LEAKWATCH

Estimating Information Leakage from Java Programs

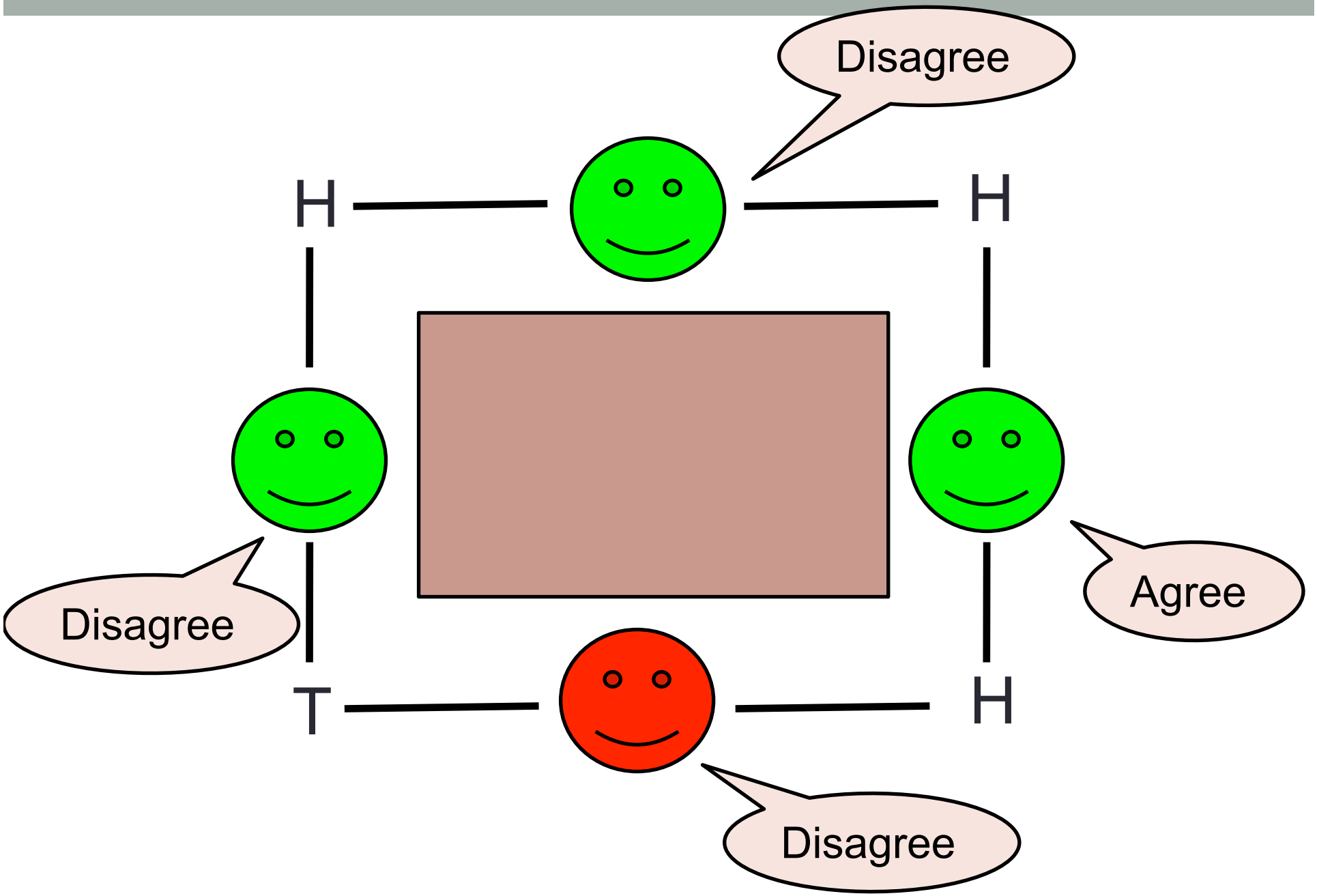
Tom Chothia
Yusuke Kawamoto
Chris Novakovic

Introduction

- Intro. to information leakage measurement:
  - Dining Cryptographers
  - Mutual Information & Min-entropy leakage

- AIM: calculate information leakage from Java Programs using statistical methods.
  - 1st Step: statistical estimation results for probabilistic mutual information and min-entropy leakage.
  - 2nd Step: a Java Framework

- RESULT: Estimation of information leakage for complex, probabilistic Java Programs (but small secret domains).


