Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs

Elad Barkan

Technion – Israel Institute of Technology

Joint work with Eli Biham and Adi Shamir

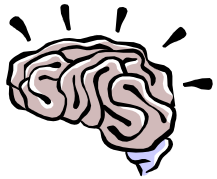
The Basic Problem of Cryptanalysis

Invert a one-way function:
$f:\{0,1,\ldots,N-1\} \rightarrow \{0,1,\ldots,N-1\}$, e.g., $f(x)=E_x(0)$

Generic inversion schemes: f - a "black-box" function,
i.e., oracle accesses to f. Given y, find x such that $y=f(x)$.

Example #1: Exhaustive search
$\forall\ x$, test if $y=f(x)$.
Worst-case time complexity $T = N$.
Memory complexity $M \approx 1$.

Example #2: Exhaustive table
Complexities $M \approx N$. $T \approx 1$.

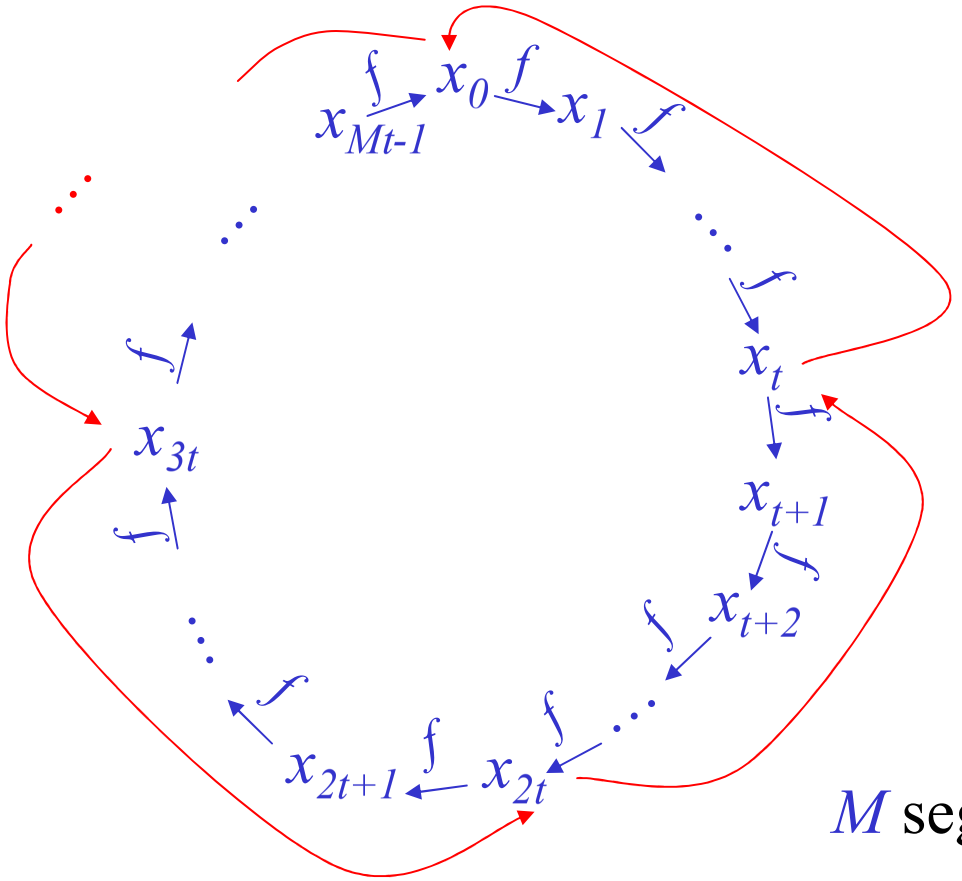


$f(x)$	x
0	172
1	539
2	304
:	:

Time/Memory tradeoffs: find a compromise between the two extremes, i.e., $M << N$ and $T << N$.

Hellman's Time/Memory Tradeoff [1980]

Basic idea: Assume f is a single-cycle permutation:



M segments

Inversion: from $y=f(x)$ complete the segment, then re-calculate segment from start.

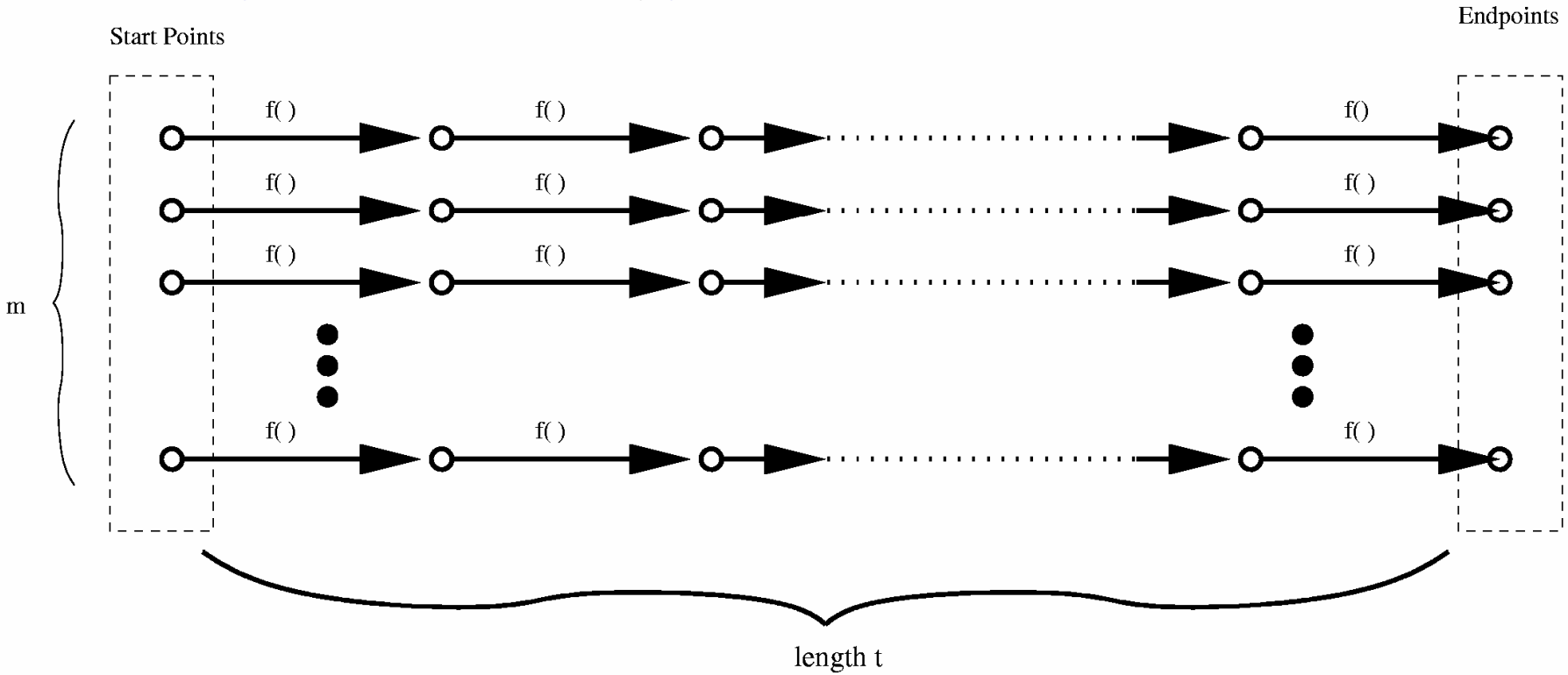
$T = t$, $M=N/t$, i.e., $TM=N$.

Hellman's T/M Tradeoff a Random *f*

Preprocessing phase:

m random starting points, chains of length *t*.

Store $(m, f^t(m))$ indexed by $f^t(m)$.

