Format-**Transforming** Encryption
(more than meets the DPI)

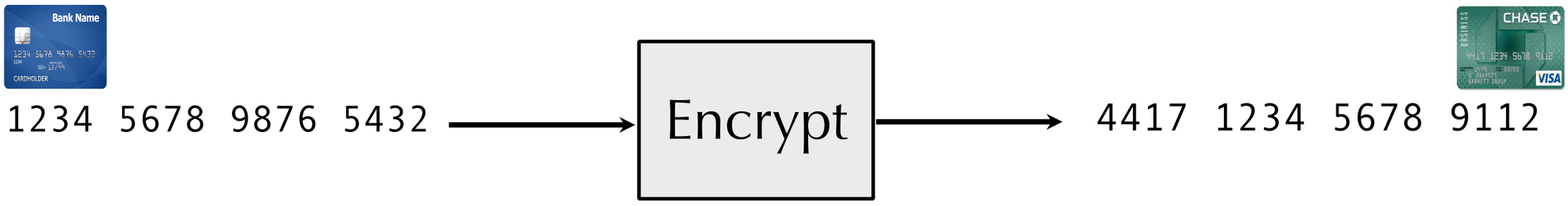
Tom Shrimpton

Florida Institute for Cybersecurity Research
University of Florida

UF Herbert Wertheim College of Engineering
Department of Computer & Information Science & Engineering
UNIVERSITY *of* FLORIDA

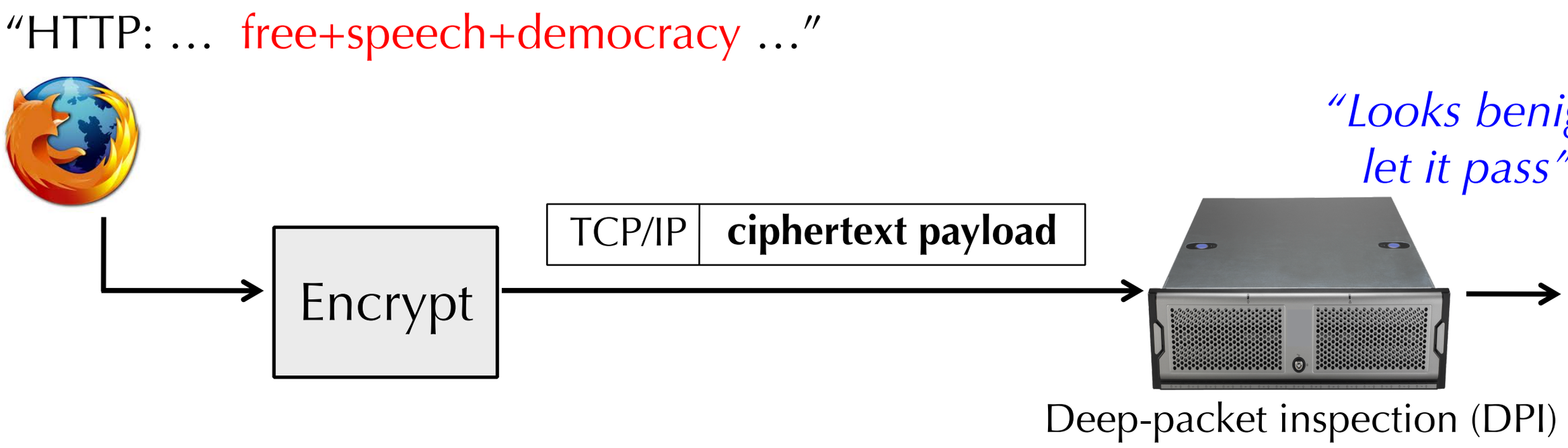
Monday

In-place encryption of CC database



Today

Circumvention of nation-state internet censorship



Traditional encryption is ill-suited for these tasks

Natively, plaintexts are bit strings

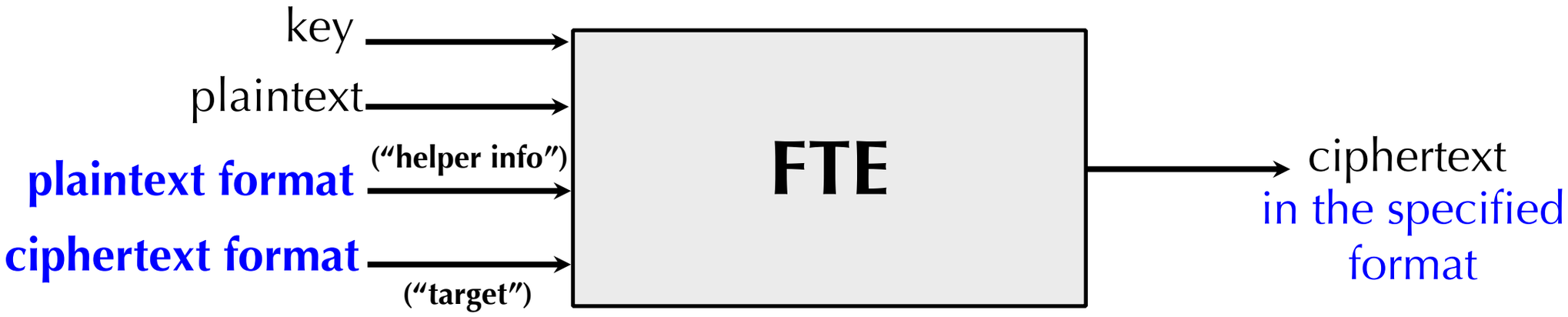
(not 16-digit decimal strings)

Traditional security goal: make ciphertexts indistinguishable from random bit strings

(not well-formatted HTTP messages or CC #s)

Format-Transforming Encryption

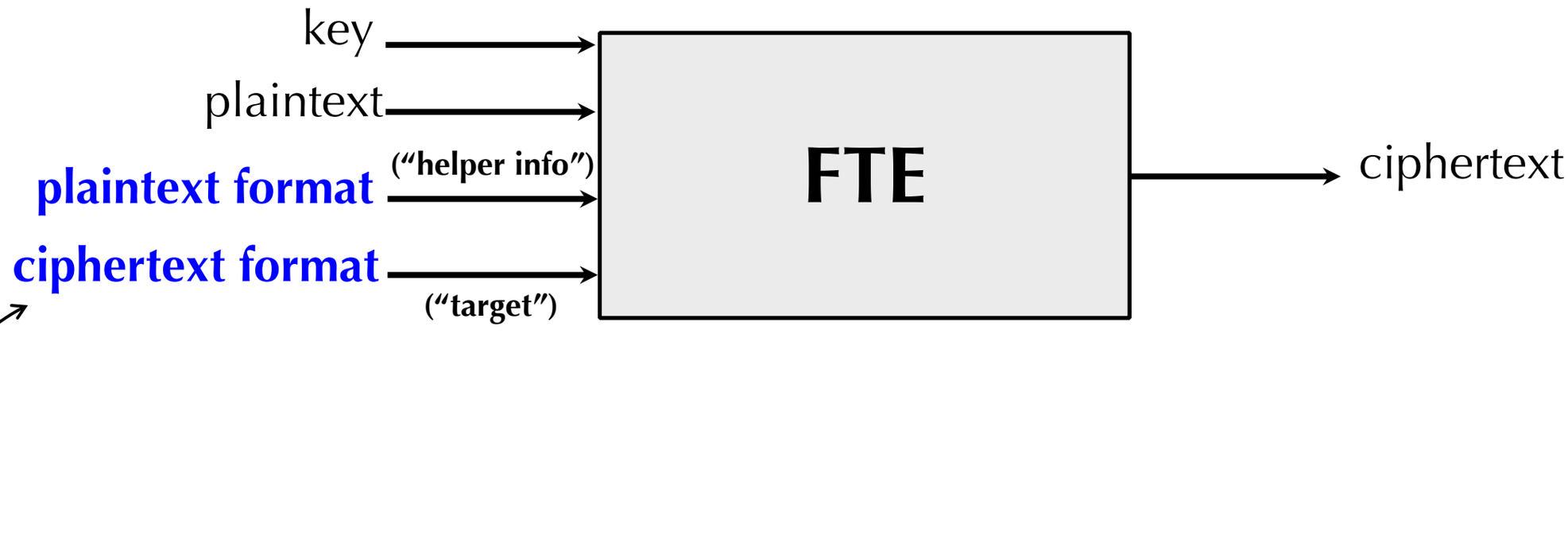
(inspired by Bellare et al. “Format-Preserving Encryption”)



A **format** is a set.

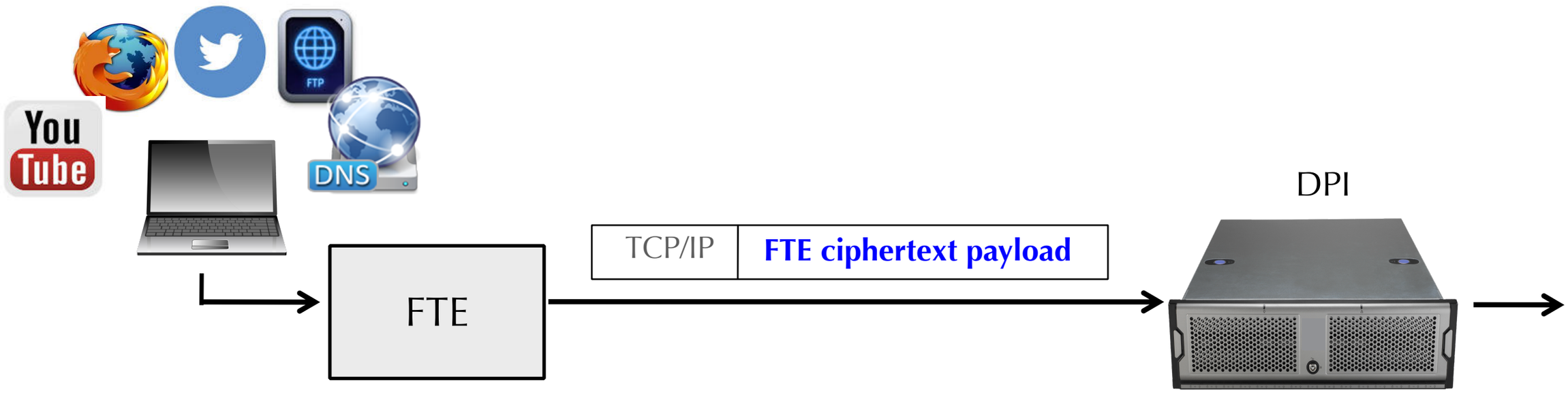
FTE is like traditional encryption, with the extra operational requirement that **ciphertexts abide by the ciphertext format**

Flexibility is “baked in” to the syntax

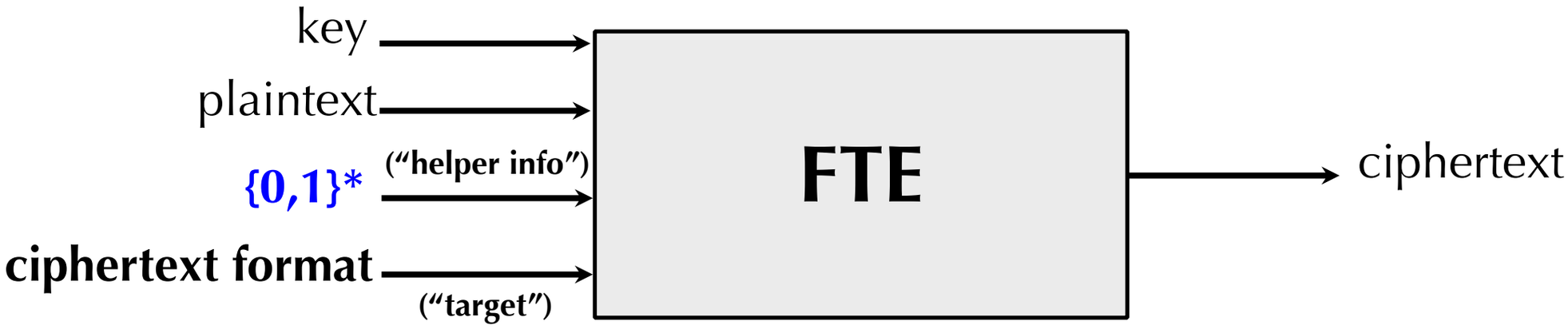


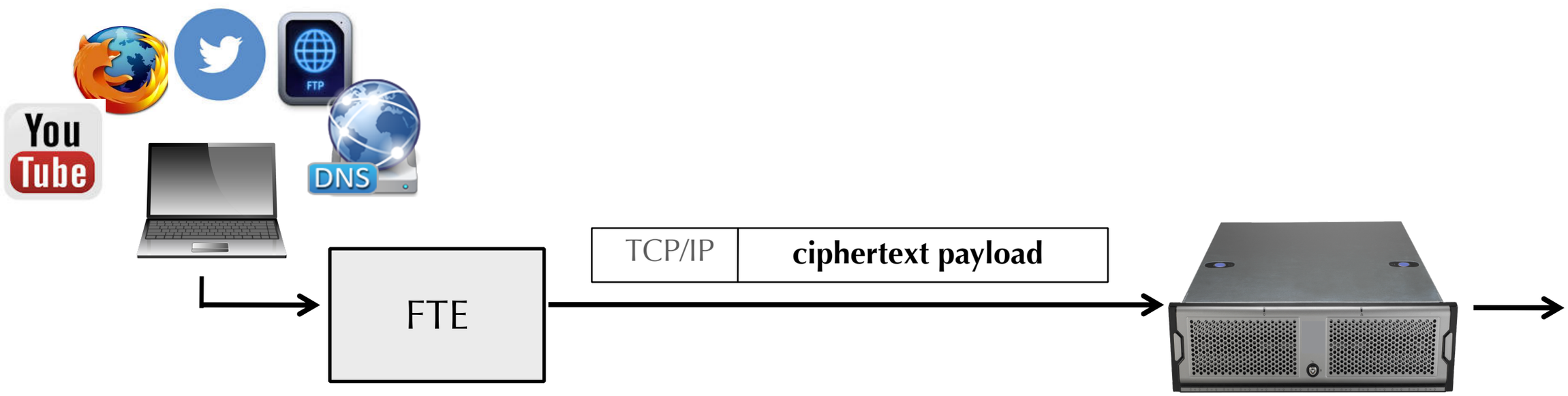
To change the “look” of ciphertexts, **just change the ciphertext format**.
The system doesn’t (necessarily) need to change.

Let’s consider the censorship-circumvention setting

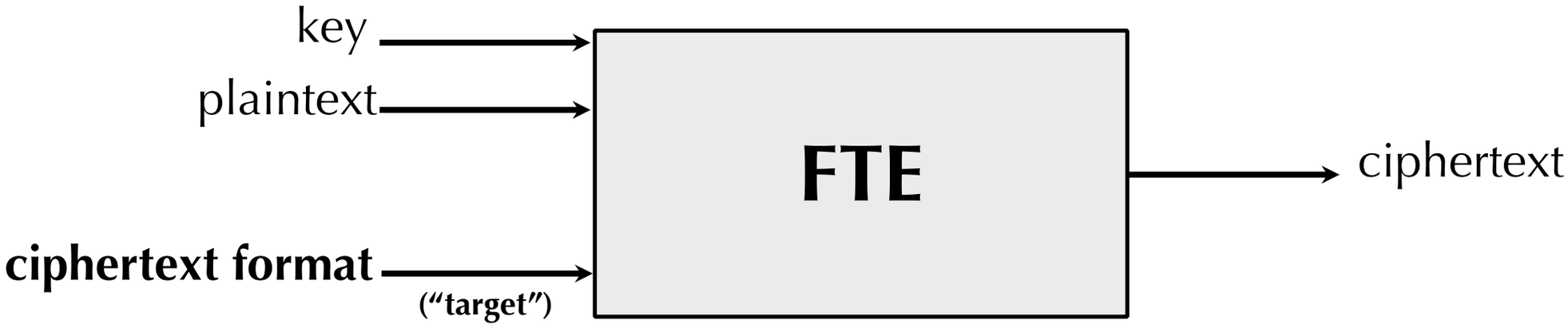


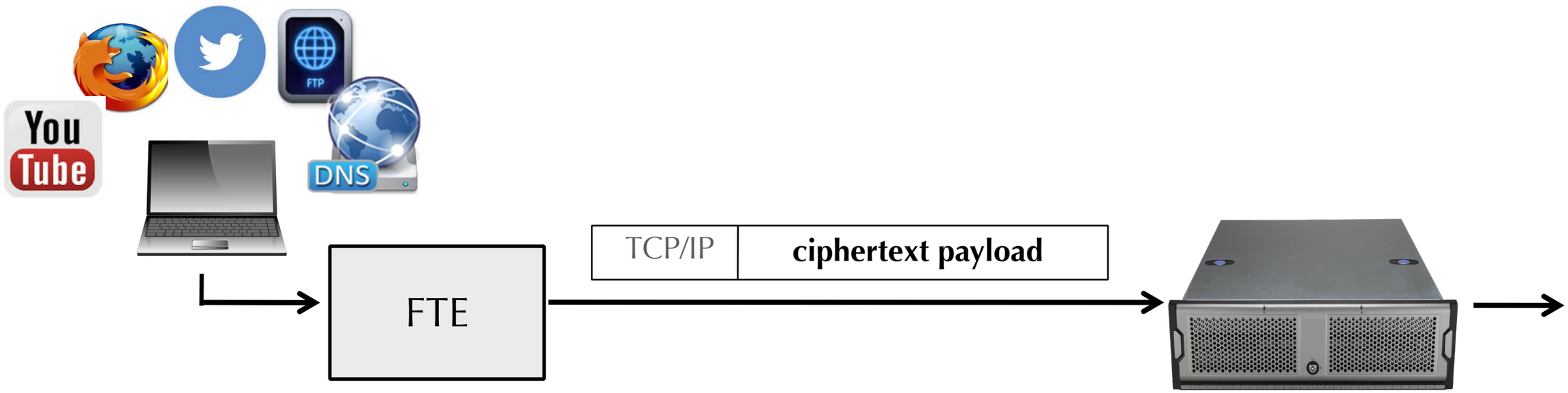
In this setting, shouldn't assume anything about plaintext formats…





