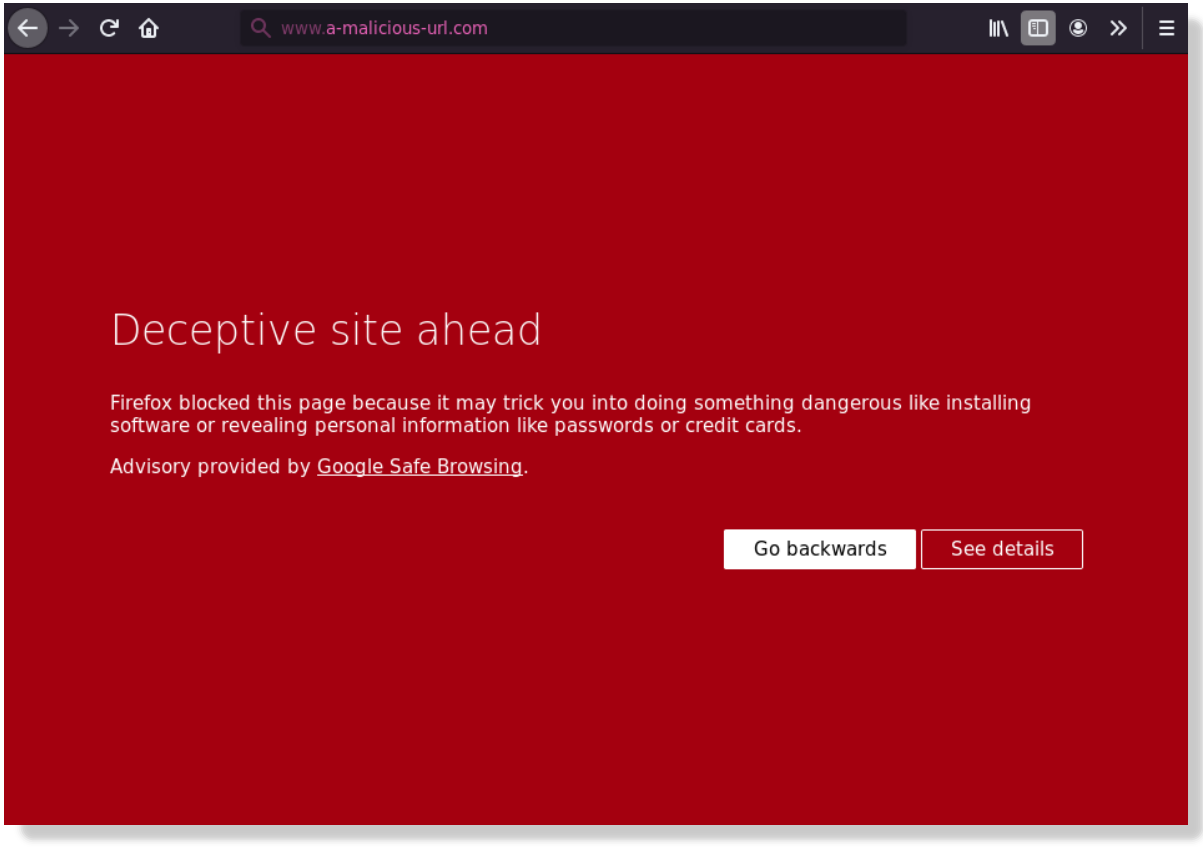
CERAMIST:
Certifying Certainty and Uncertainty

Kiran Gopinathan, Ilya Sergey

National University of Singapore

When clicking on a **malicious** url....

When clicking on a **malicious** url....

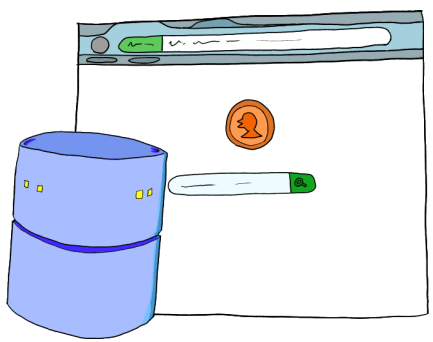


...show a **warning** to the user.

How?

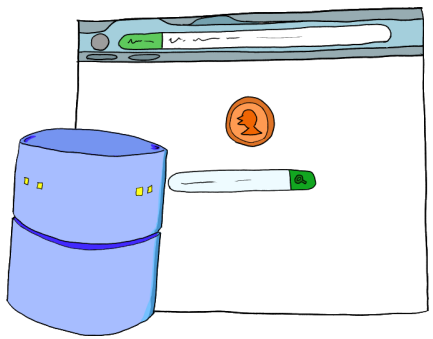
How?

Store locally?



How?

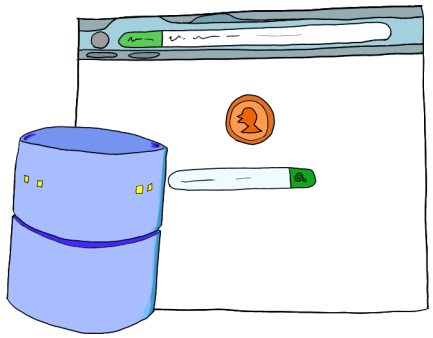
Store locally?



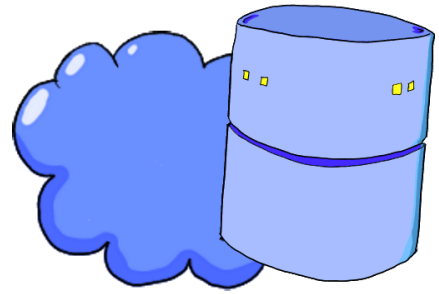
Too large!

How?

Store locally?



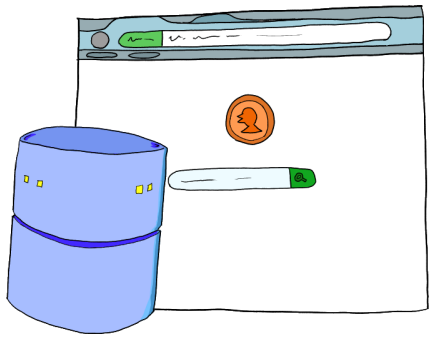
Send to server?



Too large!

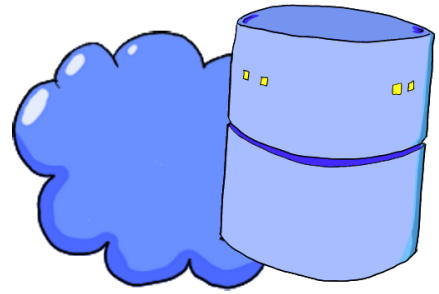
How?

Store locally?



Too large!

Send to server?

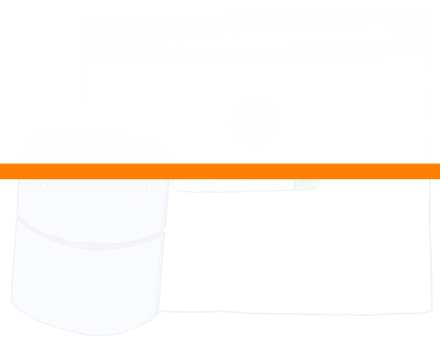


No privacy!

How?

Use a **Bloomfilter**...

Store locally?



Too large!

Send to server?

No privacy!

Use a **Bloomfilter**...

...to **approximately** track bad urls.

Key properties

1. **No False Negatives**

2. **Low False Positives**

Key properties

1. **No False Negatives**

   ... to catch **all** bad urls.

2. **Low False Positives**

Key properties

1. **No False Negatives**

   ... to catch **all** bad urls.

2. **Low False Positives**

   ... to **minimize** privacy violations.

Key properties

1. **No False Negatives**

... to catch **all** bad urls.

****Supposedly** Low **False Positives**

2. ~~**Low False Positives**~~

... to **minimize** privacy violations.

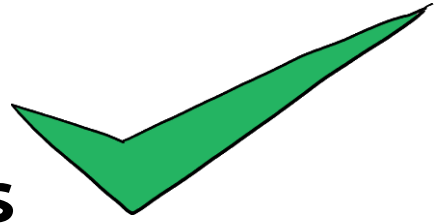
Key properties

1. **No False Negatives**

... to catch **all** bad urls.

Certified

2. **Low False Positives**



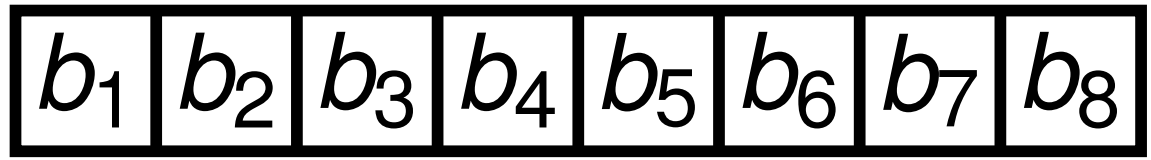
... to **minimize** privacy violations.

Roadmap

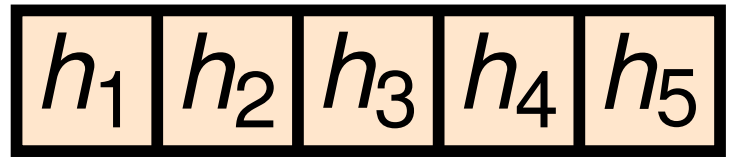
- What are Bloomfilters?
- Encoding in Coq
- Generalizing to other structures

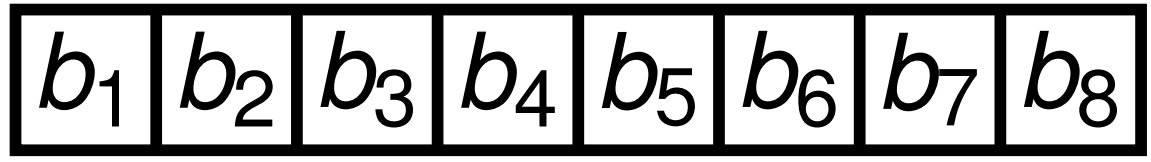
What is a Bloomfilter?

What is a Bloomfilter?



What is a Bloomfilter?

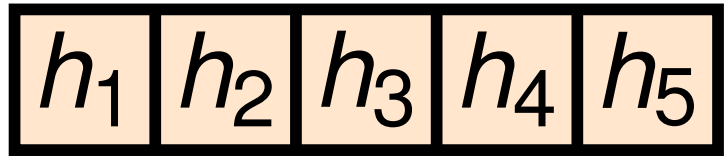


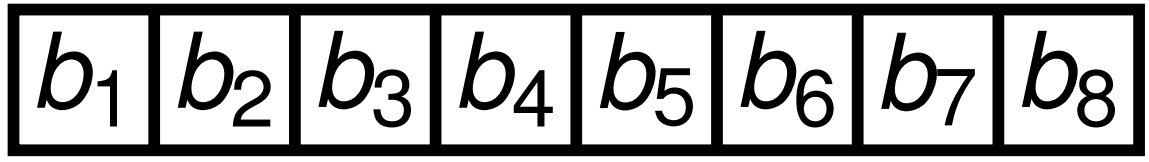


What is a Bloomfilter?

- **Insert**
- **Query**

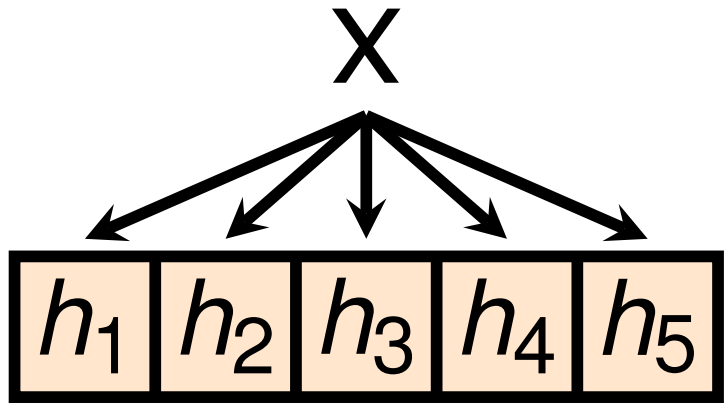
x

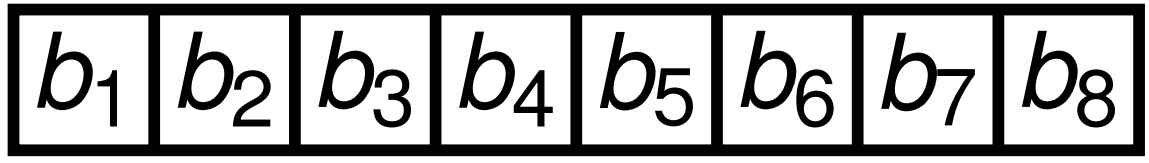




What is a Bloomfilter?

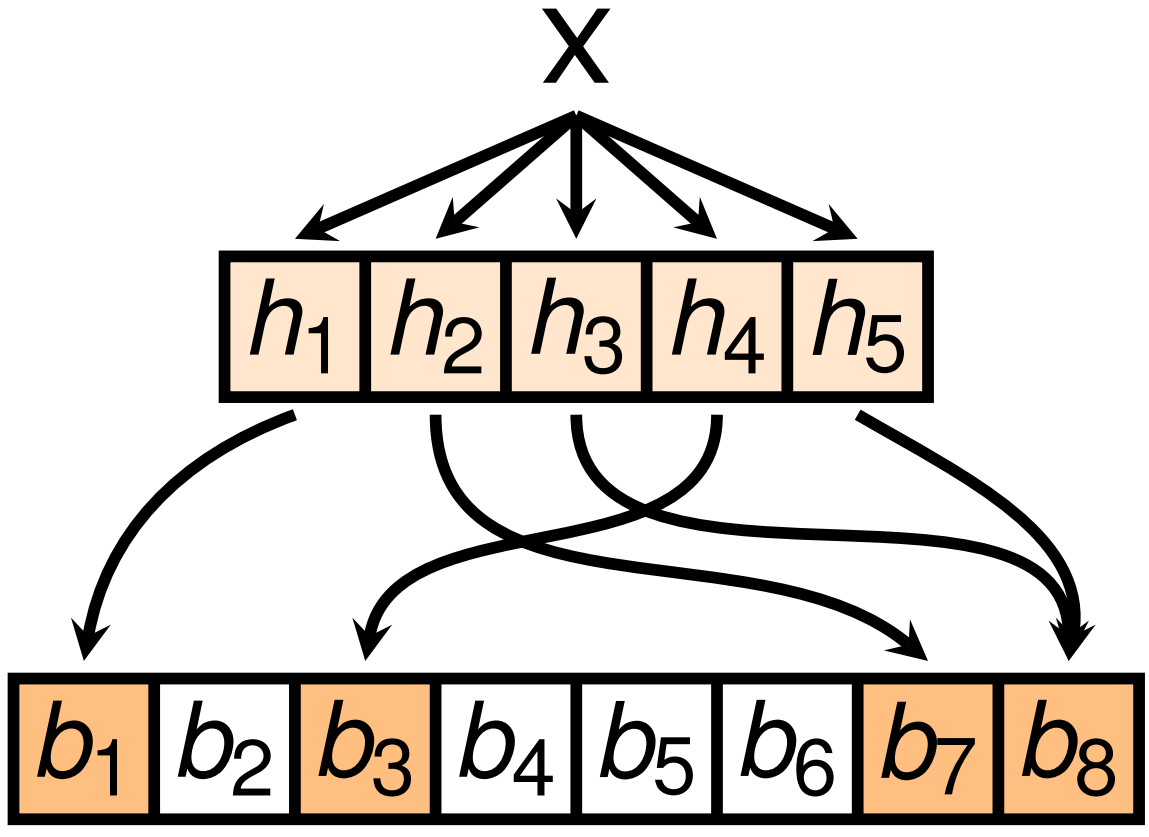
- **Insert**
- **Query**





What is a Bloomfilter?