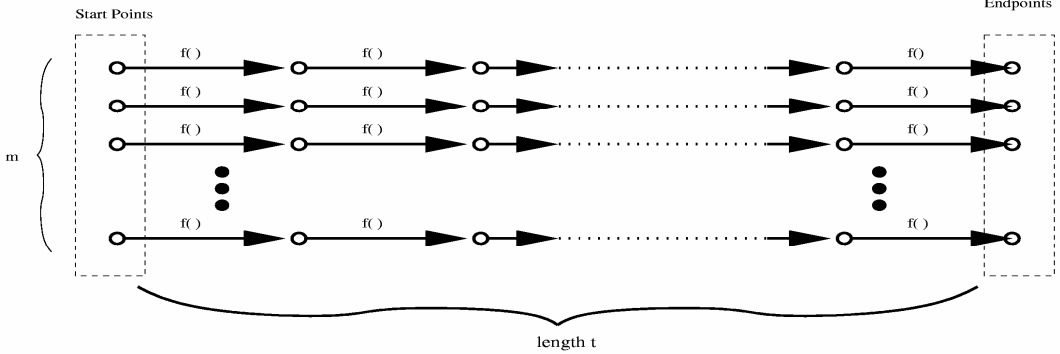


Online phase: from $y=f(x)$ complete the chain.

Find *x* by re-calculating the chain.

(Neglecting false alarms).

Hellman's T/M Tradeoff – Cont.



Problem: Difficult to cover more than N/t images with a single table.

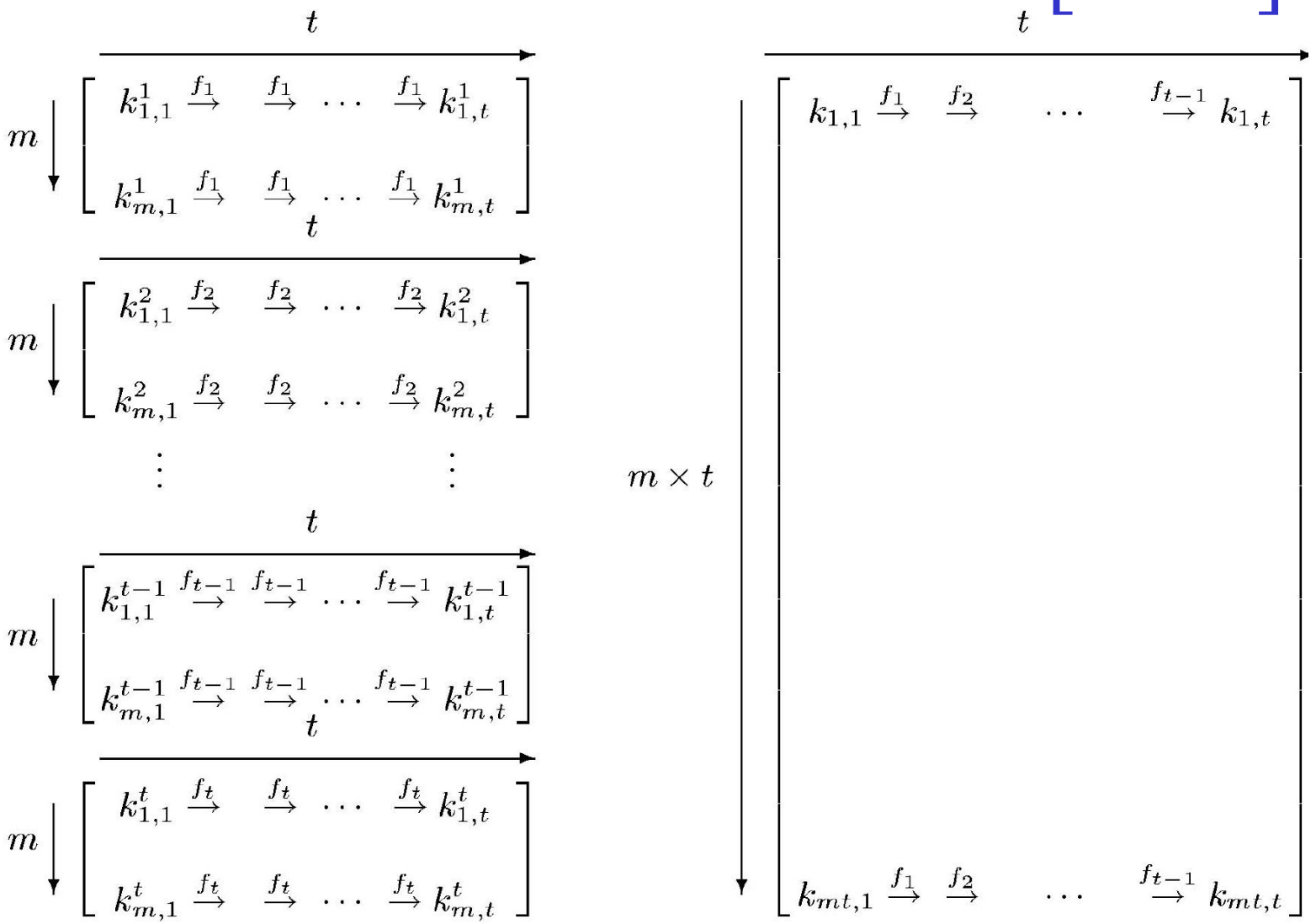
Why? Assume m rows and mt distinct images,
a new row collides with prob. 0.5 when $t \cdot mt \approx N$ (birthday paradox),
i.e., $mt = N/t \quad \Rightarrow \quad m^2t^4 = N^2$

Hellman's solution:
t "independent" tables from t "independent" $f_i = f(x \oplus i)$
– very different cycle structure, but invertion of $f_i \Rightarrow$ inversion of f.

In practice, a table covers $\approx 0.8mt$, and t tables cover $\approx 0.55 \cdot N$.

Analysis: $T = t^2$ (most time spent on wrong tables), $M = mt$, $\Rightarrow$ $TM^2 = N^2$.
#Disk accesses = T. Can be reduced to t [Rivest1982].

Oechslin's Rainbow Tables [2003]



$T=t^2/2$ instead $T=t^2$. No change to $M \Rightarrow 2TM^2=N^2$. #Disk accesses – t.

BUT coding the start points can take twice as many bits.

Schemes and Existing Bound

Hellman's tradeoff curve: $TM^2=N^2$

Rainbow tradeoff curve: $2TM^2=N^2$

Data variants [BiryukovShamir2000]: $TM^2D^2=N^2$

Is it a coincidence that the tradeoff curves are so similar?
(Did we hit a bound?)

[ShamirSpencer1981]: the coverage of "the best" Hellman table = coverage of an average Hellman table (upto logarithmic factors).

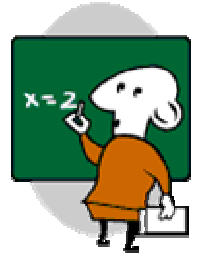
[Yao1990]: the running time of any scheme must be

$$T \geq \Omega(N(\log N)/M)^*.$$

As we saw, it is achievable for permutations,
is there a tighter lower bound for a random function?

* In Yao's model, M is measured in bits.

This Work



1. The *Stateful random graphs model* for cryptanalytic T/M tradeoffs. Evolution of paths in the graph depends on a *hidden state*.
2. Rigorously prove an upper bound on the coverage of M chains.
3. Derive a lower bound on the number of hidden states.
4. With additional natural assumptions, prove that for any T/M Tradeoff scheme:

   $$T \geq \Omega(N^2/(M^2 \ln N))$$

   for the overwhelming majority of the functions $\Rightarrow$
   Hellman and Rainbow schemes are tight up to logarithmic factors.
5. In the paper: bound on time/memory/data tradeoffs, present a new T/M scheme, improvements of T/M tradeoffs (by small factors), and time/memory/data tradeoff based on Rainbow tables.