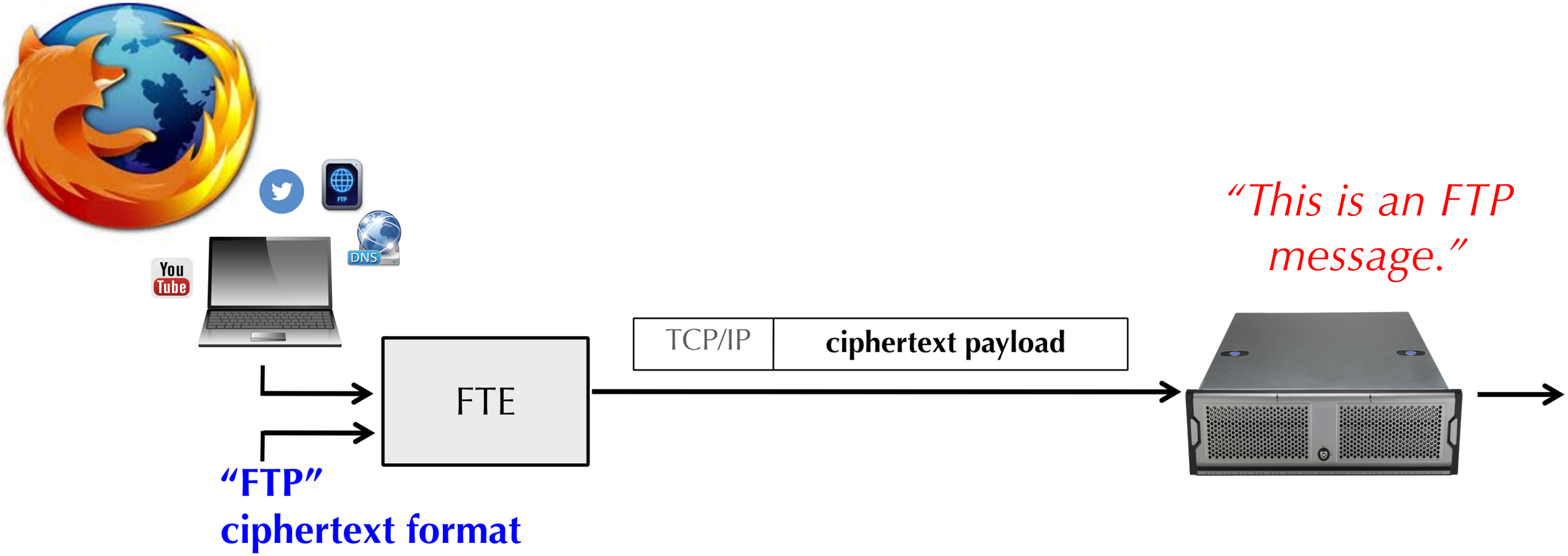
… so let's focus on this simpler API



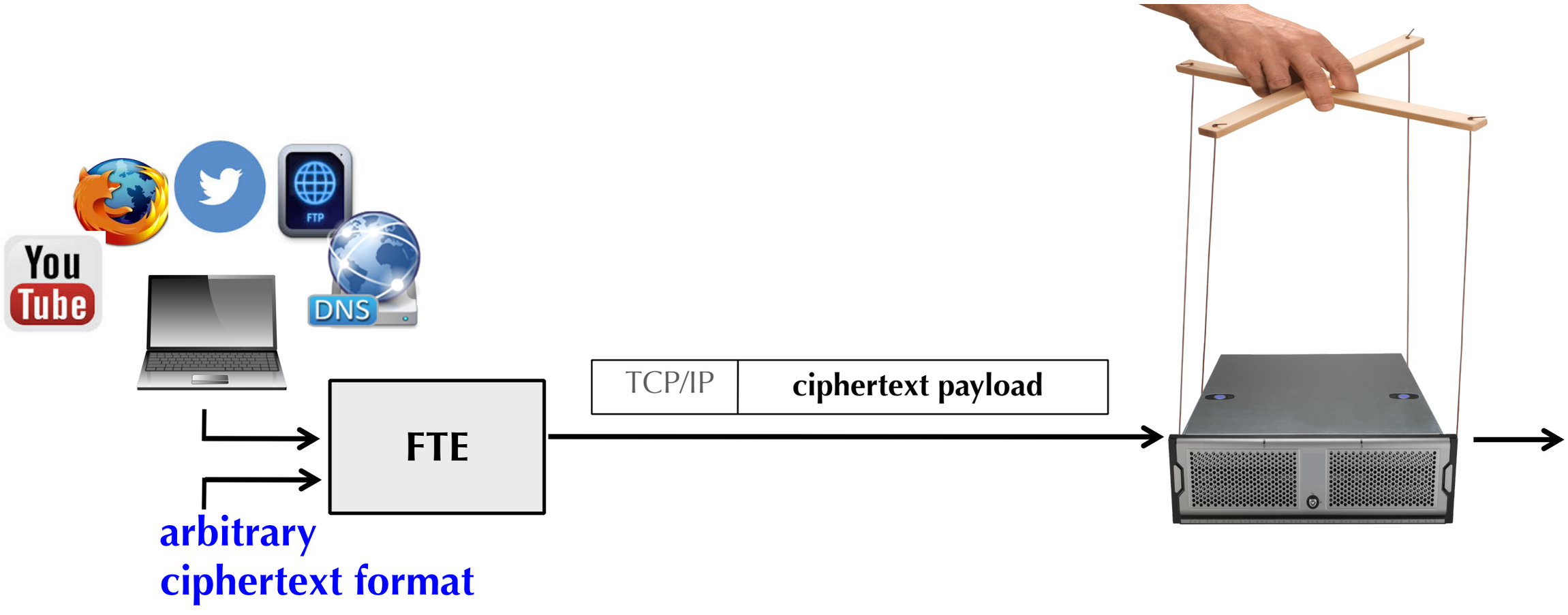


Our goal: to cause real DPI systems to reliably *misclassify* plaintext traffic

for example, HTTP misclassified as FTP



Our goal: to cause real DPI systems to reliably *misclassify* our (plaintext) traffic as whatever protocol we want



(while still having good throughput, low latency…)

We wondered:

How do real DPI devices determine to what protocol a message belongs?

"This is an _____ message."

System	Classification Tool	Price
appid		free
l7-filter		free
YAF		free
bro		free
nProbe		~300 Euros
DPI-X		~$10K

Enterprise grade DPI, well-known company

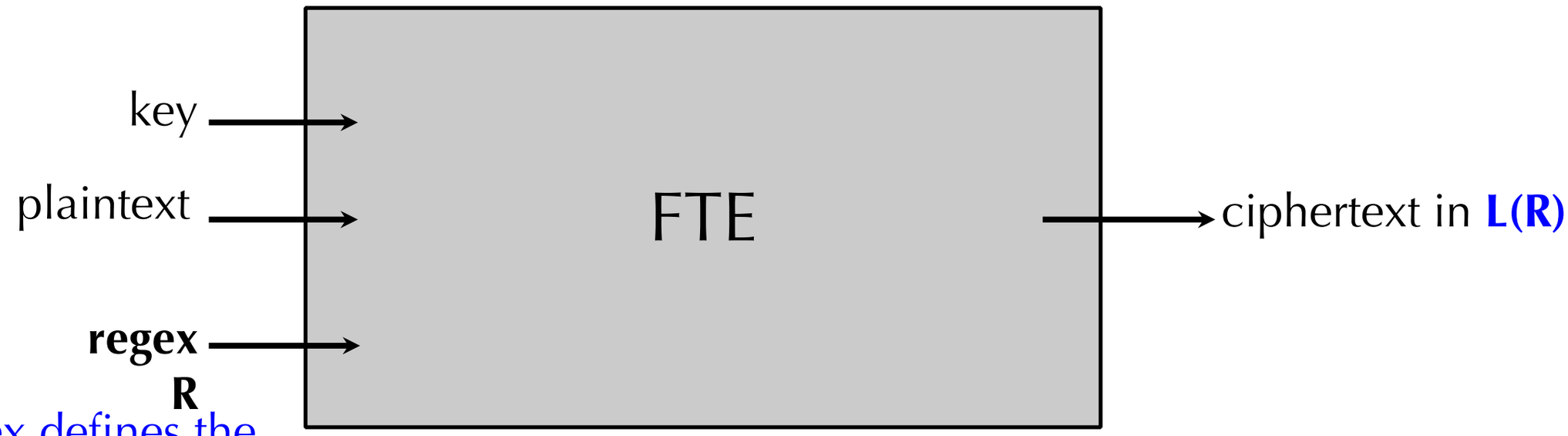
We wondered:
How do real DPI devices determine to what protocol a message belongs?

"This is an _____ message."

System	Classification Tool	Price
appid	**Regular expressions**	free
l7-filter	**Regular expressions**	free
YAF	**Regular expressions** (sometimes hierarchical)	free
bro	Simple **regular expression** triage, then additional parsing and heuristics	free
nProbe	Parsing and heuristics (many of them "**regular**")	~300 Euros
DPI-X	**???**	~$10K

Regular langauges/expressions figure heavily in state-of-the-art DPI classification tools

Regular-expression-based FTE

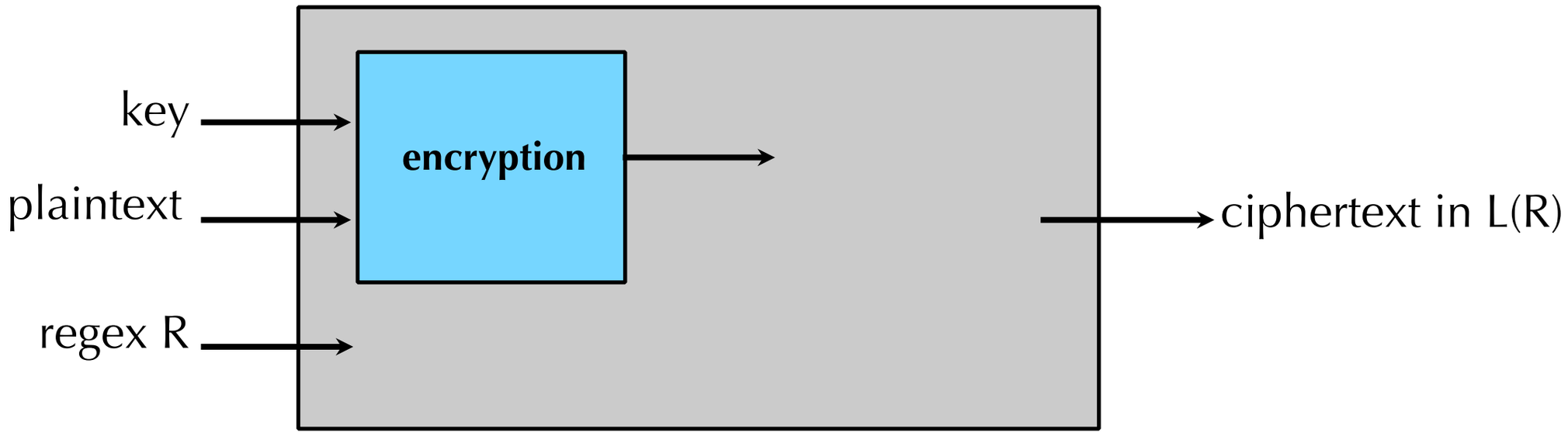


Regex defines the ciphertext format L(R)

How should we realize regex-based FTE?

We want: Cryptographic protection for the plaintext
Ciphertexts in L(R)

Realizing regex-based FTE

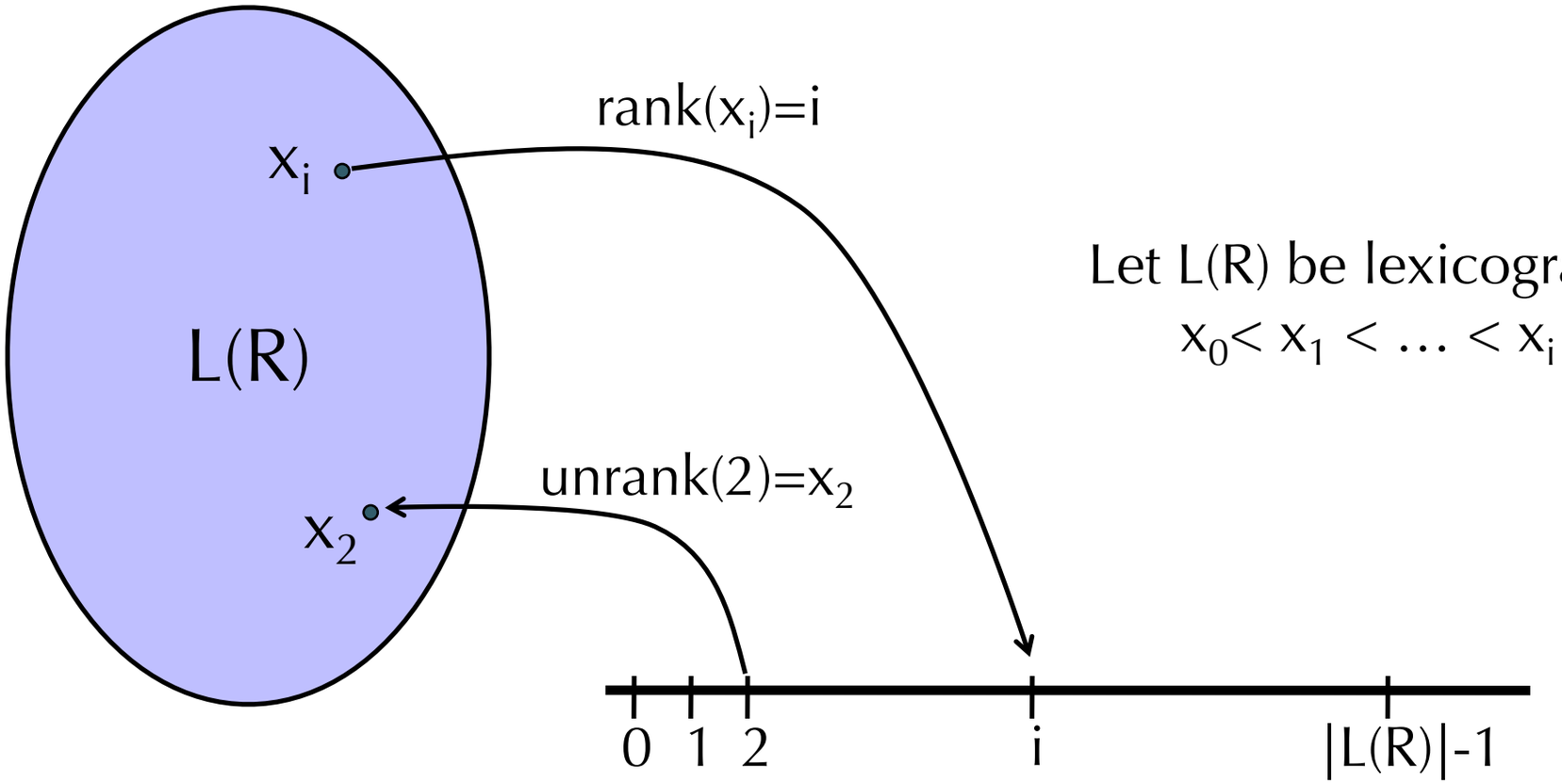


How should we realize regex-based FTE?

We want: Cryptographic protection for the plaintext
Ciphertexts in L(R)

Ranking a Regular Language

[Goldberg, Sipser '85]
[Bellare et al. '09]



Let L(R) be lexicographically ordered

$x_0 < x_1 < \ldots < x_i < \ldots < x_{|L(R)-1|}$

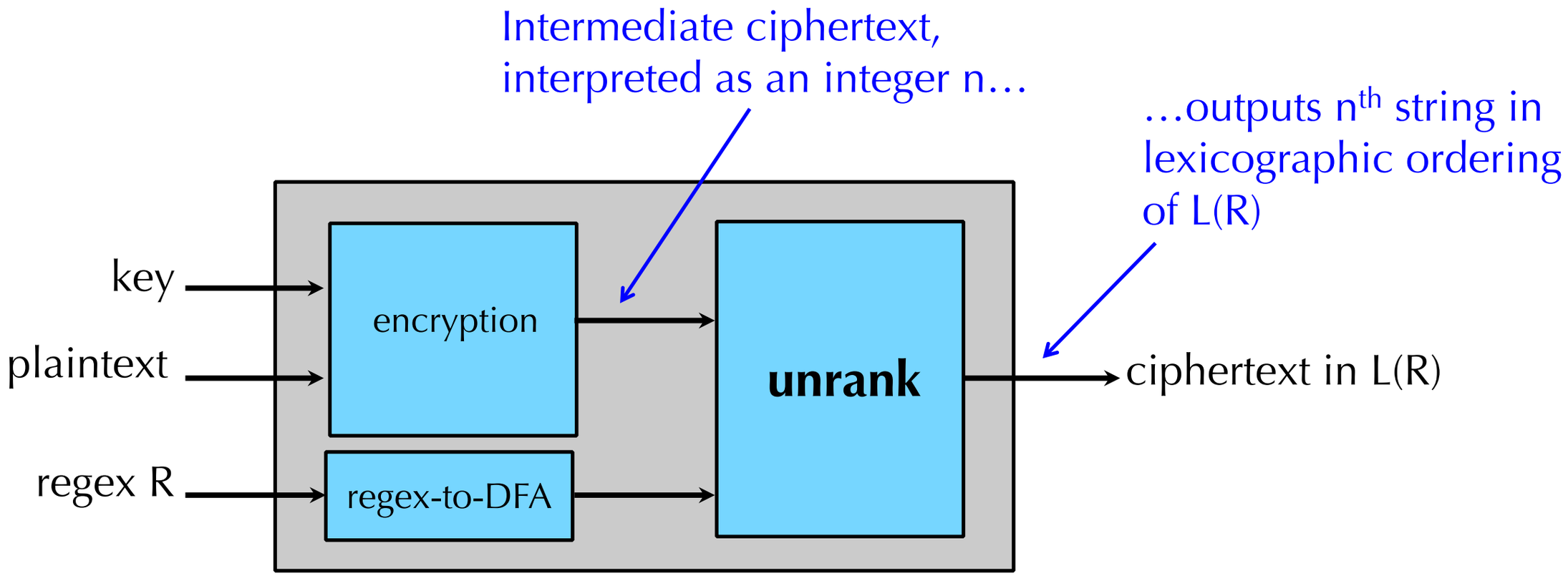
Given a **DFA** for L(R), there are efficient algorithms

rank: L(R) $\longrightarrow$ {0,1,…,|L(R)|-1}
unrank: {0,1,…,|L(R)|-1} $\longrightarrow$ L(R)