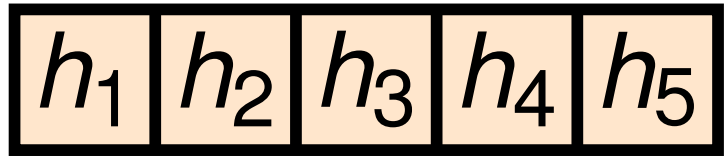
- **Insert**
- **Query**

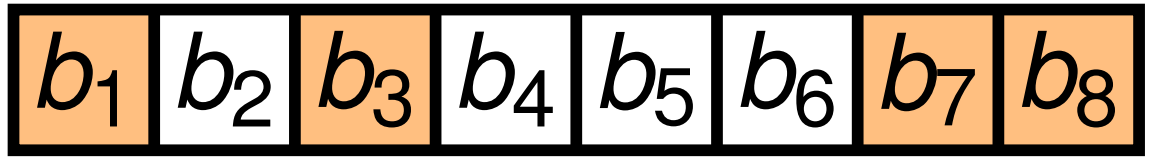


What is a Bloomfilter?

- **Insert**
- **Query**

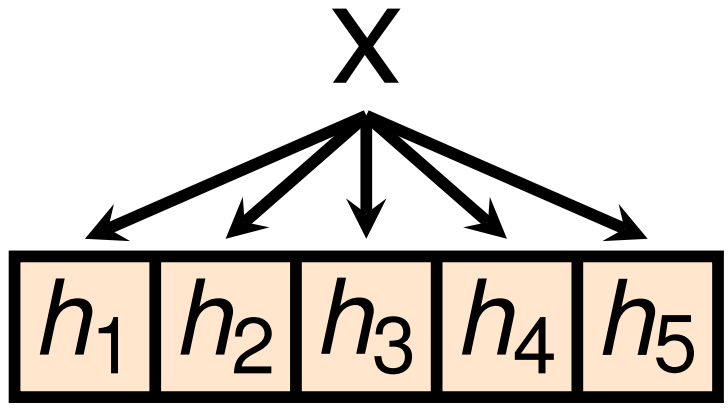
x

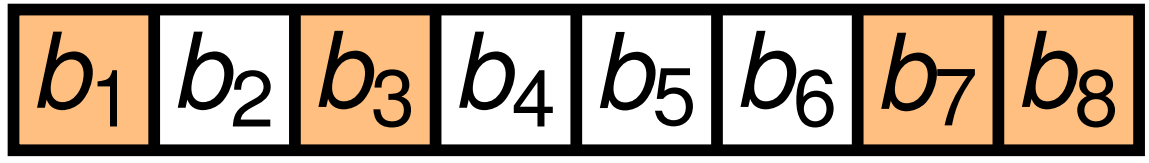




What is a Bloomfilter?

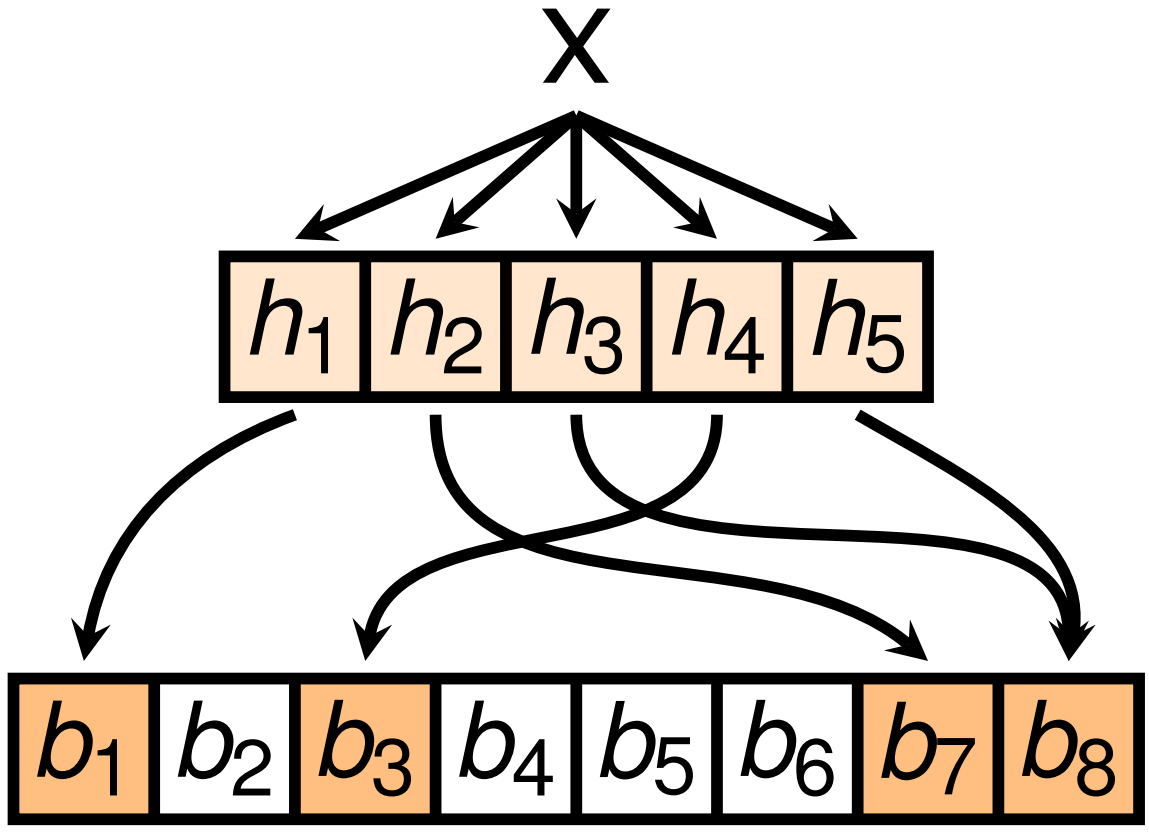
- **Insert**
- **Query**





What is a Bloomfilter?

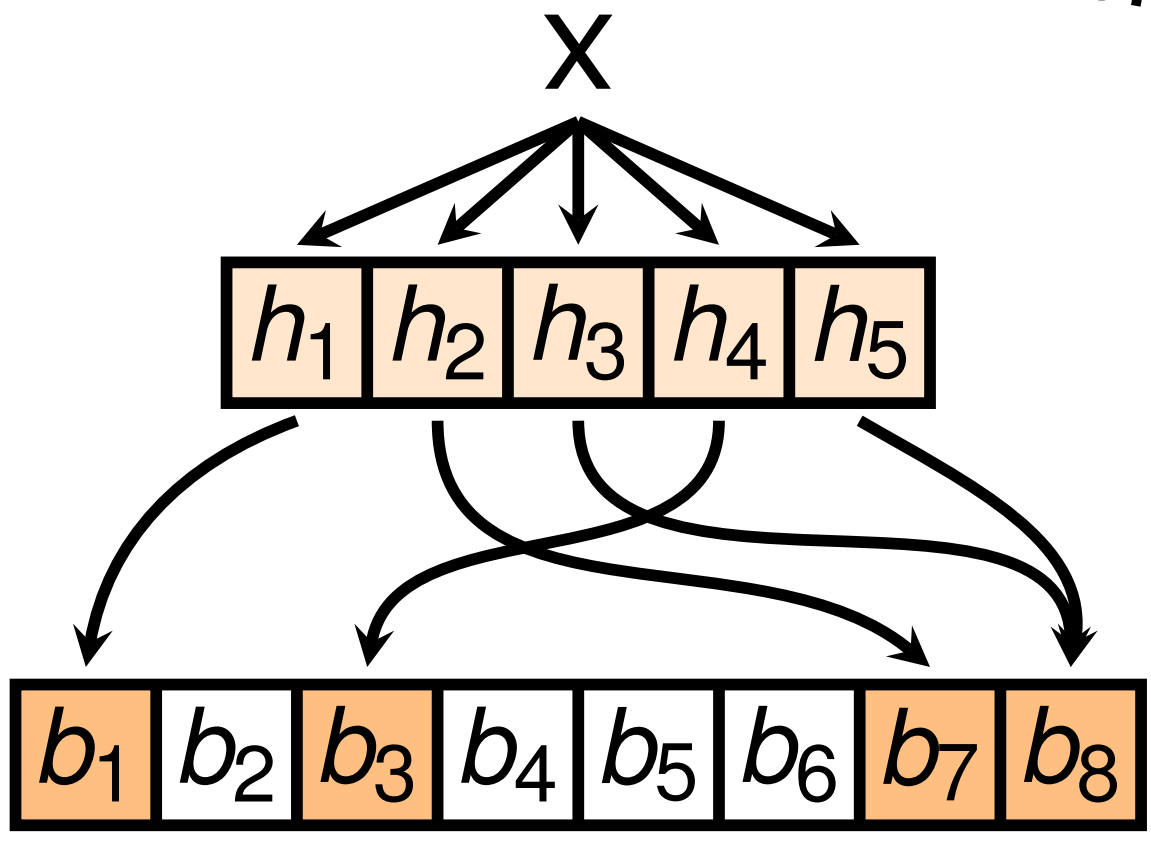
- **Insert**
- **Query**



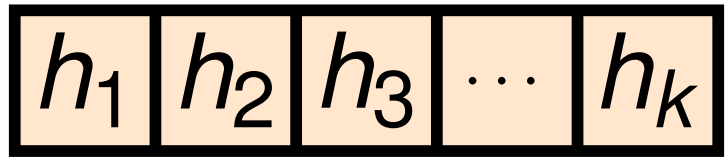
What is a Bloomfilter?

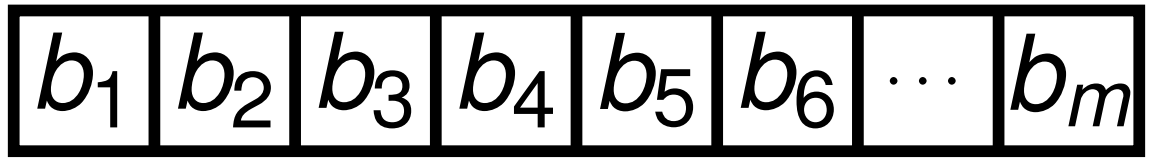
No False Negatives!

