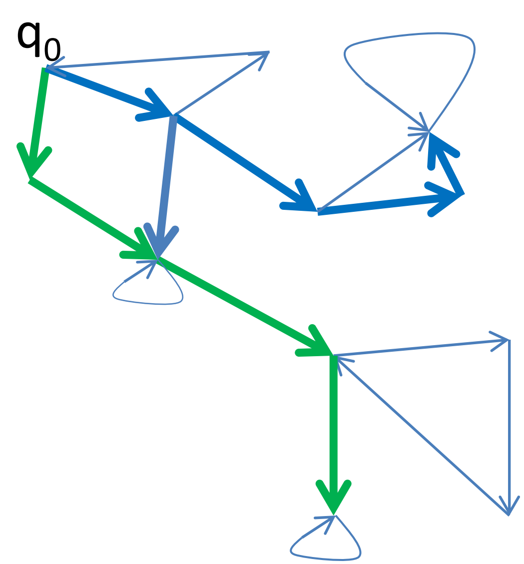


graph for δ ⊆ Q x Q   D

NFA:

- N = (Q, ∑, δ, q_0, F)
  - δ is **not** deterministic
  - count accepting paths
  - multiple accepting paths possible

  small



graph for δ ⊆ Q x Q   N

LibFTE: NFA-based ranking

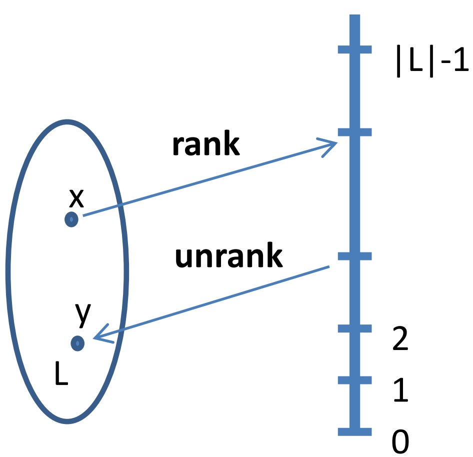
DFA:

- D = (Q, ∑, δ, q_0, F)
  - δ is deterministic
  - count accepting paths

  rank/unrank bijection

  unank(rank(x)) = x

  rank(unrank(n)) = n



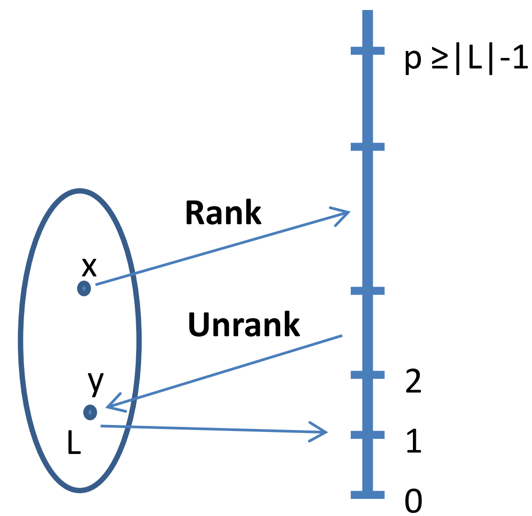
NFA:

- D = (Q, ∑, δ, q_0, F)
  - δ is **not** deterministic
  - count accepting paths

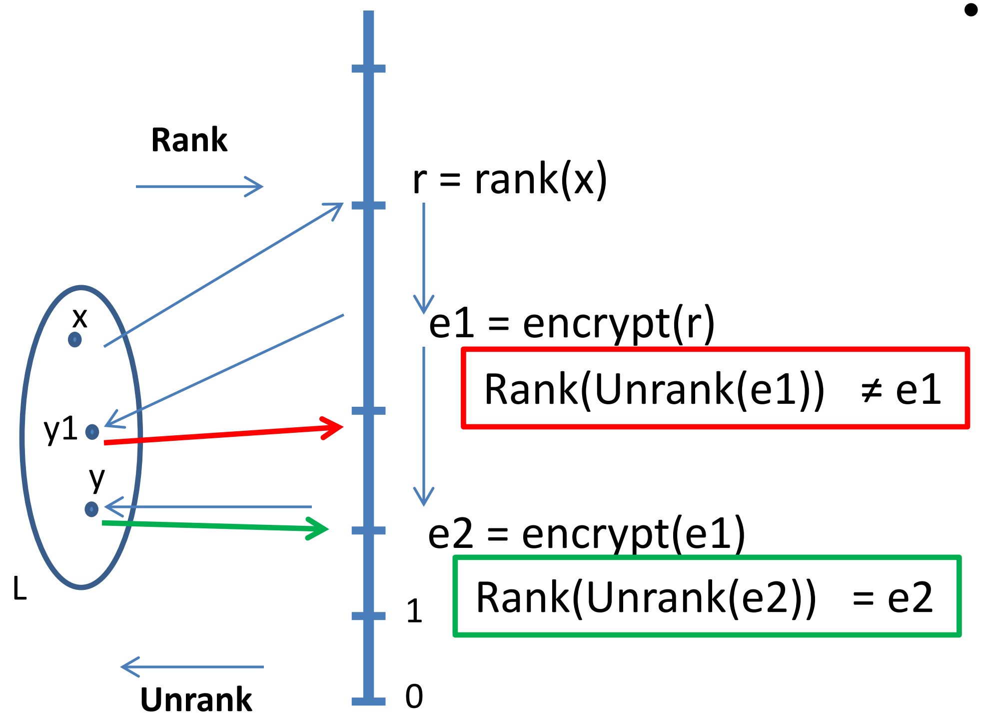
  Rank injective/ Unrank

  Unank(Rank(x)) = x

  Rank(Unrank(n)) $\stackrel{?}{=}$ n may not hold



Cycle Walking



NFA:

- D = (Q, ∑, δ, q_0, F)
  - δ is **not** deterministic
  - count accepting paths

Rank injective/ Unrank

Unank(Rank(x)) = x

Rank(Unrank(n)) $\stackrel{?}{=}$ n may not hold

Rank And Encipher with Injective Ranking

Adaptations:

- Cycle walk
- Nondeterministic encryption
- Language sizes are relevant

- Details in the paper *(Luchaup et. al., 2013)*

New Work: LibFTE

- Public implementation
- Generic framework, simple specification
  - regular expression, size ranges
- Fast
  - Improved DFA ranking
  - Relaxed ranking
  - NFA ranking
  - Choice of DFA/NFA ranking transparent to user
- Configuration
  - Input/Output language selection
  - Tool to help user reasoning about configuration choices
- Performance analysis
- Applications:
  - In browser encryption
  - DB encryption and compression

QUESTIONS?

luchaup@cs.wisc.edu