Recent Advances in Analysis of HMAC

Jian Guo

Nanyang Technological University, Singapore

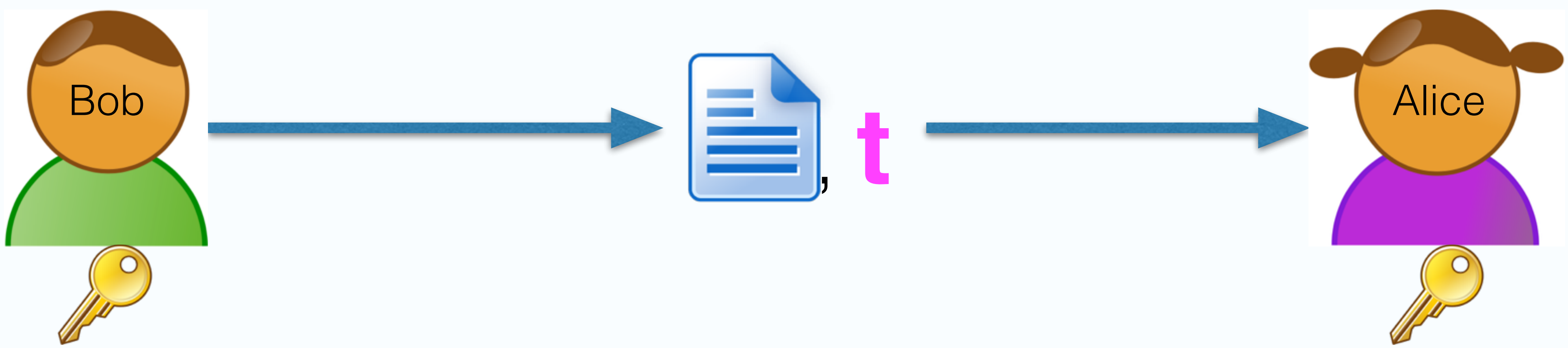
22 Dec, ASK 2014 @ Chennai, India

Overview

- Introduction to HMAC
- Pollard Rho Method and Functional Graph
- Distinguishers, Forgeries and Key Recovery Attacks
- Applications to HMAC-Whirlpool

Introduction to MAC

Message Authentication Code (MAC) is a short string used to provide integrity and authenticity.



1. Alice and Bob share a key **k**
2. Bob sends t = MAC_k(M), and M
3. Alice receives (M*, t*), she computes t'=MAC_k(M*)
4. Alice checks if t* = t', and confirms the message M* is consistent with M, i.e., M* = M, and it was indeed from Bob

NANYANG TECHNOLOGICAL UNIVERSITY

MAC constructions

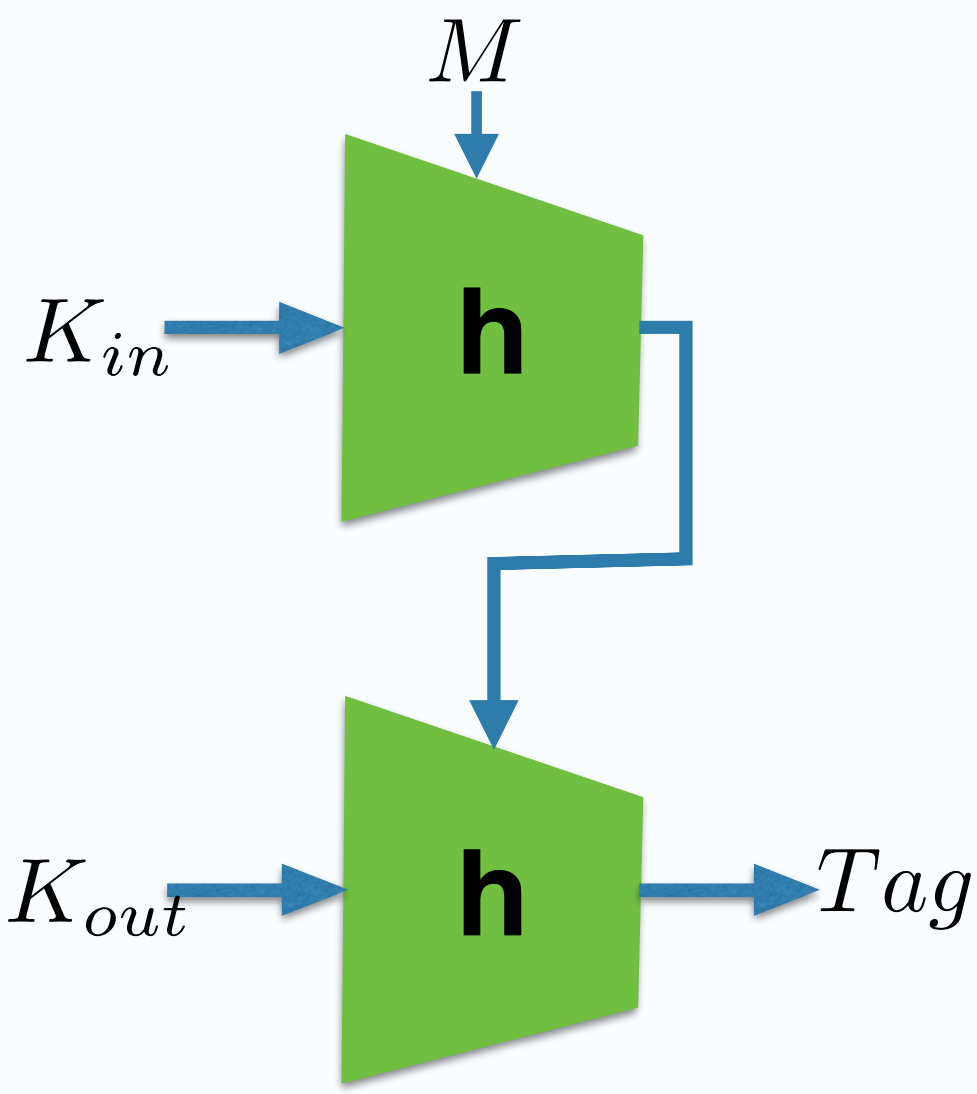
- Dedicated designs
  - Pelican-MAC, SQUASH, SipHash
- From universal hash functions
  - UMAC, VMAC, Poly1305
- From block ciphers
  - CBC-MAC, CMAC, OMAC, PMAC
- From hash functions
  - **HMAC**, Sandwich-MAC, Envelope-MAC

Introduction to HMAC

- Designed by Mihir Bellare, Ran Canetti and Hugo Krawczyk at CRYPTO 1996
- Standardized by ANSI, IETF, ISO, NIST from 1997
- **The** most widely deployed hash-based MAC construction, implemented in SSL, TLS, IPSec, etc.