- **Insert**
- **Query**

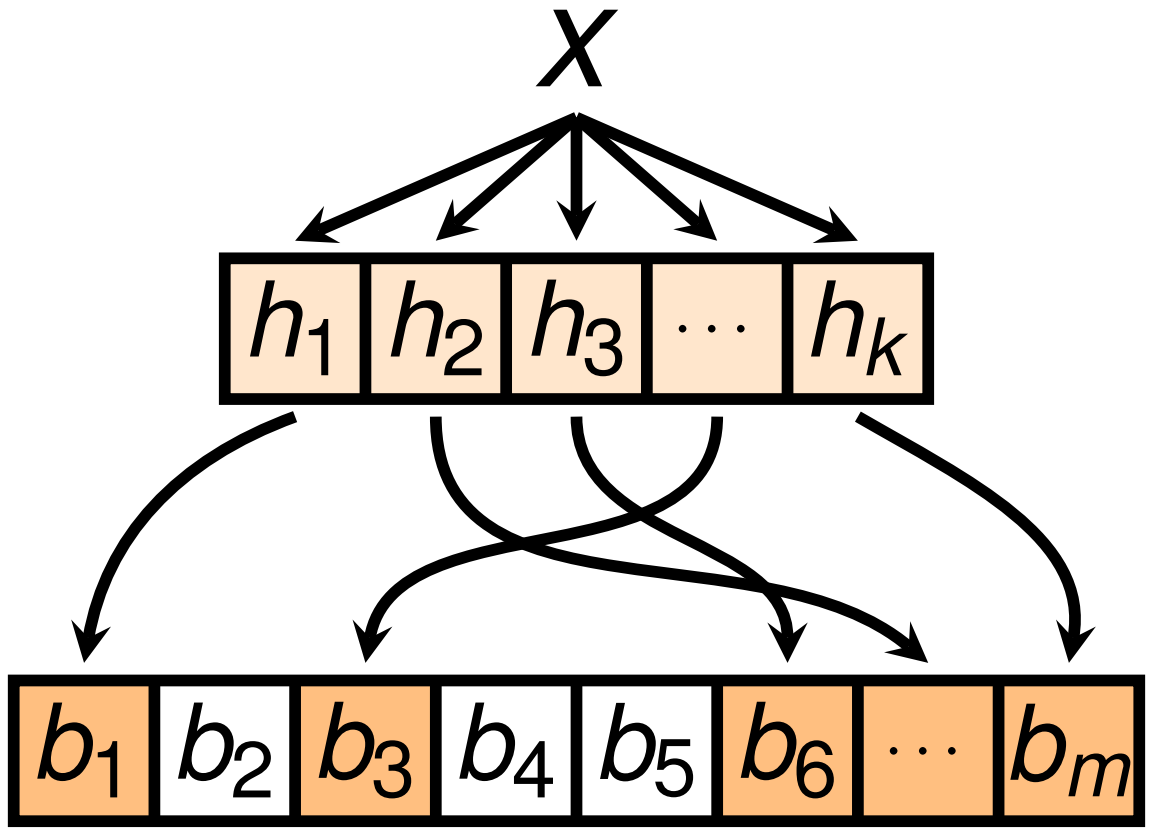


x



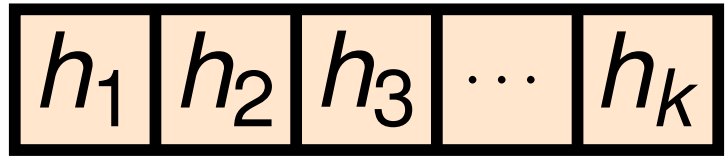


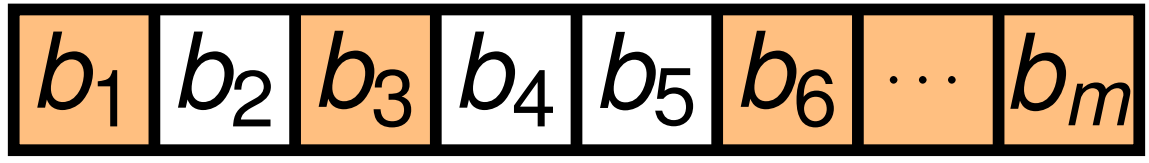
False positives



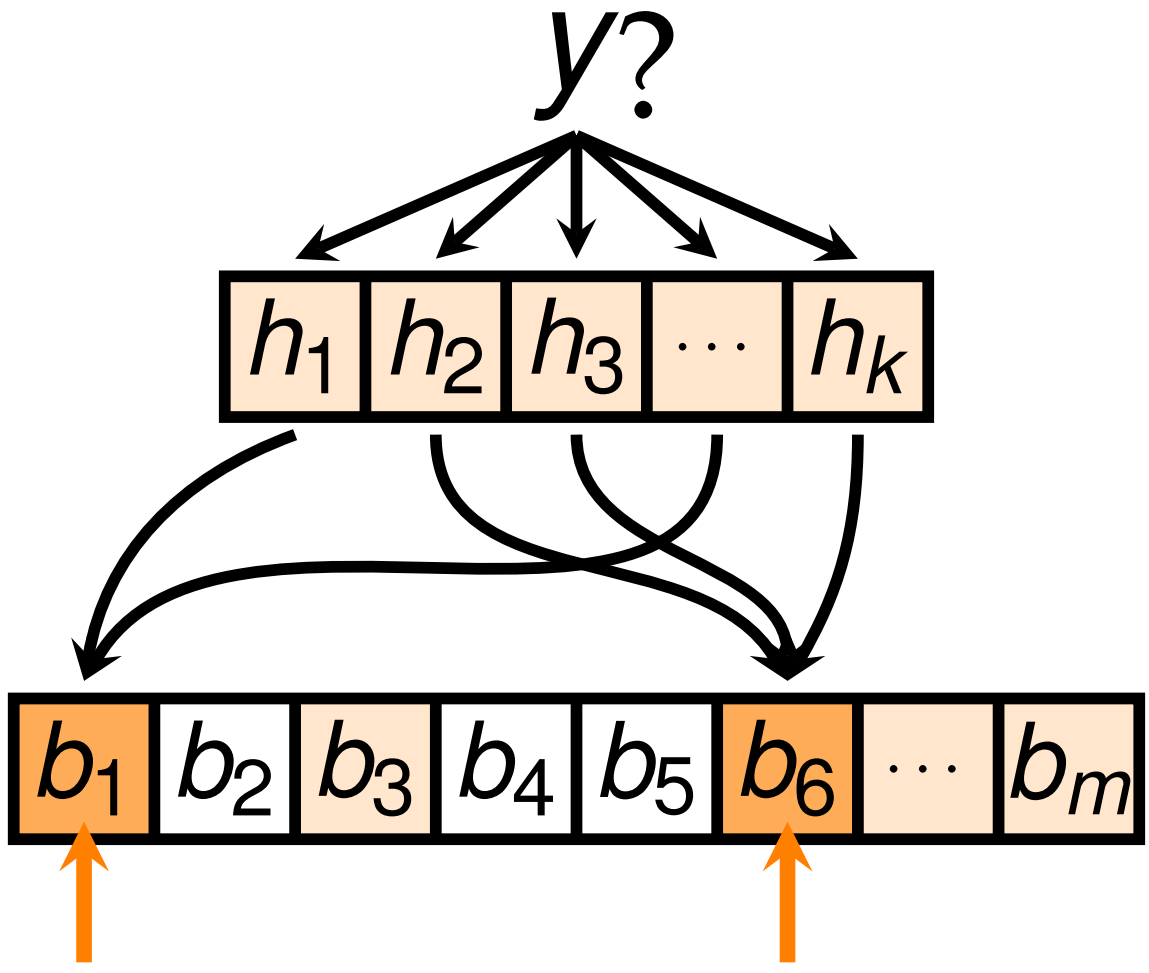
False positives

y?