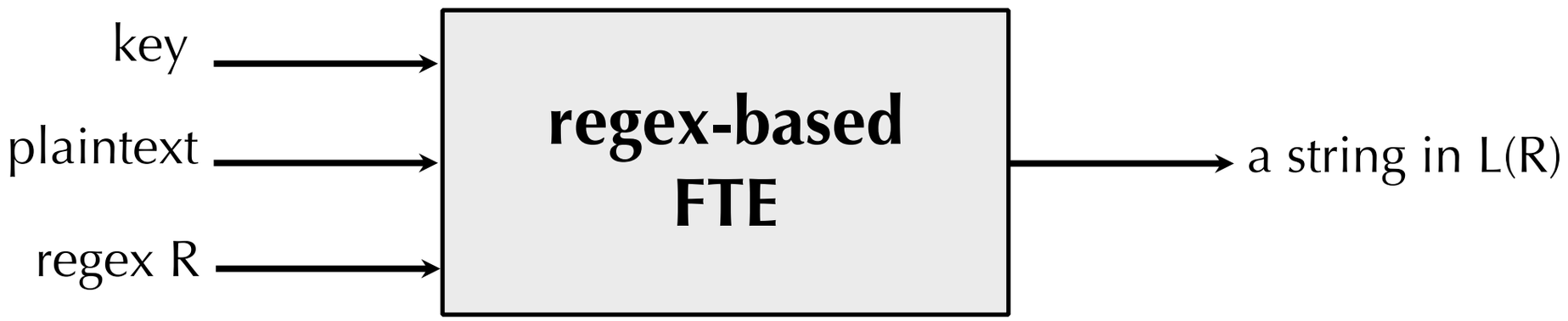
such that rank(unrank(i)) = i
and unrank(rank(x_i)) = x_i

With precomputed tables, rank, unrank are *O*(n)

Realizing regex-based FTE



Now all we need are good regular expressions



We considered three options :

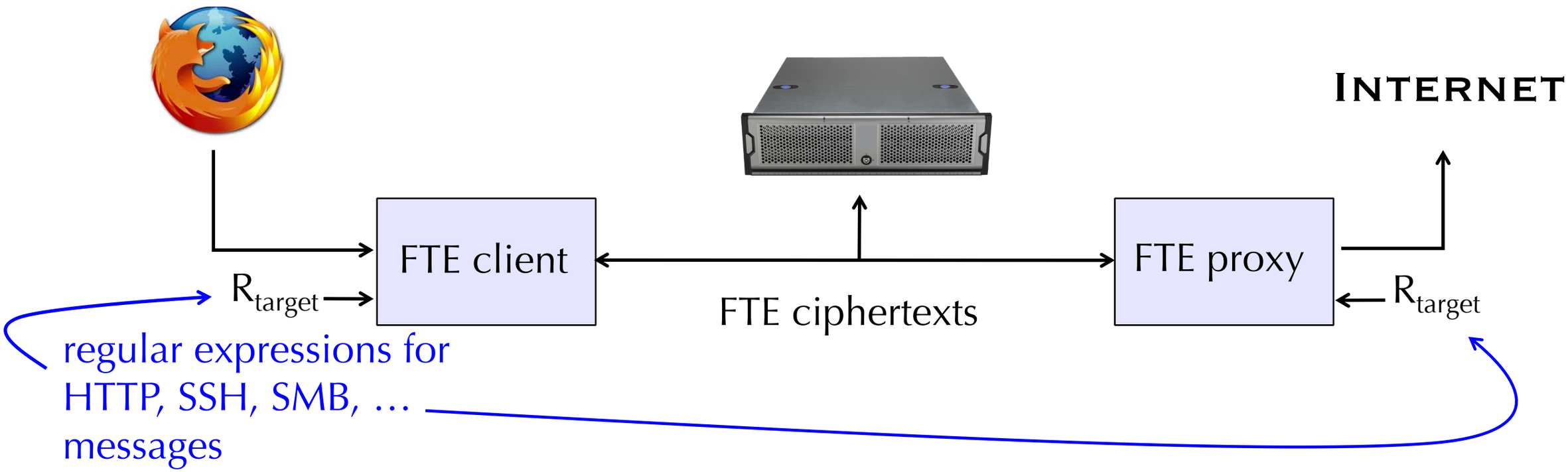
1. If the DPI is open source (appid, l7-filter, YAF), try to **extract them**, directly!

2. **Build them manually**, using RFCs and (when possible) DPI source code.

3. **Learn them from traffic** that was allowed by the DPI.

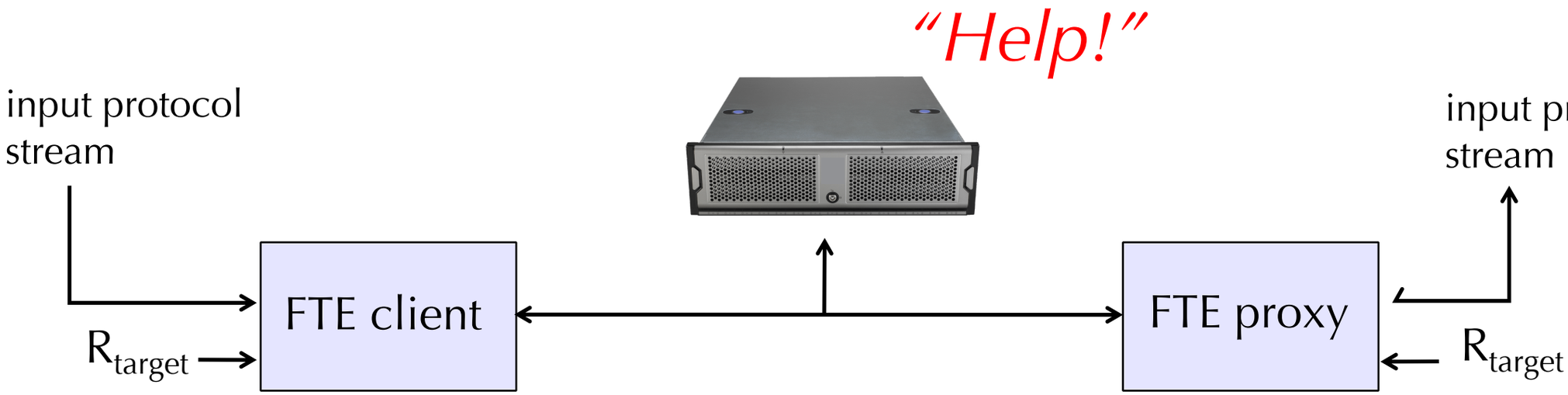
Use case: Browsing the web through an FTE tunnel

FTE "wins" if the DPI classifies the stream it sees as the target protocol



Using each "target" format, we visited each of the Top 50 websites five times.

Punchline: regex-based FTE can make *real* DPI say whatever we want it to ~100% of the time.



Browser experience
through FTE tunnel

≈

Browser experience
through SSH tunnel

FTE library is open-source, runs on multiple platforms/OS,
and is fully integrated with major circumvention efforts

Eric Schmidt gave us a sizable
unsolicited research gift

A field test…

Without FTE tunnel, we tried Facebook, YouTube, Tor website, banned search queries…

With FTE tunnel, we tried Facebook, YouTube, Tor website, banned search queries…

Used FTE to download Tor bundle:

Tor without FTE: "active blacklisting" attack on proxy
Tor through FTE: no problems

Ran various tests every 5 minutes for one month,
no sign of detection in logs. (We shut it down after that.)

What about in-place encryption of CC database?

key

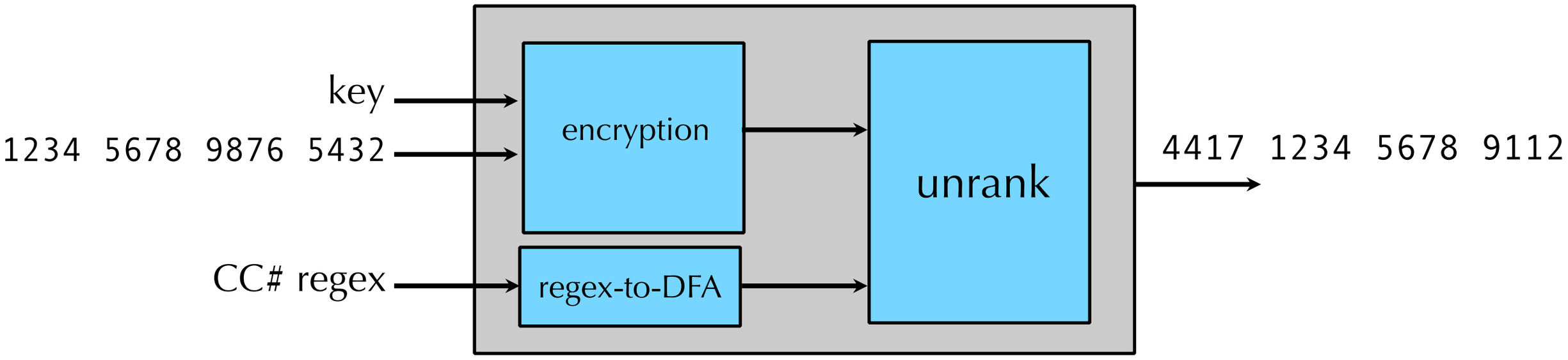
1234 5678 9876 5432

regex for language of
16-decimal digit CC #s

regex-based
FTE

4417 1234 5678 9112

Not quite handled by “simpler” FTE construction



1) valid 16-digit number in, valid 16-digit number out

|plaintext language| = |ciphertext language|

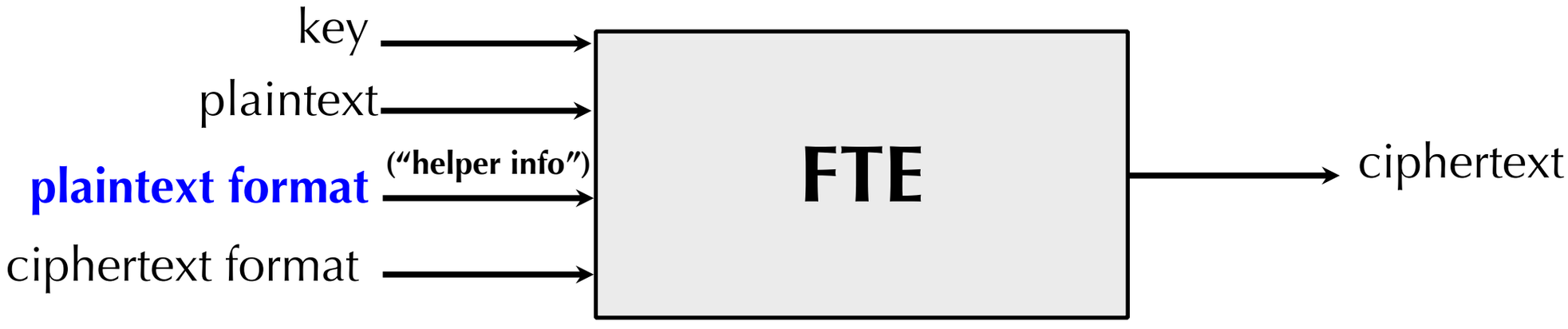
2) conventional encryption takes **bit strings** as input

encoding of valid 16-digit strings into bitstrings
expands the effective plaintext space

3) conventional encryption has ciphertext **stretch**

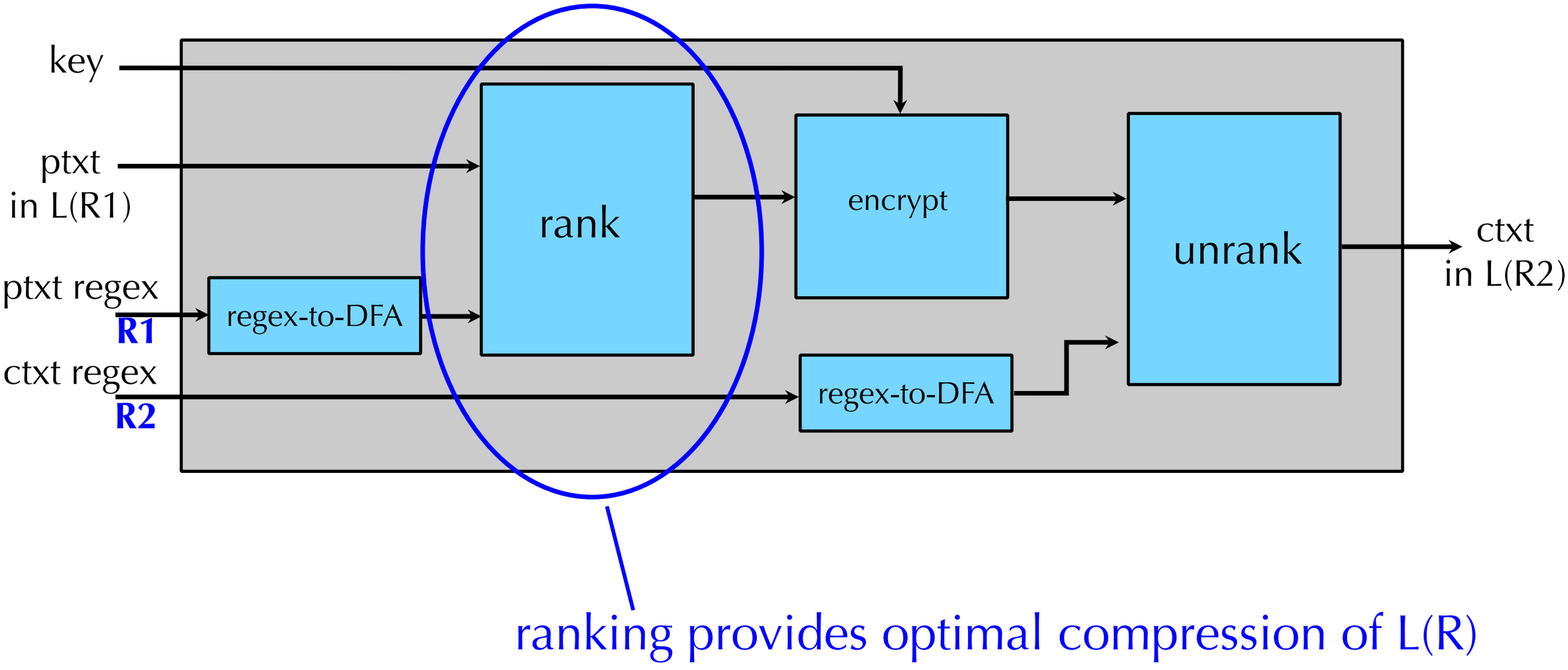
can have exponential number of AE ciphertexts that cannot be unranked!

Recall the full FTE API...

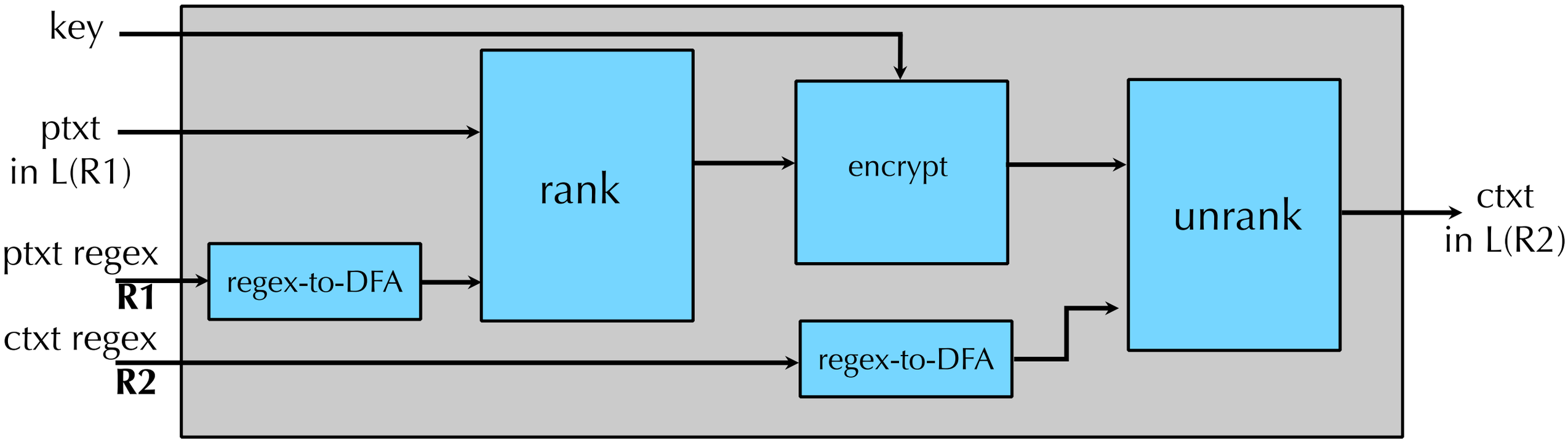


"rank-encrypt-unrank" FTE construction

(generalization of Bellare et al. SAC'09)



“rank-encrypt-unrank” FTE construction



Great potential… but developers face many hard questions:

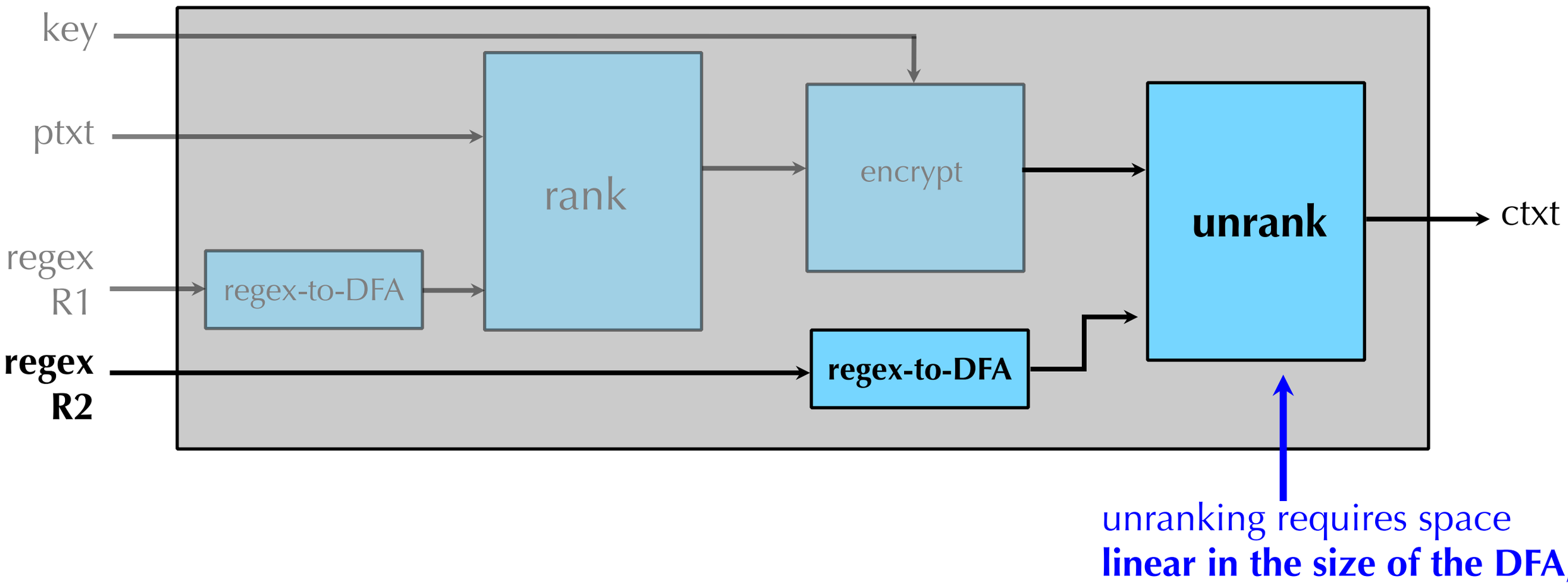
-- Can I even use R1 and R2 together? (Requires $|L(R1)| \leq |L(R2)|$)

-- Should “encrypt” be deterministic (i.e. a cipher) or can I use traditional encryption?