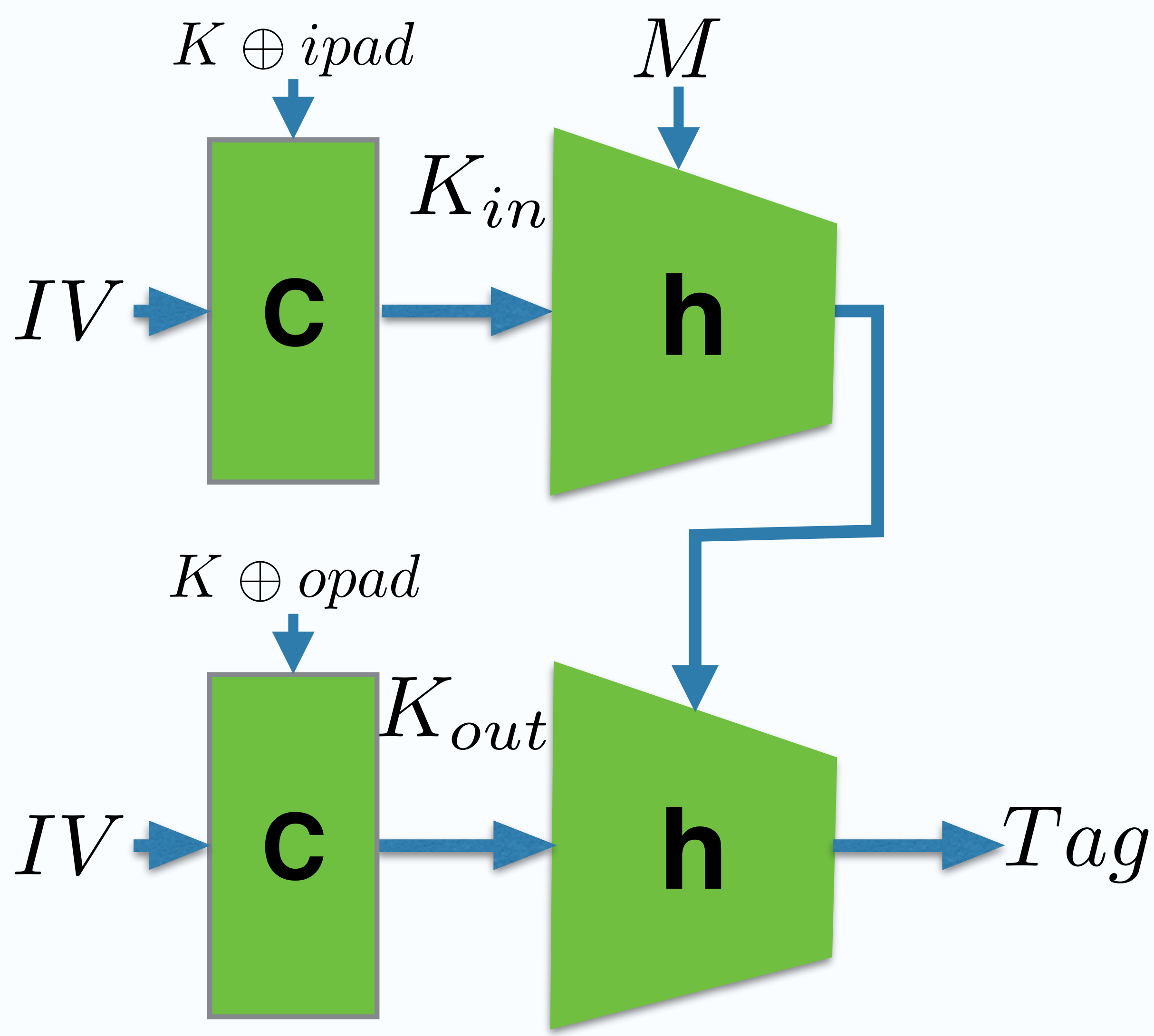
NMAC construction

- **2** Independent Keys
- Proven security up to $2^{l/2}$ with l for internal state size



HMAC construction

- Based on NMAC, generate inner and outer keys from a single master key ***K***
- Security bounds remain the same as for NMAC



Attack Models against MAC

- Distinguishers
  - Distinguishing-R: distinguish the MAC function from random oracle
  - Distinguishing-H: distinguish a MAC instantiated with some hash function from a MAC instantiated with a random function.
- Forgeries: given one or more valid ($\mathbf{M_i}$, $\mathbf{t_i}$) pairs, attacker shows another valid pair ($\mathbf{M_j}$, $\mathbf{t_j}$) where $\mathbf{M_j}$ has never been queried.
  - Existential Forgery: attacker controls both provided message $\mathbf{M_i}$'s and the forged one $\mathbf{M_j}$
  - Selective Forgery: forgery applies to a pre-selected message set of $\mathbf{M_i}$'s
  - Universal Forgery: forgery applies to any message $\mathbf{M_i}$
- Key Recovery: forgery at will, impersonate and more….
  - Master key or equivalent keys

Results in last 3 years

1. Thomas Peyrin, Yu Sasaki, Lei Wang: Generic Related-Key Attacks for HMAC. ASIACRYPT 2012
2. Gaëtan Leurent, Thomas Peyrin, Lei Wang: New Generic Attacks against Hash-Based MACs. ASIA CRYPT 2013
3. Jian Guo, Yu Sasaki, Lei Wang, Shuang Wu: Cryptanalysis of HMAC/NMAC-Whirlpool. ASIACRYPT 2013
4. Thomas Peyrin, Lei Wang: Generic Universal Forgery Attack on Iterative Hash-Based MACs. EUROCRYPT 2014
5. Jian Guo, Thomas Peyrin, Yu Sasaki, Lei Wang: Updates on Generic Attacks against HMAC and NMAC. CRYPTO 2014
6. Itai Dinur, Gaëtan Leurent: Improved Generic Attacks against Hash-Based MACs and HAIFA. CRYPTO 2014
7. Jian Guo, Yu Sasaki, Lei Wang, Meiqin Wang, Long Wen, Equivalent Key Recovery Attacks against HMAC and NMAC with Whirlpool Reduced to 7 Rounds. FSE 2014

NANYANG TECHNOLOGICAL UNIVERSITY

Results in last 3 years

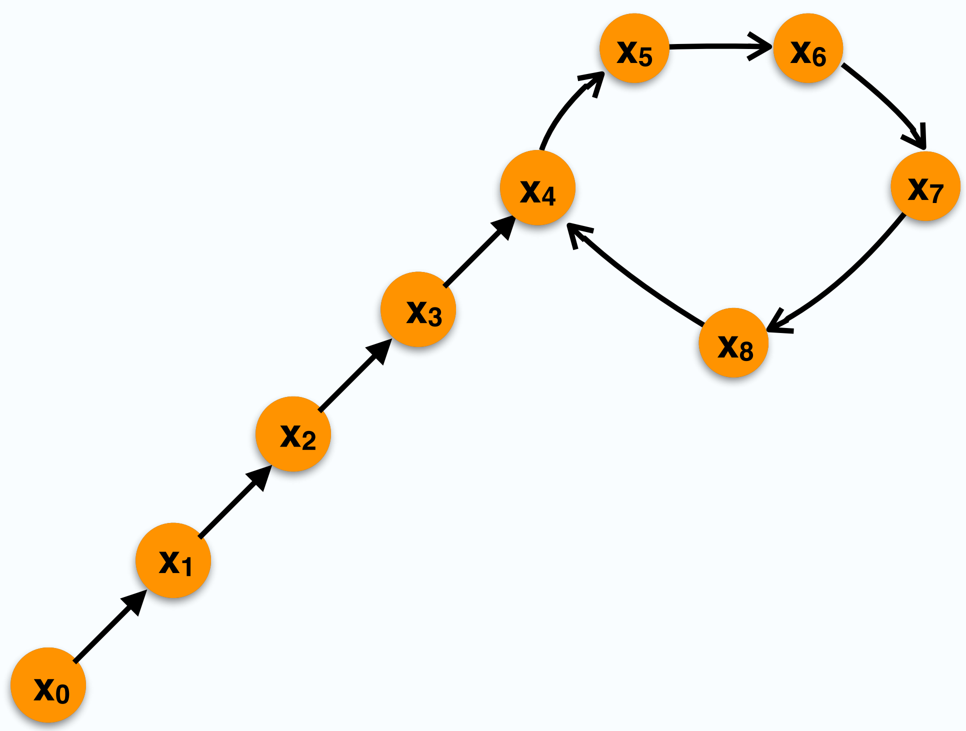
Attack Types	Proven Bound	Generic Attacks	Recent Result	Remark
distinguishing-R	l/2	l/2	[1,2]	tight
distinguishing-H	l/2	l/2	[1,2]	tight
existential forgery	l/2	l/2	[2]	tight
selective forgery	l/2	l/2 ~ l	[5]	hash dependent
universal forgery	**l/2**	**3l/4**	**[4,5,6]**	**gap**
key recovery	k	3l/4, l	[3,5,7]	**TMD tradeoff**