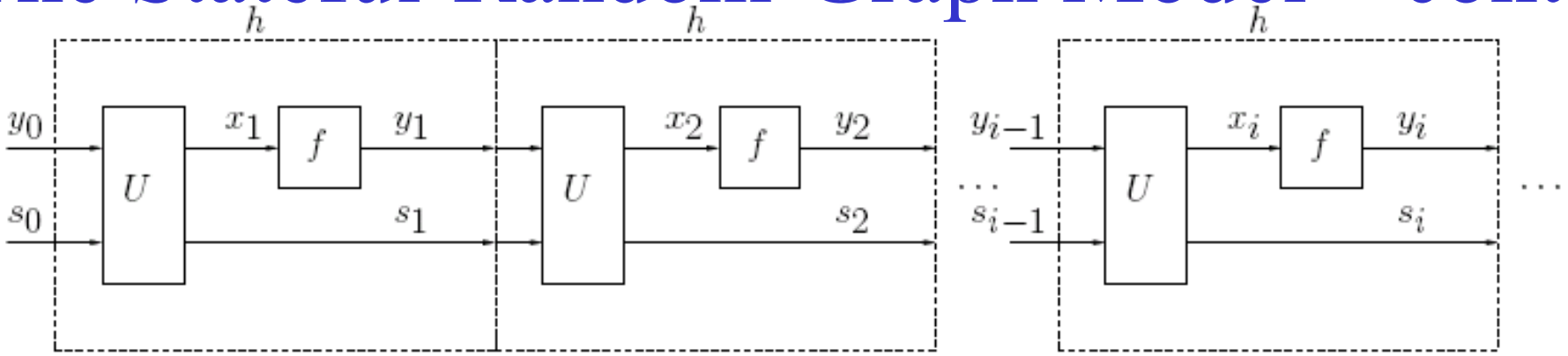
The Stateful-Random-Graph Model – cont

1. Adversary commits on a generic scheme with oracle accesses to f.
2. The actual choice of f is revealed to the adversary.
3. Adversary performs an unbounded *precomputation,* constructs the best collection of M chains.
4. Only M starting points and end points are passed to the *online algorithm* (memory complexity is M).
5. Chain-based *online algorithm*, succeeds in inverting y with prob. > 0.5.

Generous to adversary, no memory limit on size of the scheme.

Would not be fair to switch between steps 1 and 2.

The Stateful-Random-Graph Model – cont



We consider the most general model:

All oracle accesses to *f* (both in online and preprocessing) must be performed through a single a sub-algorithm *U*.

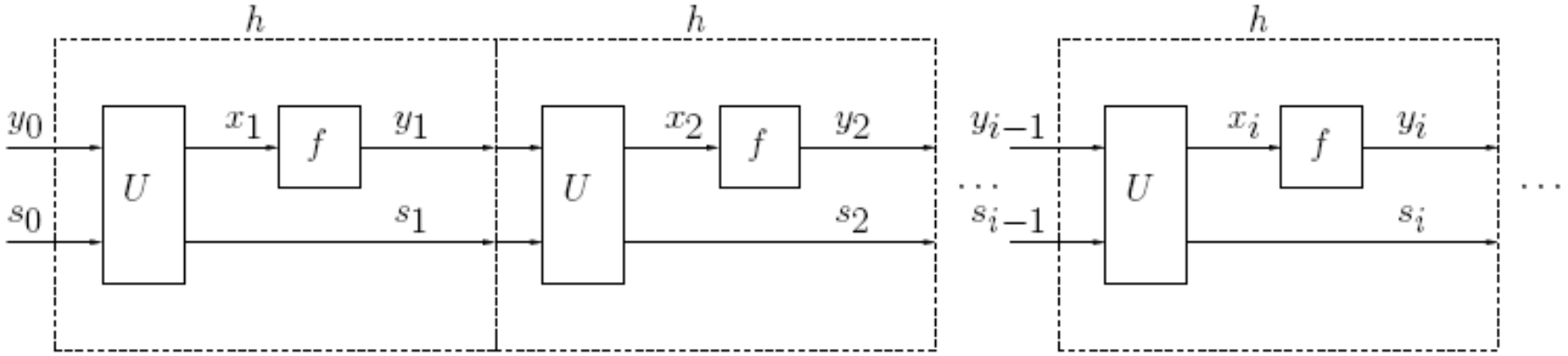
chain – a series of such accesses (link is created by *h)*.

hidden state – s_i – the internal state of the sub-algorithm (without the input/output of *f*).

Typical online algorithms: exhaustively try hidden states, waste most time on recovering the value of the hidden state. Only a square root of the running time spent on actual inversion.

U and *f* define a *Stateful Random Graph* – the evolution of (y_i,s_i).

The Stateful-Random-Graph Model – cont

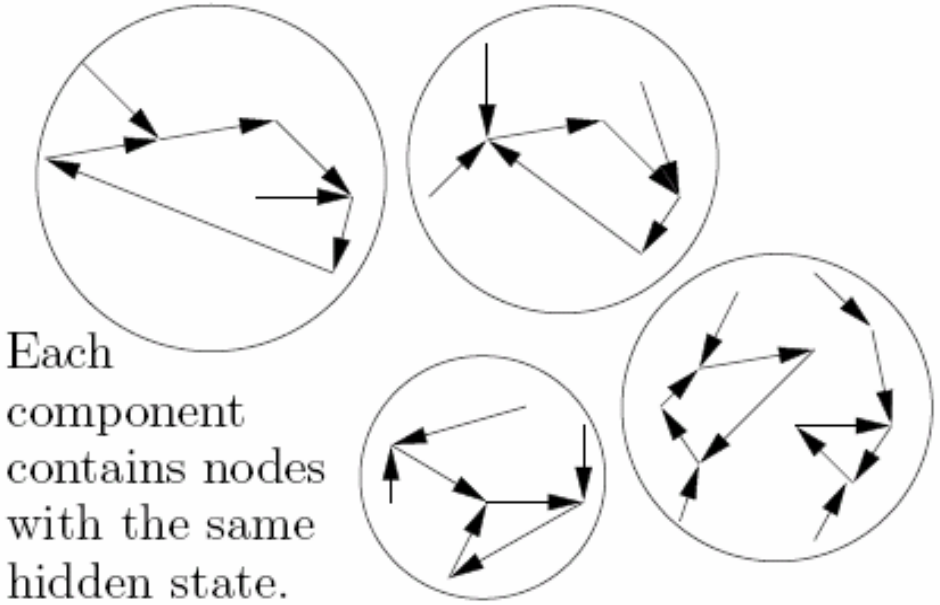


Examples for hidden state are
# of tables in Hellman or
# of colors in Rainbow.

In Hellman:

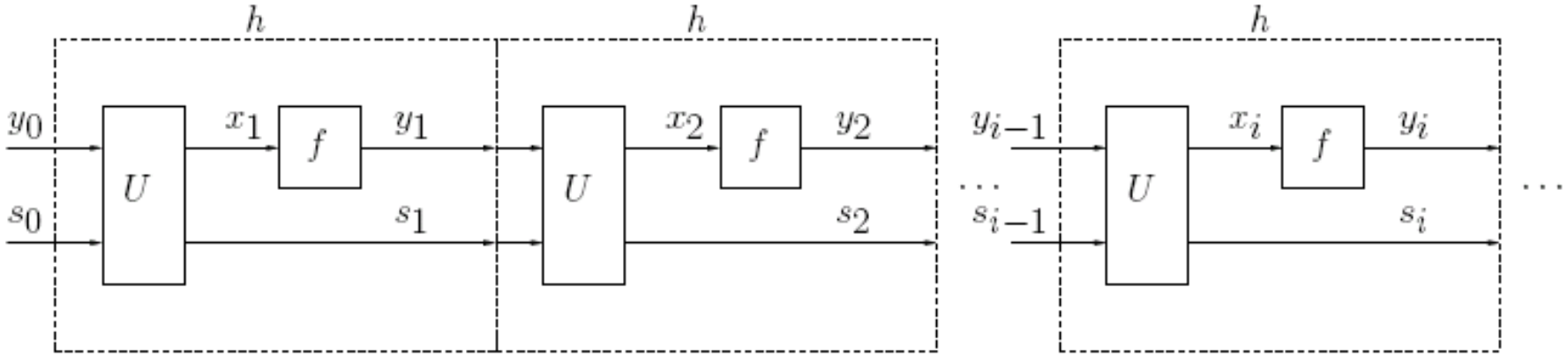
$x_i = y_{i-1} \oplus s_{i-1}$

$s_i = s_{i-1}$



Each component contains nodes with the same hidden state.