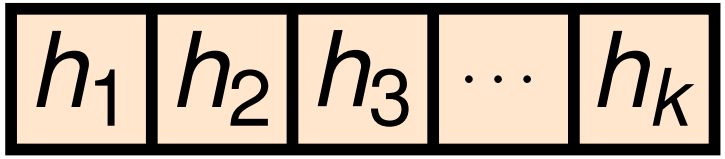


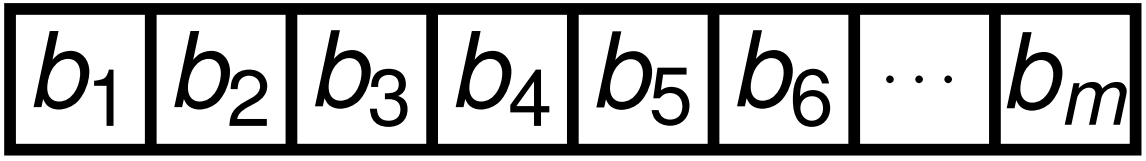
False positives



False positives

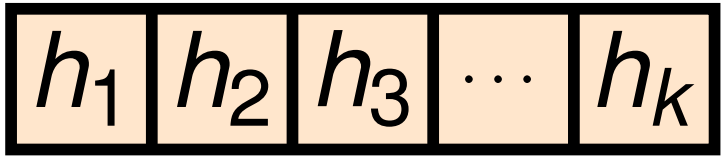
$x_1 \cdots x_n$

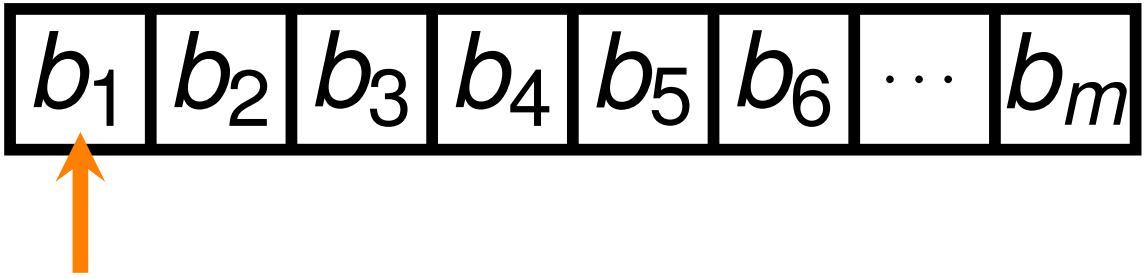




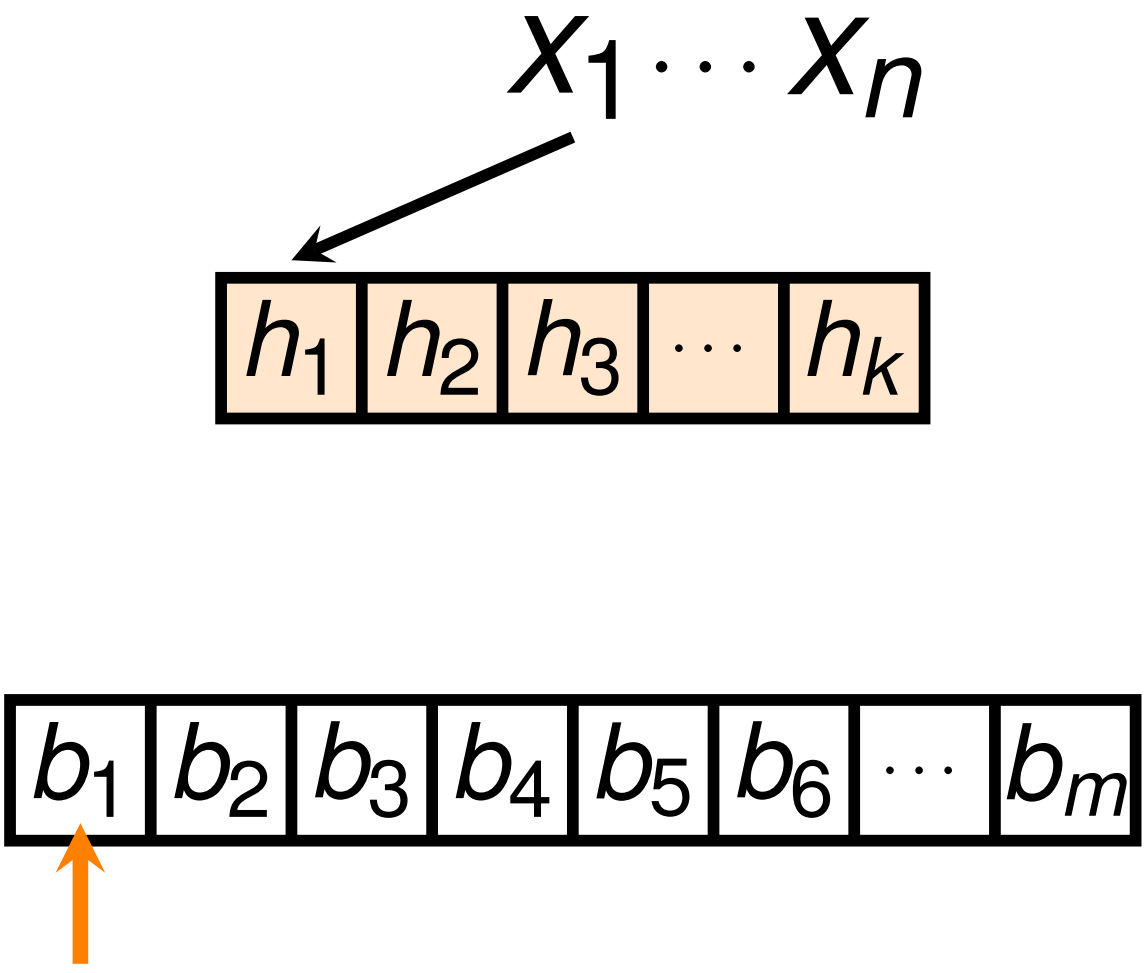
Bloom's Analysis

$x_1 \cdots x_n$

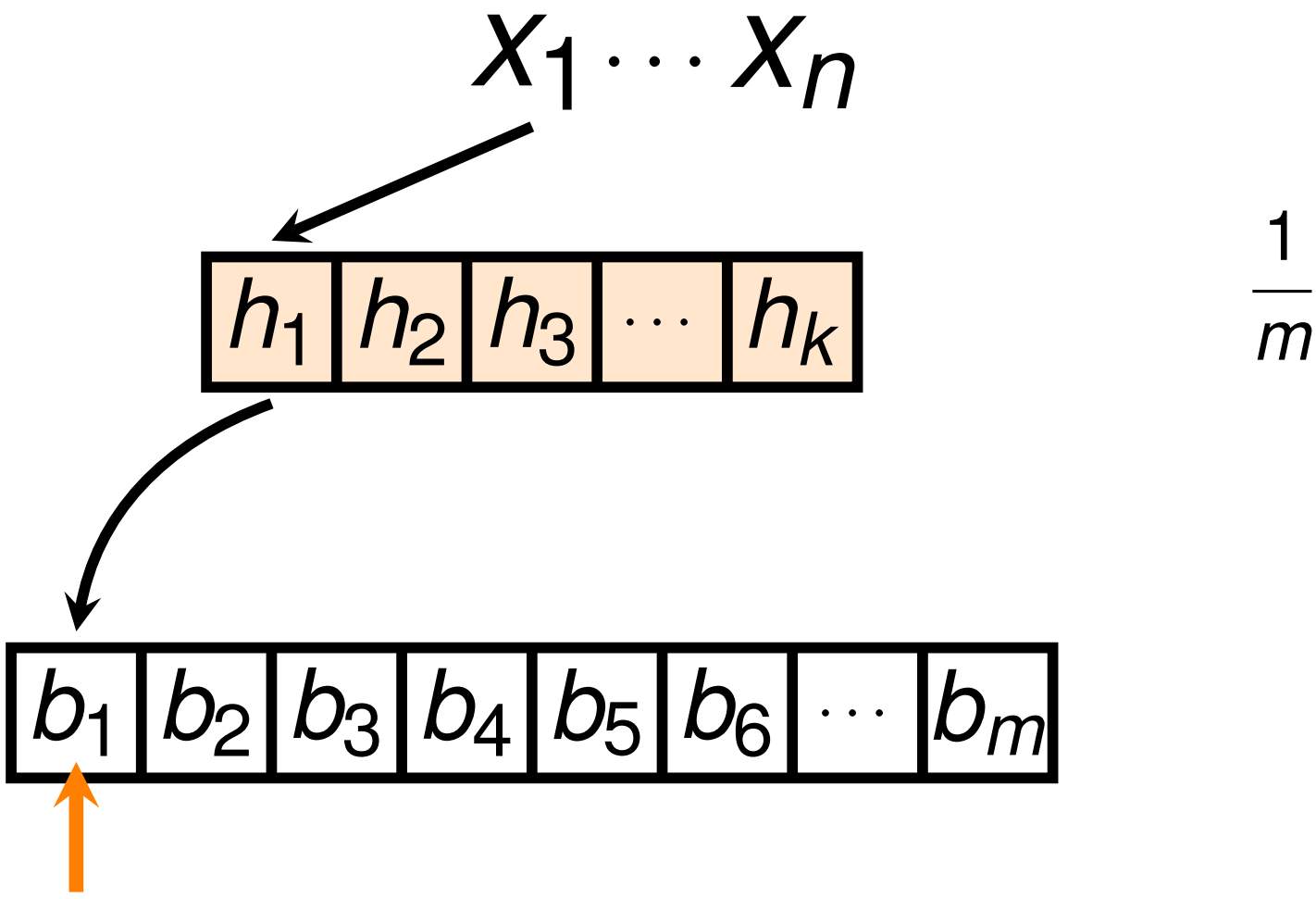




Bloom's Analysis

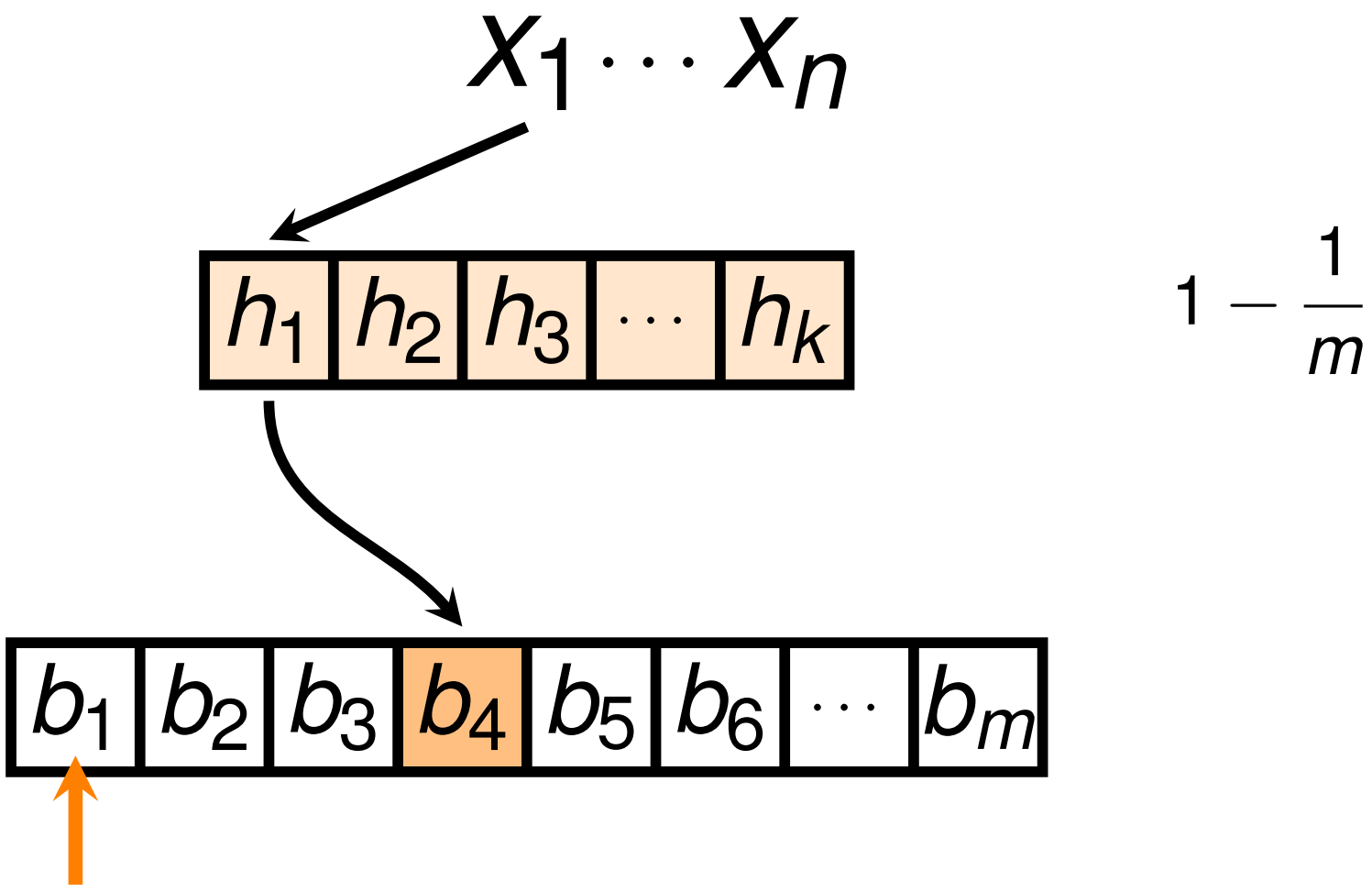


Bloom's Analysis



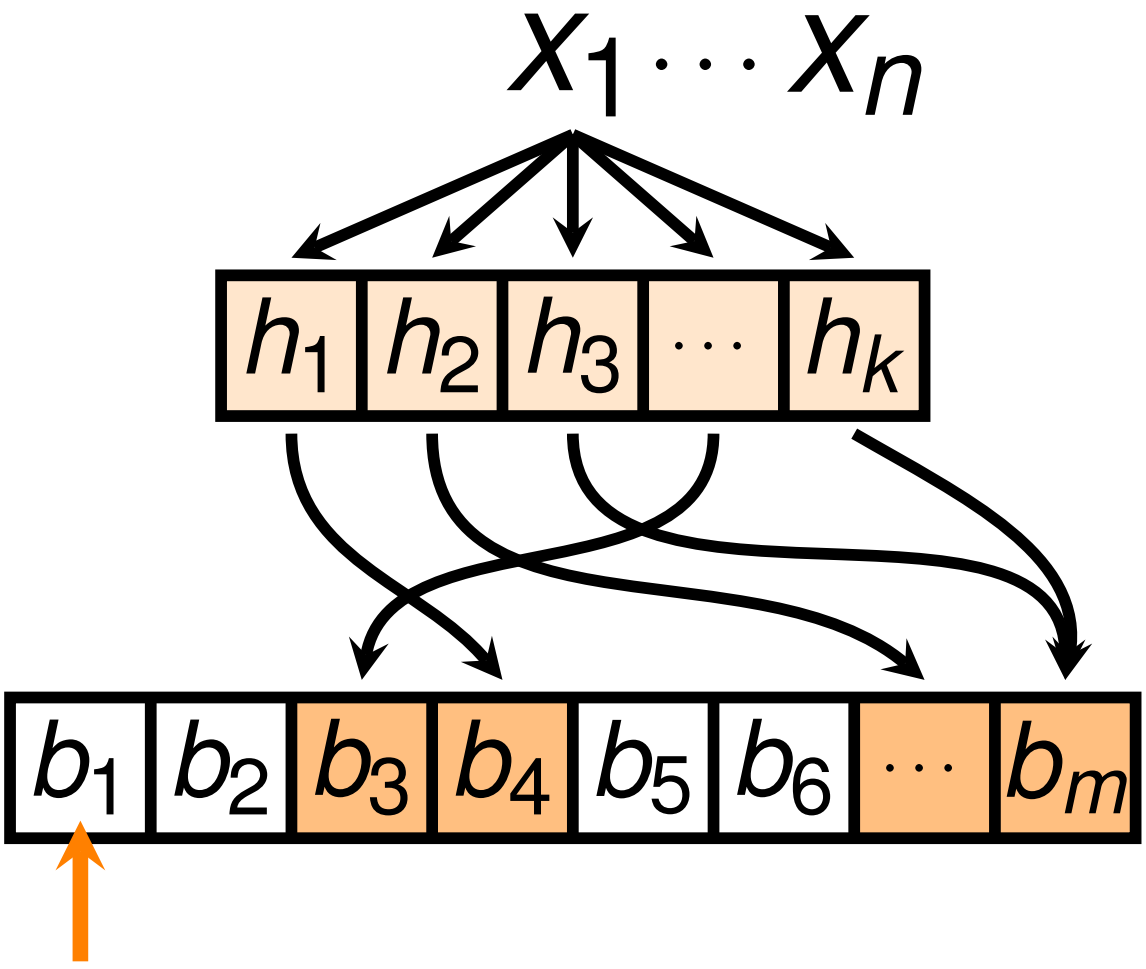
$$\frac{1}{m}$$

Bloom's Analysis



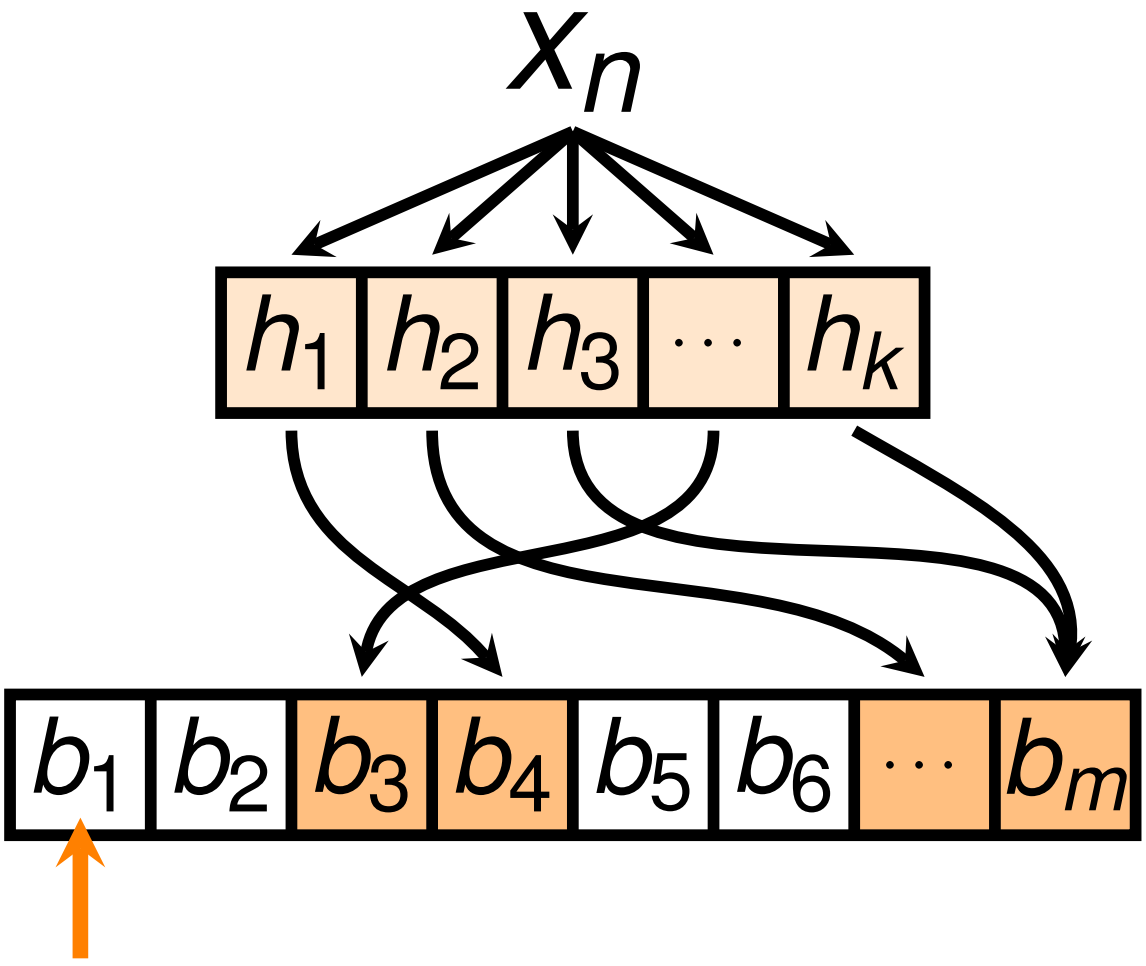
$$1 - \frac{1}{m}$$

Bloom's Analysis



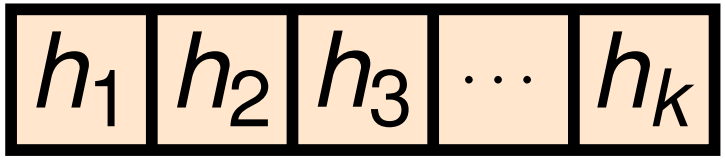
$$\left(1 - \frac{1}{m}\right)^k$$

Bloom's Analysis

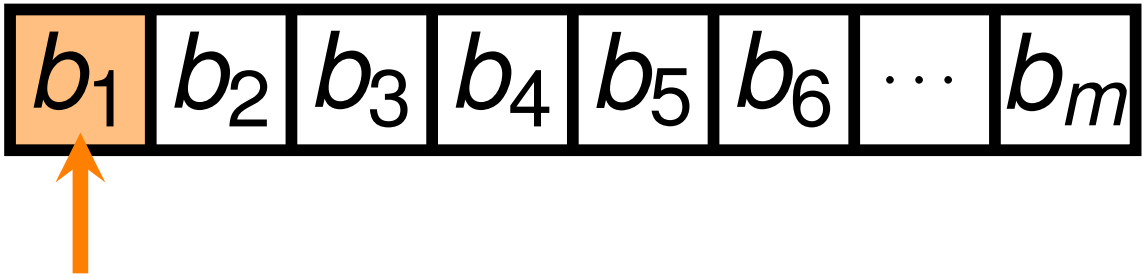


$$\left(1 - \frac{1}{m}\right)^{kn}$$

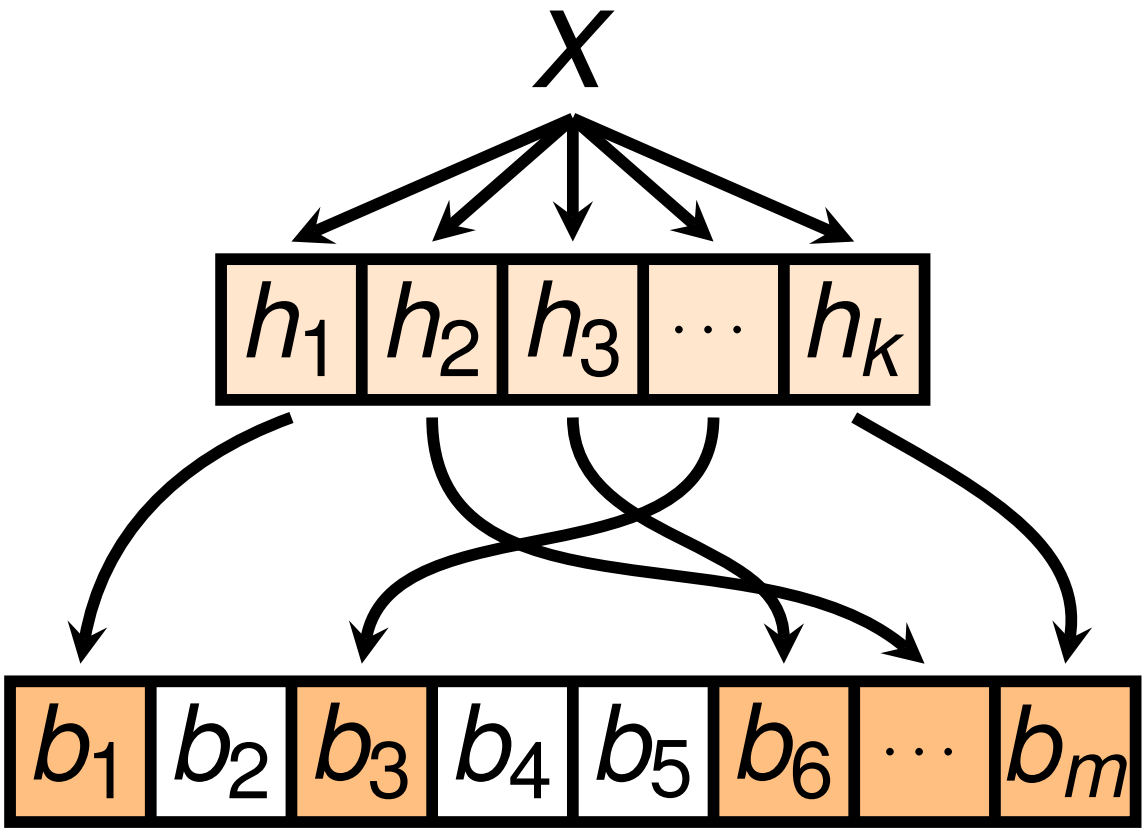
Bloom's Analysis



$$1 - \left(1 - \frac{1}{m}\right)^{kn}$$

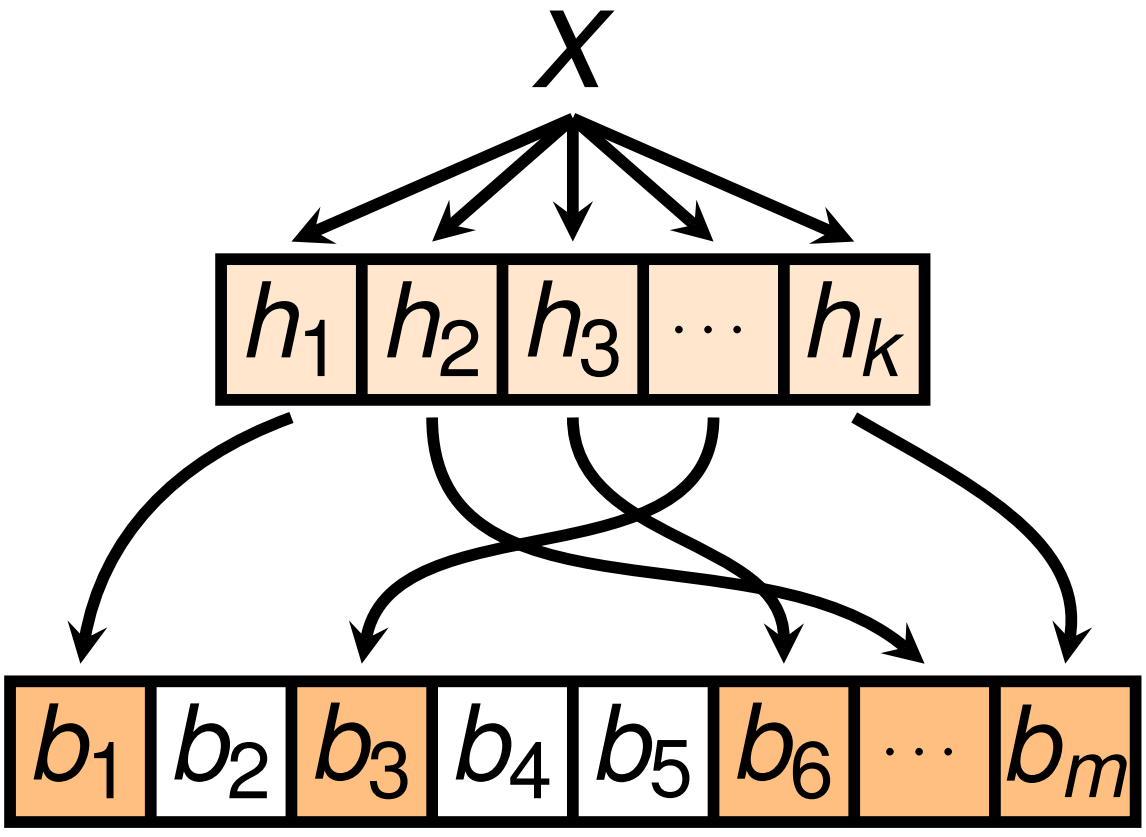


Bloom's Analysis



$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

Bloom's Analysis

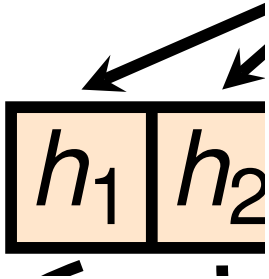


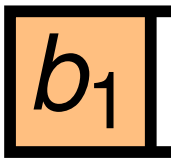
$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

Bloom's bound