-- **Will both R1 and R2 admit time/space efficient implementations of (un)ranking?**

-- …

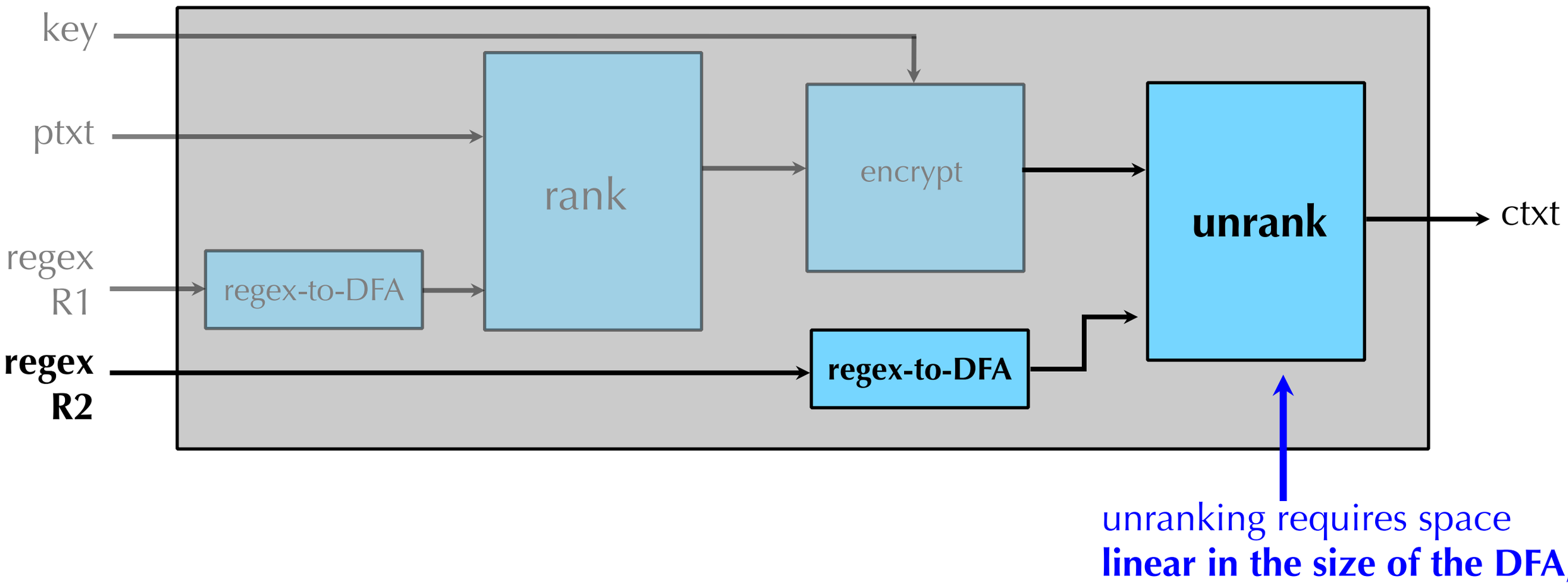
The space/memory issue



For some regular expressions, this works out just fine…

regex → NFA → DFA

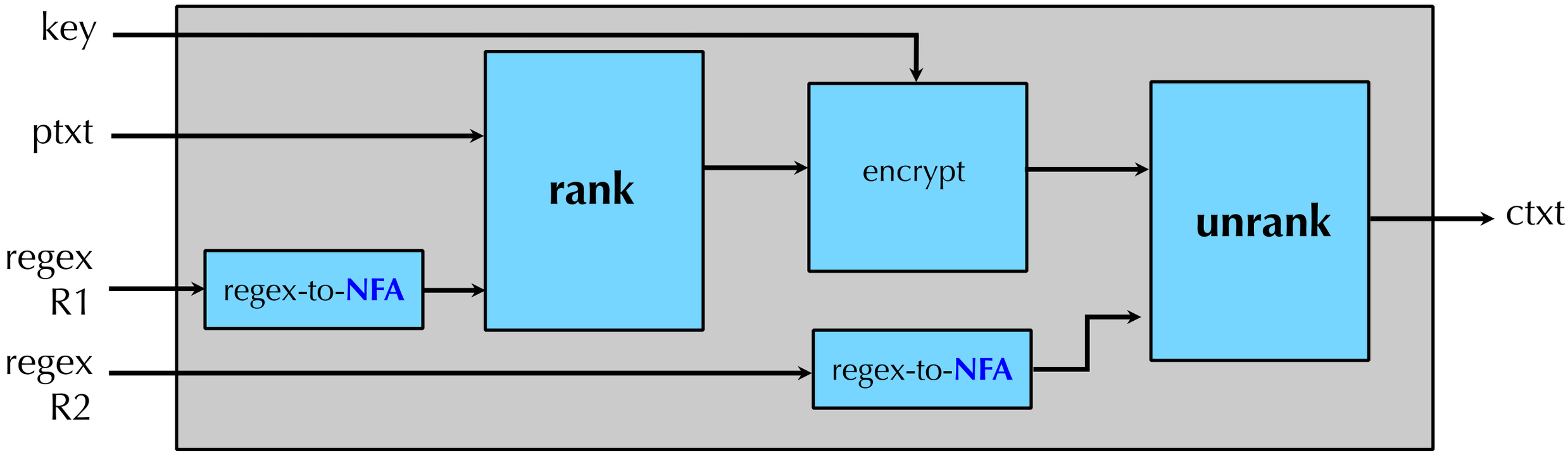
The space/memory issue



…for others, you can have an **exponential** space blow-up

regex → NFA → DFA

The space/memory issue

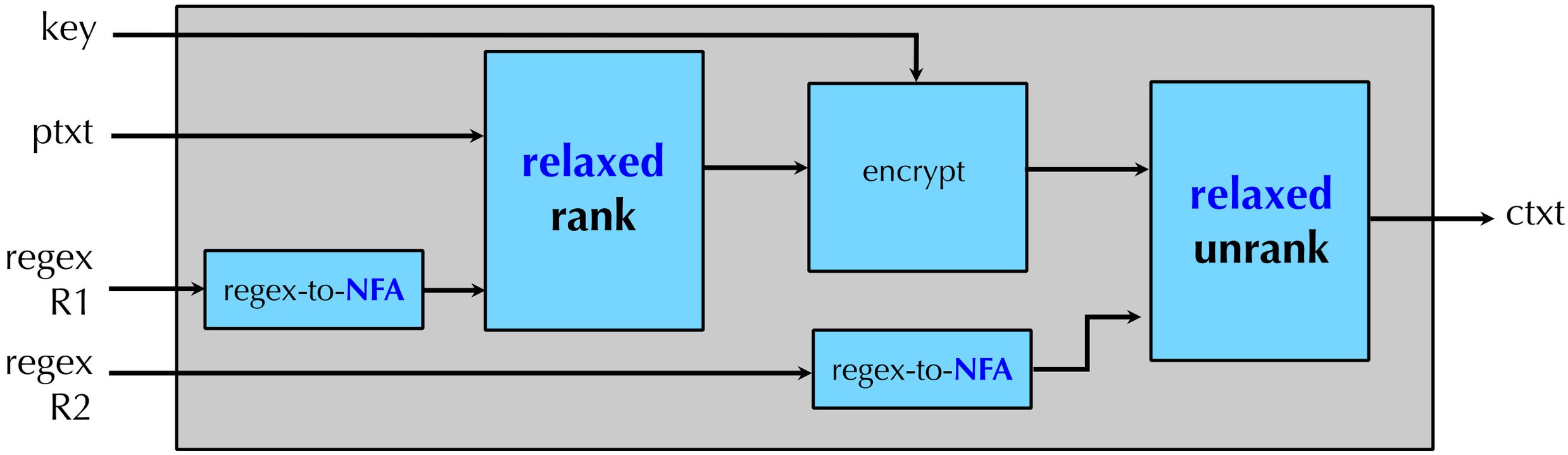


Wanted: efficient (un)ranking methods that work directly from the NFA representation

regex → **NFA** → DFA

Problem: (un)ranking from NFAs (or directly from a regex) is **PSPACE-complete**

relaxed rank-encrypt-unrank FTE construction



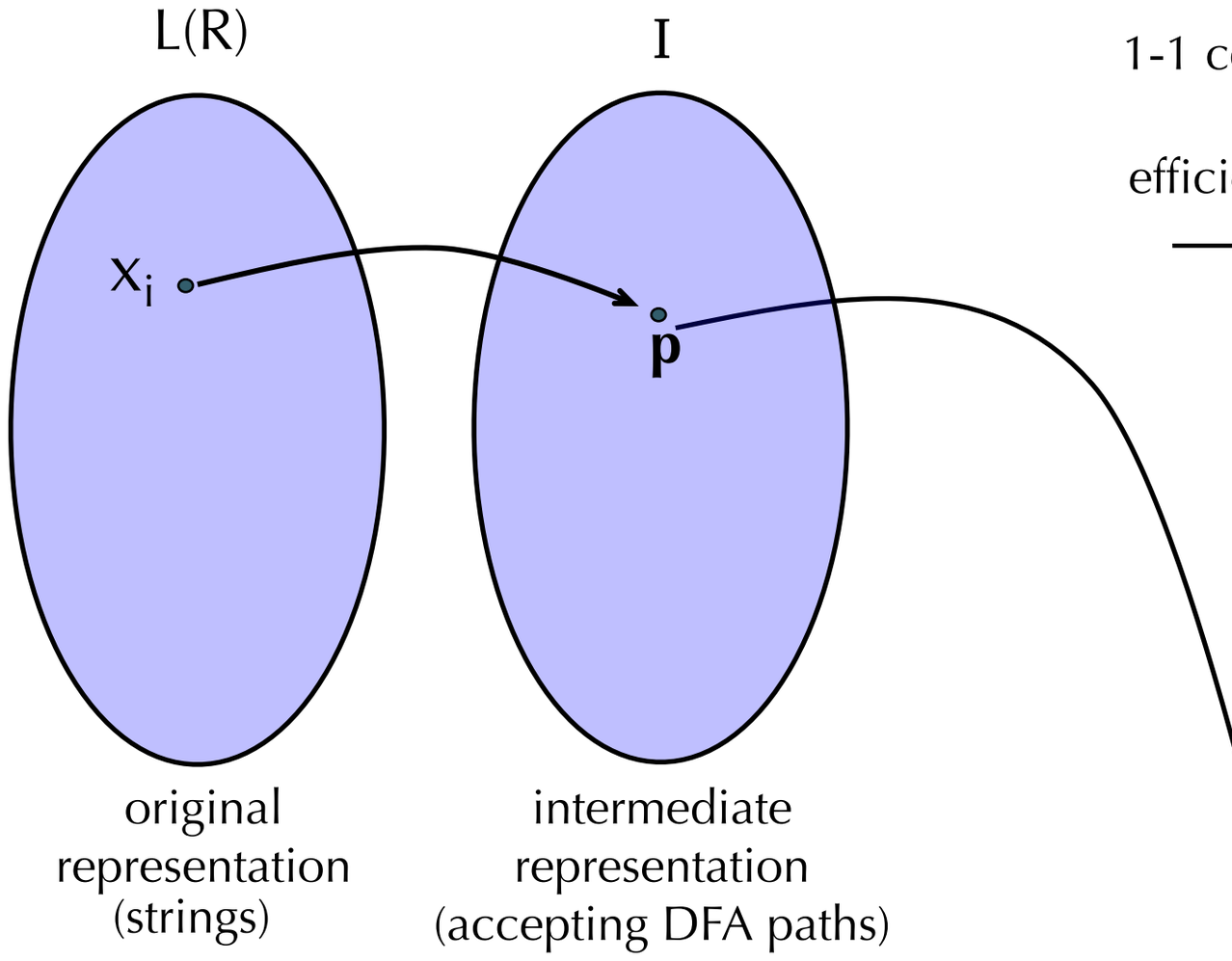
Wanted: efficient (un)ranking methods that work directly from the NFA representation

regex → **NFA** → DFA

Problem: (un)ranking from NFAs (or directly from a regex) is **PSPACE-complete**

We side-step this by developing a new "relaxed ranking" algorithm

Ranking of a language from a DFA



1-1 correspondence between strings and accepting paths

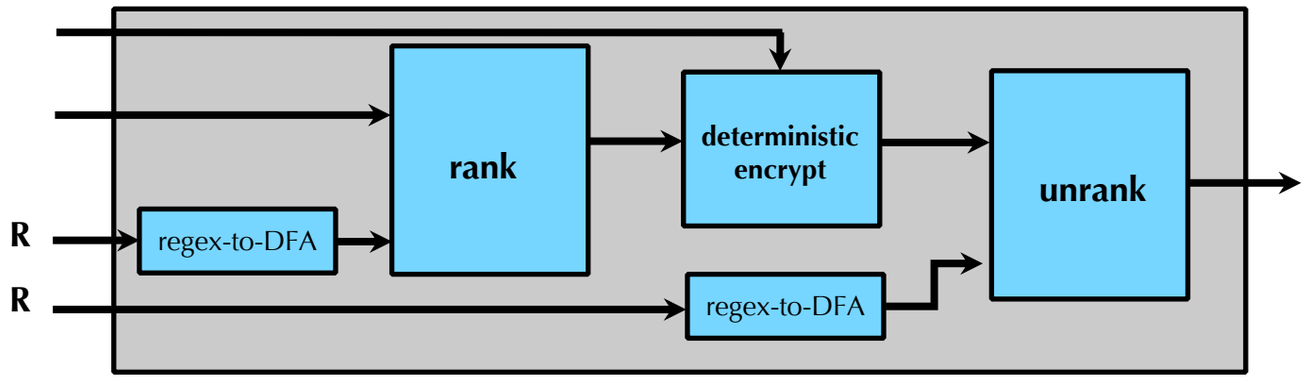
+

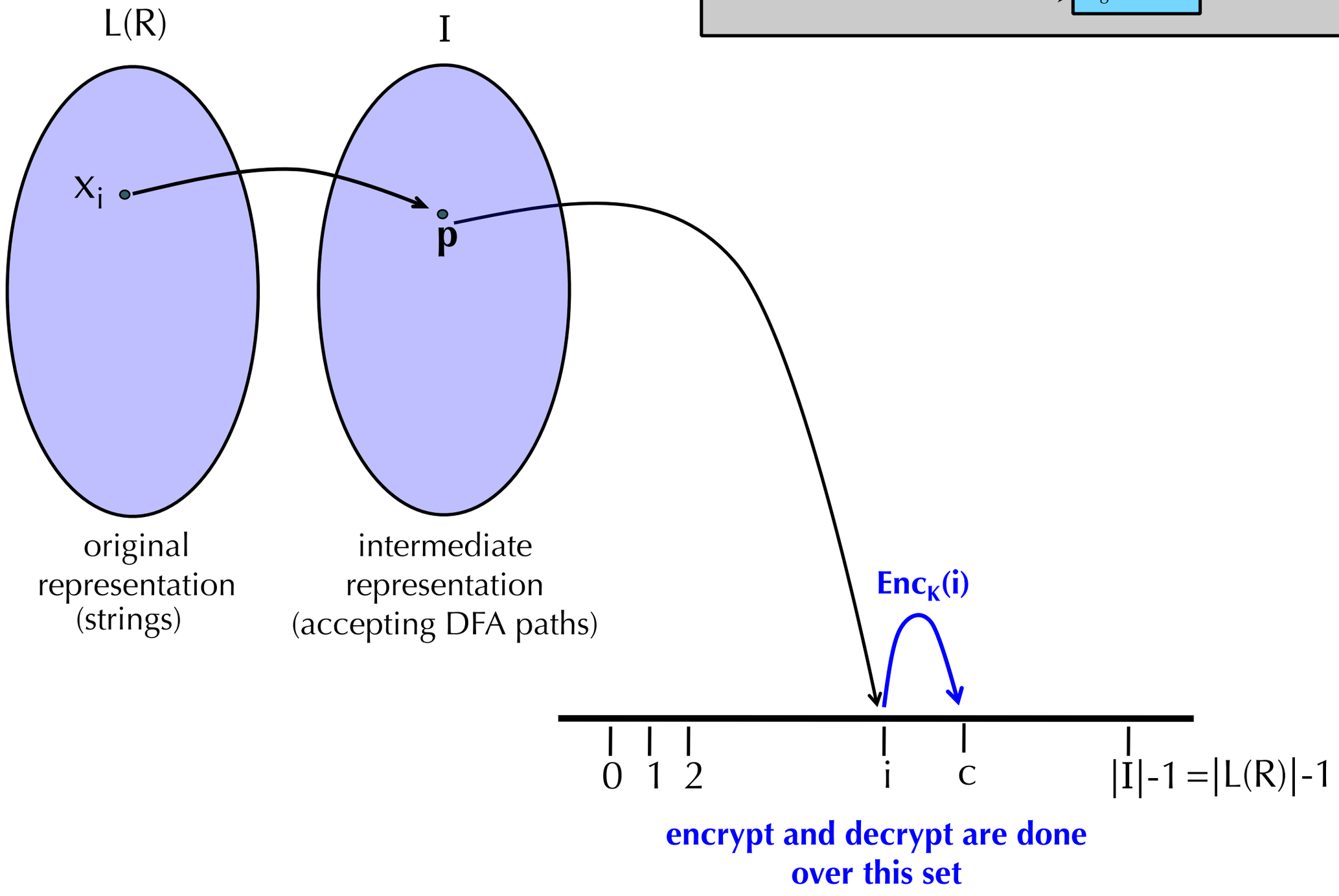
efficient alg. for 1-1 mapping between paths and integers