Disagree
H
H
Disagree
Agree
T
H
Disagree

Leakage

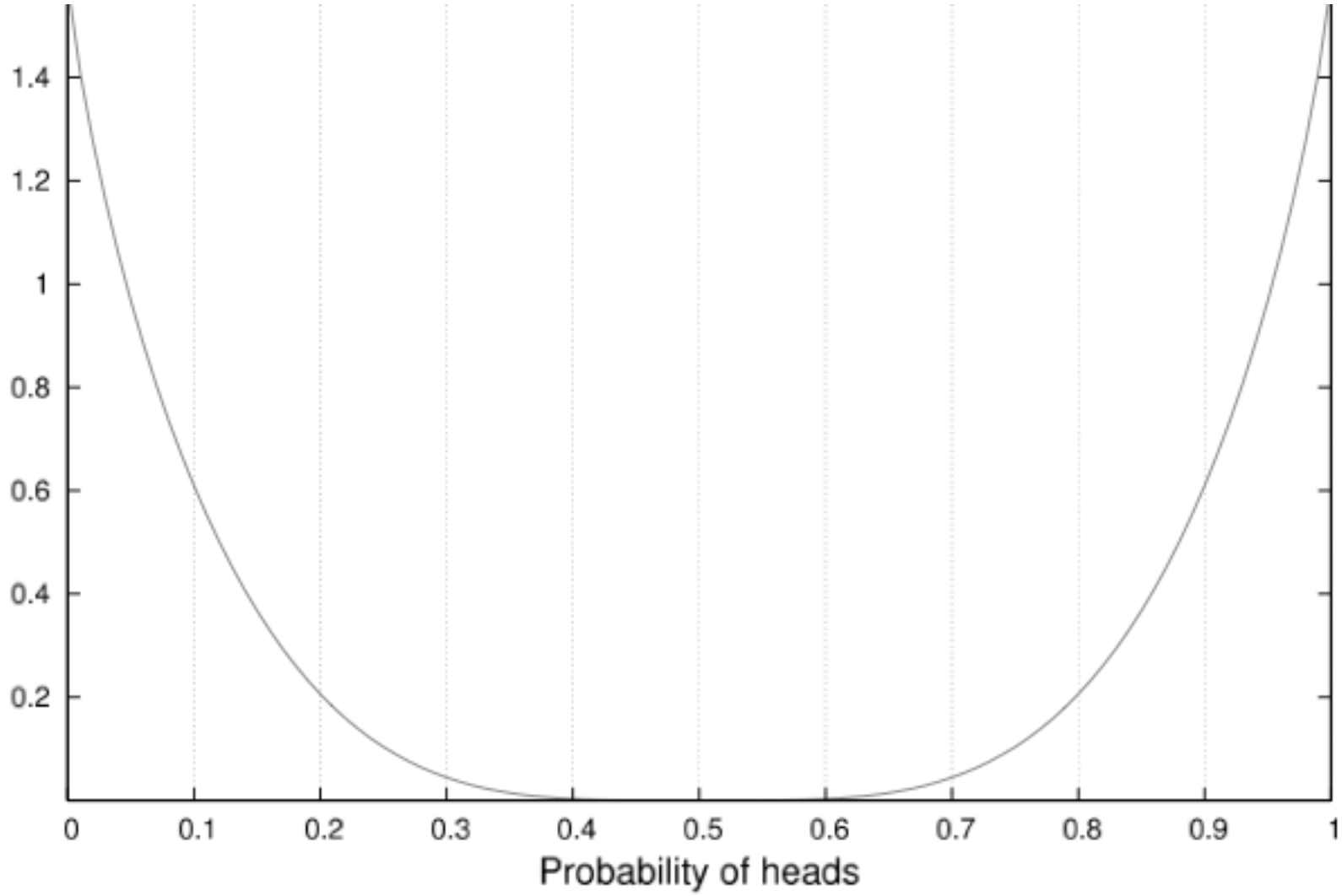
- We assume some distribution on secrets S (p(x)) and the observables O (p(y)).
- The system then defines p(y|x) & p(x,y)
- We can then measure the leakage from S to O, using mutual information:

$$I(X;Y) = \sum p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

- Or min-entropy leakage:

$$L(X;Y) = \log \sum_{Y} p(y) \max_{X} p(x \mid y) - \log \max_{X} p(x)$$

What happens if the coins are bias?



Summary

- Intro. to information leakage measurement:
  - Dining Cryptographers
  - Mutual Information & Min-entropy leakage

- AIM: calculate information leakage from Java Programs using statistical methods.
  - 1st Step: statistical estimation results for probabilistic mutual information and min-entropy leakage.
  - 2nd Step: a Java Framework

- RESULT: Estimation of information leakage for complex, probabilistic Java Programs (but small secret domains).

Estimating mutual information

- We need to know how the estimated values of mutual information relates to the real value.