(1970)

Bloom's Analysis





Space/Time Trade-offs in Hash Coding with Allowable Errors

BURTON H. BLOOM
Computer Usage Company, Newton Upper Falls, Mass.

Let ϕ'' represent the expected proportion of bits in the hash area of N'' bits still set to 0 after n messages have been hash stored, where d is the number of distinct bits set to 1 for each message in the given set.

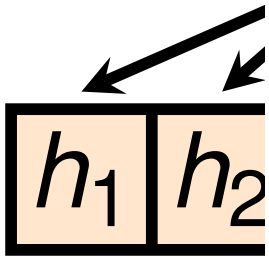
$$\phi'' = (1 - d/N'')^n. \tag{16}$$

A message not in the given set will be falsely accepted if all d bits tested are 1's. The expected fraction of test messages, not in M, which result in such errors is then

$$P'' = (1 - \phi'')^d. \tag{17}$$

$\left(\right)^{kn}$ k

ound



Space/Time Trade-offs in Hash Coding with Allowable Errors

Burton H. Bloom

Computer U..., Mass.

Let ϕ'' ...

in the

e been

set to

...ound

(16)

f all

ges,

Network Applications of Bloom Filters: A Survey

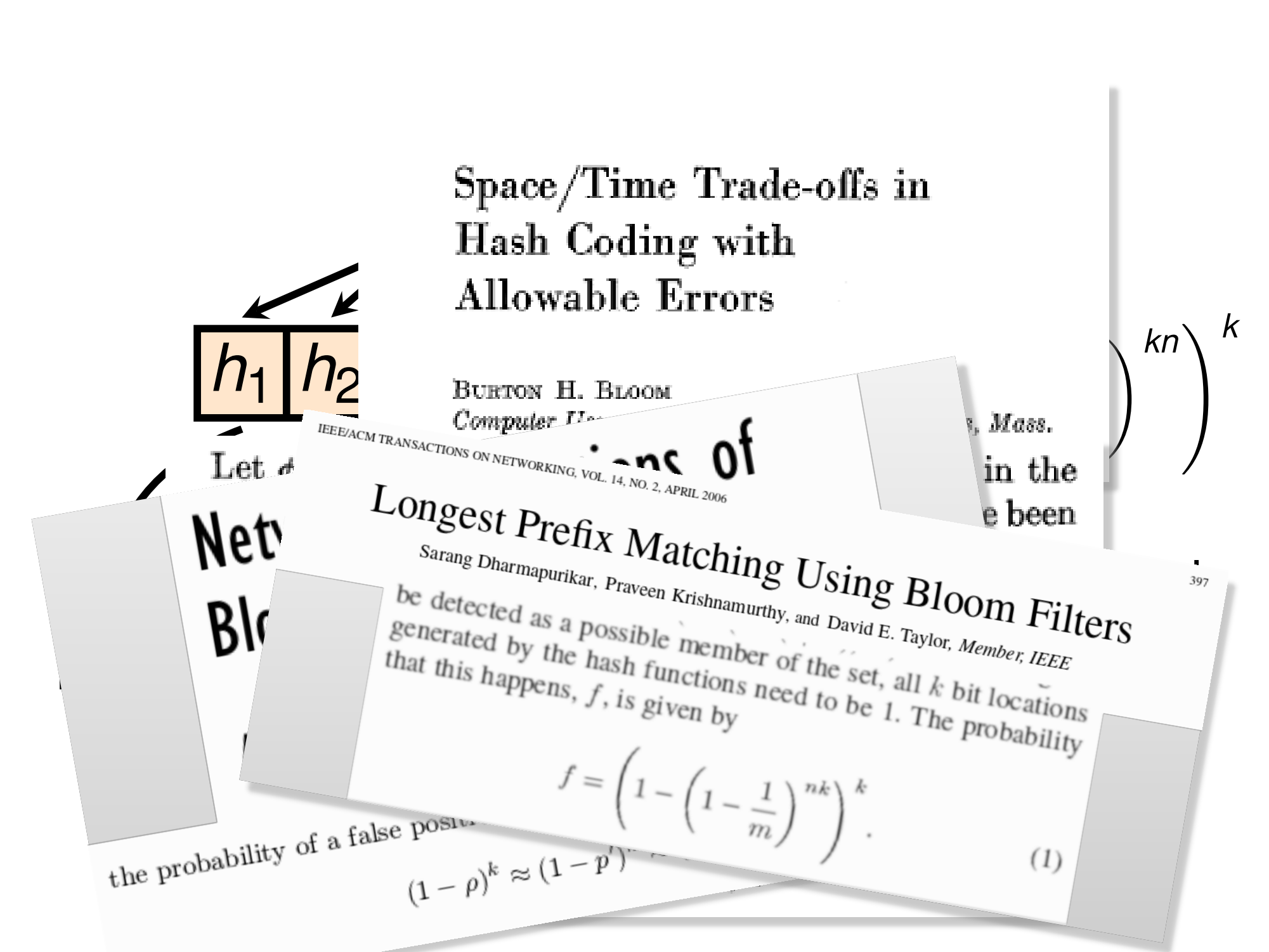
Andrei Broder and Michael Mitzenmacher

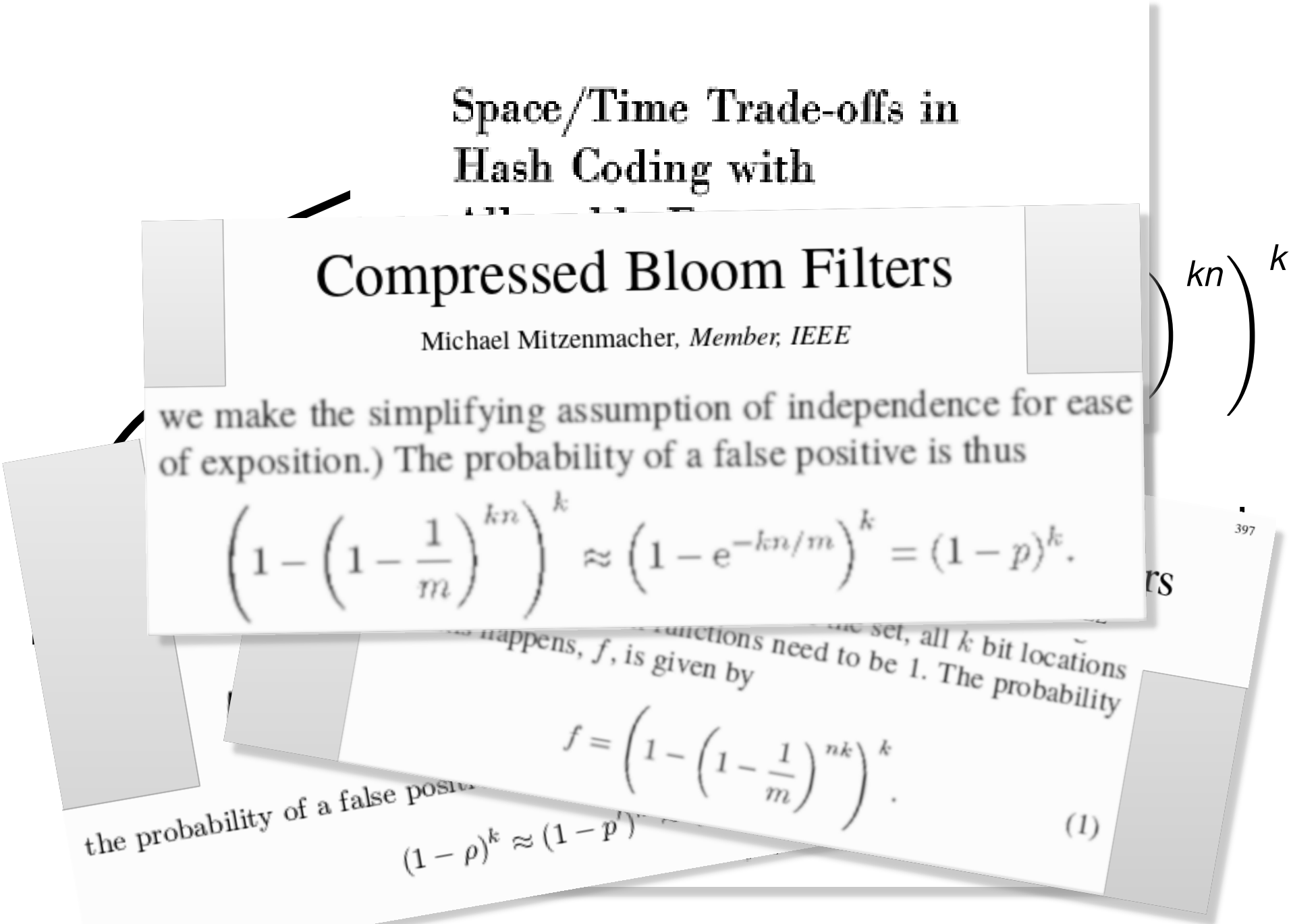
the probability of a false positive is

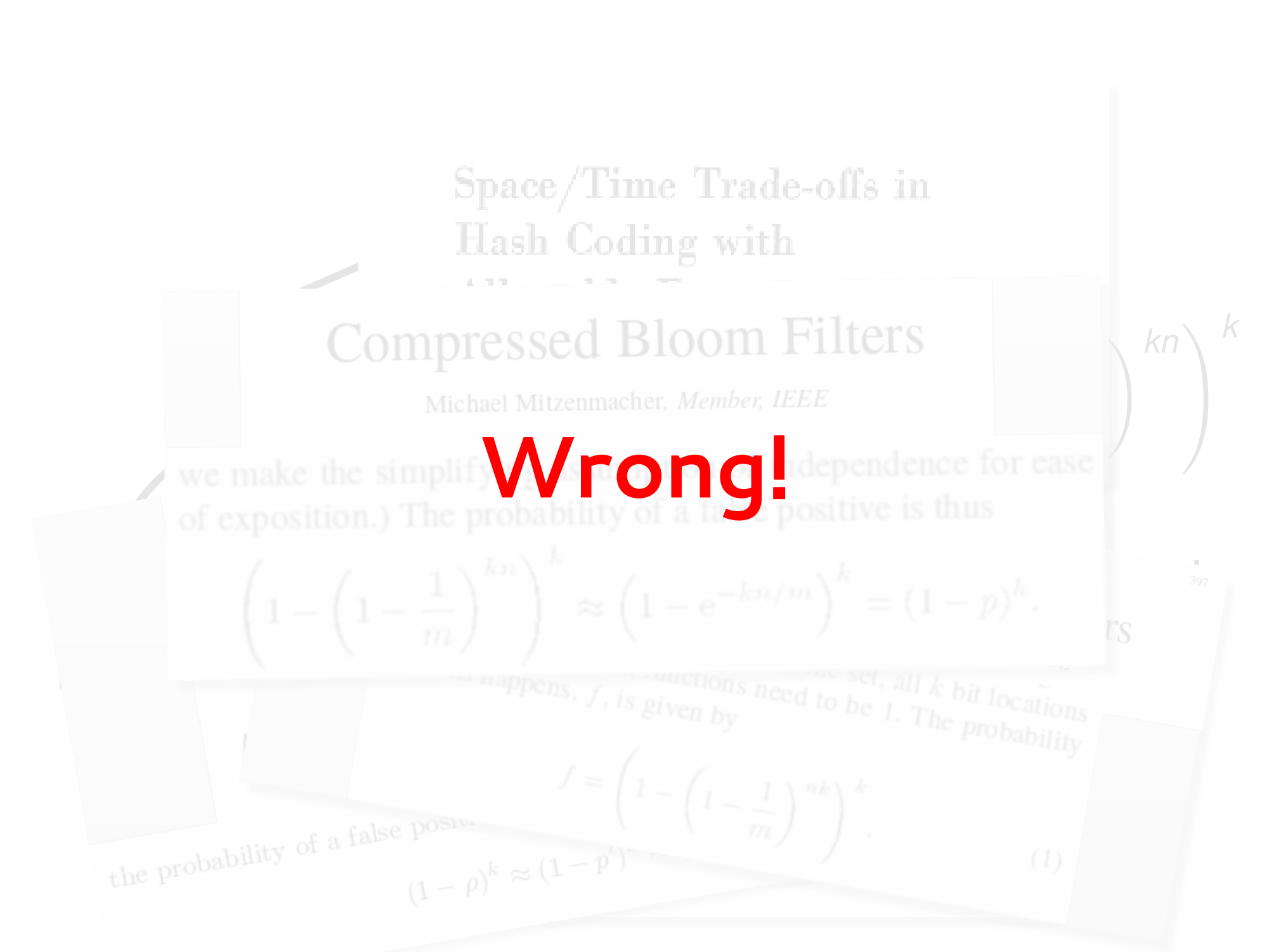
$$(1-\rho)^k \approx (1-p')^k \approx (1-p)^k.$$