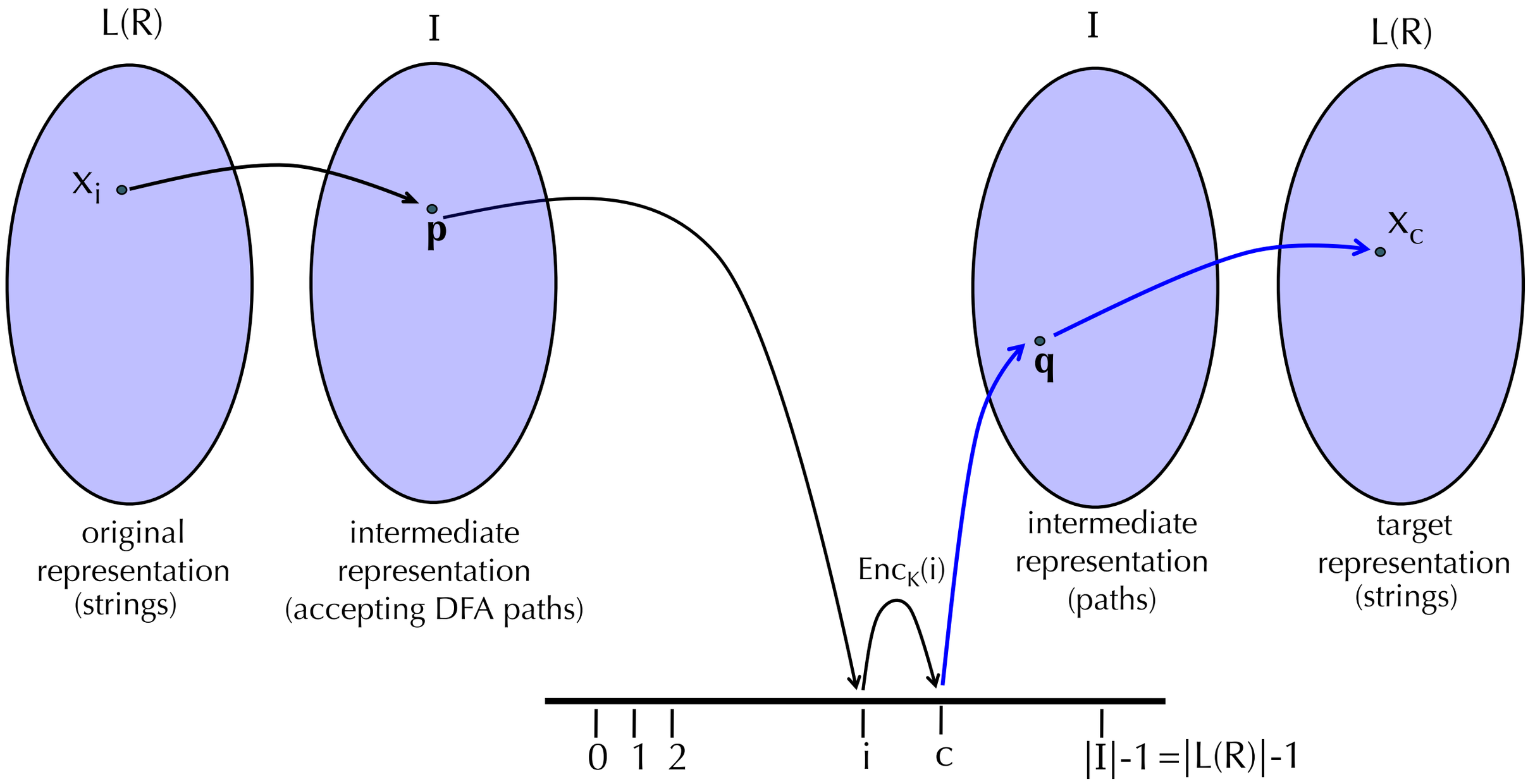
---

rank(x_i)=i and unrank(i)=x_i

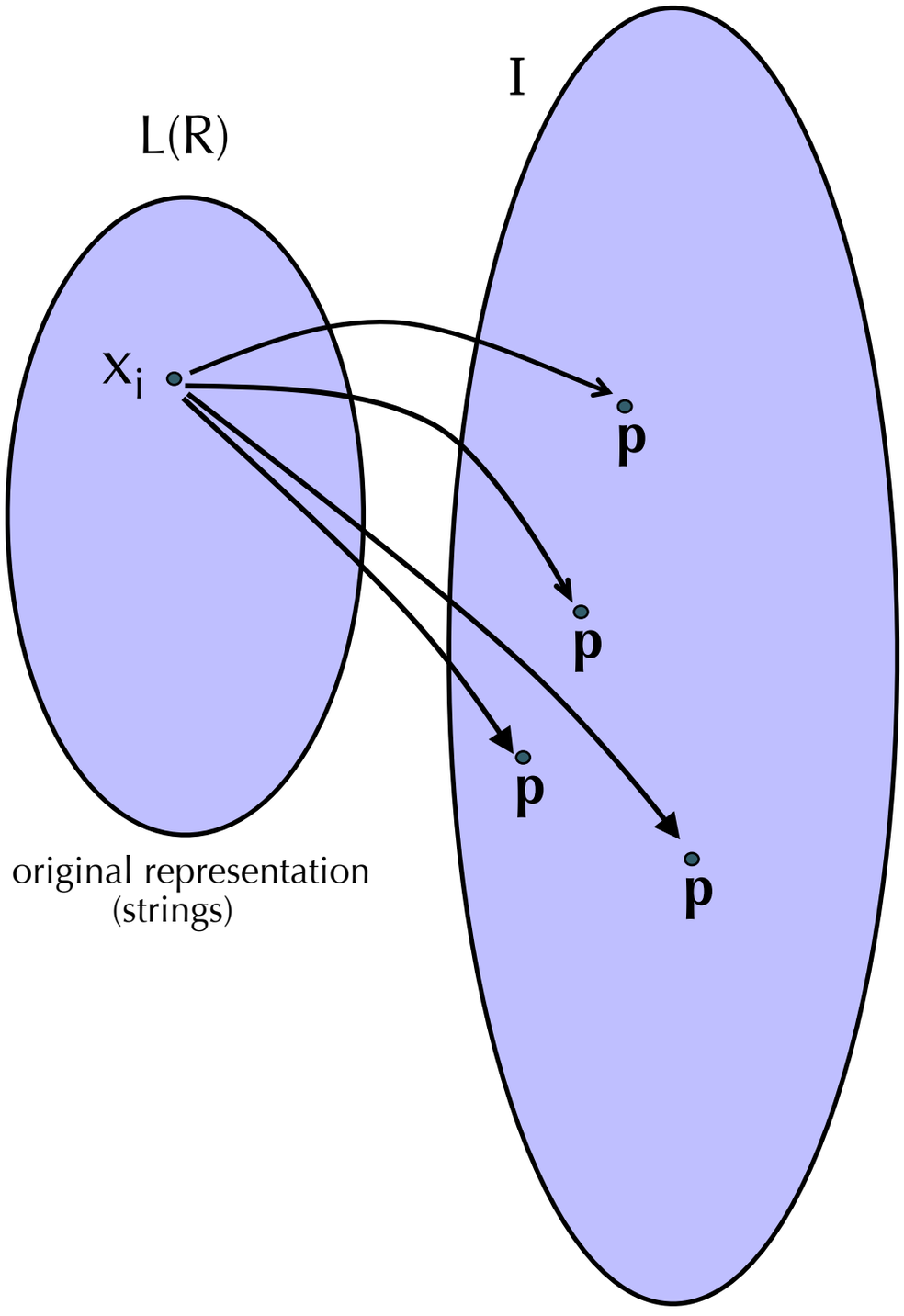
"Rank"

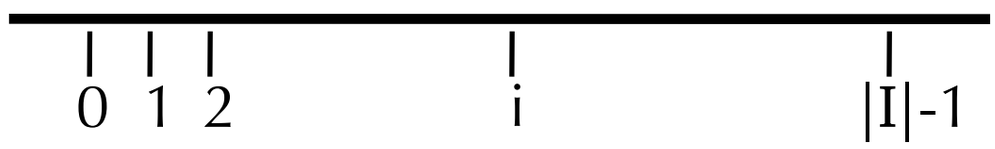

R
R
regex-to-DFA
rank
deterministic encrypt
regex-to-DFA
unrank


L(R)
I
x_i
p
original representation (strings)
intermediate representation (accepting DFA paths)
$Enc_K(i)$
0 1 2
i
c
|I|-1 = |L(R)|-1
encrypt and decrypt are done over this set


"rank"
"encrypt"
"unrank"
L(R)
I
I
L(R)
x_i
p
q
x_C
original representation (strings)
intermediate representation (accepting DFA paths)
$Enc_K(i)$
intermediate representation (paths)
target representation (strings)
0 1 2
i
c
|I|-1 =|L(R)|-1

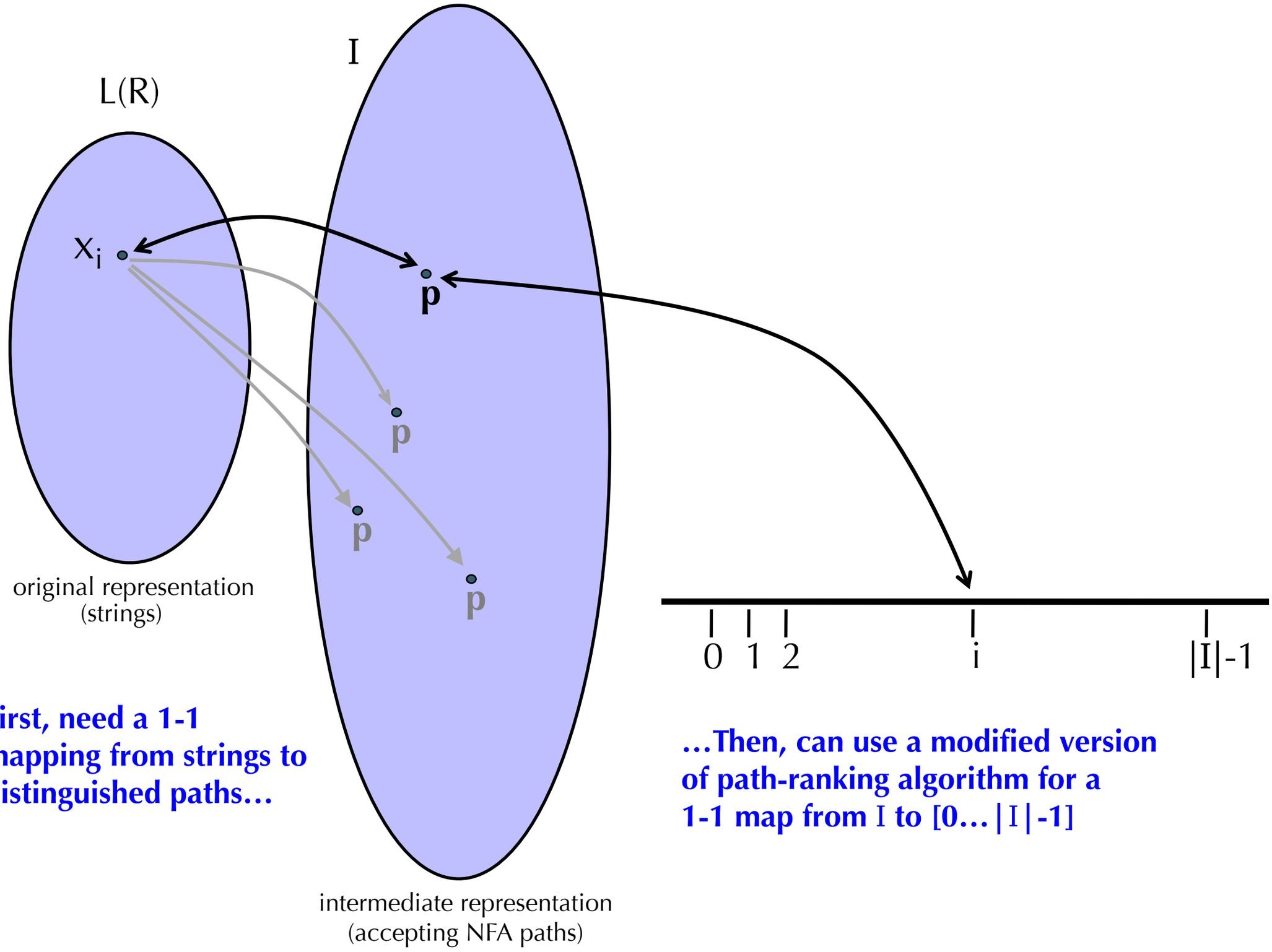
Ranking of a language from an NFA





Relaxed

Ranking of a language from an NFA

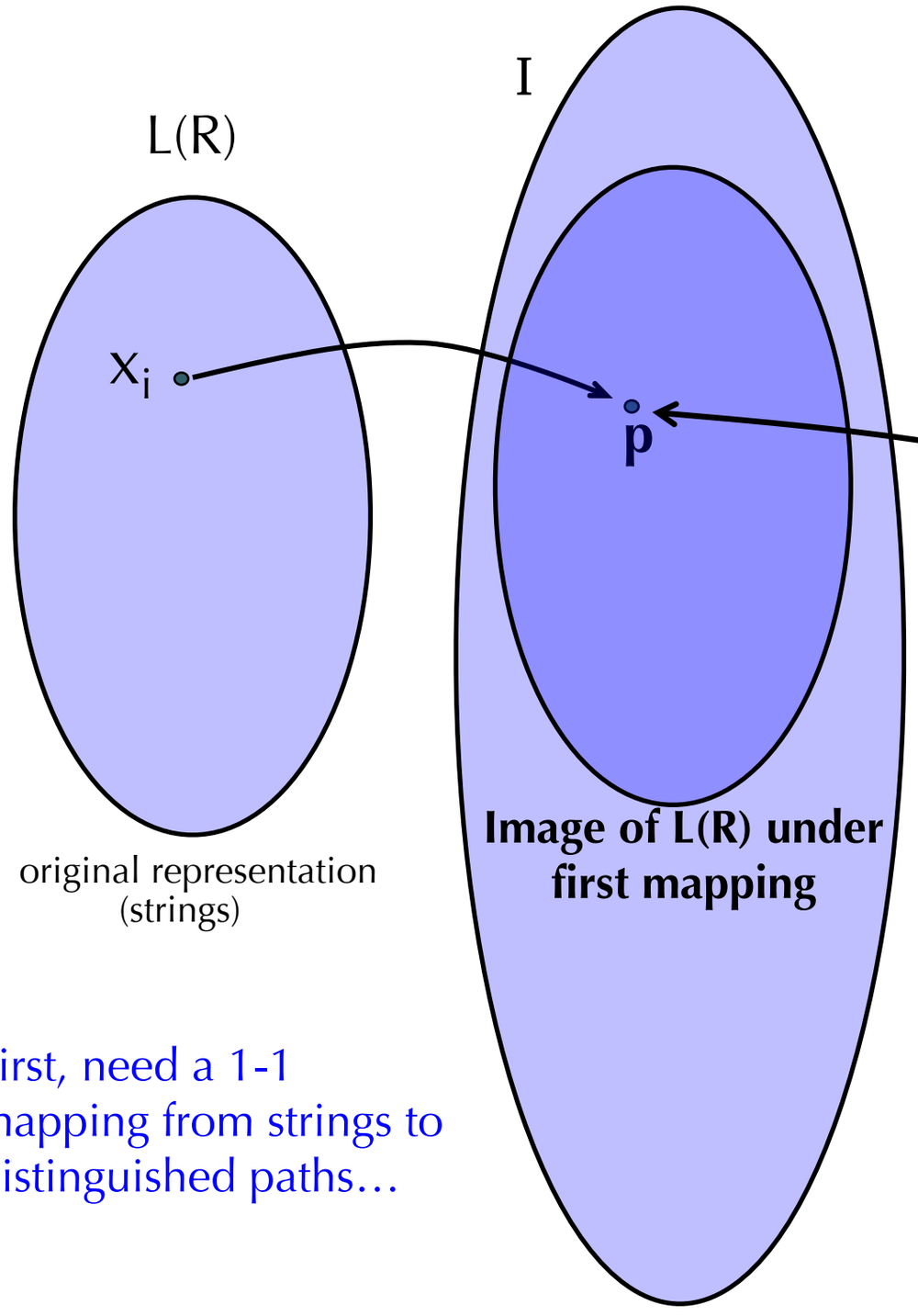


First, need a 1-1 mapping from strings to distinguished paths…

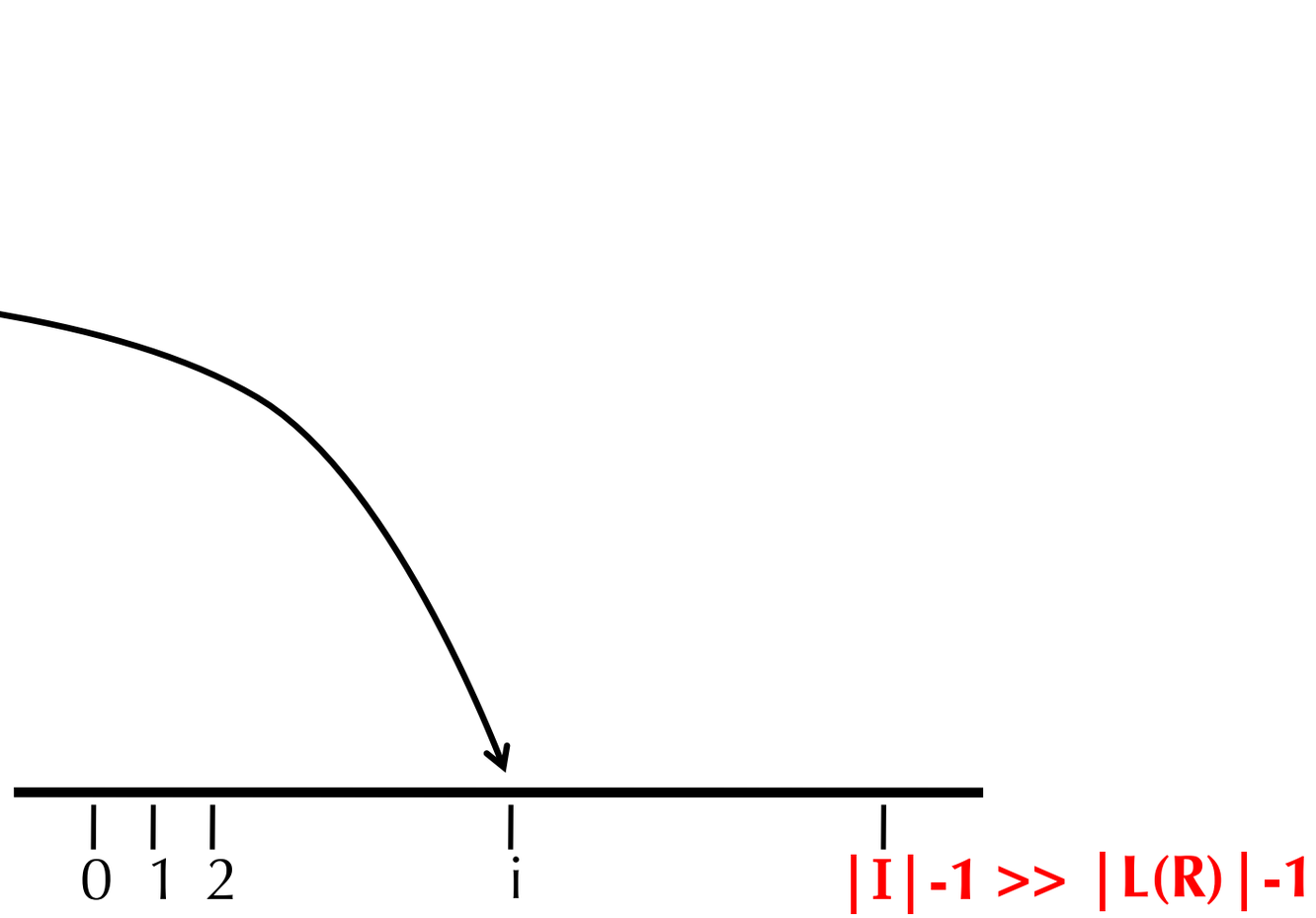
…Then, can use a modified version of path-ranking algorithm for a 1-1 map from I to [0…|I|-1]