- Past result: Estimates $\hat{I}(X;Y)$ have:

Mean

$$I(X;Y) + \frac{(\#\mathcal{X}-1)(\#\mathcal{Y}-1)}{2n} + O\left(\frac{1}{n^2}\right)$$

Variance

$$\frac{1}{n}\left(\sum_{x,y}\hat{P}_{XY}(x,y)\log^2\left(\frac{\hat{P}_{XY}(x,y)}{\hat{P}_X(x)\hat{P}_Y(y)}\right) - \left(\sum_{x,y}\hat{P}_{XY}(x,y)\log\left(\frac{\hat{P}_{XY}(x,y)}{\hat{P}_X(x)\hat{P}_Y(y)}\right)\right)^2\right) + O\left(\frac{1}{n^2}\right)$$

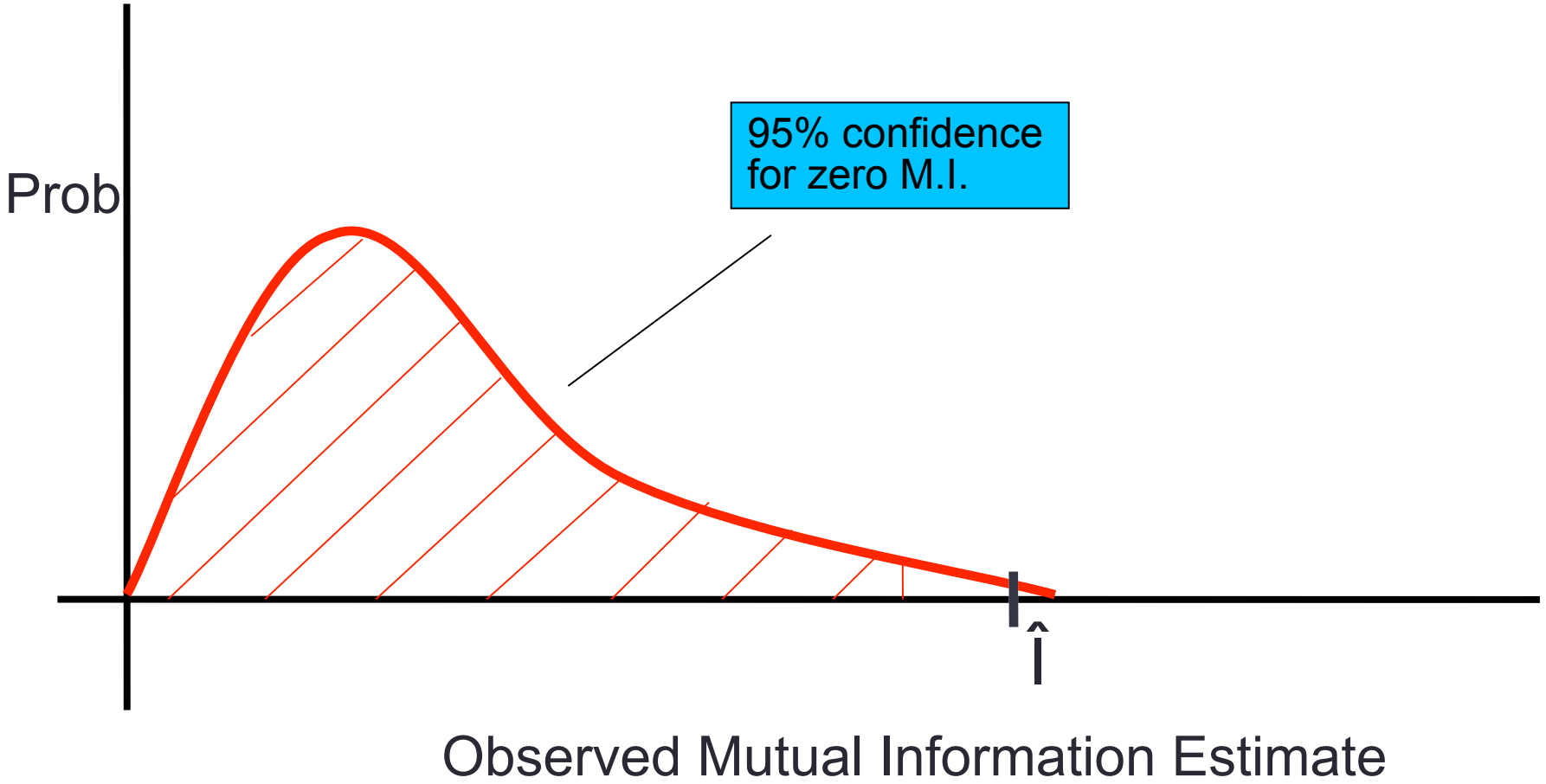
If there is no leakage …

- If there in no information leakage, i.e., I(X;Y) = 0
- The $O(n^{-2})$ term cancels out and the estimate comes from a Chi-squared distribution:

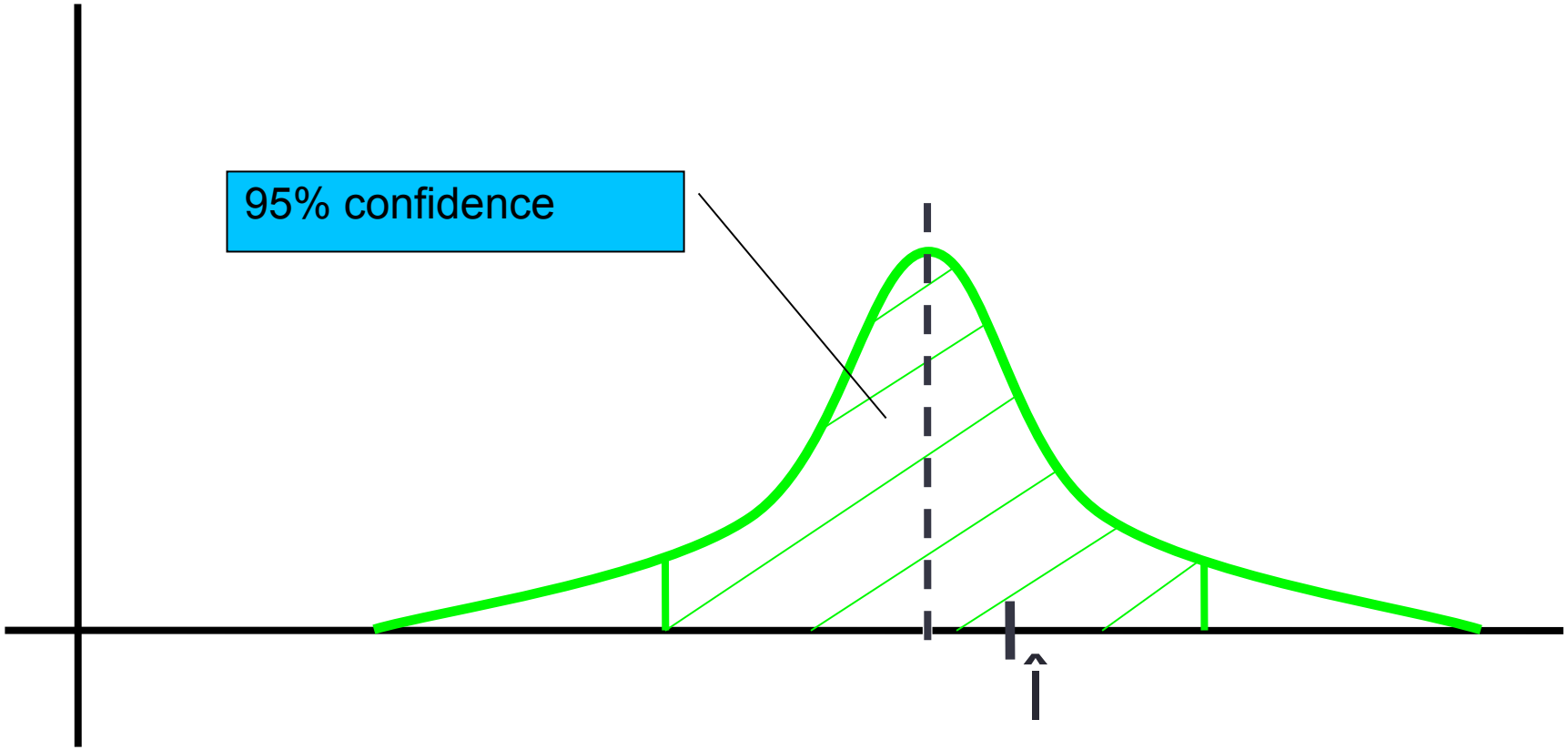
When $I(X;Y) = 0$

$$\hat{I}(X;Y) \propto \frac{\chi^2((\#\mathcal{X}-1)(\#\mathcal{Y}-1))}{2n}$$

Using the Distributions



Using the Distributions



Observed Mutual Information Estimate

Estimation of Min-Entropy Leakage

$$L(X;Y) = \log \sum_{Y} \max_{X} p(x,y) - \log \max_{X} \sum_{Y} p(x,y)$$

The max makes estimation hard

e.g.

p(0,0) = 0.3

p(0,1) = 0.3

p(1,0) = 0.3

p(1,1) = 0.1

We can't take the confidence interval around the max p(x,y)

We can't treat all p(x,y) as independent as they sum to 1.