NANYANG TECHNOLOGICAL UNIVERSITY

Pollard Rho Method

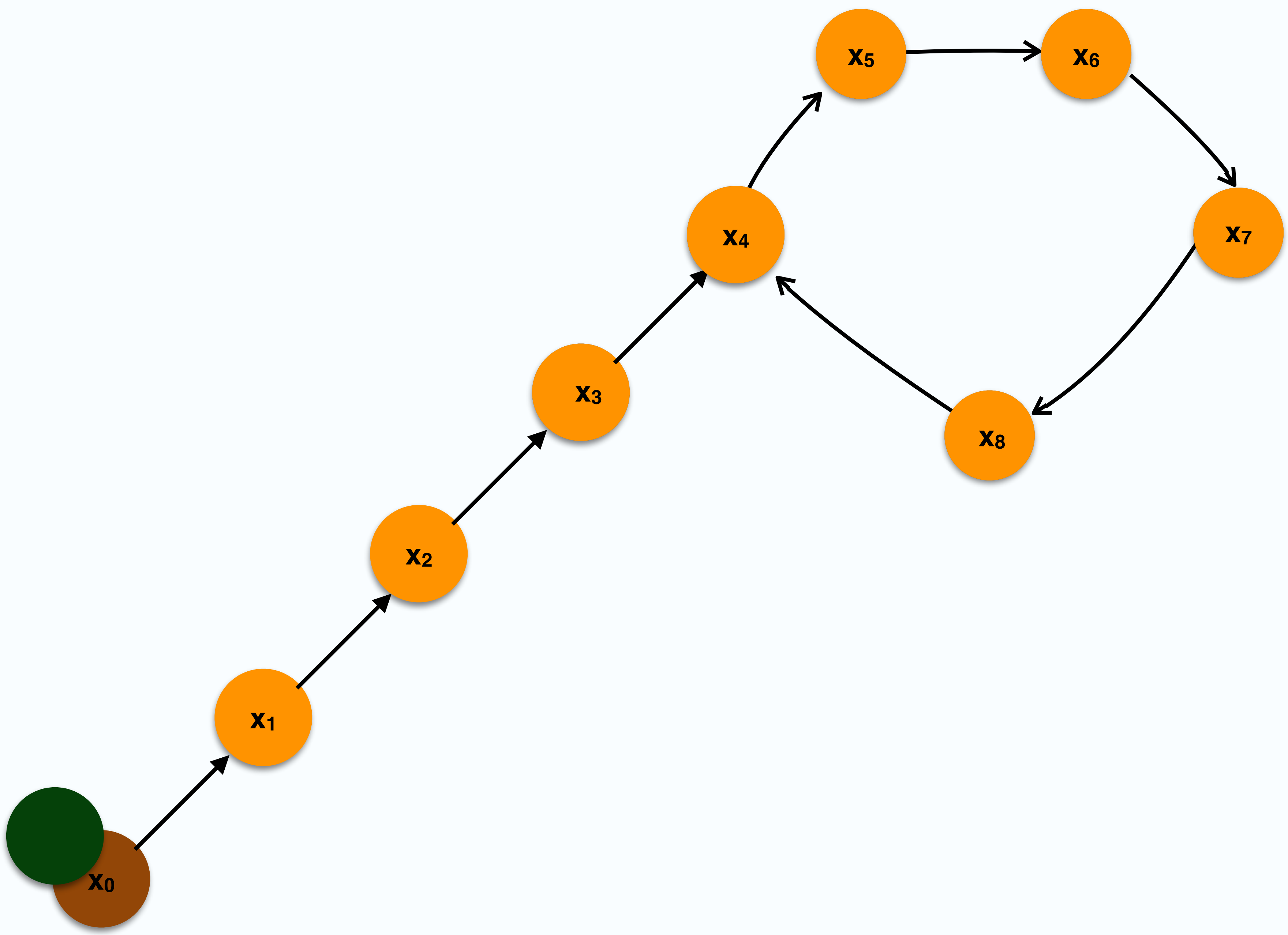
- node: value;
  arrow: function f,
  with $x_{i+1} = f(x_i)$

- Two threads, one evaluate f once at each step, the other two f evaluations at each step, collision will be detected inside the cycle.