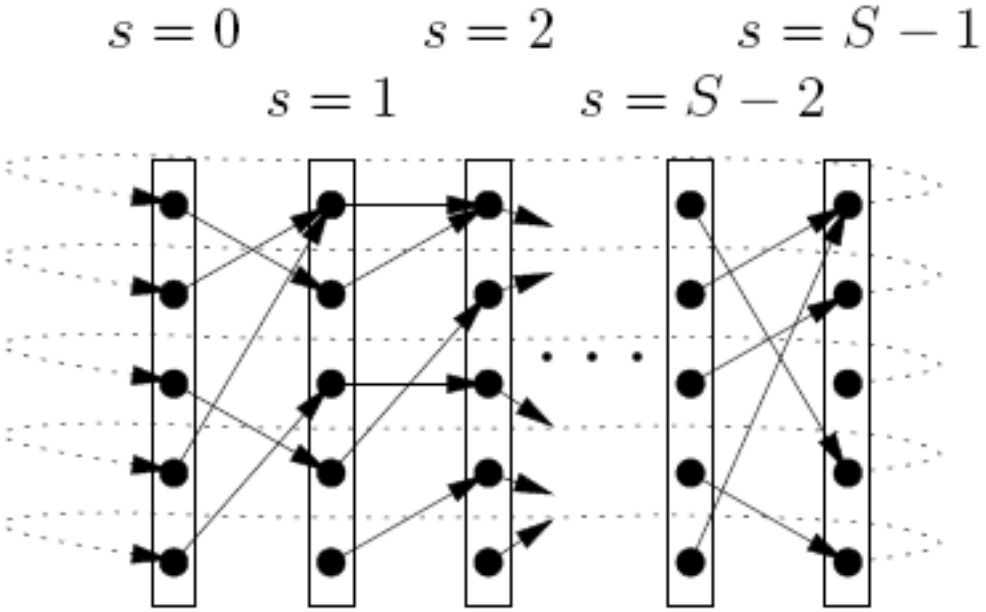
The Stateful-Random-Graph Model – cont



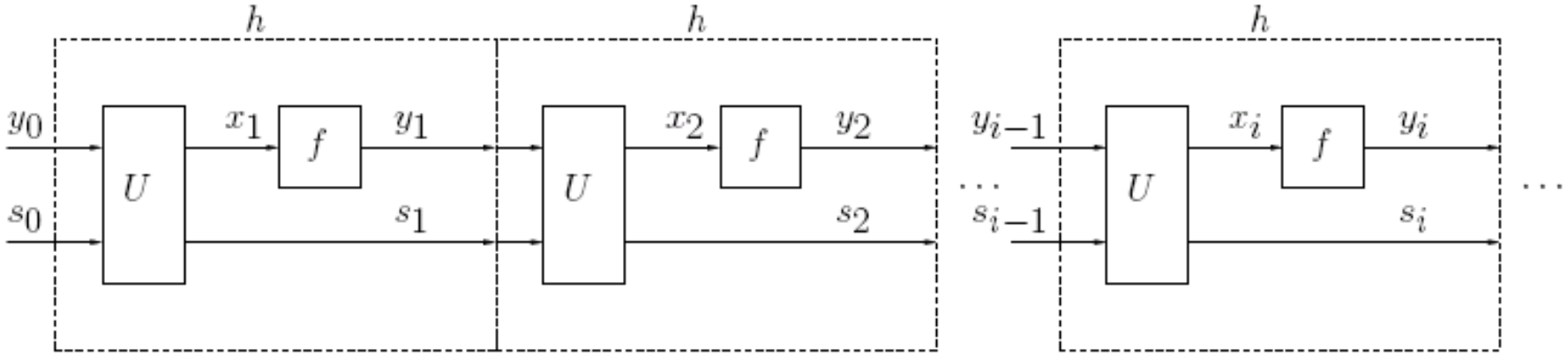
In Raibow a table with *S* colors:

$x_i = y_{i-1} \oplus s_{i-1}$

$s_i = s_{i-1} + 1 \bmod S$.



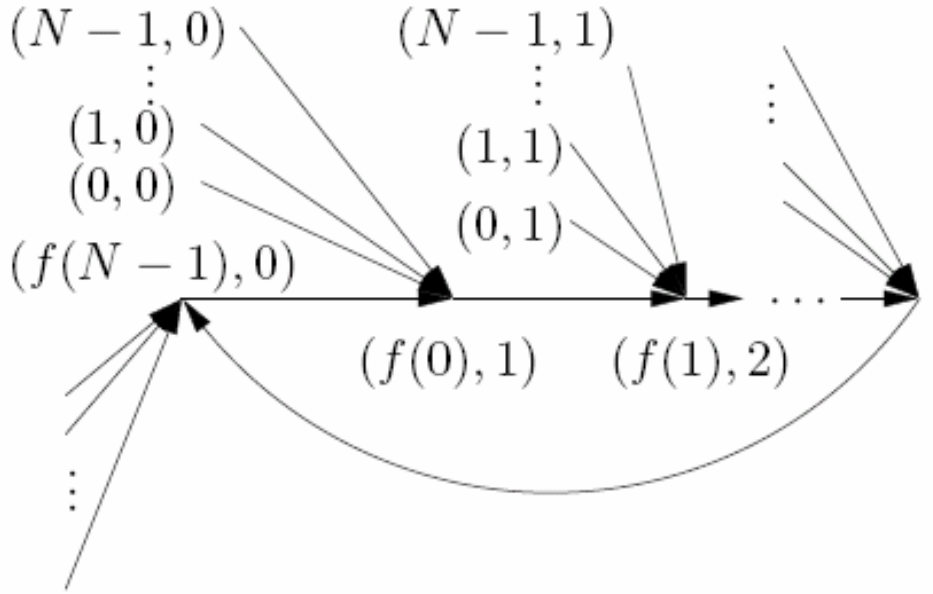
The Stateful-Random-Graph Model – cont



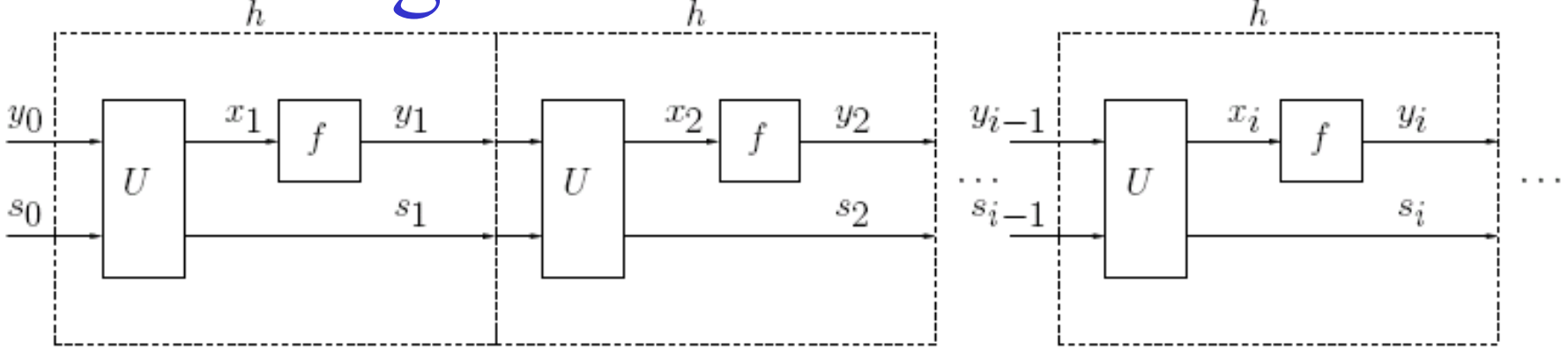
As the model is general, it is extremely flexible.
For example, we can choose:

$x_i = s_{i-1}$
$s_i = s_{i-1} + 1 \bmod N$,

which goes over
all the preimages
of f in a large cycle:

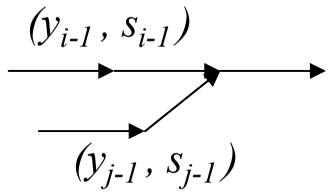


Coverage and Collision of Paths



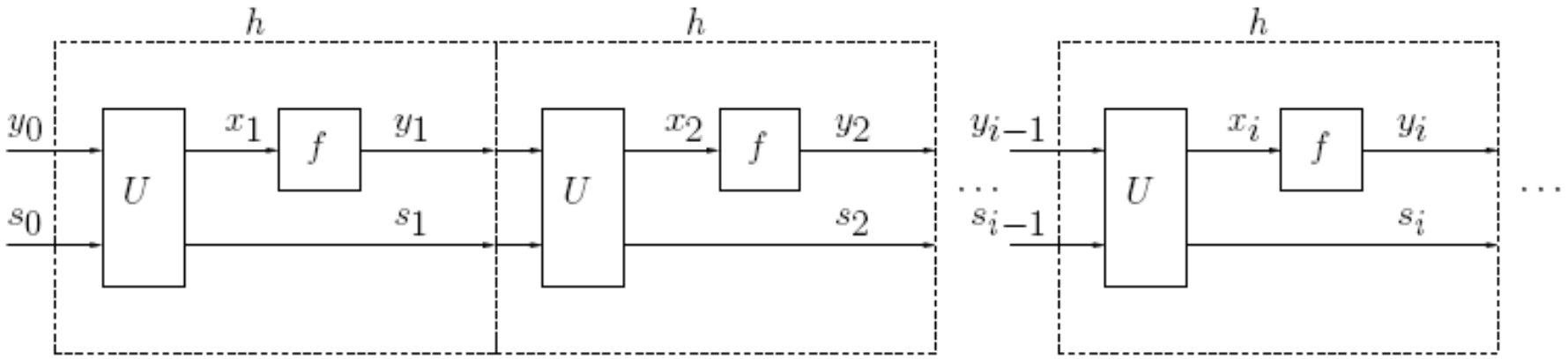
Goal: upper bound the size of
net coverage – the set of images y_i covered by the M paths.
Different from, the
gross coverage – set of points (y_i, s_i) covered by the M paths.

Two paths *collide* if
(y_i, s_i) in the first path =
(y_j, s_j) in the second path ($y_i = y_j$, and $s_i = s_j$).
No collision if only $y_i = y_j$ (but $s_i \neq s_j$).



Simplification: chains of unbounded length
(finite – each chain loops eventually or collides with another chain).

Coverage Theorem (Main Theorem)



Let $A = \sqrt{SNM \ln(SN)}$,
where $M = N^{\alpha}$, for any $0 < \alpha < 1$.
For Any U with S hidden states,
with overwhelming probability over the choices of f,
the net coverage of any collection of M paths of any length
in the stateful random graph is bounded from above by $2A$.

Interestingly, the components of the internal state of the sub-algorithm S and N play the same role in the bound.

We can limit the discussion to $S < N$.

Reduction of the Best Choice of Paths to the Average Choice of Paths.

A Huge Table W:

	M_1	M_2	$\cdots$	$M_{\binom{NS}{M}}$
f_1	0	0		
f_2	1	0		
$\vdots$			$\ddots$	
f_{N^N}				

For the choice of U,
$W_{i,j}=1$ if the net coverage of f_i and M_j is larger than $2A$
(and 0 otherwise).

It suffices to show that $\#1$'s$<<\#r \Leftrightarrow \text{Prob}(W_{i,j}=1)\cdot\#r\#c << \#r \Leftrightarrow$ $\text{Prob}(W_{i,j}=1)\cdot\#c$ is very close to zero.

We first upper bound $\text{Prob}(W_{i,j}=1)$, and then multiply by $\#c$.

Upper Bounding Prob($W_{i,j}=1$)

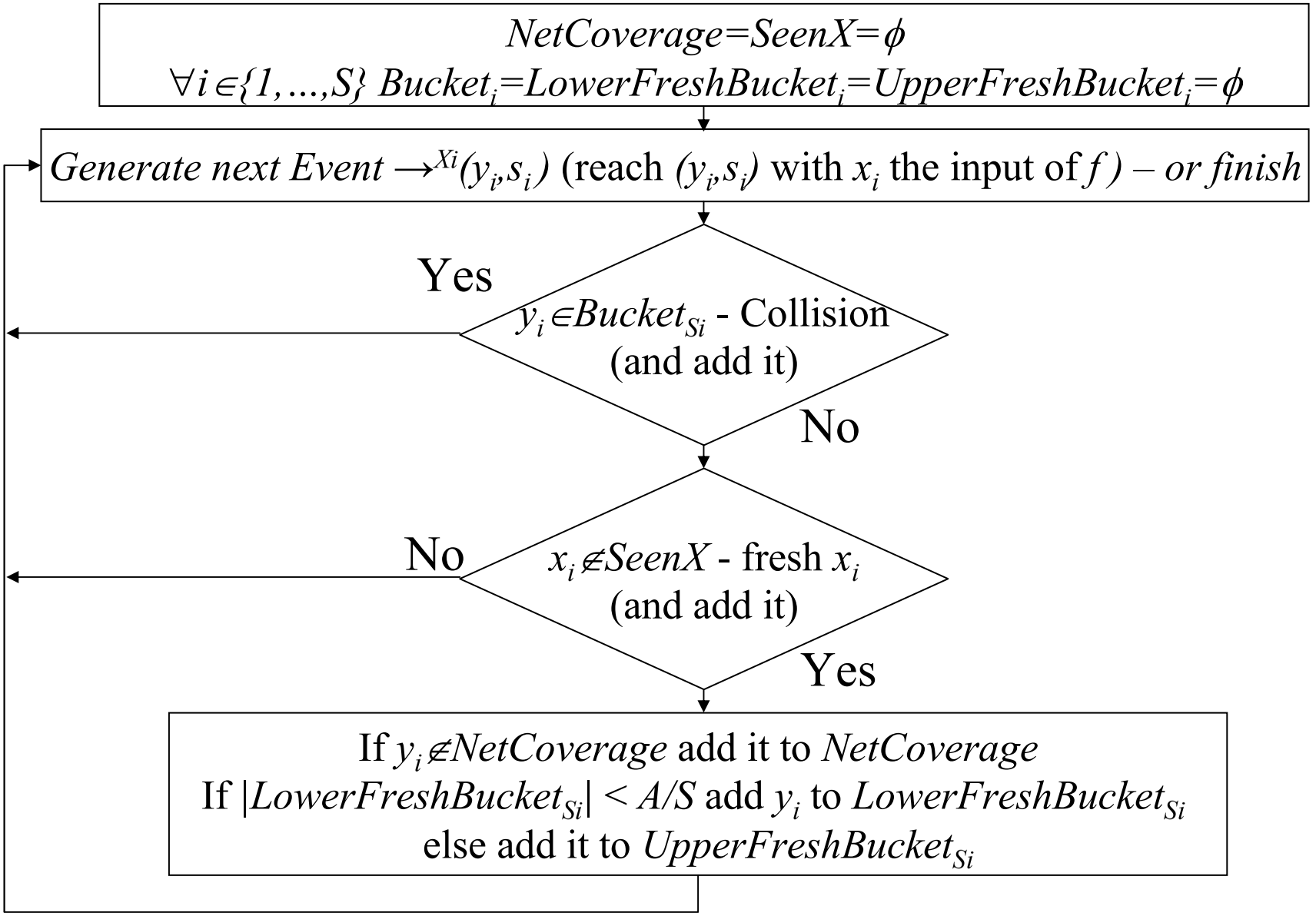
Method:

- An algorithm counts the net coverage for f_i and M_j.
- Analyze prob. that the counted coverage > $2A$, *i.e.*, Prob($W_{i,j}=1$).