Using the Chi-square Statistic

- The chi-square statistic tells us now close an estimate of a distribution is to the true distribution

- For N samples:

$$\chi^2_{(0.05,(\#X.\#Y)-1)} = \sum_{x,y} \frac{(\hat{p}(x,y)N - p(x,y)N)^2}{p(x,y)N}$$

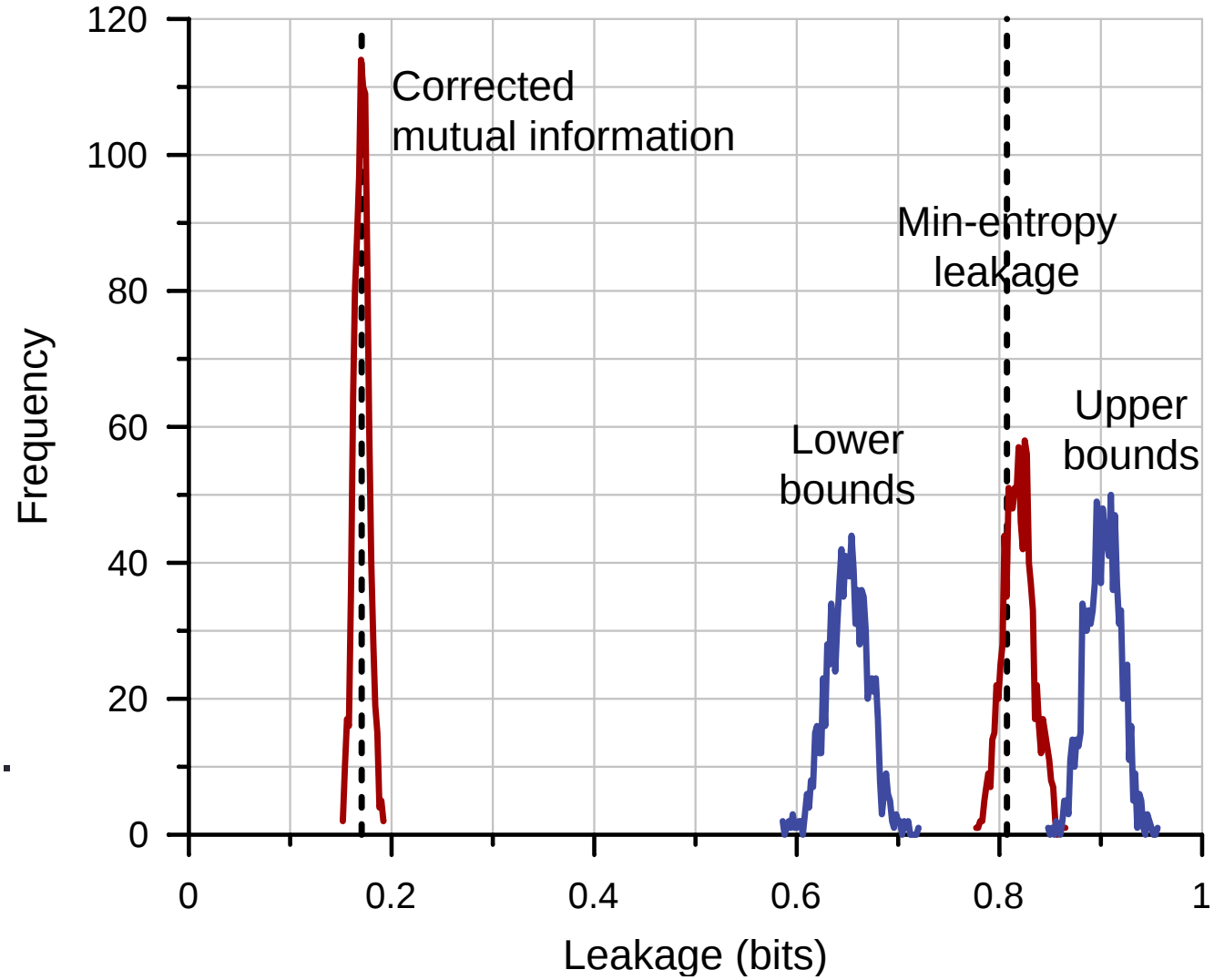
- Our tools estimates $\hat{p}(x,y)$ we then find the most extreme $p(x,y)$ which maximizes for x.

- This is an over estimate, but it provides useful bounds.

Estimation of Min-Entropy Leakage

For the DC protocol with with bias coins

Estimating ME is hard.