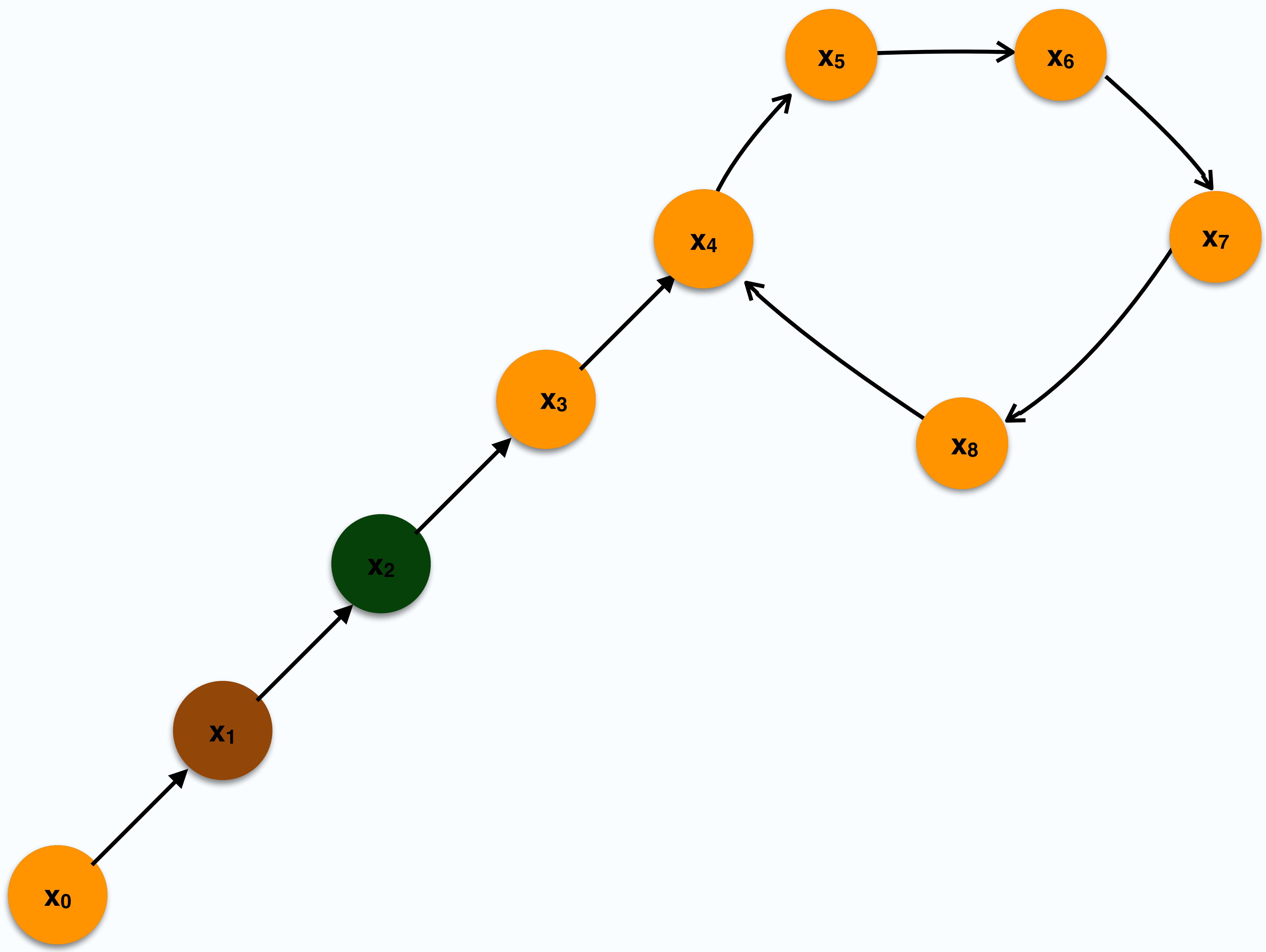


NANYANG TECHNOLOGICAL UNIVERSITY

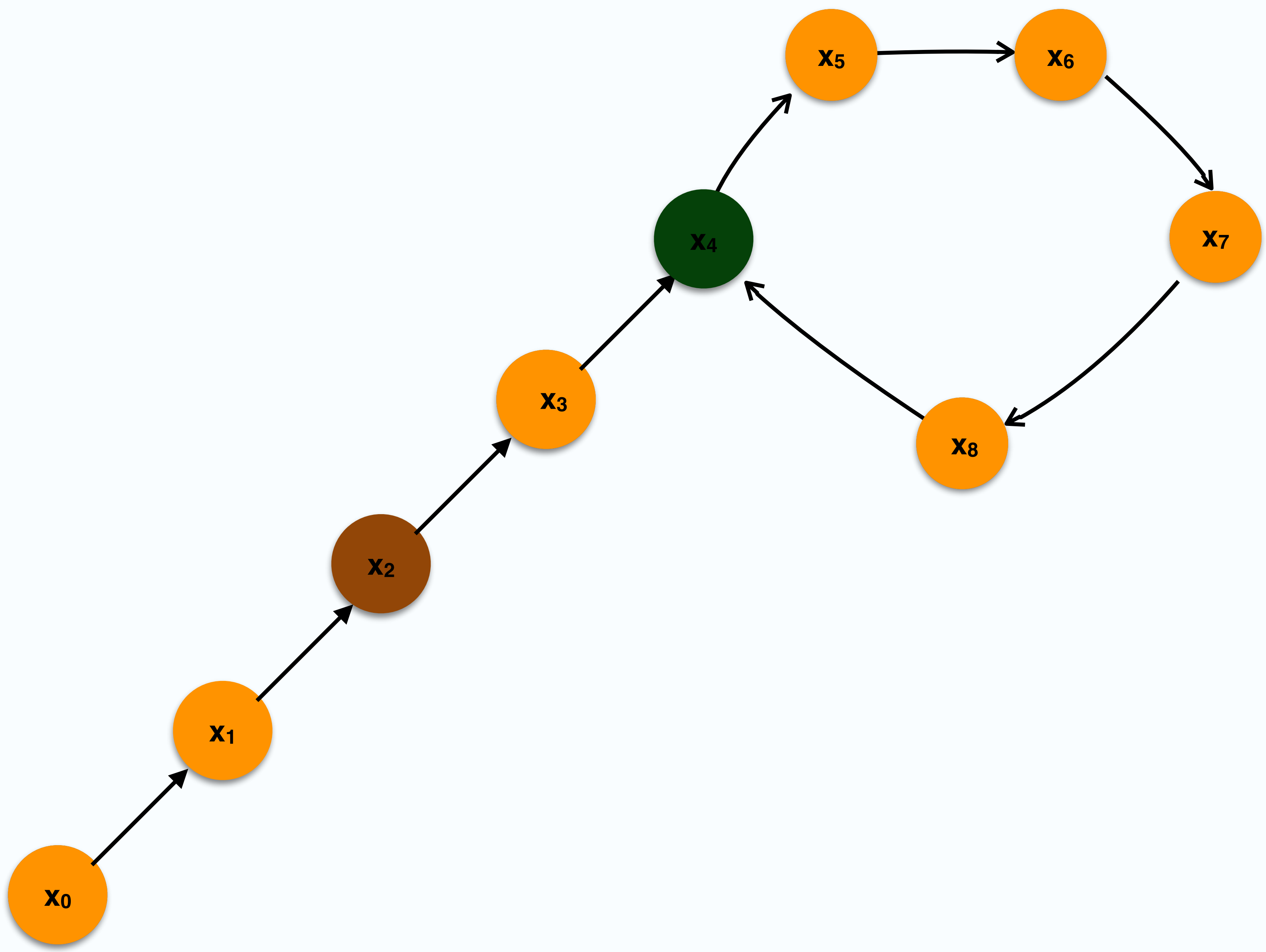
Pollard Rho Method
Detection - 0



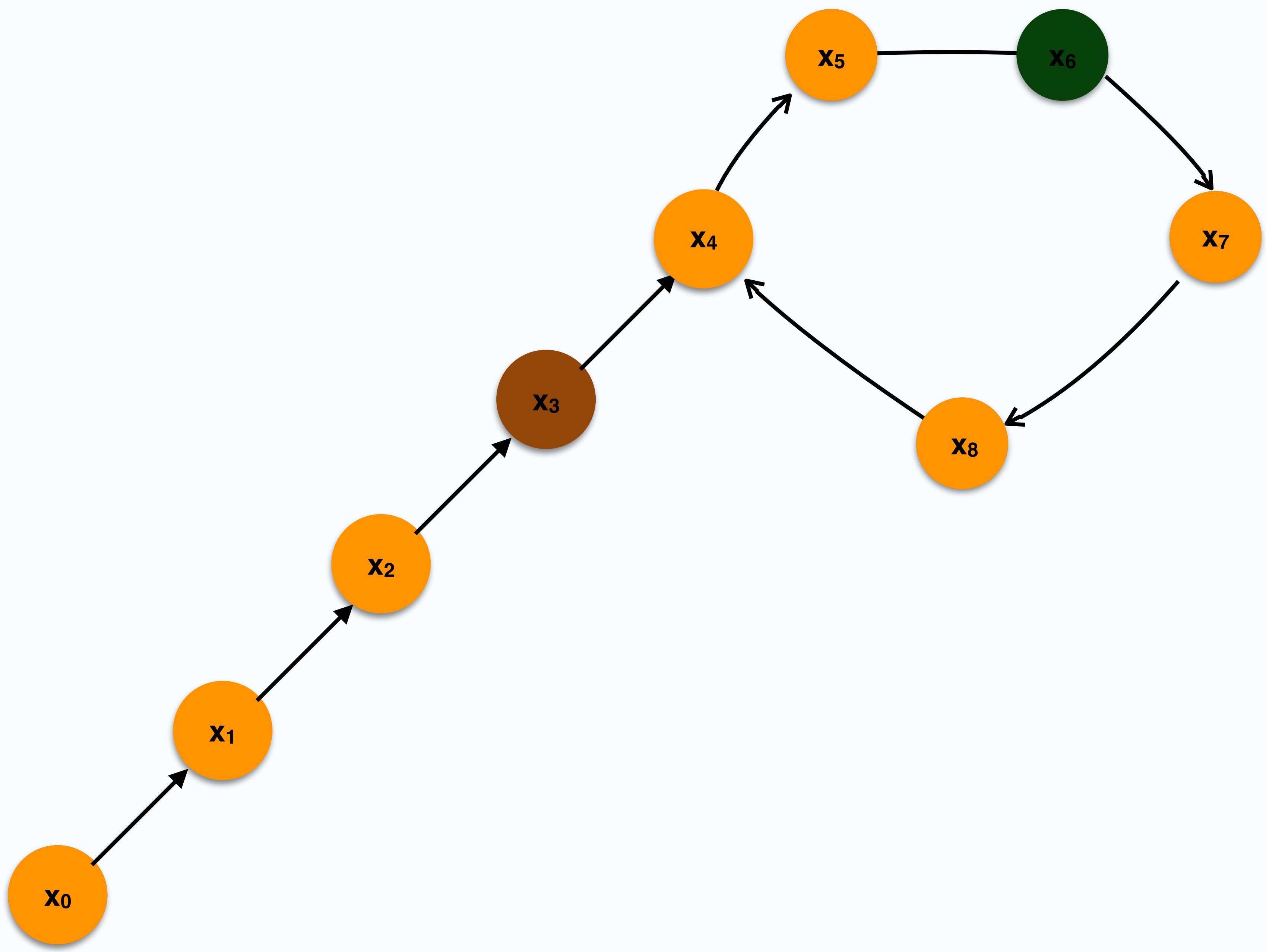
Pollard Rho Method
Detection - 1



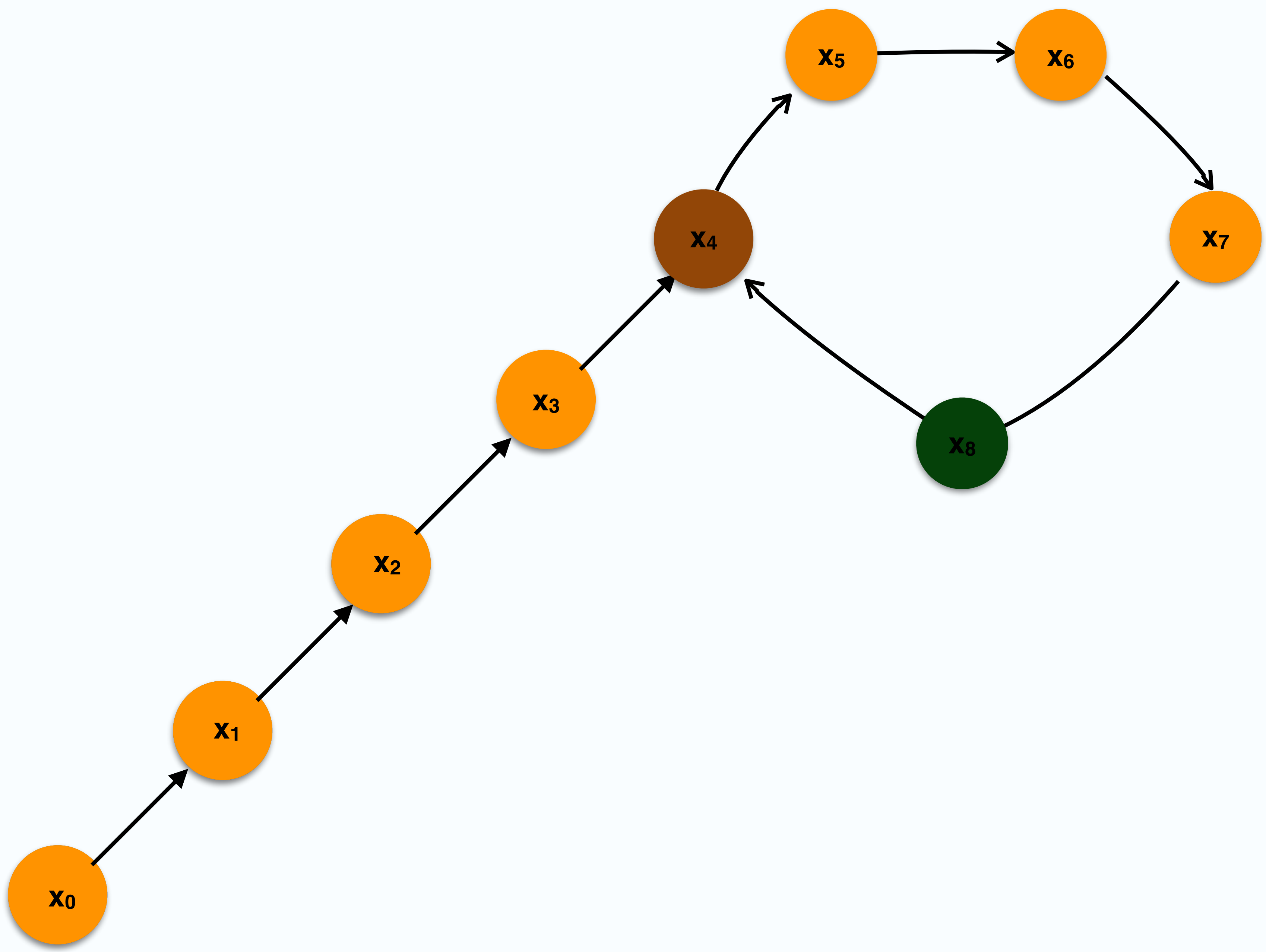
Pollard Rho Method