(17)

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$$


Space/Time Trade-offs in Hash Coding with Allowable Errors
BURTON H. BLOOM
h1
h2
kn
k
IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 14, NO. 2, APRIL 2006
Longest Prefix Matching Using Bloom Filters
Sarang Dharmapurikar, Praveen Krishnamurthy, and David E. Taylor, Member, IEEE
be detected as a possible member of the set, all k bit locations
generated by the hash functions need to be 1. The probability
that this happens, f, is given by
$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k}.$$
(1)
397
the probability of a false
$$(1-\rho)^k \approx (1-p')$$


Space/Time Trade-offs in
Hash Coding with
Compressed Bloom Filters
Michael Mitzenmacher, *Member, IEEE*
we make the simplifying assumption of independence for ease of exposition.) The probability of a false positive is thus
$$\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k \approx \left(1-e^{-kn/m}\right)^k = (1-p)^k.$$
kn
k
397
rs
set, all k bit locations
functions need to be 1. The probability
happens, f, is given by
$$f = \left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^k. \quad (1)$$
the probability of a false posi
$(1-\rho)^k \approx (1-p')$


Space/Time Trade-offs in
Hash Coding with
Compressed Bloom Filters
Michael Mitzenmacher, Member, IEEE
Wrong!

In 2008:

ON THE FALSE-POSITIVE RATE OF BLOOM FILTERS

Prosenjit Bose Hua Guo Evangelos Kranakis Anil Maheshwari Pat Morin
Jason Morrison Michiel Smid Yihui Tang

School of Computer Science
Carleton University
{jit,hguo2,kranakis,maheshwa,morin,morrison,michiel,y_tang}@scs.carleton.ca

In 2008:

ON THE FALSE-POSITIVE RATE OF BLOOM FILTERS

Prosenjit Bose Hua Guo Evangelos Kranakis Anil Maheshwari Pat Morin
Jason Morrison Michiel Smid Yihui Tang

School of Computer Science
Carleton University
{jit,hguo2,kranakis,maheshwa,morin,morrison,michiel,y_tang}@scs.carleton.ca

***still had errors!**

Encoding in Coq

- Probability Monad
- Hash functions as random oracles

Encoding in Coq

- Probability Monad

$$\text{dist } A \to (A \to \text{dist } B) \to \text{dist } B$$

- Hash functions as random oracles

Encoding in Coq

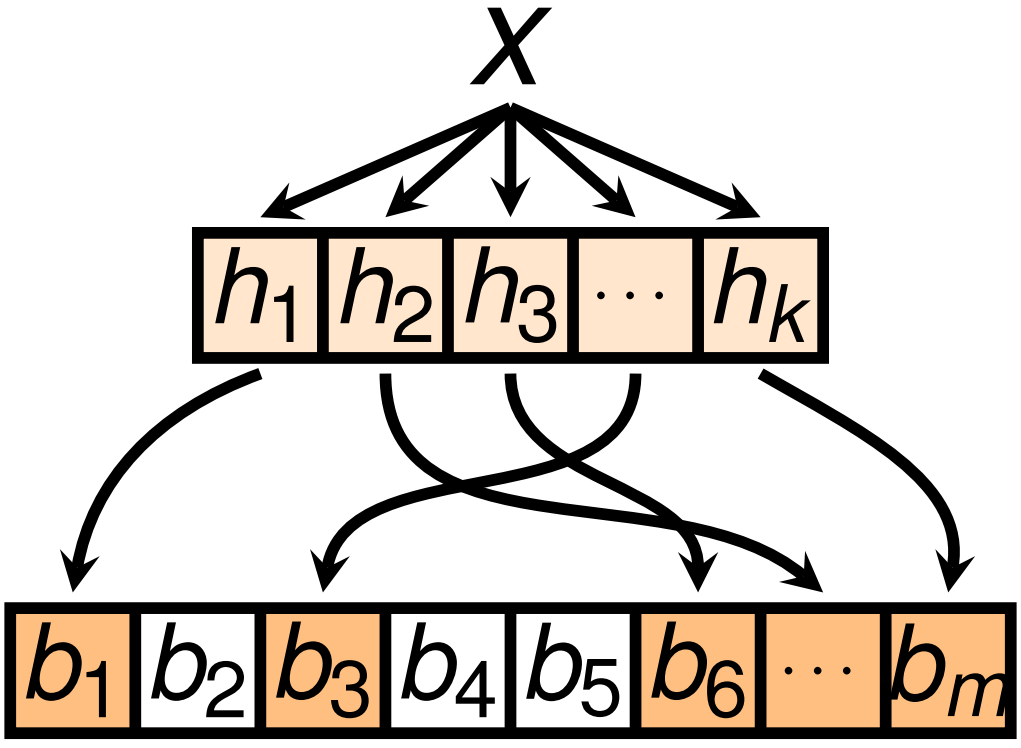
- Probability Monad

$$\text{dist}\ A \to (A \to \text{dist}\ B) \to \text{dist}\ B$$

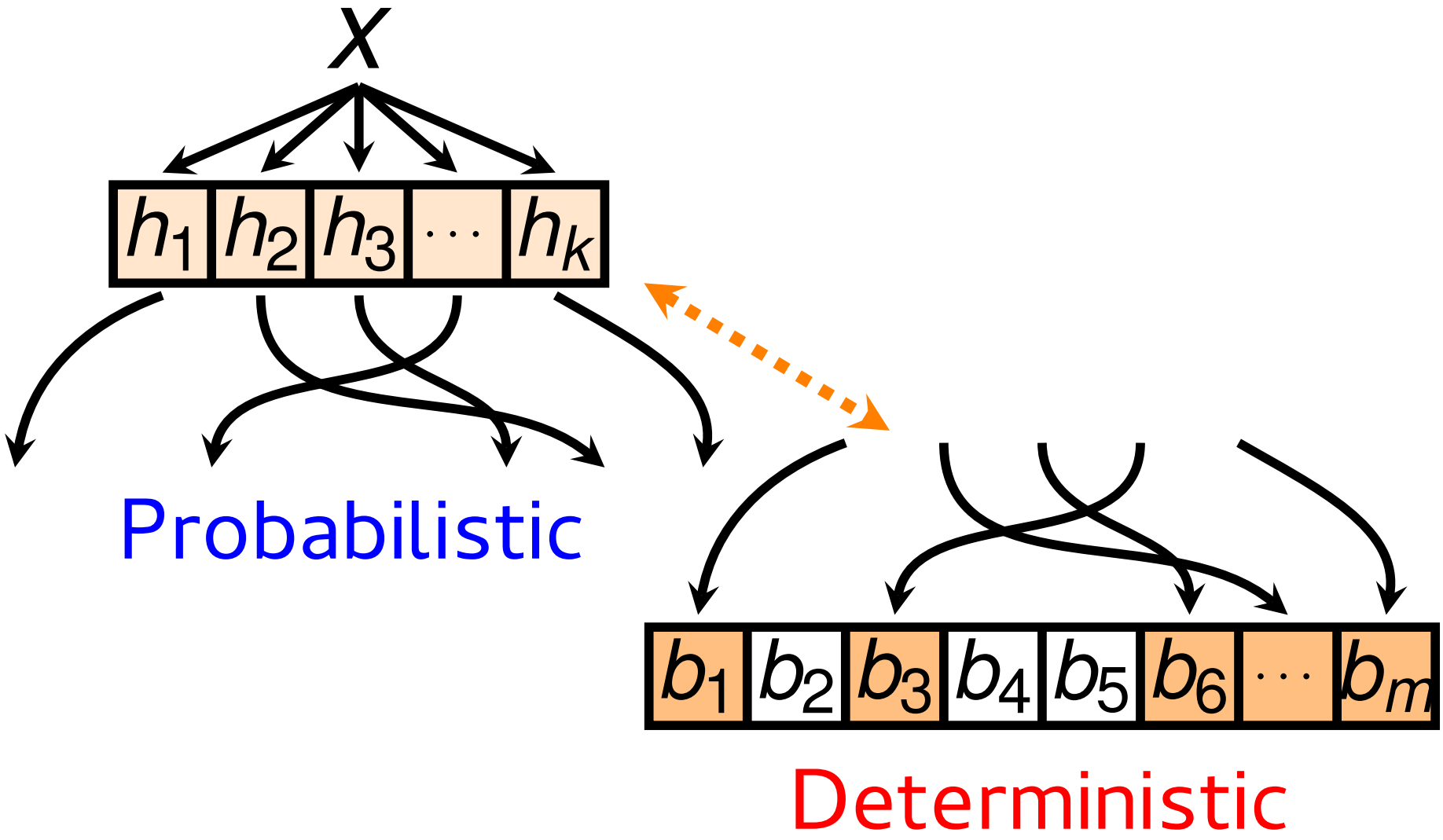
- Hash functions as random oracles

$$\text{hash} : B \to \text{Hash}\ B \to \text{dist}\ (\text{Hash}\ B * \text{seq}\ \text{Int})$$

Encoding in Coq



Encoding in Coq

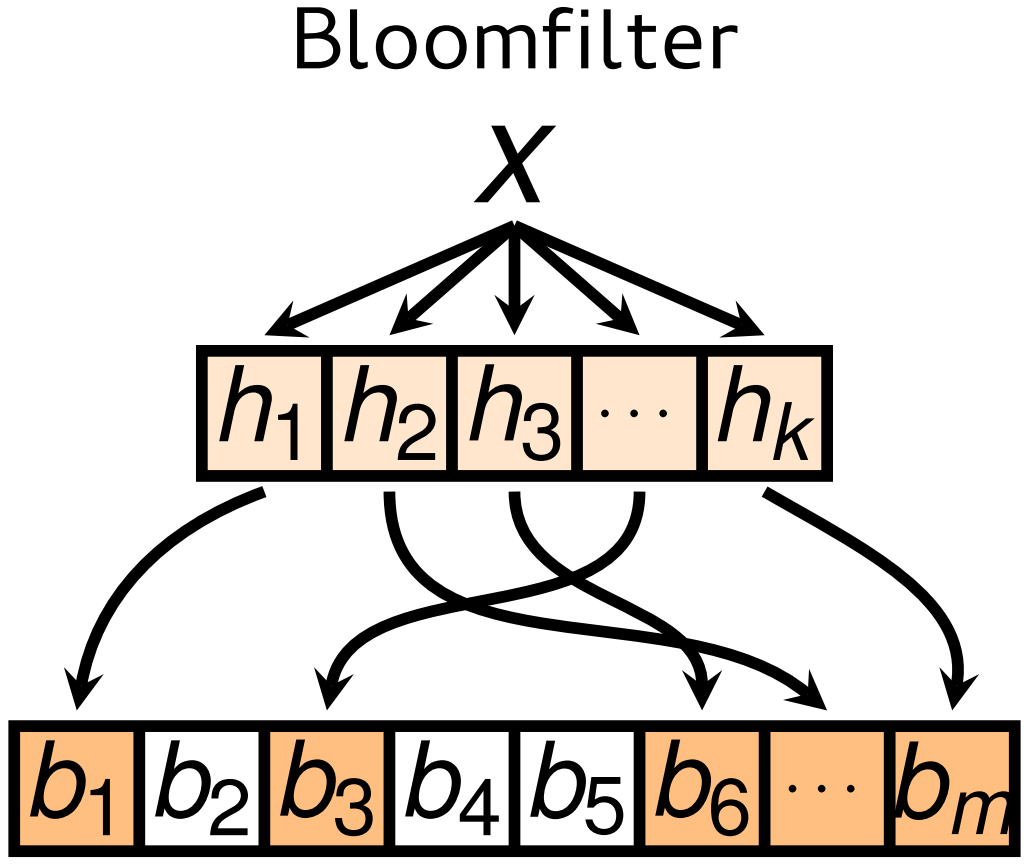


Certified

False positive rate of Bloomfilters:

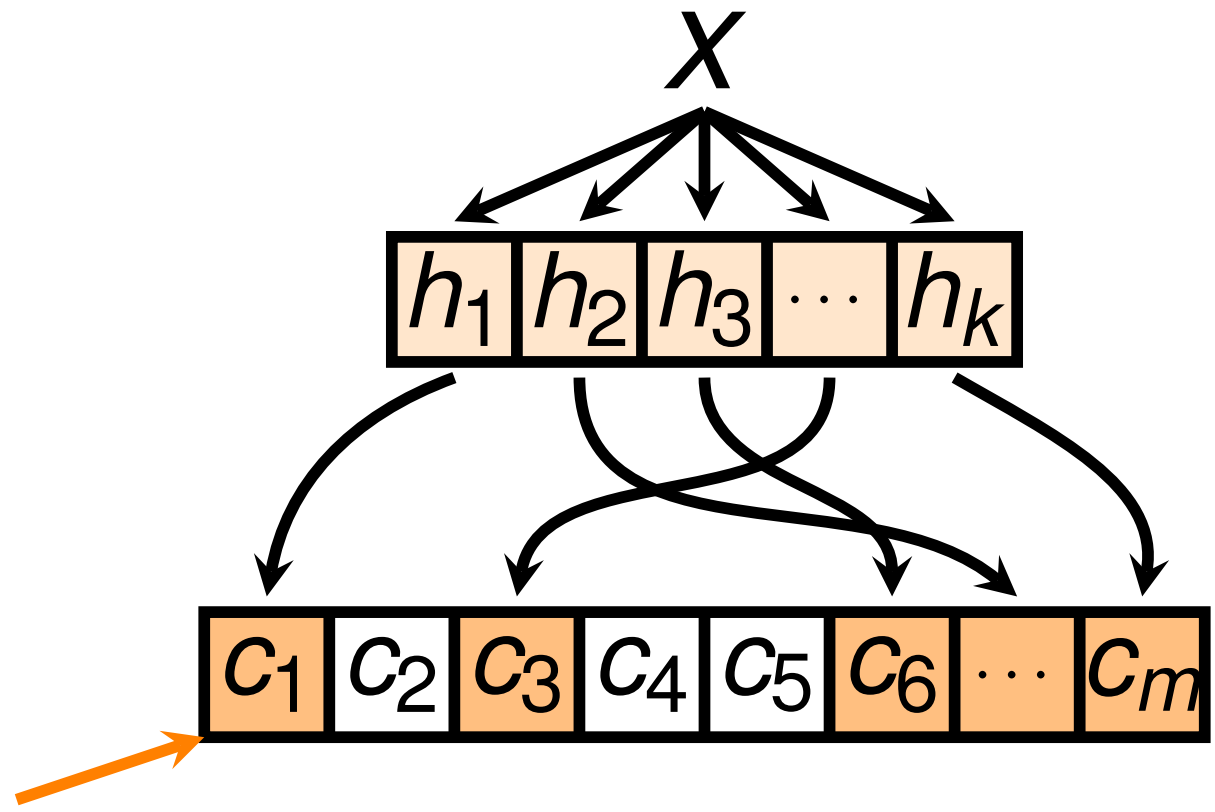
$$\frac{1}{m^{k(l+1)}} \sum_{i=1}^{m} i^k \, i! \left(\begin{array}{c} m \\ i \end{array} \right) \left\{ \begin{array}{c} kl \\ i \end{array} \right\}$$

Can we generalize BFs?