Relaxed

Ranking of a language from an NFA

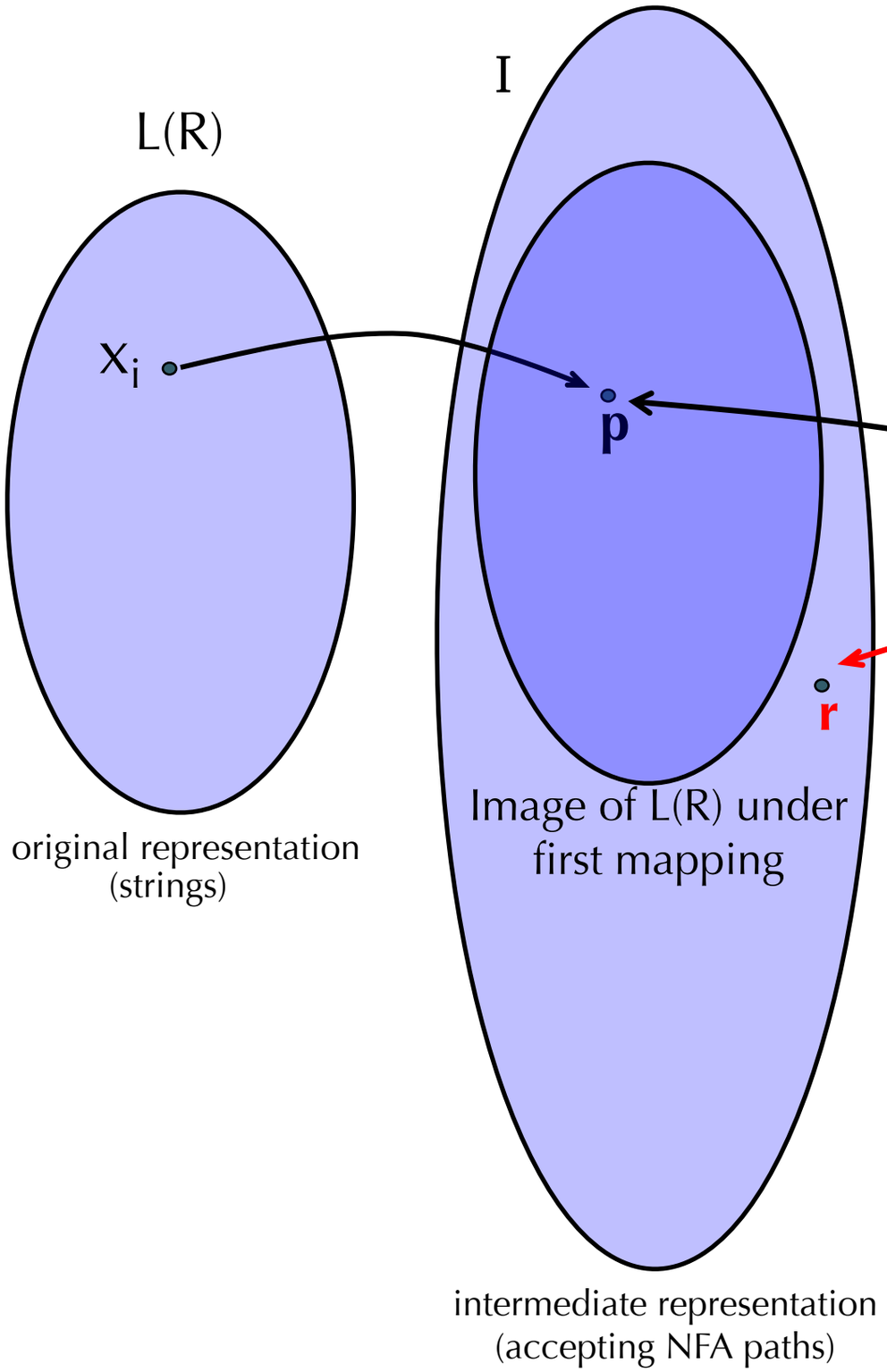


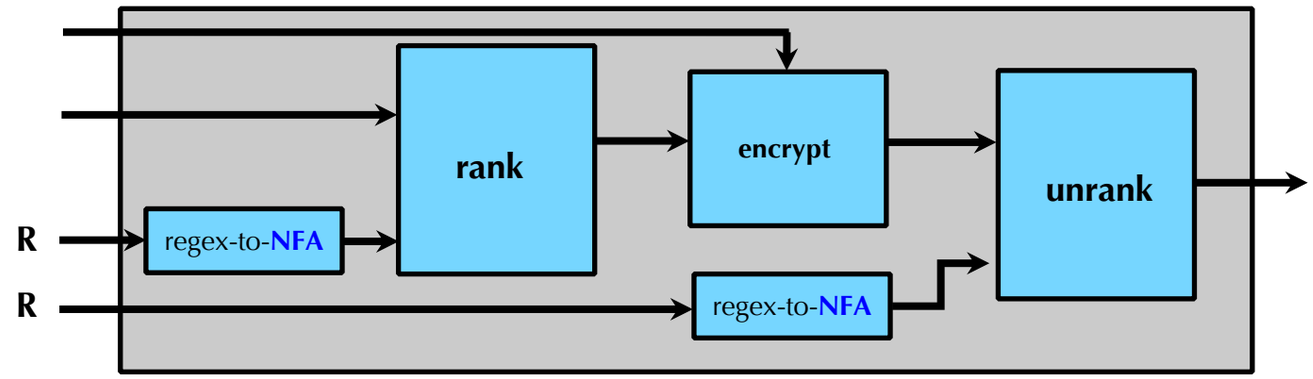
First, need a 1-1 mapping from strings to distinguished paths…

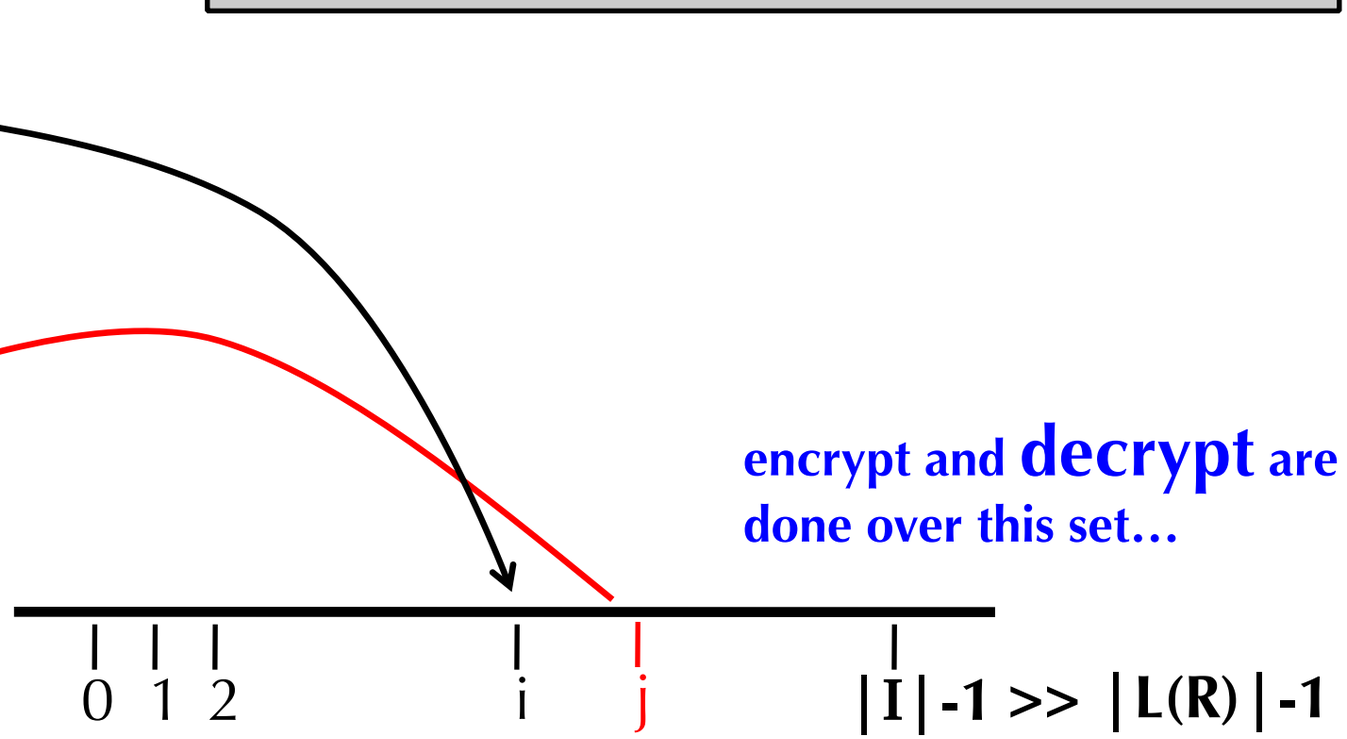


…Then, can use a modified version of path-ranking algorithm for a 1-1 map from I to [0…|I|-1]

Relaxed
Ranking of a language from an NFA

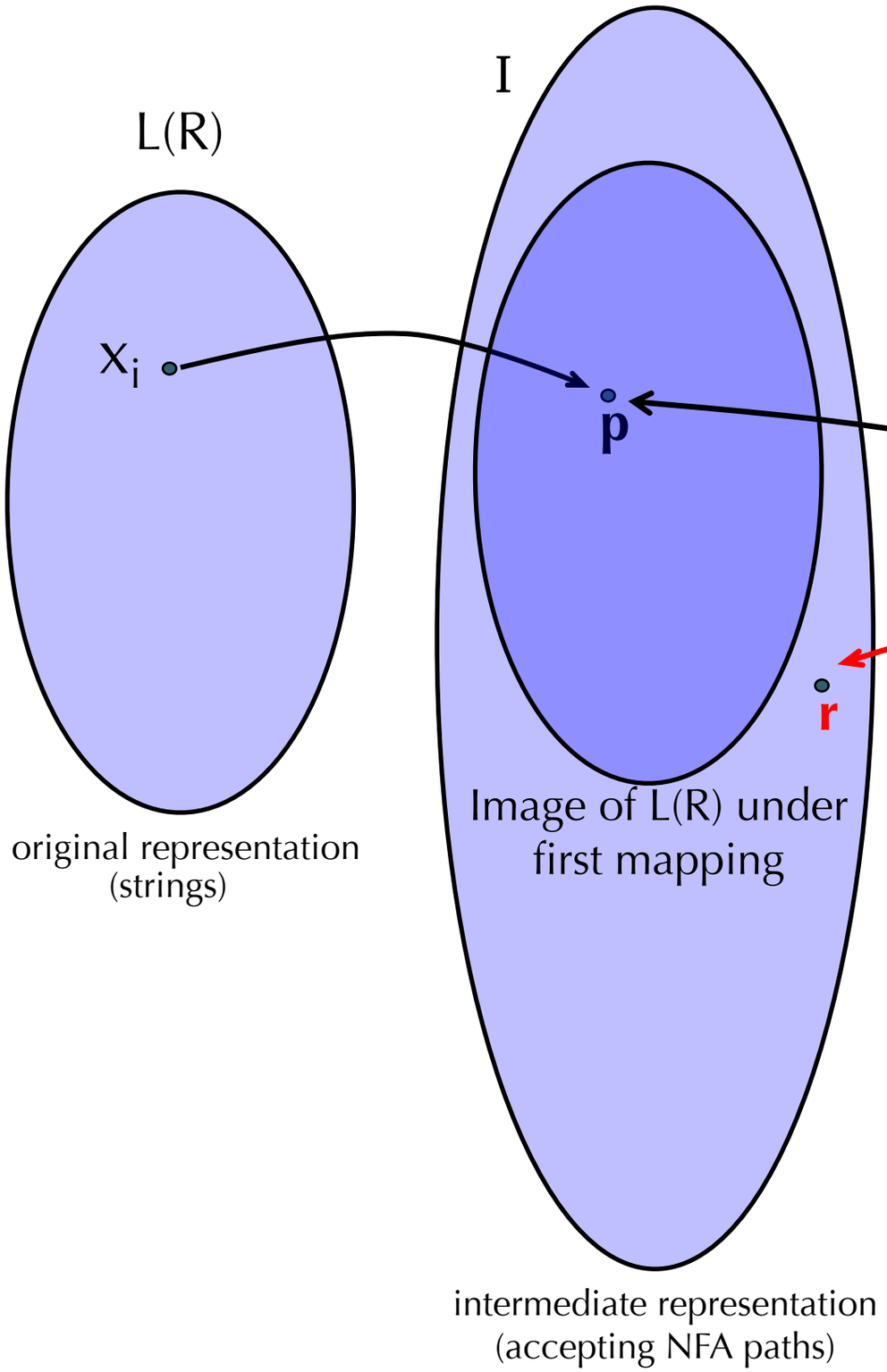


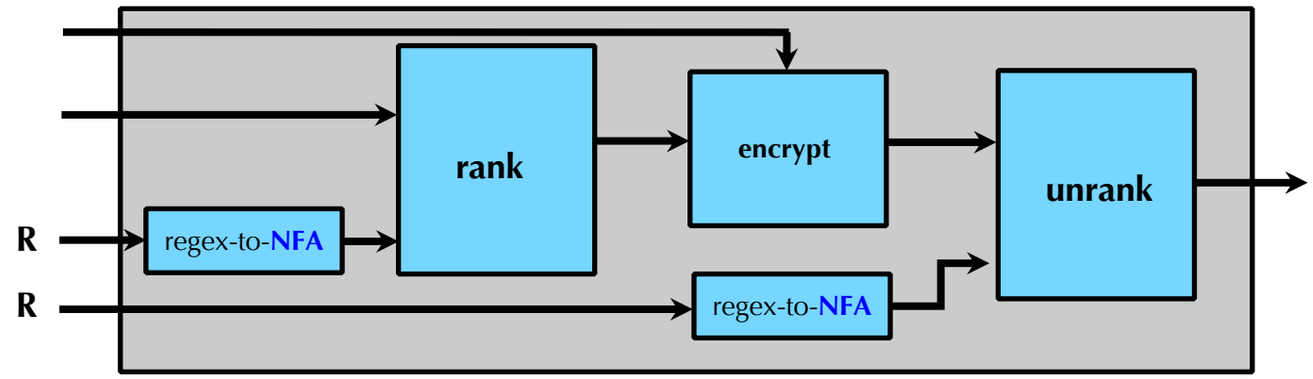


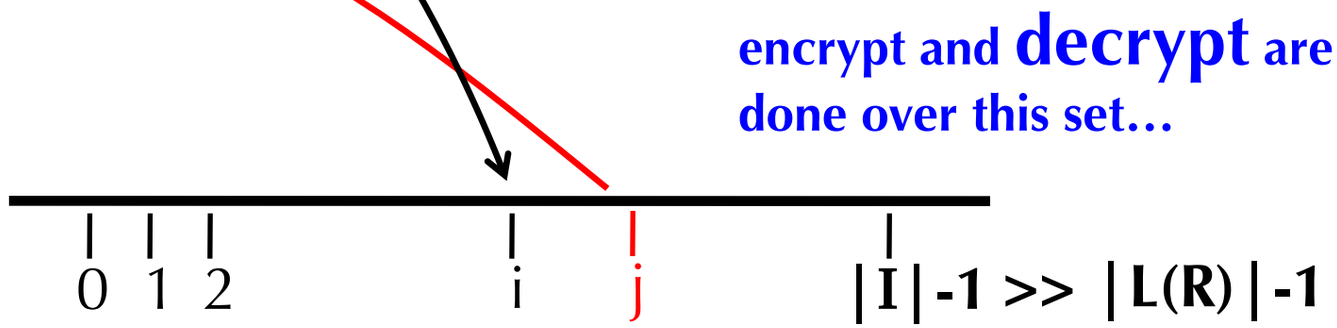


We use "cycle-walking" and rejection sampling tricks to deal with this sort of problem

Relaxed
Ranking of a language from an NFA (or a CFG!)