Detection - 2



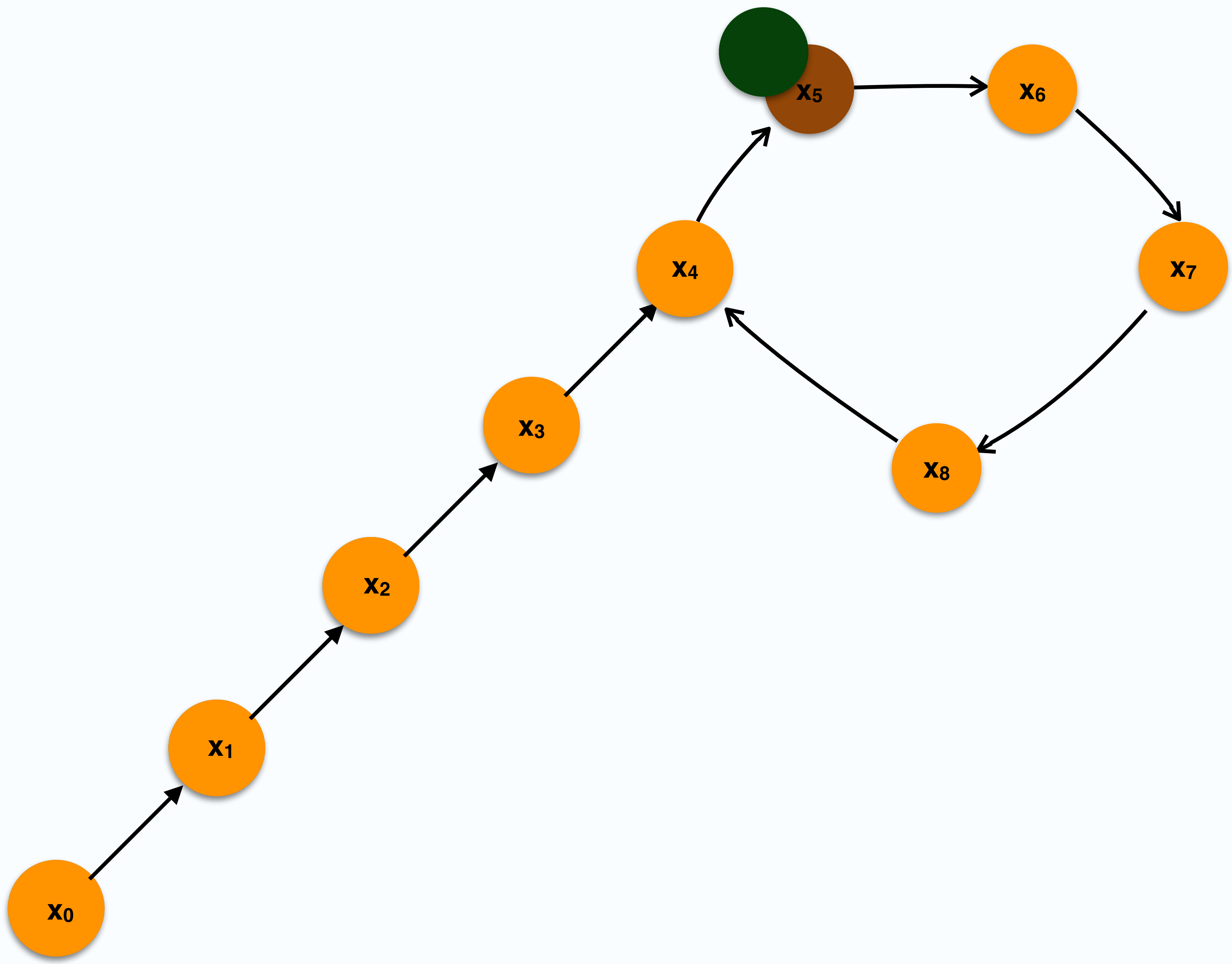
Pollard Rho Method
Detection - 3



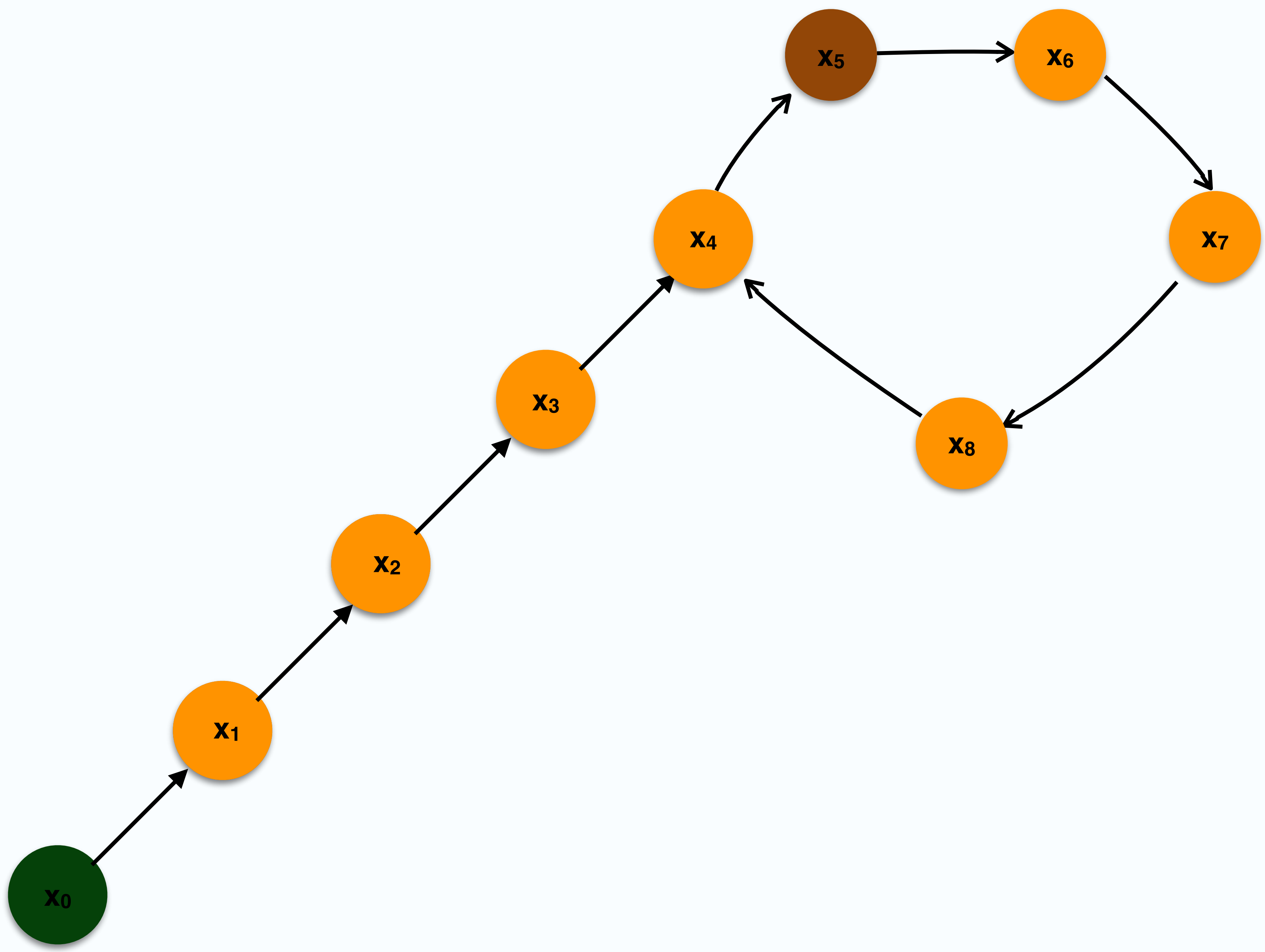
Pollard Rho Method
Detection - 4



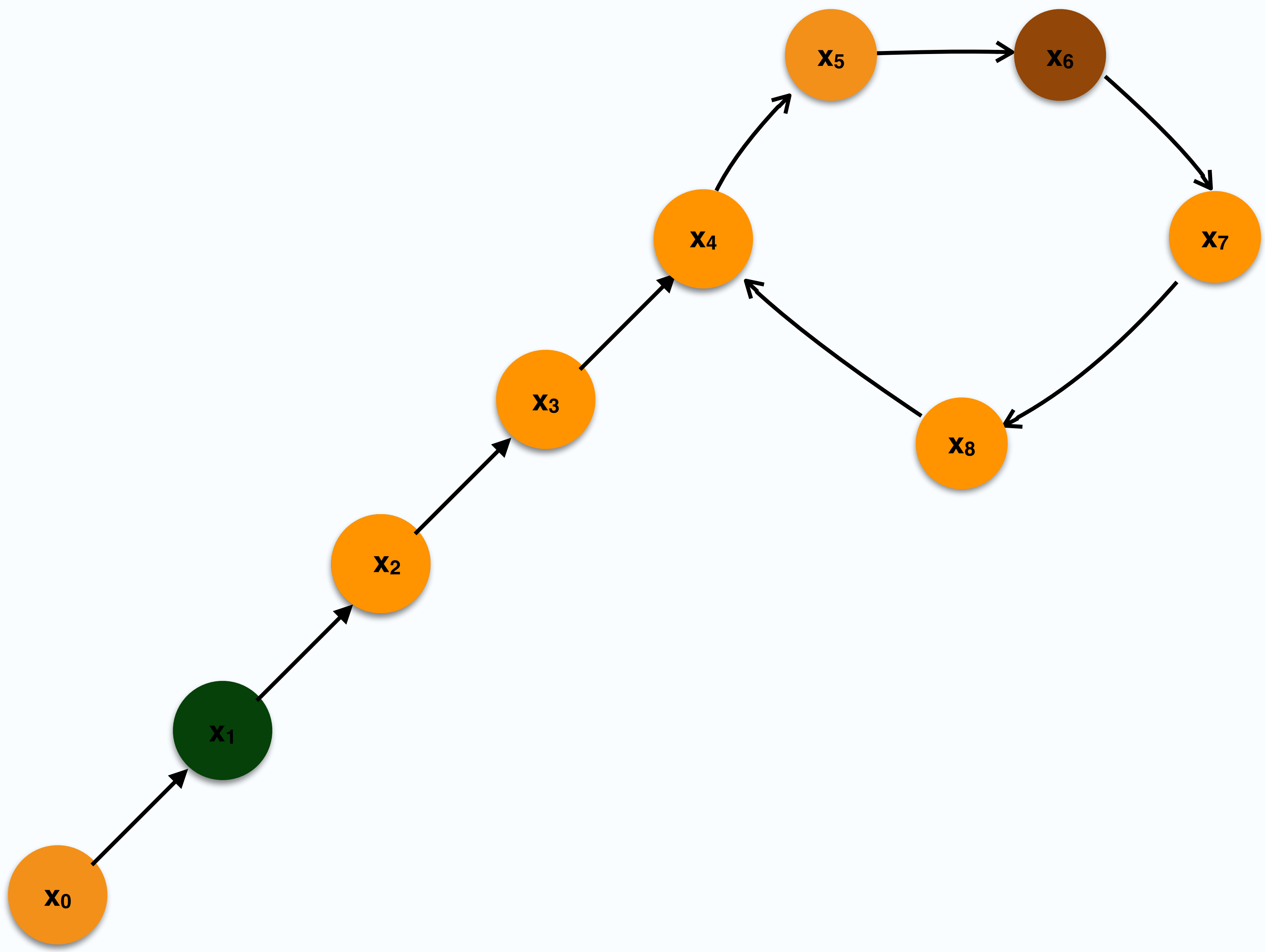
Pollard Rho Method
Detection - 5



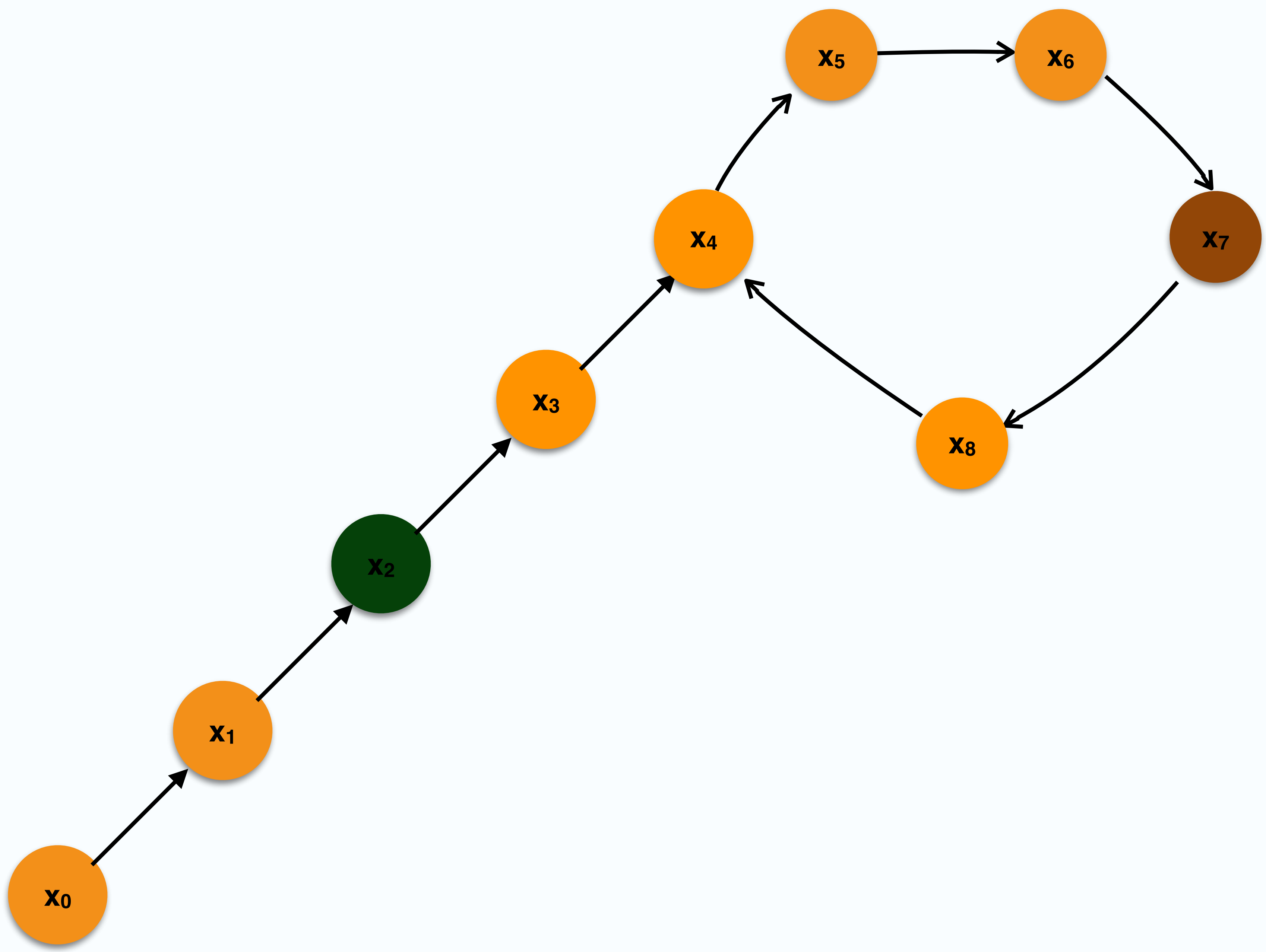