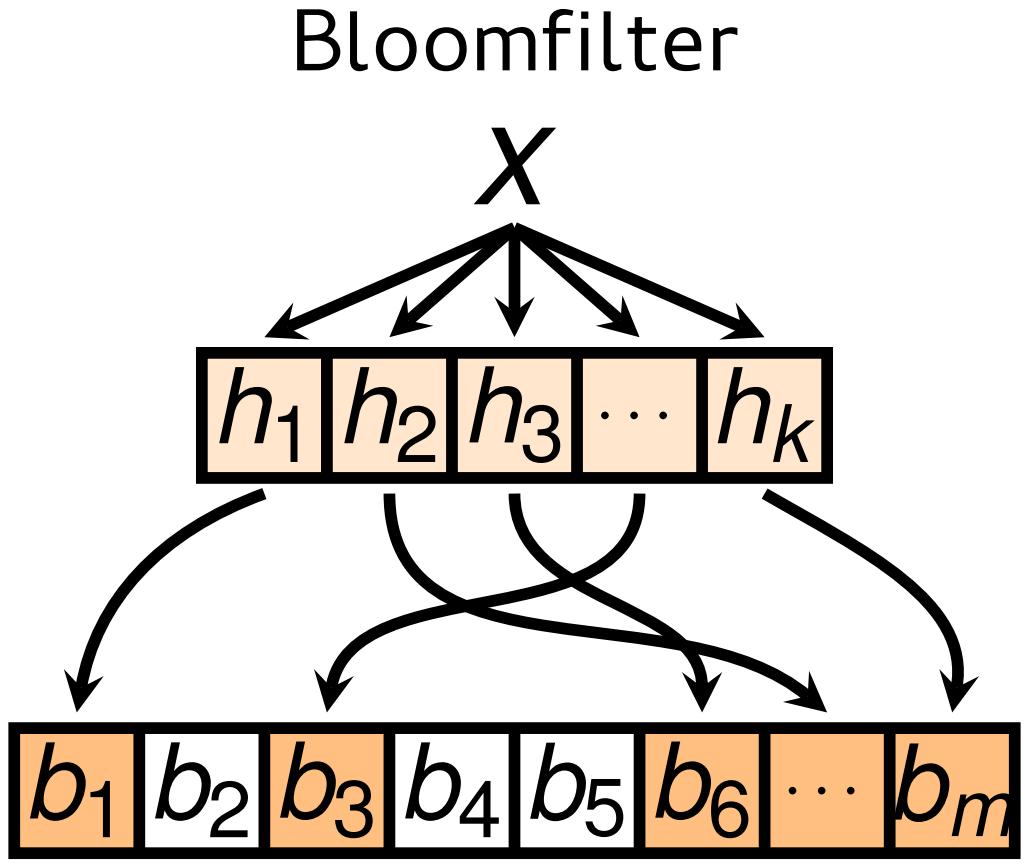
Can we generalize BFs?

Counting Bloomfilter



Bit $\rightarrow$ Counter

Approximate Membership Queries



Approximate Membership Queries

Bloomfilter

Approximate Membership Queries

Counting
Bloomfilters

Bloomfilter

Approximate Membership Queries

Counting
Bloomfilters

Bloomfilter

Blocked
Bloomfilters

Approximate Membership Queries

Quotient Filters

Counting Bloomfilters

Bloomfilter

Blocked Bloomfilters

Approximate Membership Queries

Quotient Filters

Counting Bloomfilters

Bloomfilter

Blocked Quotient Filter

Blocked Bloomfilters

Approximate Membership Queries

Quotient Filters

Counting Bloomfilters

Bloomfilter

Verification?

Blocked Quotient Filter

Blocked Bloomfilters

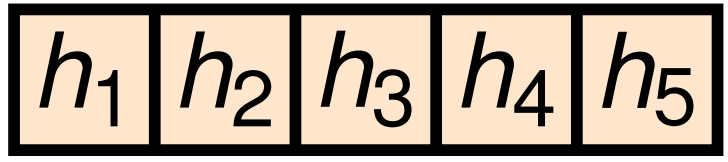
Verifying AMQs

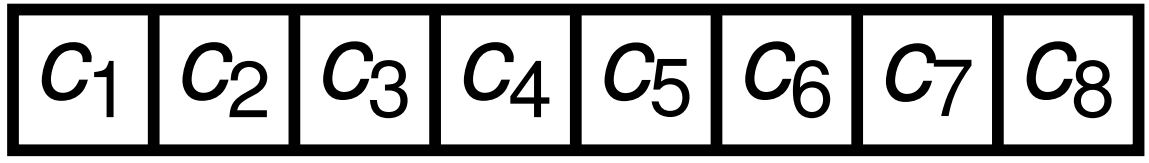
- Decomposition can be generalized
- Massive proof reuse
- Properties for free