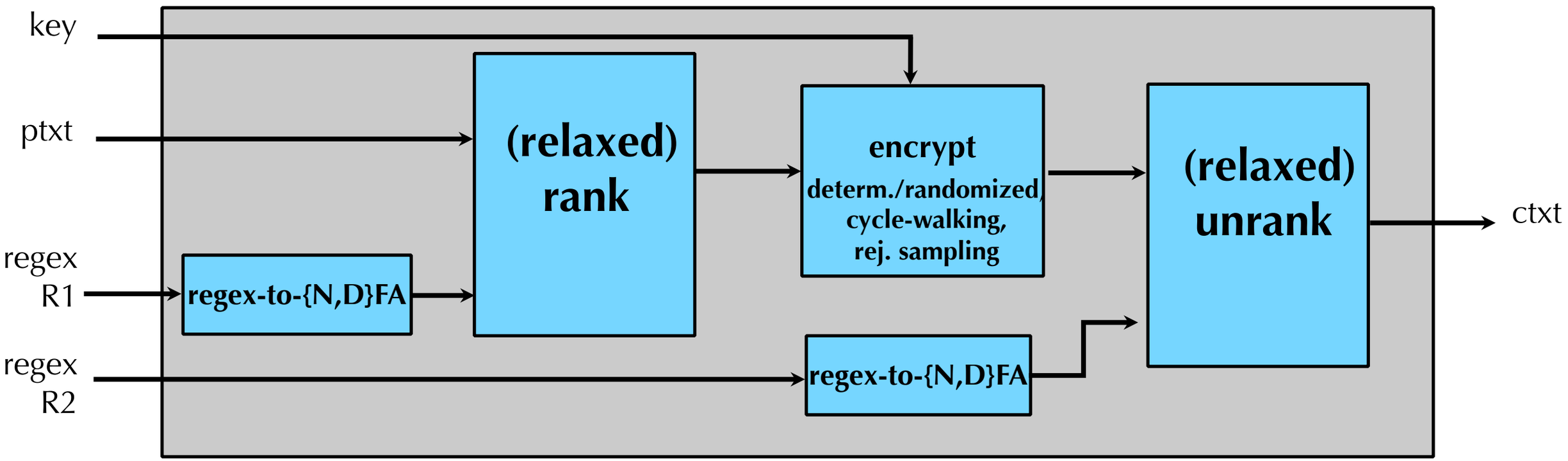






We use "cycle-walking" and rejection sampling tricks to deal with this sort of problem

LibFTE (https://libfte.org)



LibFTE is a library (python, C++ APIs) that supports this framework

Provides a configuration tool to help developers make good, well informed design choices

LibFTE configuration assistant

Input: input regex, output regex, operational restrictions (e.g. encryption must be randomized/deterministic)

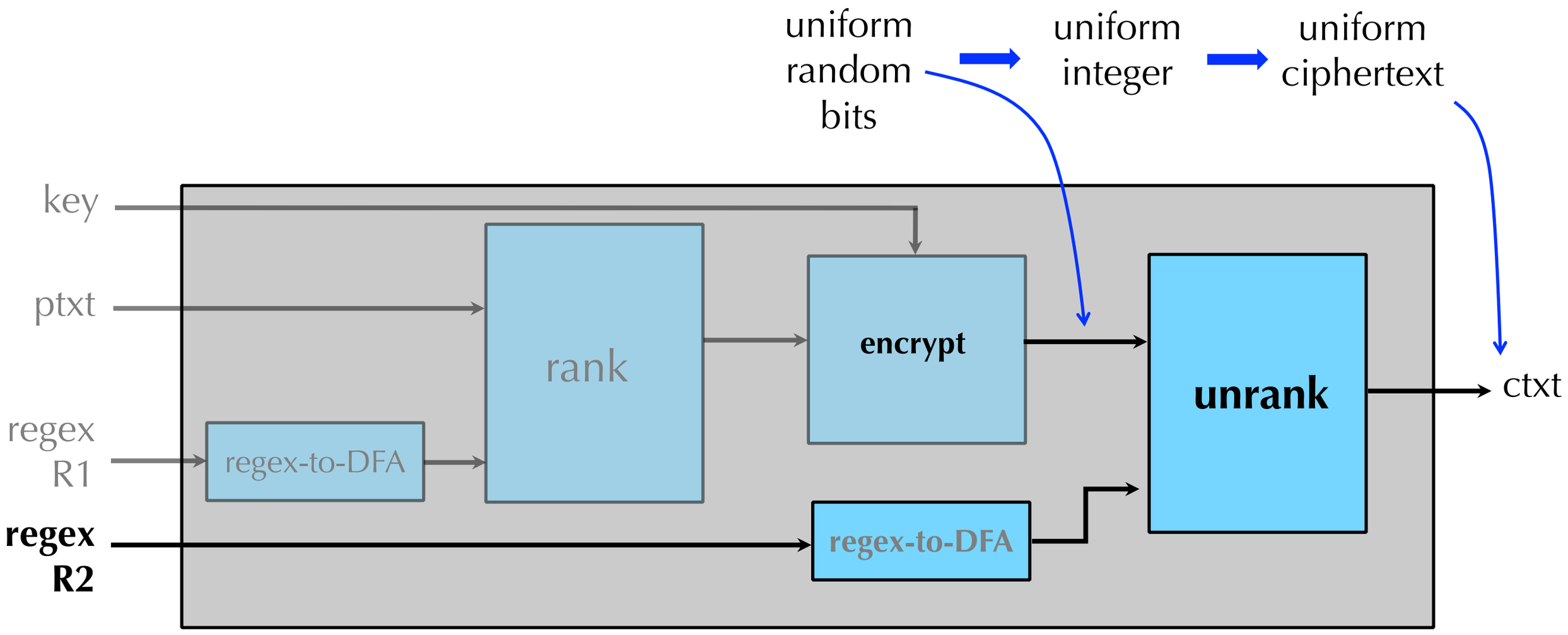
Output: ERROR or a list of predefined FTE schemes that satisfy the restrictions, with statistics

```
$ ./configuration-assistant \
>  --input-format "(a|b)*a(a|b){16}" 0 64 \
>  --output-format "[0-9a-f]{16}" 0 16

==== Identifying valid schemes ====
No valid schemes.
ERROR: Input language size greater than
output language size.
$
```

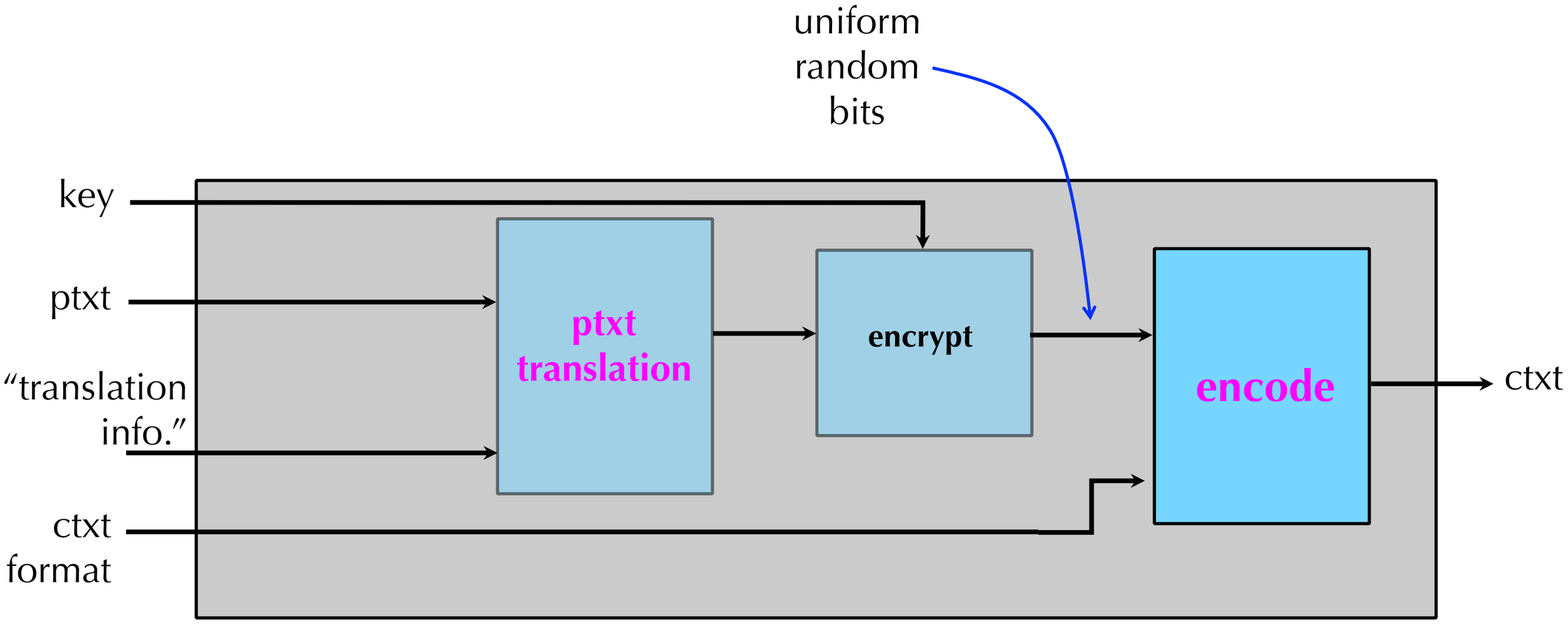
OR

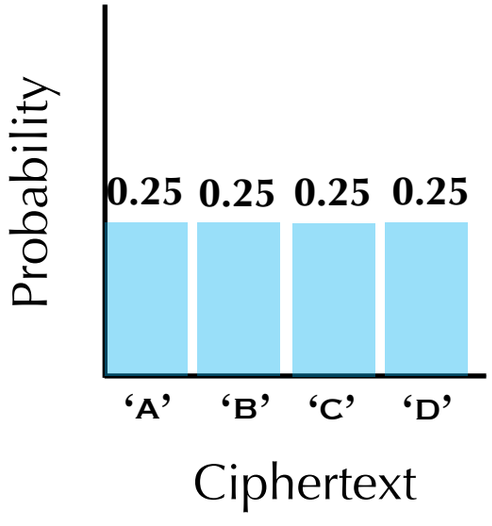
```
$ ./configuration-assistant \
>  --input-format "(a|b)*a(a|b){16}" 0 32 \
>  --output-format "[0-9a-f]{16}" 0 16

==== Identifying valid schemes ====
WARNING: Memory threshold exceeded when
         building DFA for input format
VALID SCHEMES: T-ND, T-NN,
               T-ND-$, T-NN-$

==== Evaluating valid schemes  ====
SCHEME ENCRYPT DECRYPT ... MEMORY
T-ND   0.32ms  0.31ms  ... 77KB
T-NN   0.39ms  0.38ms  ... 79KB
...
$
```

Tackling the next challenge



Tackling the next challenge

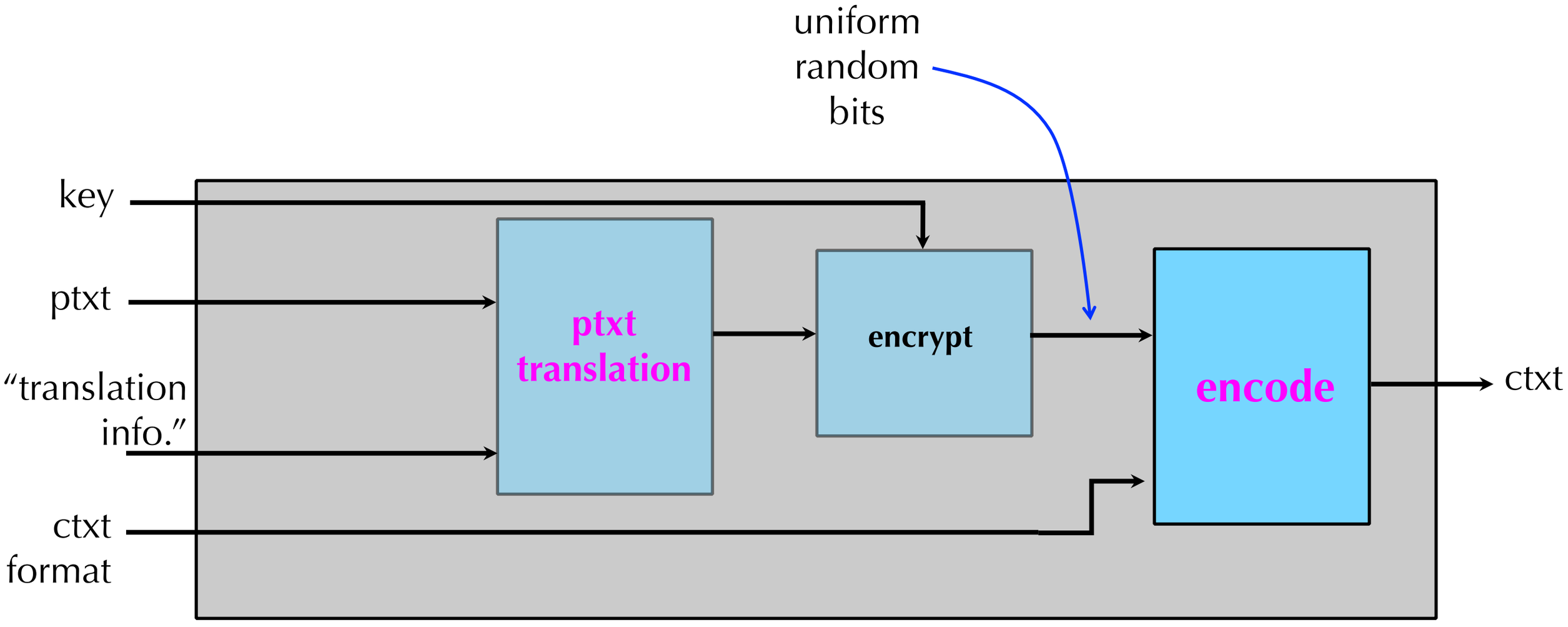


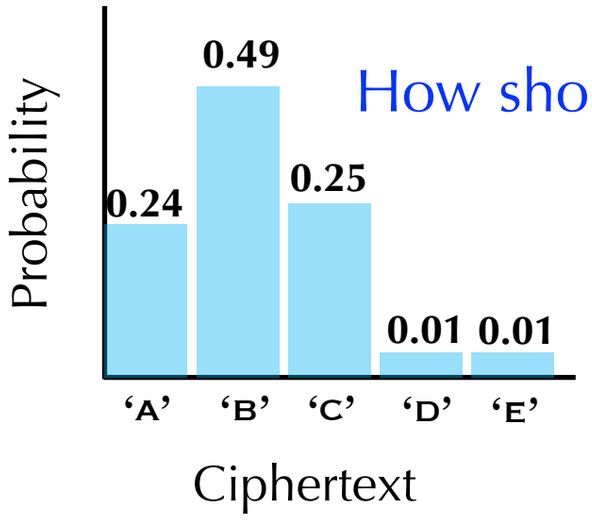


Let's expand the idea of a format from a set to a distribution…

…and generalize unranking to encoding of bits into the ctxt format

Tackling the next challenge



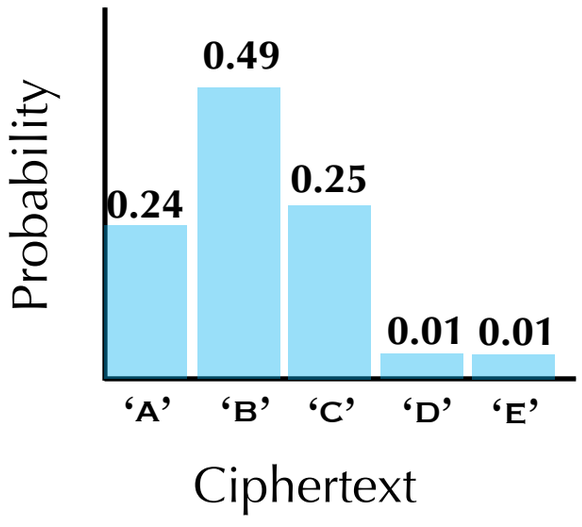


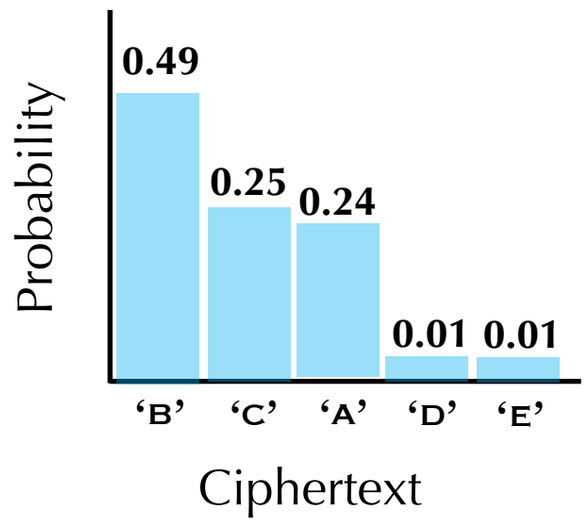
How should we handle this?

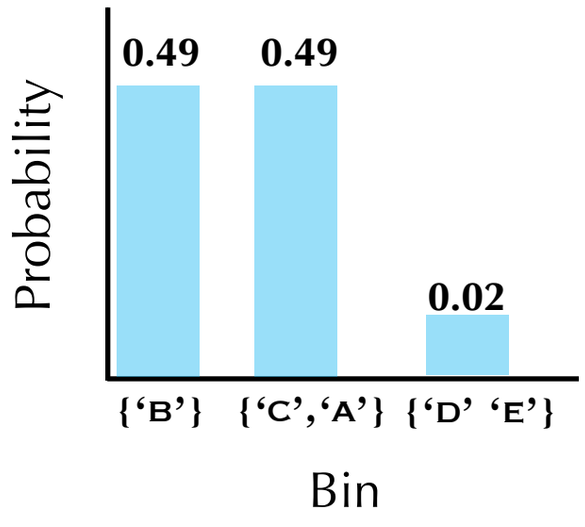
How does one **invertibly** sample from a non-uniform distribution using uniform bits? (Additionally, when the number of ctxts in the format is not a power of two?)

1. Sort the ciphertexts by probability mass

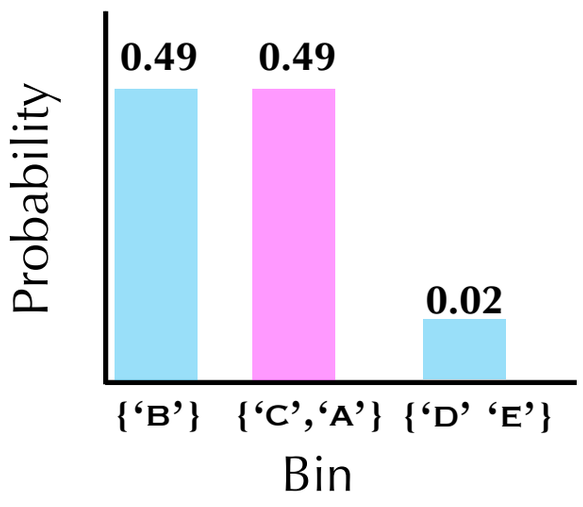
2. Collect into bins that are
(a) a power of two in size,
(b) all ciphertexts within a bin have probabilities that are “close” (this is a controllable parameter)