Collecting enough samples (heuristic)

- We have shown:

$$\hat{I}(X;Y) = I(X;Y) + \frac{(\#X-1)(\#Y-1)}{2n} + O(n^{-2})$$

- For large n the $O(n^{-2})$ term is small. But how large is large enough?

$$\hat{I}(X;Y) - \frac{(\#X-1)(\#Y-1)}{2n} = I(X;Y) + O(n^{-2})$$

- If $O(n^{-2})$ is not affecting the result this will be constant as no. of samples (n) increases.

- We collect samples until $\hat{I}(X;Y) - \frac{(\#X-1)(\#Y-1)}{2n}$ stops increasing

Summary

- Intro. to information leakage measurement:
  - Dining Cryptographers
  - Mutual Information & Min-entropy leakage

- AIM: calculate information leakage from Java Programs using statistical methods.
  - 1st Step: statistical estimation results for probabilistic mutual information and min-entropy leakage.
  - 2nd Step: a Java Framework

- RESULT: Estimation of information leakage for complex, probabilistic Java Programs (but small secret domains).

Collecting data from programs

- Java or C? We picked Java.
- Collecting samples: Rewrite “secret” and “observable” and execute the program many time??
  - No!!
- We found that starting a JVM is VERY slow.
  - This stops calling “`java program`” many times from being practical.

Collecting data from Java

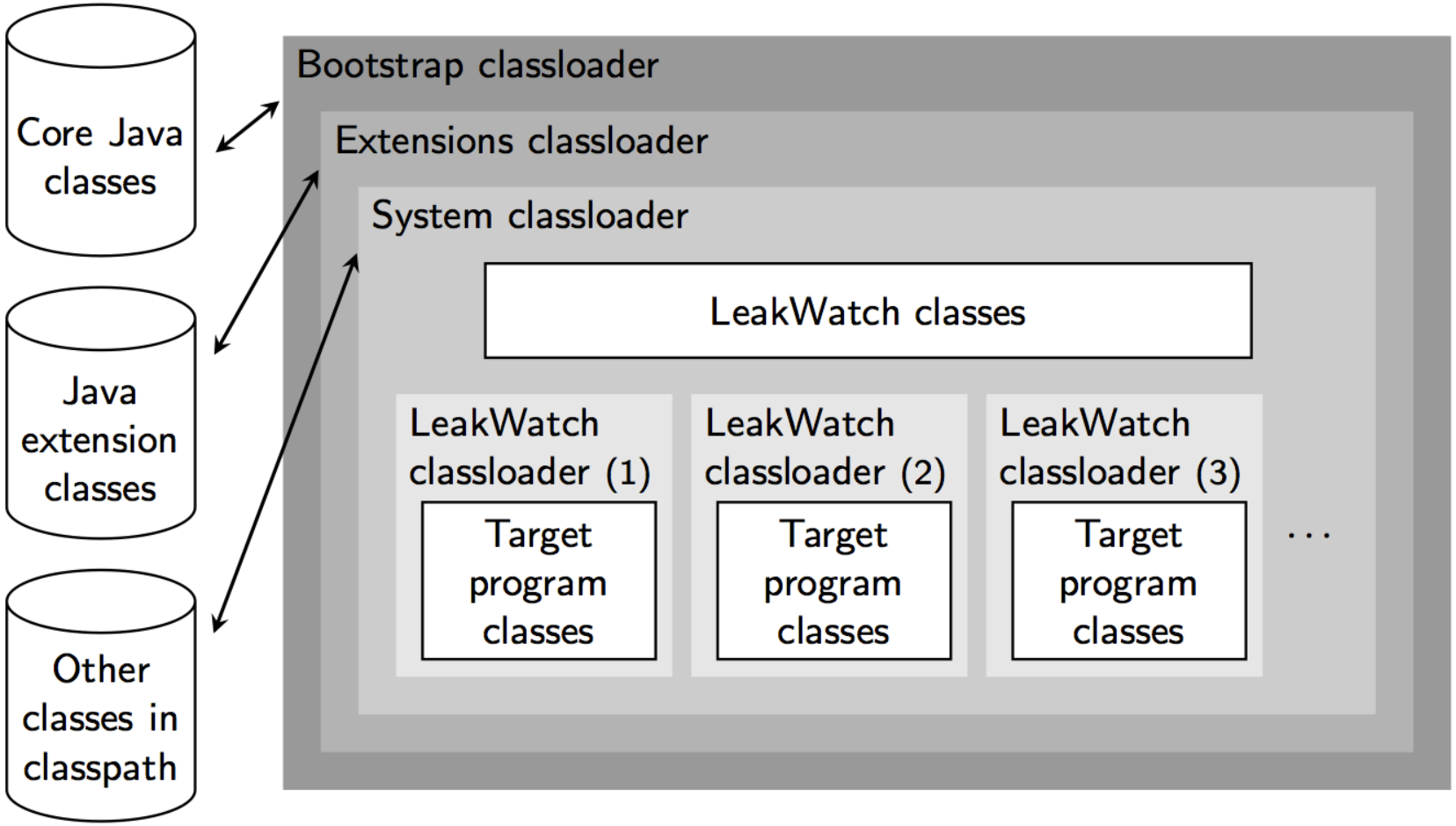
- So we used the Java Classloader to repeatedly load and execute the Java main method.