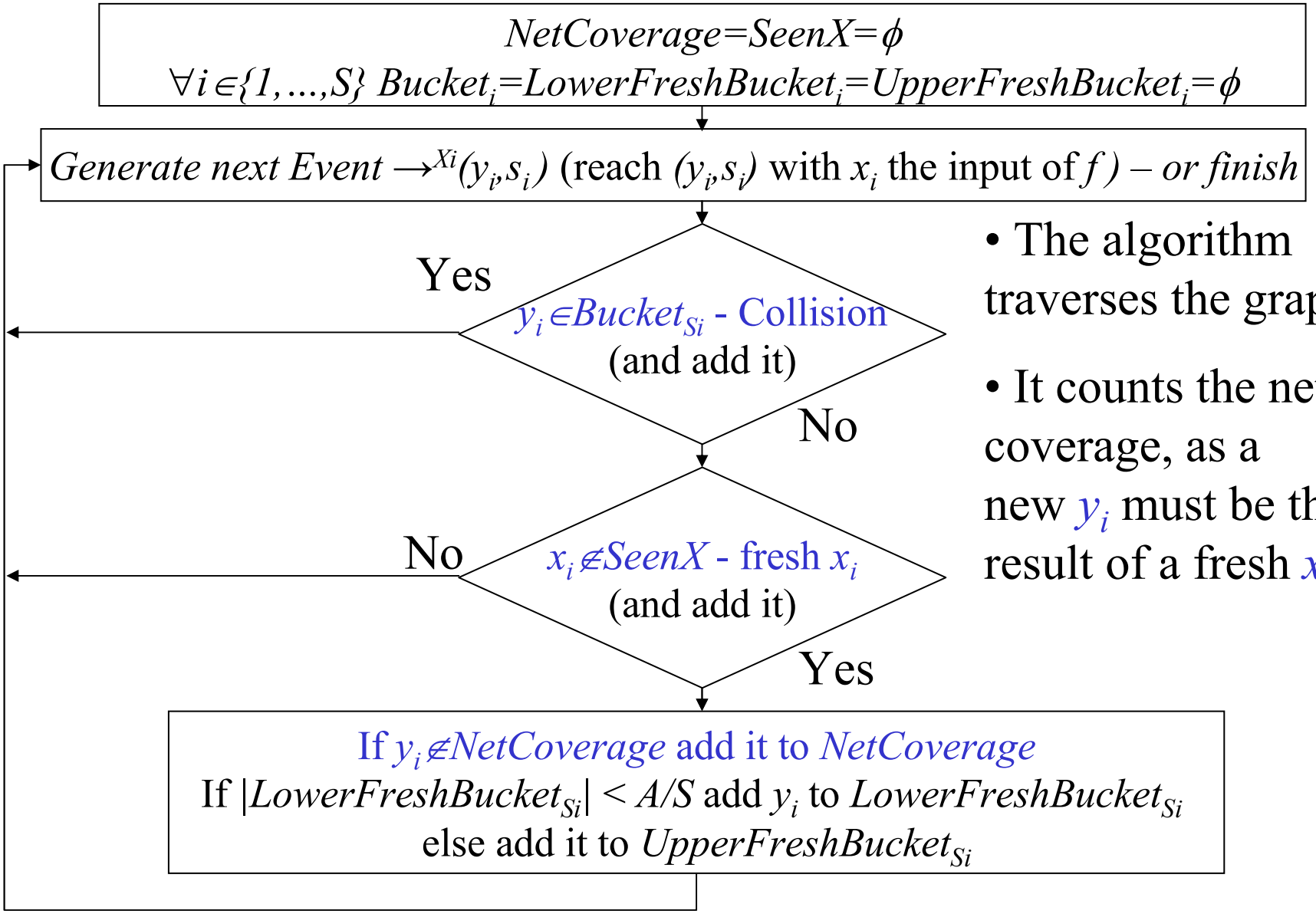
Prob($W_{i,j}=1$) is taken over a random and uniform choice of start points M_j and a function f_i,
convenient to view the output of f as a truly random number $\{0,\ldots,N-1\}$.

View is justified only the first time f is applied on a value (*fresh* value).
Otherwise, the output of f is already known.

An Algorithm that Counts the NetCoverage (and few other things)

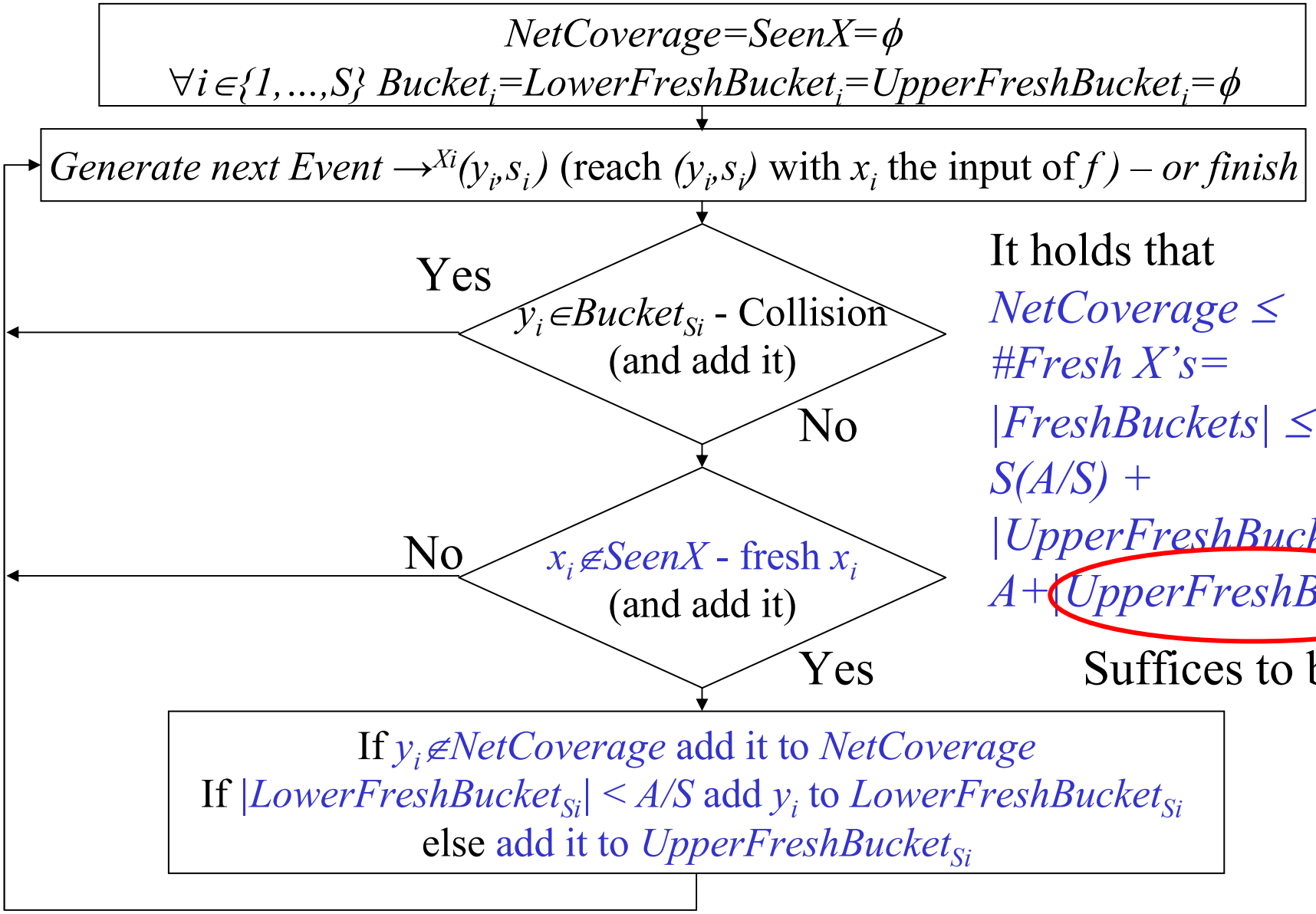


An Algorithm that Counts the NetCoverage – cont



- The algorithm traverses the graph.
- It counts the net coverage, as a new y_i must be the result of a fresh x_i.