Pollard Rho Method
Locating - 0



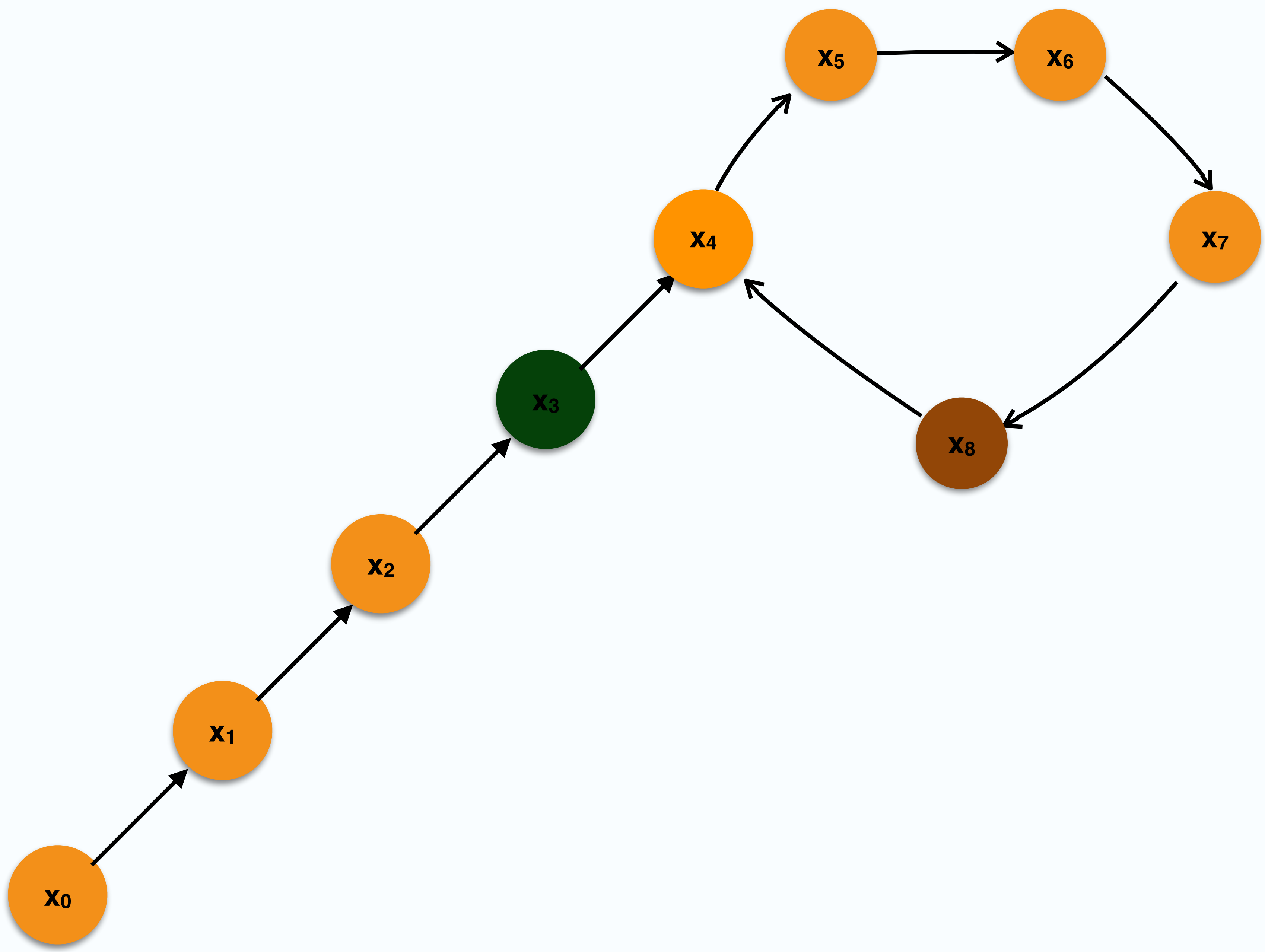
Pollard Rho Method
Locating - 1



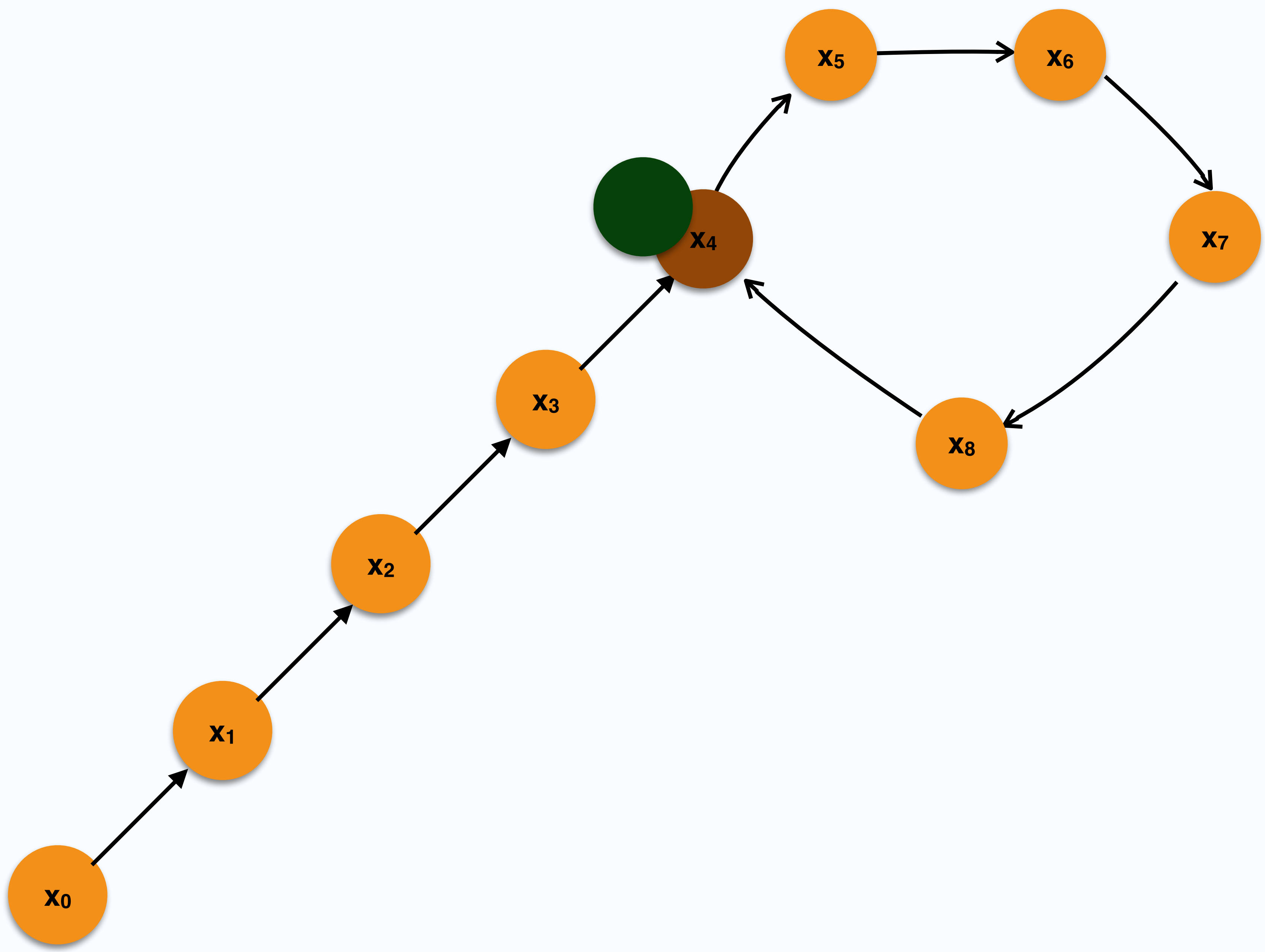
Pollard Rho Method
Locating - 2



Pollard Rho Method
Locating - 3



Pollard Rho Method
Locating - 4



Pollard Rho Method

- Pollard Rho Method detects and finds collisions in time $O(2^{l/2})$ and memory complexity $O(1)$, i.e., removes the memory requirement from the original birthday attacks.

- Remarks:

    - cycle-length: number of nodes in the cycle
    - height: number of steps away from the cycle