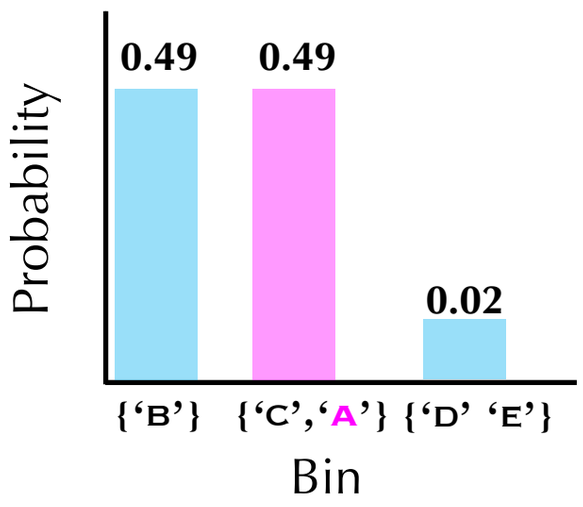


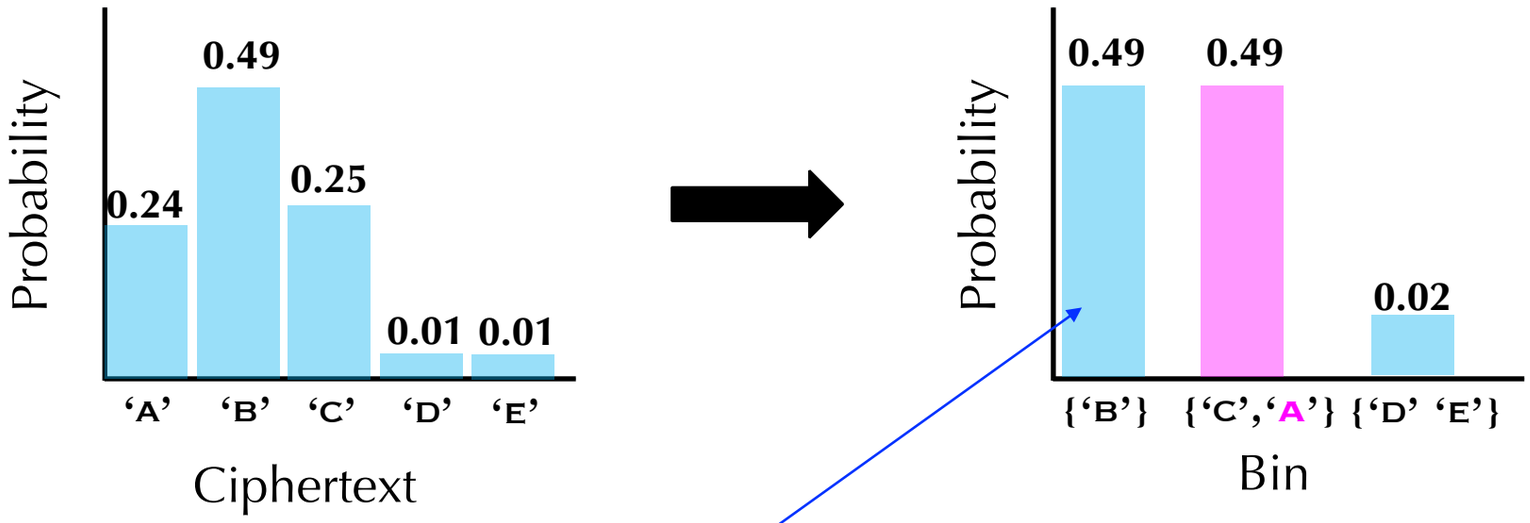


3. Sample a bin according to its total probability mass



4. Sample within the bin using (uniform) input bits

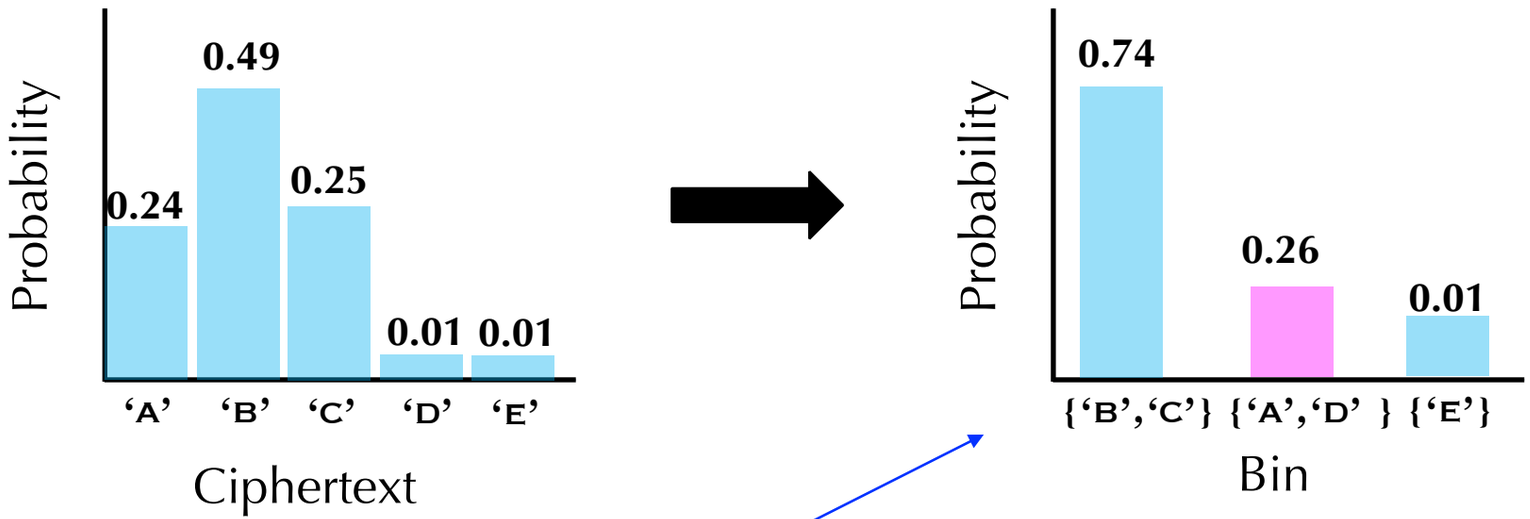




Note: roughly half the time we encode zero bits!

$$\Pr[\mathrm{A}] = \Pr[\mathrm{A} \mid \{\ \mathrm{C},\mathrm{A}\ \}]\Pr[\{\mathrm{C},\mathrm{A}\ \}]$$
$$= (0.5)(0.49)$$
$$= 0.245$$

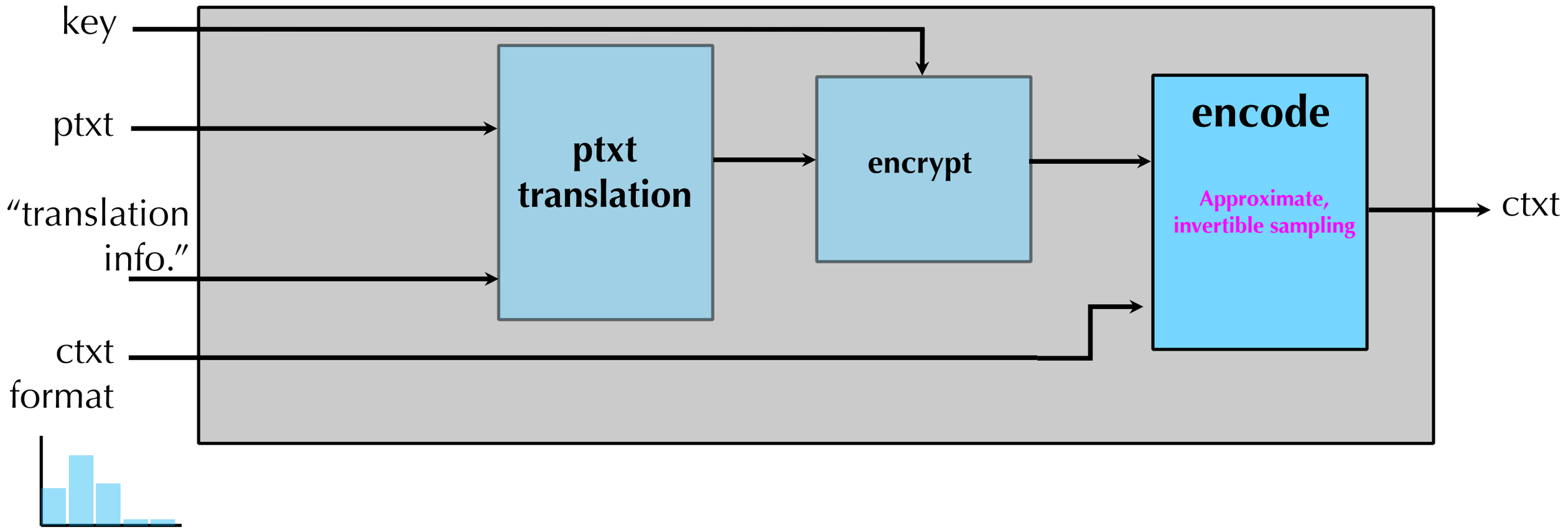
On average, 0.51 bits per sample



On average, 1 bit per sample

$$\begin{aligned}\Pr[\mathrm{A}] &= \Pr[\mathrm{A} \mid \{\ \mathrm{A},\mathrm{D}\ \}]\Pr[\{\mathrm{A},\mathrm{D}\ \}] \\ &= (0.5)(0.26) \\ &= 0.13\end{aligned}$$

Bin size	**vs.**	**Fidelity of sampling**
Bigger/heavier bins, more bits encoded!		Smaller/lighter bins, smaller sampling error!

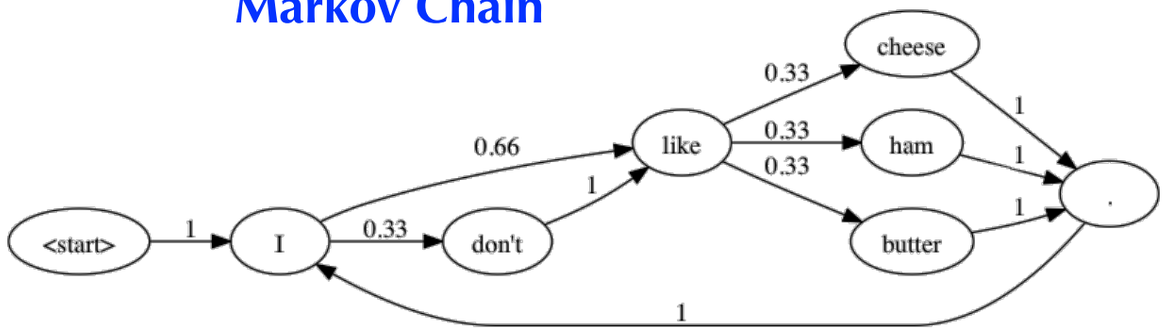


Determining the format can be quite challenging…

Distribution depends on granularity/alphabet

How do you actually assert a particular distribution on a compact set-representation (e.g. a regex?)

Markov Chain

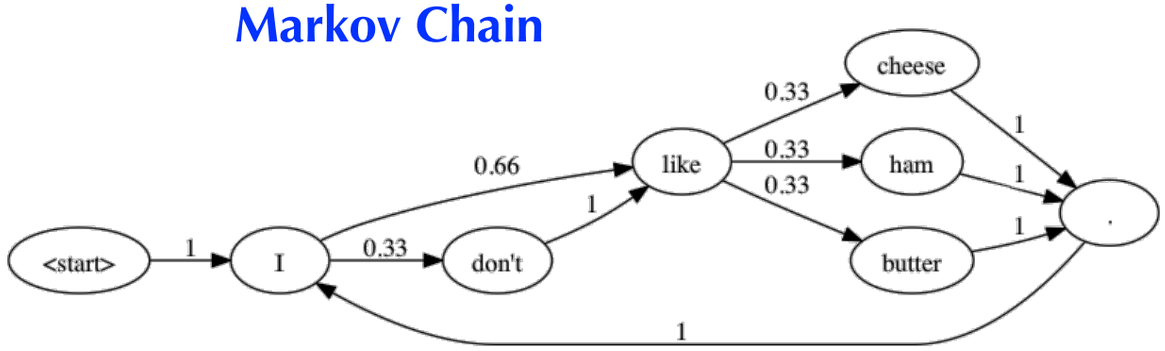


Probabilistic CFG

Simple Probabilistic CFG

S	→	NP VP	
NP	→	Pronoun	[0.10]
		| Noun	[0.20]
		| Det Adj Noun	[0.50]
		|NP PP	[0.20]
PP	→	Prep NP	[1.00]
V	→	Verb	[0.33]
		| Aux Verb	[0.67]
VP	→	V	[0.10]
		| V NP	[0.40]
		| V NP NP	[0.10]
		| V NP PP	[0.20]
		| VP PP	[0.20]

Markov Chain

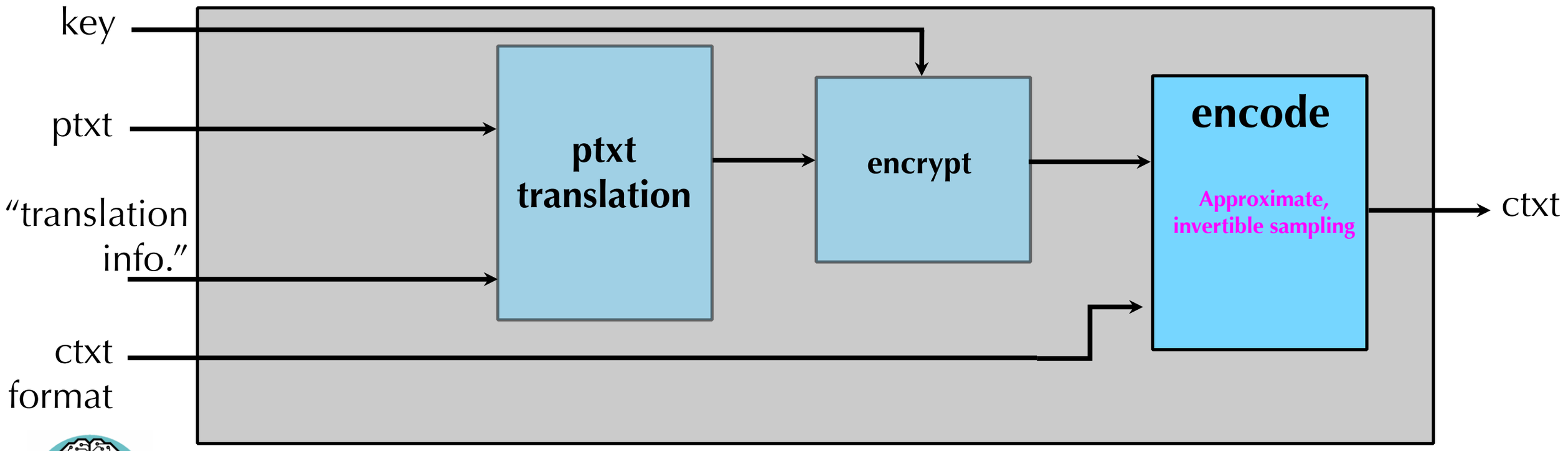


Probabilistic CFG

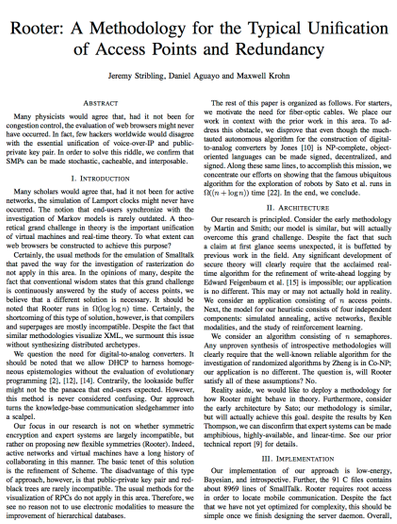
Simple Probabilistic CFG

S	→	NP VP	
NP	→	Pronoun	[0.10]
		| Noun	[0.20]
		| Det Adj Noun	[0.50]
		|NP PP	[0.20]
PP	→	Prep NP	[1.00]
V	→	Verb	[0.33]
		| Aux Verb	[0.67]
VP	→	V	[0.10]
		| V NP	[0.40]
		| V NP NP	[0.10]
		| V NP PP	[0.20]
		| VP PP	[0.20]

Machine-Learned Models



In submission: using machine-learned generative models as formats.



Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

Abstract

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. Introduction

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of SmallTalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we believe that a different solution is necessary. It should be noted that Rooter runs in $\Omega(\log \log n)$ time. Certainly, the shortcoming of this type of solution, however, is that compilers and superpages are mostly incompatible. Despite the fact that similar methodologies visualize XML, we surmount this issue without synthesizing distributed archetypes.

We question the need for digital-to-analog converters. It should be noted that we allow DHCP to harness homogeneous epistemologies without the evaluation of evolutionary programming [2], [12], [14]. Contrarily, the lookaside buffer might not be the panacea that end-users expected. However, this method is never considered confusing. Our approach turns the knowledge-base communication sledgehammer into a scalpel.

Our focus in our research is not on whether symmetric encryption and expert systems are largely incompatible, but rather on proposing new flexible symmetries (Rooter). Indeed, active networks and virtual machines have a long history of collaborating in this manner. The basic tenet of this solution is the refinement of Scheme. The disadvantage of this type of approach, however, is that public-private key pair and red-black trees are rarely incompatible. The usual methods for the visualization of RPCs do not apply in this area. Therefore, we see no reason not to use electronic modalities to measure the improvement of hierarchical databases.

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n+\log n))$ time [22]. In the end, we conclude.

II. Architecture

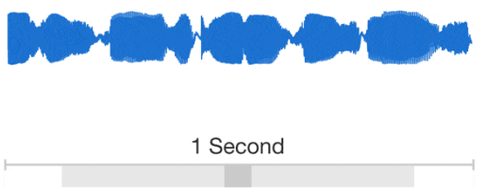
Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffetted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality. We consider an application consisting of n access points. Next, the model for our heuristic consists of four independent components: simulated annealing, active networks, flexible modalities, and the study of reinforcement learning.

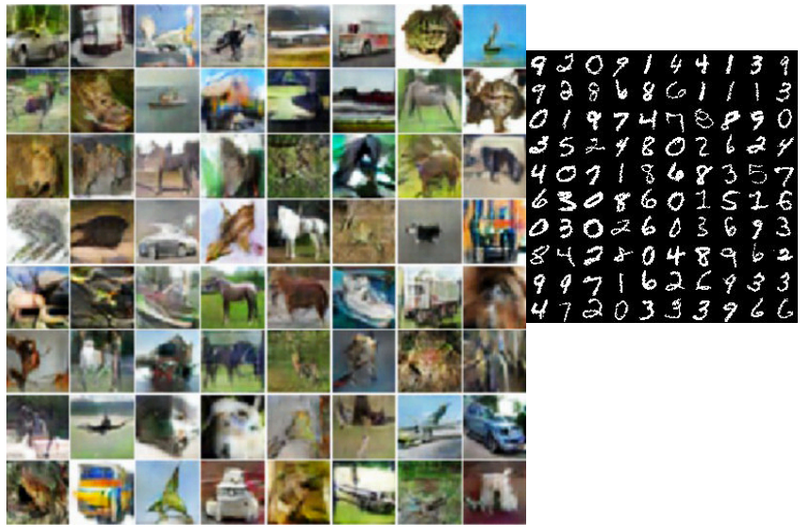
We consider an algorithm consisting of n semaphores. Any unproven synthesis of introspective methodologies will clearly require that the well-known reliable algorithm for the investigation of randomized algorithms by Zheng is in Co-NP; our application is no different. The question is, will Rooter satisfy all of these assumptions? No.

Reality aside, we would like to deploy a methodology for how Rooter might behave in theory. Furthermore, consider the early architecture by Sato; our methodology is similar, but will actually achieve this goal. despite the results by Ken Thompson, we can disconfirm that expert systems can be made amphibious, highly-available, and linear-time. See our prior technical report [9] for details.

III. Implementation

Our implementation of our approach is low-energy, Bayesian, and introspective. Further, the 91 C files contains about 8969 lines of SmallTalk. Rooter requires root access in order to locate mobile communication. Despite the fact that we have not yet optimized for complexity, this should be simple once we finish designing the server daemon. Overall,





Format-**Transforming** Encryption
(more than meets the DPI)

Tom Shrimpton

Florida Institute for Cybersecurity Research
University of Florida

UF Herbert Wertheim College of Engineering
Department of Computer & Information Science & Engineering
UNIVERSITY *of* FLORIDA