An Algorithm that Counts the NetCoverage – cont

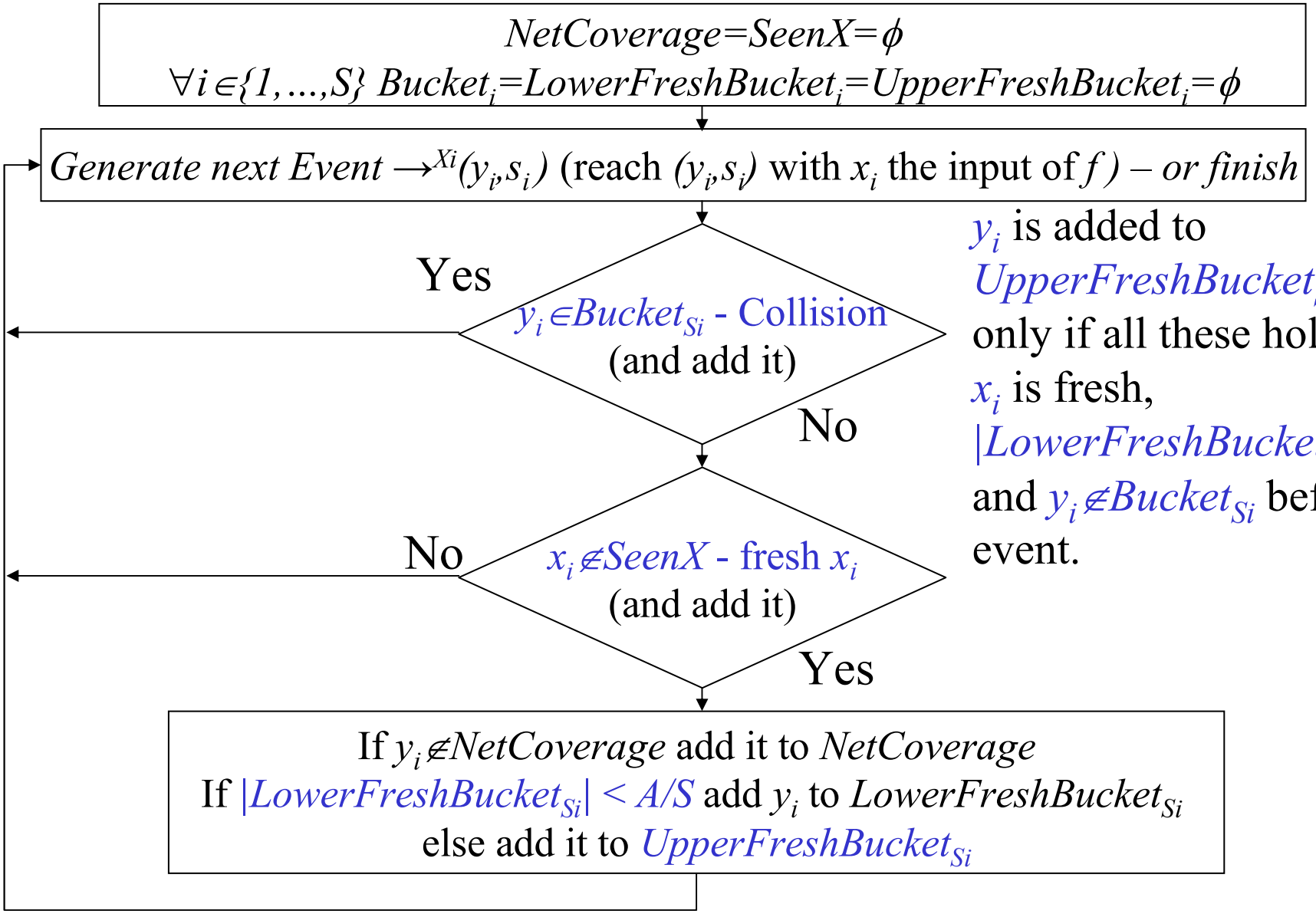


It holds that

$NetCoverage \leq \#Fresh\ X's = |FreshBuckets| \leq S(A/S) + |UpperFreshBuckets| = A + |UpperFreshBuckets|$

Suffices to bound

An Algorithm that Counts the NetCoverage – cont



y_i is added to $UpperFreshBucket_{Si}$ if and only if all these hold: x_i is fresh, $|LowerFreshBucket_{Si}|=A/S$ and $y_i \notin Bucket_{Si}$ before the event.

An Algorithm that Counts the NetCoverage – cont

If these two conditions hold, then we call the event a "coin toss".

y_i is added to $UpperFreshBucket_{Si}$ if and only if all these hold:

- x_i is fresh,
- $|LowerFreshBucket_{Si}|=A/S$
- and $y_i \notin Bucket_{Si}$ before the event.

Clearly, it holds that $|UpperFreshBuckets| \leq$ # Coin Tosses.

If $y_i=f(x_i) \in LowerFreshBucket_{Si}$ before the event, and the event is a coin toss, then we call it a "successful coin toss".

A successful coin toss is a collision (not necessarily vice versa).
There cannot be too many coin tosses, as some will be successful.

An Algorithm that Counts the NetCoverage – cont

What is the probability that a coin toss is a successful coin toss?

Exactly $q=A/(SN)$, *independently* of any previous event (justifying the name "coin toss").

Proof: coin toss $\Rightarrow$ x_i is fresh $\Rightarrow$ $y_i=f(x_i)$ is truly random and independent of previous events.
Moreover, coin toss $\Rightarrow$ $|LowerFreshBucket_{Si}| = A/S$.
Therefore, prob. that $f(x_i) \in LowerFreshBucket_{Si}$ is exactly $(A/S)/N$.

Reminder:

coin toss $\left\{ \begin{array}{l} x_i \text{ is fresh, and} \\ |LowerFreshBucket_{Si}|=A/S \end{array} \right.$

A coin toss is successful
when $y_i \in LowerFreshBucket_{Si}$ before the event.

An Algorithm that Counts the NetCoverage – cont

We saw: $NetCoverage \leq \#Fresh\ X's \leq |FreshBuckets| = A+|UpperFreshBuckets| \leq A+\#Coin\ Tosses$

Coin toss is succesful with prob. $q=(A/SN)$ *independent of past events.*

What is prob. that $W_{i,j}=1$ (= prob. $NetCoverage > 2A$) ?
≤ prob. that $A+\#Coin\ Tosses > 2A$,
i.e., $Prob(W_{i,j}=1) \leq Prob(\#Coin\ Tosses > A)$.

How many successful coin tosses can there be?
- At most M, as each successful coin toss
means a collision, and a collision ends a path (there are only M paths).

$\#Coin\ Tosses>A$ means that after A coin tosses #successes < M.
Happens with prob. ≤ $Prob(B(A,p)<M)$,
where $B(A,p)$ is a binomial random variable, with $p=A/(SN)$.
Therefore, $Prob(W_{i,j}=1) \leq Prob(B(A,p) < M)$

An Algorithm that Counts the NetCoverage – cont

So far: $\text{Prob}(W_{i,j}=1) \leq \text{Prob}(\text{B}(A,p) < M)$

How to choose A?

A too large – upper bound will not be tight (most of the W table is 0's).

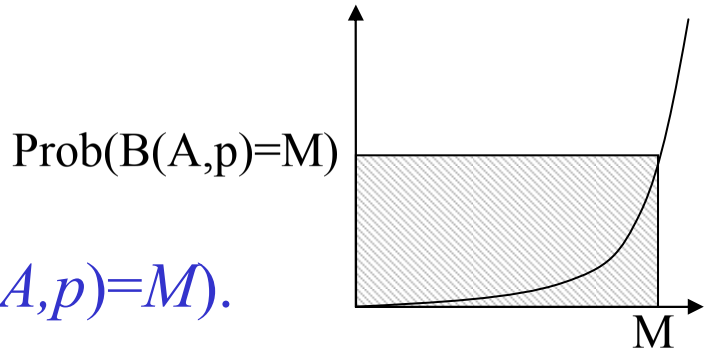
A too small – no bound (most of the W table will be 1's).

We choose A such that the expected number of successful coin tosses is $Aq=M\ln(NS)$. Thus, $A=\sqrt{SNM\ln(SN)}$.

Binomial distribution is rising for this choice of A and q with respect to M.

Therefore,

$\text{Prob}(W_{i,j}=1) \leq \text{Prob}(B(A,p)<M) \leq M\cdot\text{Prob}(\text{B}(A,p)=M)$.