- Problem?
  - Java caches the Object that contains the main method.
  - Static variables are presented from one run to the next.

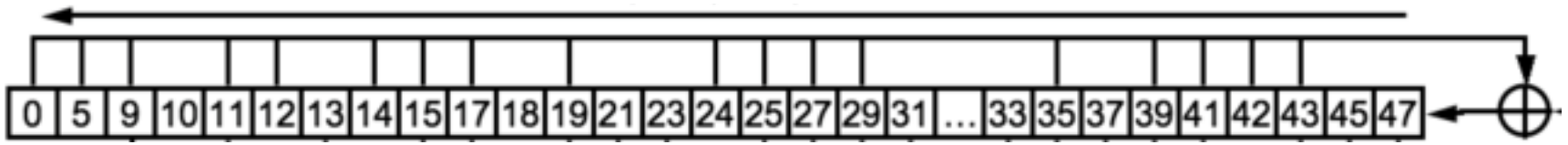
LeakWatch

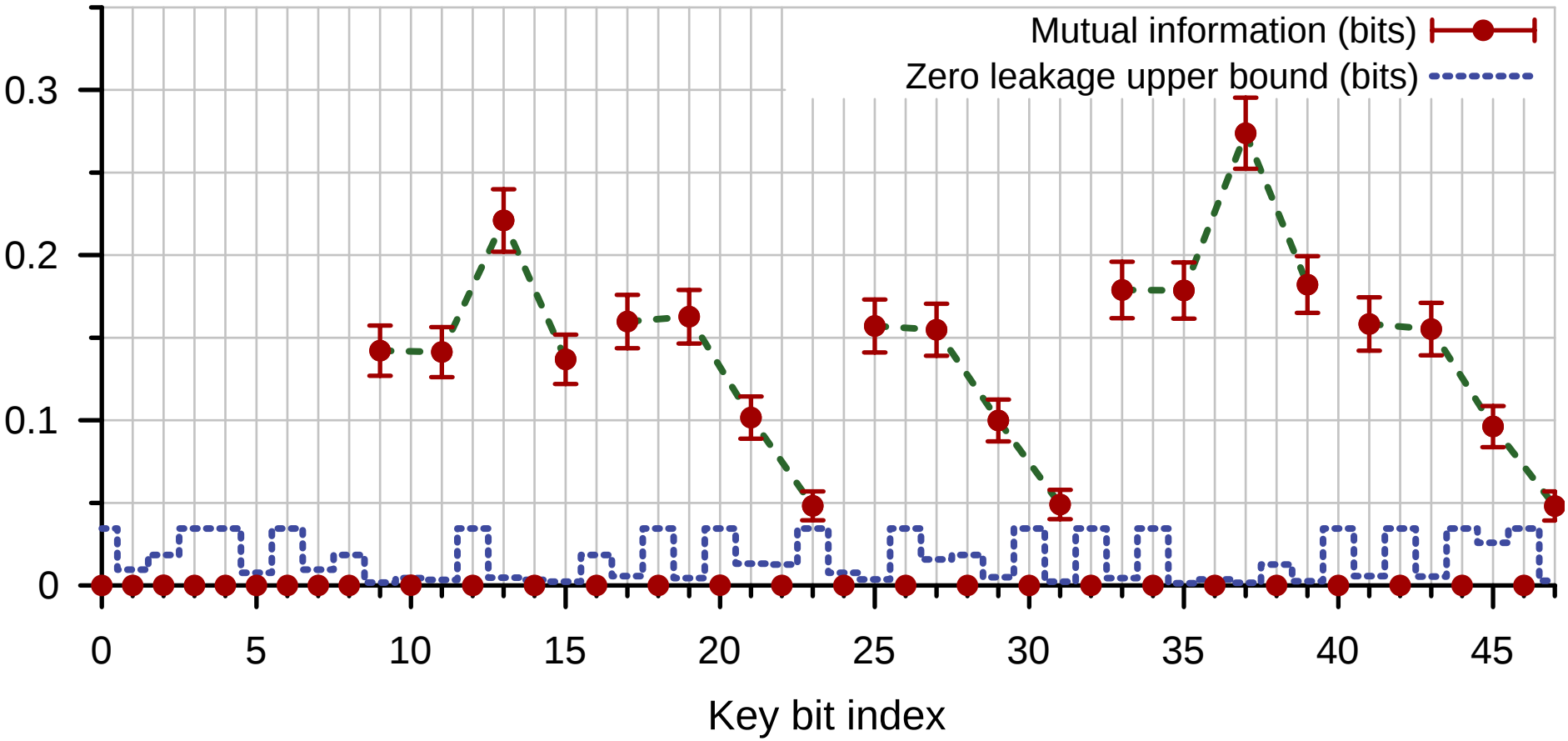
- We have written our own custom Java classloader.
- Class being tested are always reload (not cached).
- System classes with no static values are cached.
- Copies of the test program are run in parallel to take advantage of multicore machines.