Verifying AMQs : Counting Bloom Filter

- Insert
- Query

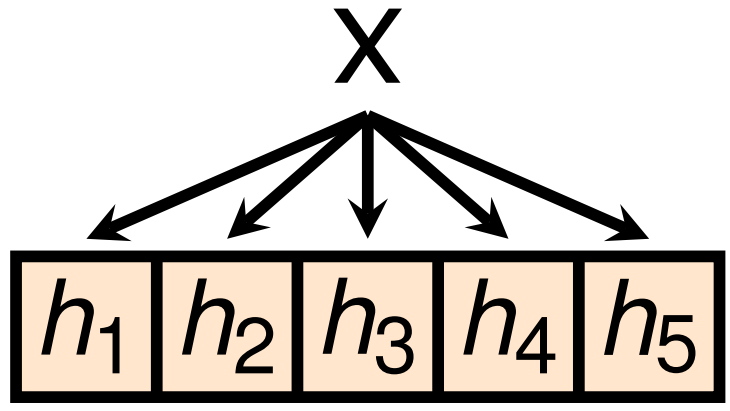
X

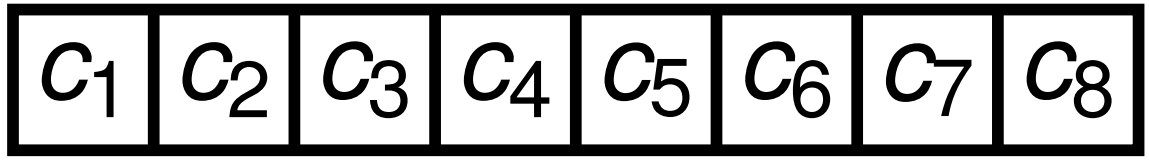




Verifying AMQs : Counting Bloom Filter

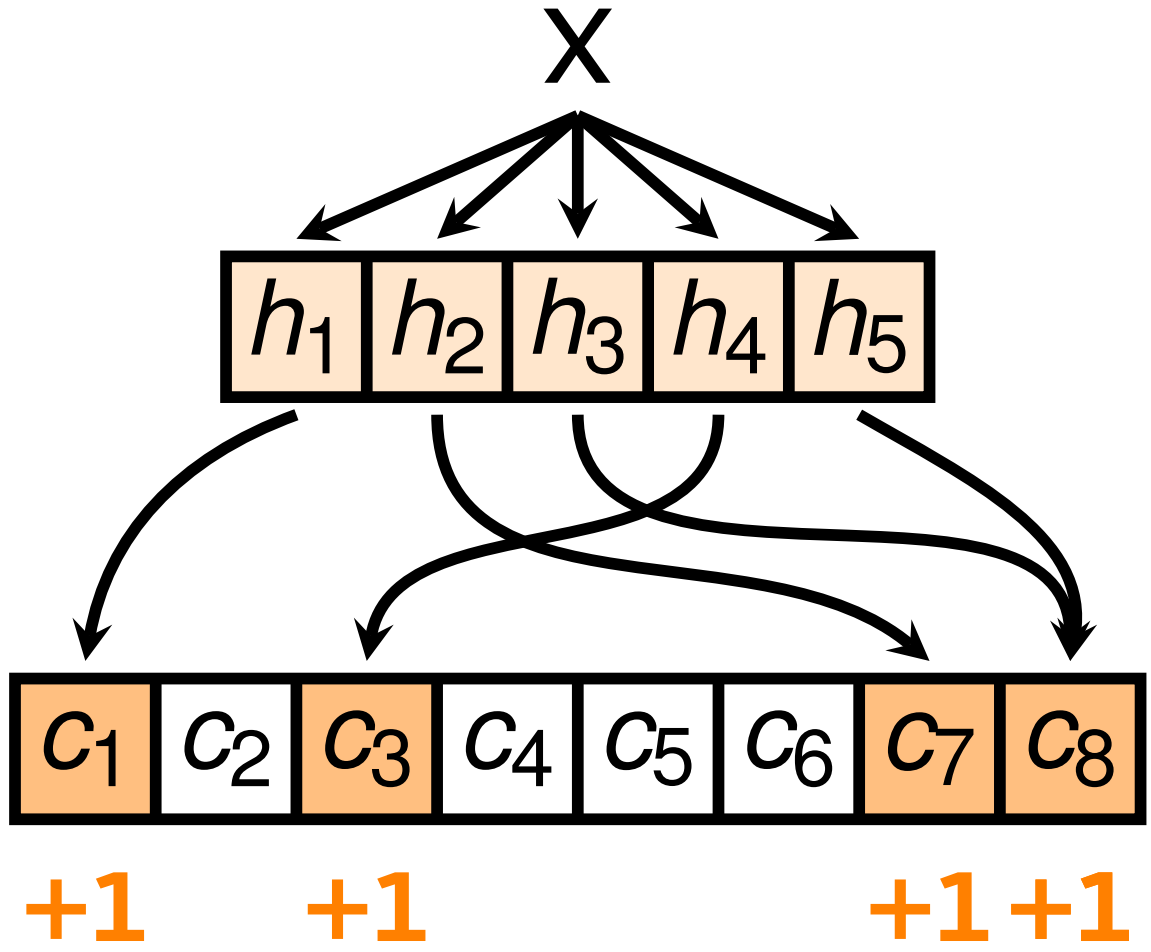
- Insert
- Query





Verifying AMQs : Counting Bloom Filter

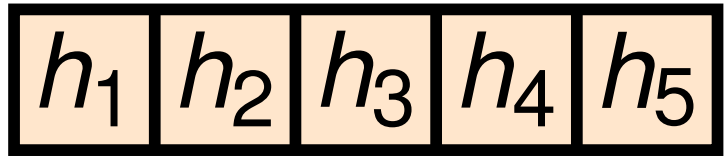
- Insert
- Query

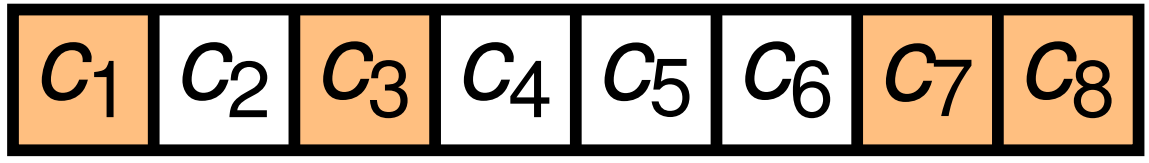


Verifying AMQs : Counting Bloom Filter

- Insert
- Query

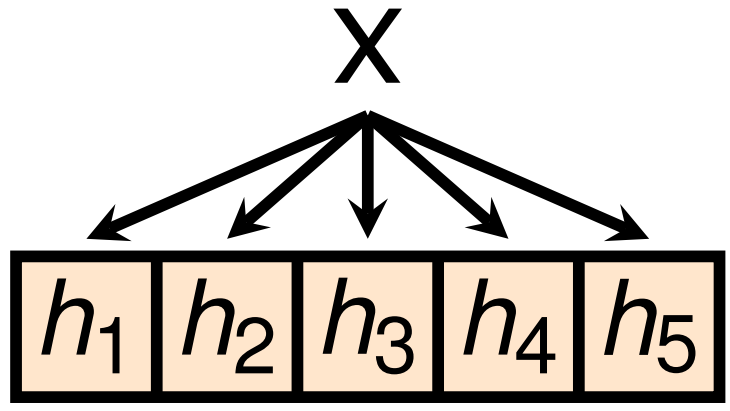
X

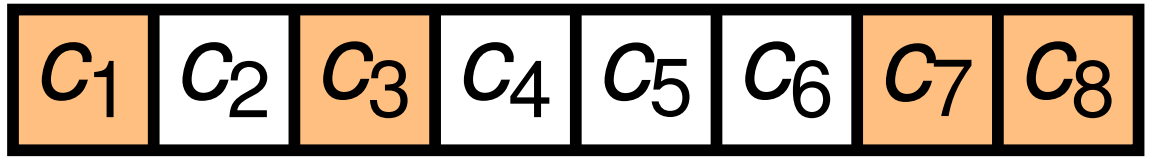




Verifying AMQs : Counting Bloom Filter

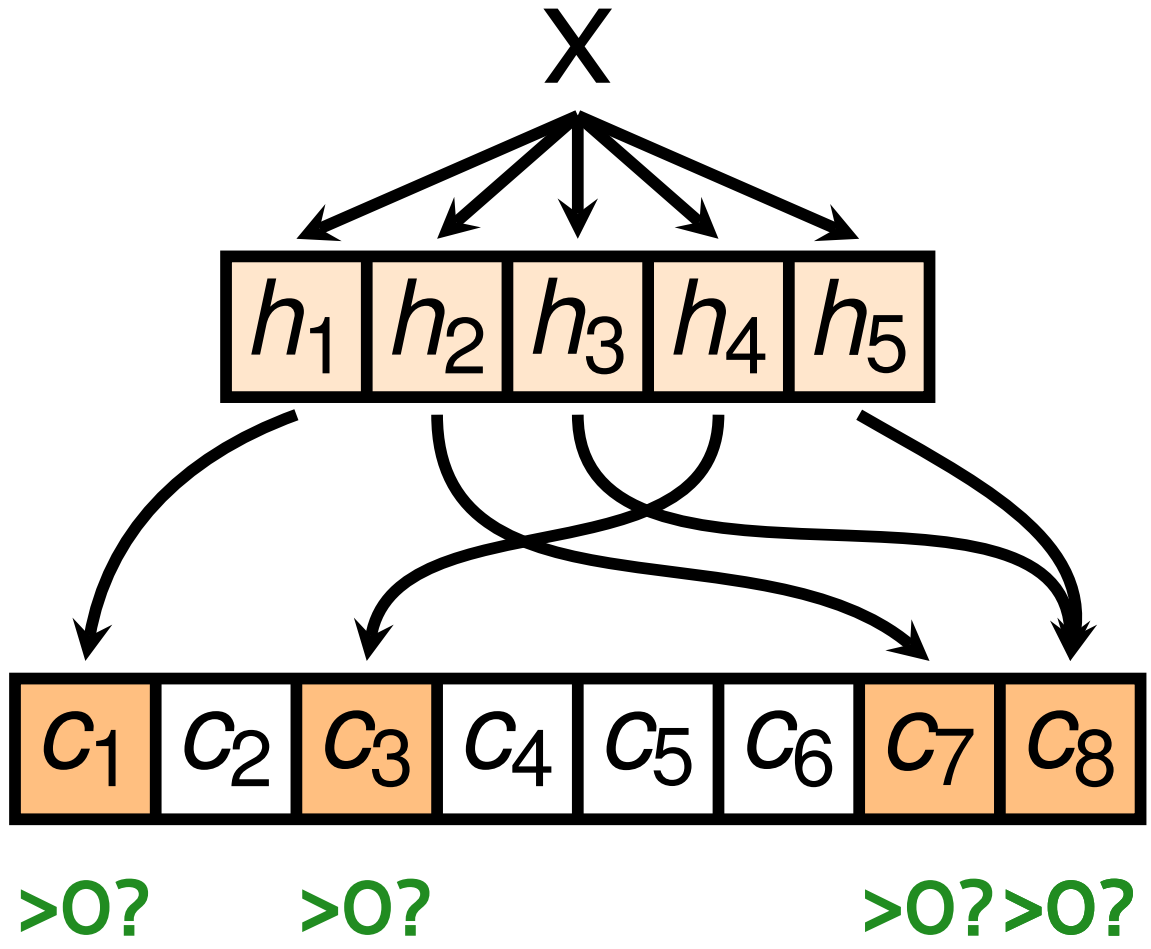
- Insert
- Query



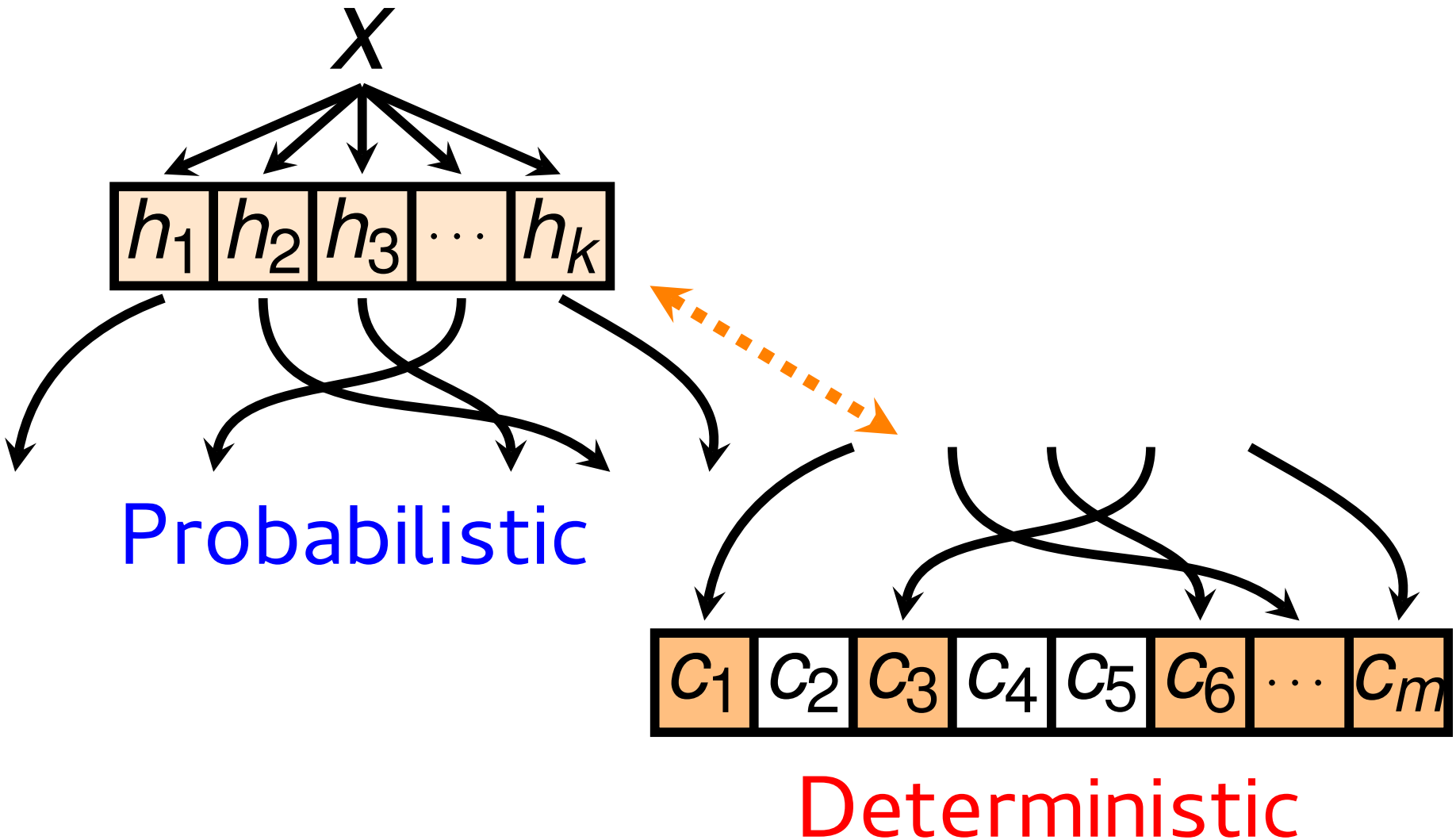


Verifying AMQs : Counting Bloom Filter

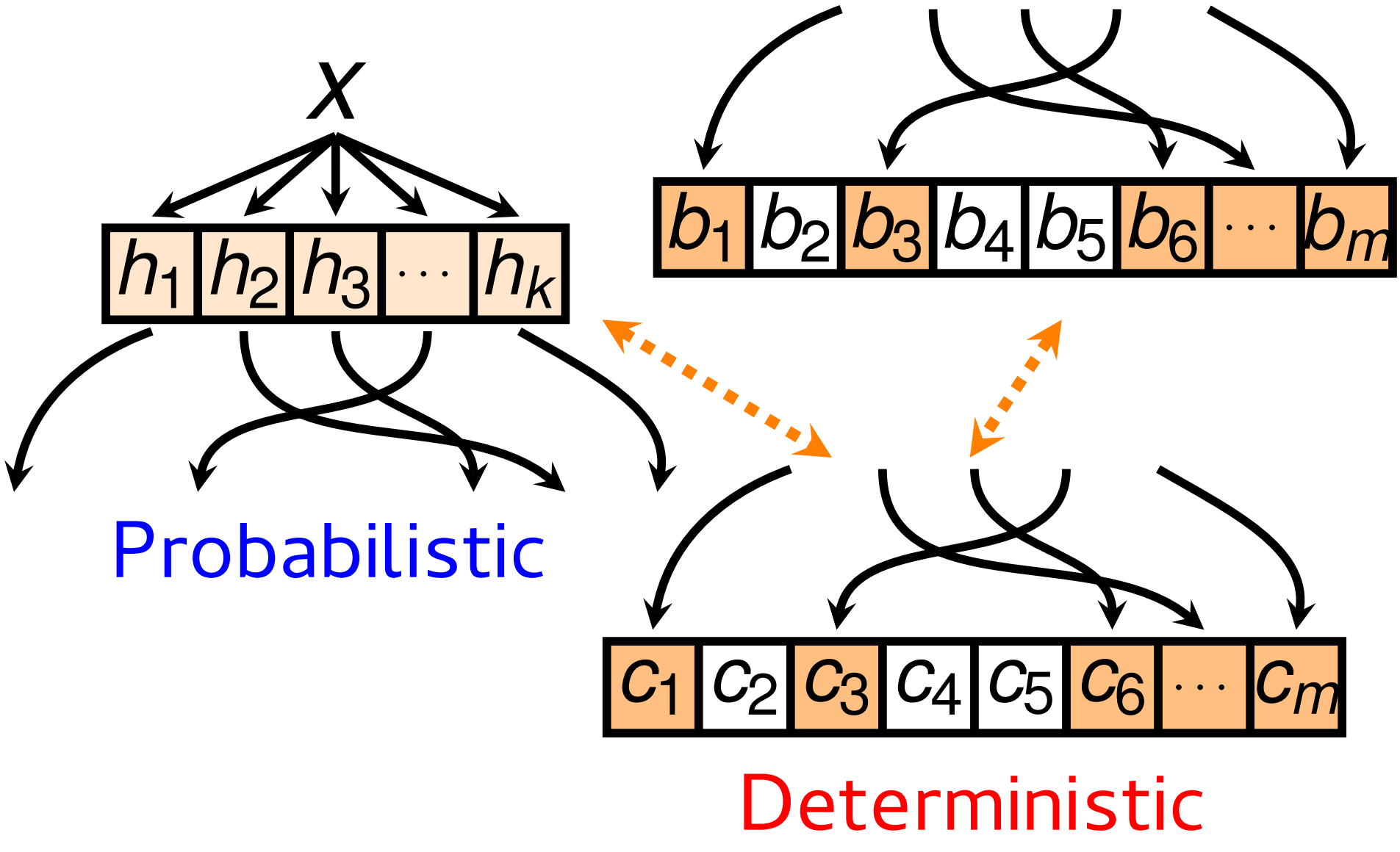
- Insert
- Query



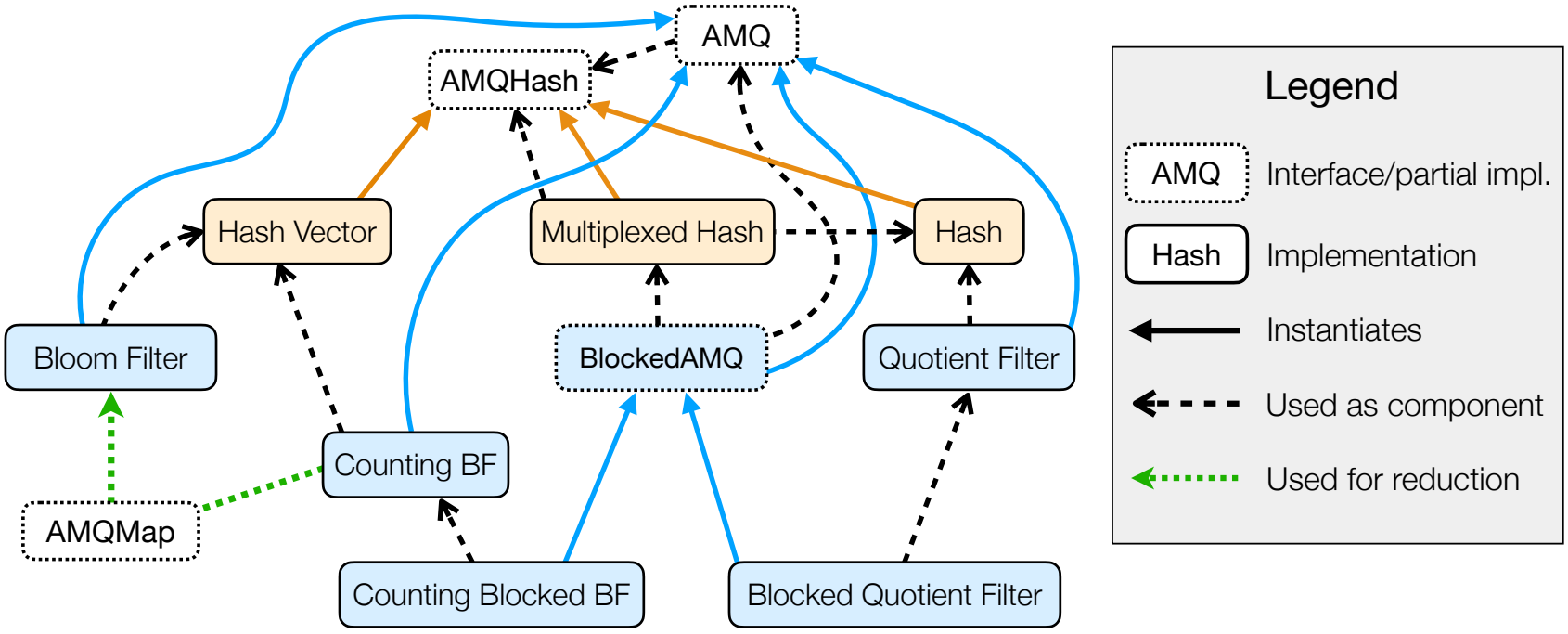
Verifying AMQs : Counting Bloom Filter



Verifying AMQs : Counting Bloom Filter



Verifying AMQs



Future work

- Support other structures (Count Min Sketch, etc.)
- Automatic Bloomfilter synthesis

The End