Concluding the proof:

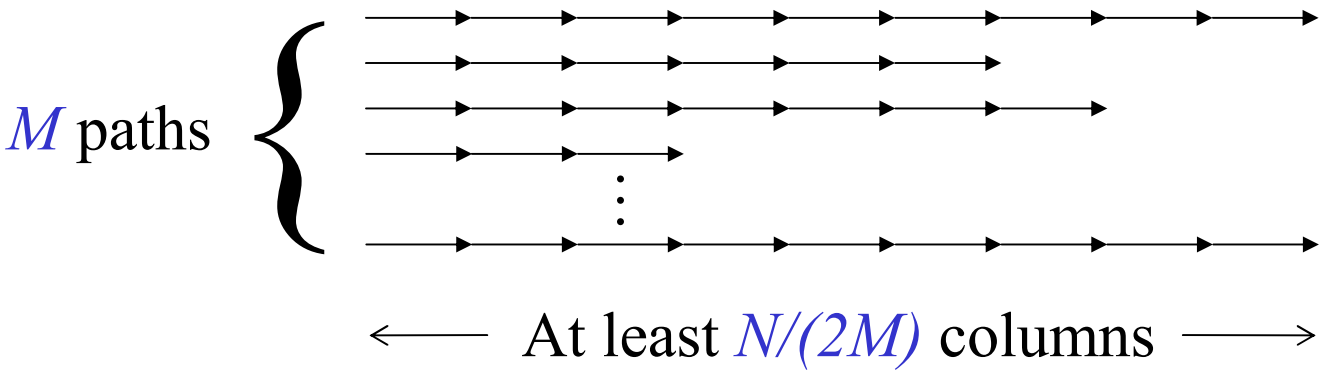
$\text{Prob}(W_{i,j}=1)\cdot\#\text{c} \leq M\cdot\text{Prob}(\text{B}(A,p)=M)\cdot\#\text{c} \leq \ldots \rightarrow 0$

when N grows to infinity. QED.

Lower Bound on the Number of Hidden States S

Net coverage of at least $N/2$ $\Rightarrow$
for the overwhelming majority of functions: $2A = 2\sqrt{SNM\ln(SN)} \geq \frac{N}{2}$,

and therefore, $S \geq \frac{N}{16M\ln(SN)} \geq \frac{N}{32M\ln(N)}$.



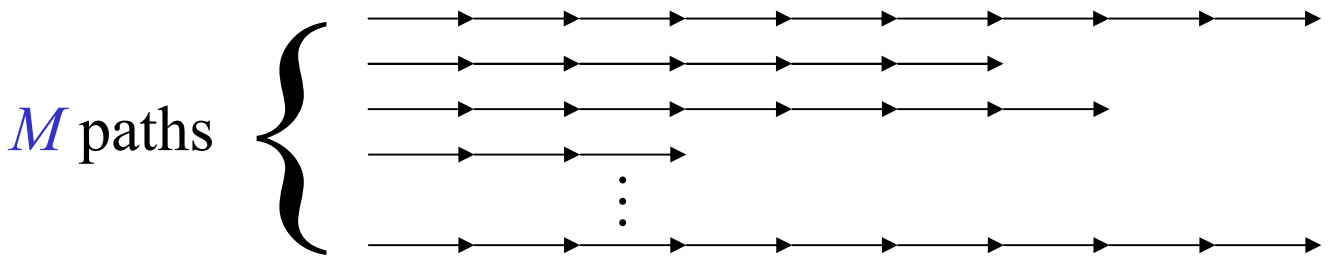
"Tempting" to use the lower bound on S, and say that for each hidden state the algorithm runs on average at least $N/(4M)$ (half the matrix), and therefore, $T \geq \frac{N}{4M}\frac{N}{32M\ln(N)} \geq \frac{N^2}{128M^2\ln(N)}$.

Incorrect, e.g., a single large hidden state close to the endpoints.

Lower Bound on the Worst-Case Time Complexity

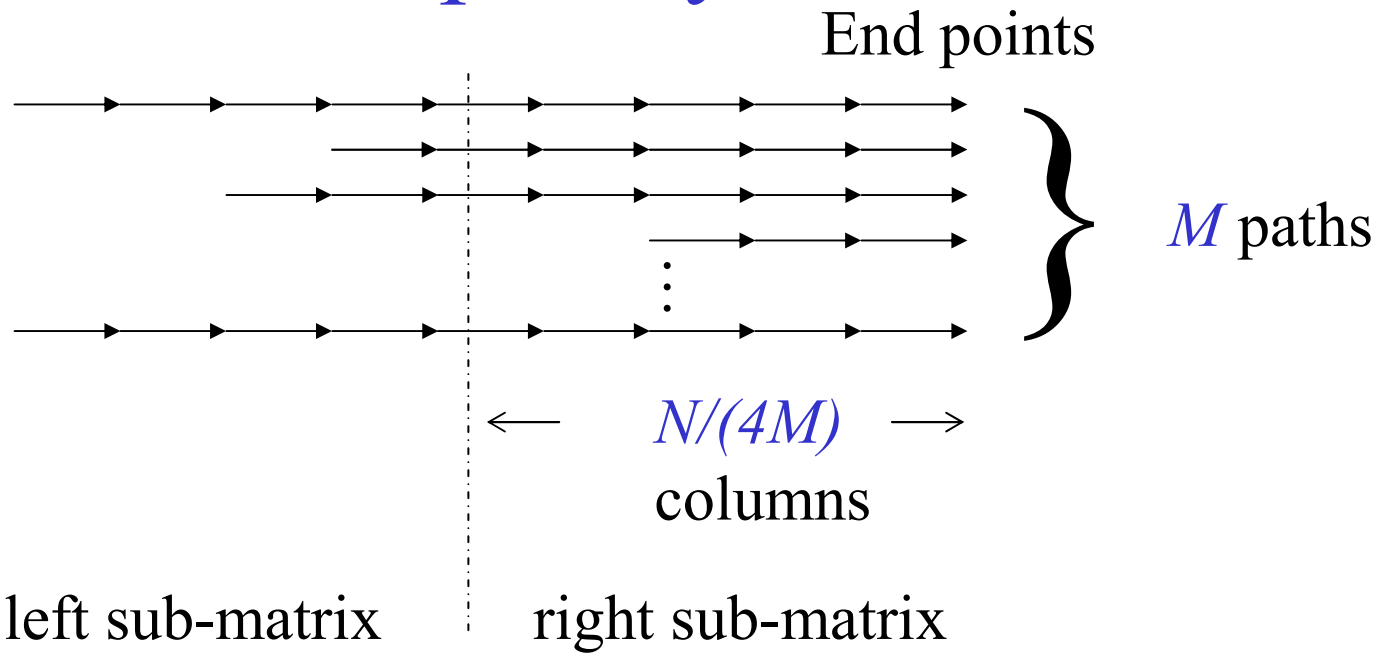
Make the following natural assumption:
To invert y, the online algorithm sequentially tries the hidden states (in any order).

For hidden state s, it applies h on (y,s) at least t_s times in case (y,s) is not covered by the table.
t_s is the largest distance from any point with hidden state s in the matrix to its corresponding end point.



Note: the t_s values can depend on the specific matrix that results from the precomputation (and thus depend on f).

Lower Bound on the Worst-Case Time Complexity – cont



The right sub-matrix contains at most $N/4$ images.
Therefore, the left sub-matrix contains at least $N/4$ images
(as the total number of images has to be at least $N/2$).

Consider the set of hidden states S_L that appear in the left sub-matrix.
Using our (slightly modified) coverage theorem, it follows that $S_L \geq \frac{N}{256M\ln(N)}$.

Therefore, $T \geq \frac{N}{4M}\frac{N}{256M\ln(N)} \geq \frac{N^2}{1024M^2\ln(N)}$ (for the overwhelming majority of functions f).

Summary

- The new model for time/memory tradeoffs based on a new concept of a stateful random graph.
- Rigorous combinatorial proof gives a lower bound on the number of hidden states required in order to cover at least $N/2$ images.
- Lower bound on the worst-case time complexity.
- In the full paper we also:
  - Provide a similar lower bound for time/memory/data tradeoffs.
  - Show that Rainbow tables are not as efficient as Hellman's method (taking into account the representation of start points and end points).
  - Present another new time/memory tradeoff.
  - Present a time/memory/data tradeoff based on Rainbow tables.
  - Show how we can gain small factors in the time complexity by performing a deeper preprocessing.

Thank You

Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs

Elad Barkan

Technion – Israel Institute of Technology

Joint work with Eli Biham and Adi Shamir