Leakwatch Overview

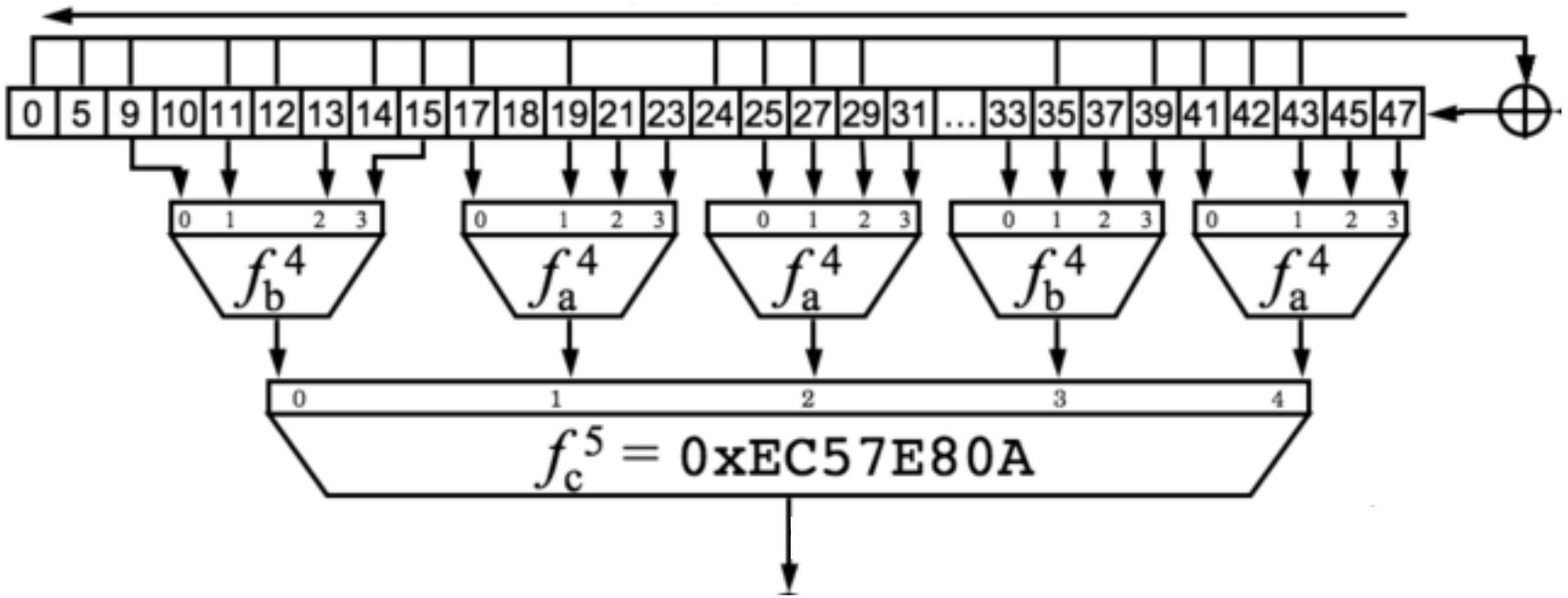


Reverse Engineering the Crypto1 Cipher




Mutual information (bits)
Zero leakage upper bound (bits)
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
Key bit index

Reverse Engineering the Crypto1 Cipher



Conclusion

- AIM: calculate information leakage from Java Programs using statistical methods.
  - 1st Step: statistical estimation results for probabilistic mutual information and min-entropy leakage.
  - 2nd Step: a Java Framework
- RESULT: Estimation of information leakage for complex, probabilistic Java Programs (but small secret domains).
- More examples including an analysis of PGP encryption in the paper and on our website:

`http://www.cs.bham.ac.uk/research/projects/infotools/leakwatch/`