Functional Graph

$f : N \longrightarrow N$ is a random function

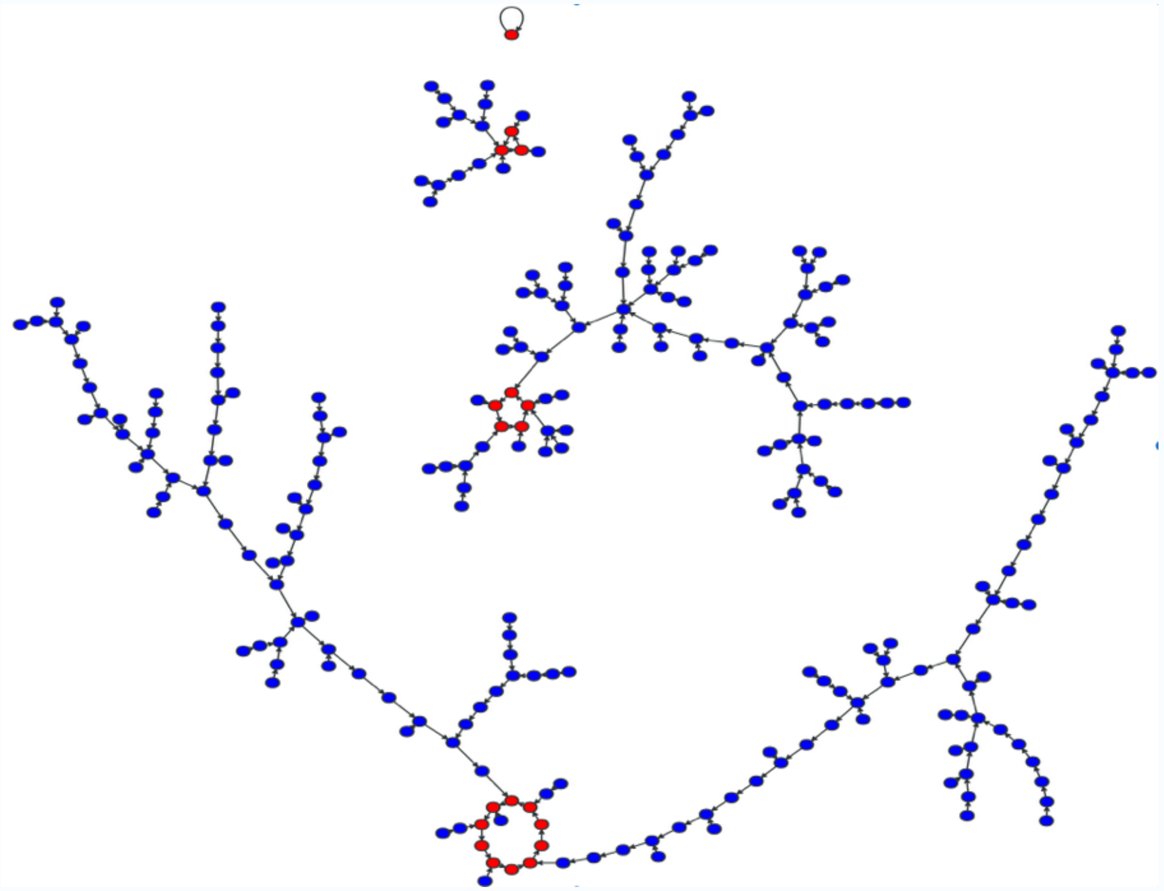
Trail Length (λ) : $\sqrt{\pi N/8}$

Cycle Length (μ) : $\sqrt{\pi N/8}$

Rho Length ($\rho = \lambda + \mu$) : $\sqrt{\pi N/2}$

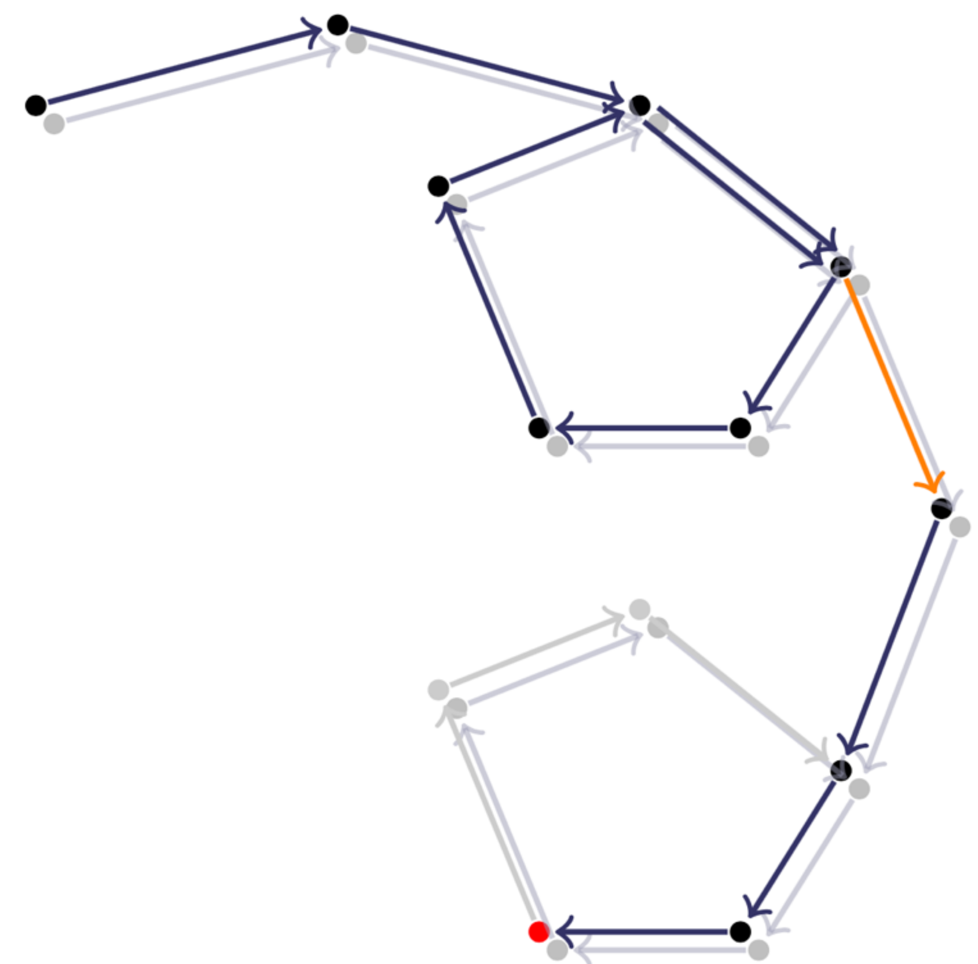
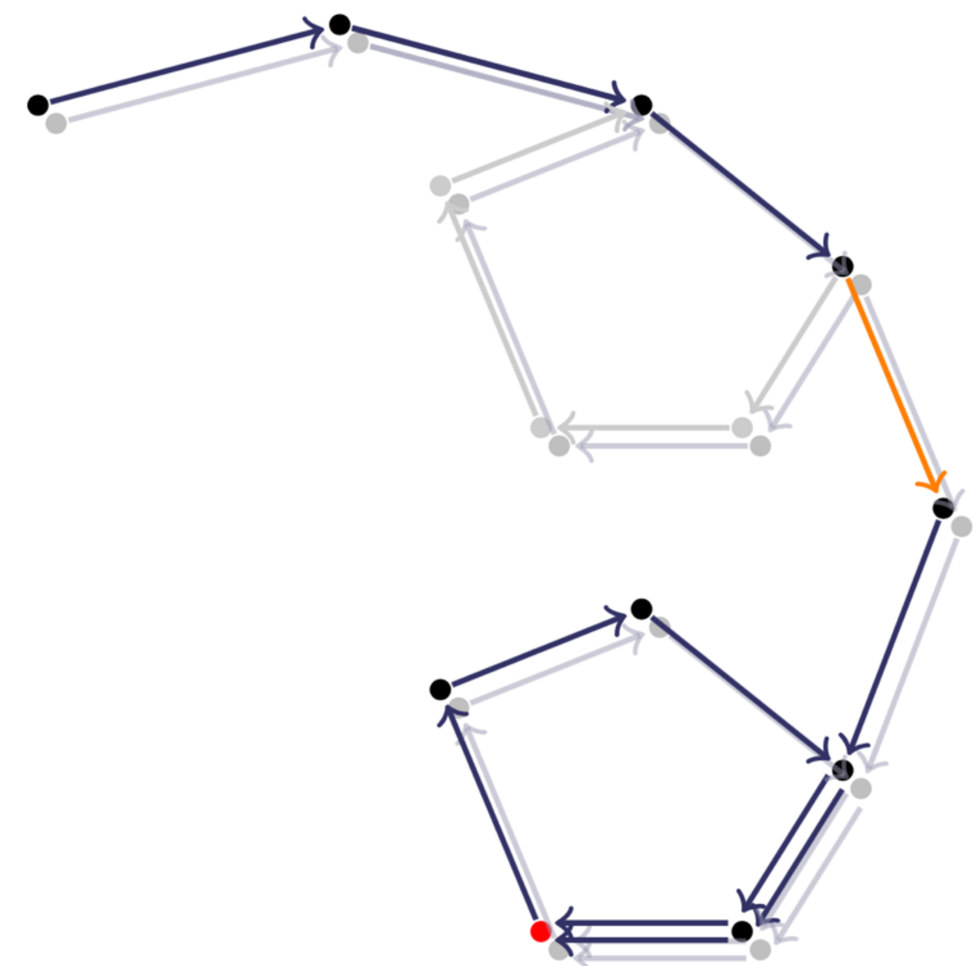
Tree Size : $N/3$

Component Size : $2N/3$



NANYANG TECHNOLOGICAL UNIVERSITY

HMAC: Existential Forgery

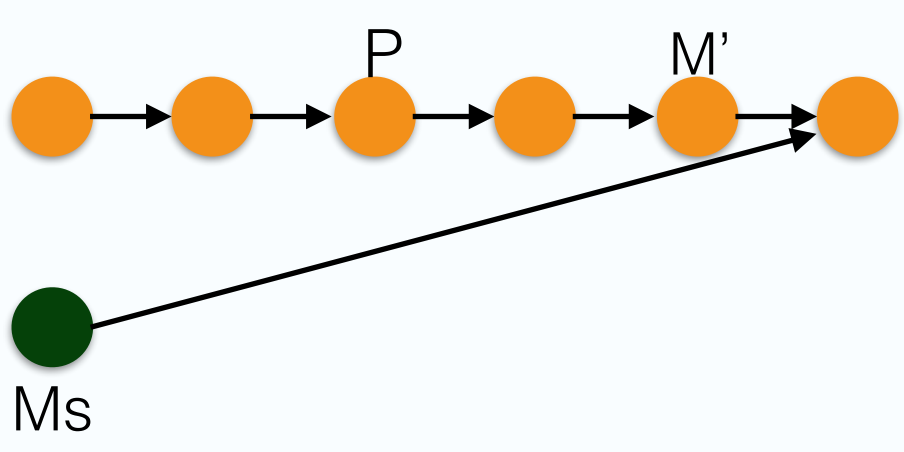
$M_1 = r \,\|\, [0]^{2^{l/2}+L} \,\|\, [1] \,\|\, [0]^{2^{l/2}}$ $\qquad$ $M_2 = r \,\|\, [0]^{2^{l/2}} \,\|\, [1] \,\|\, [0]^{2^{l/2}+L}$

- It is likely both cycles are the cycle of the largest component.
  L is the cycle length of the largest component.

HMAC: State Recovery

- Test for the smallest X (by a binary division approach) such that:
  $M_1 = r \,||\, [0]^{X+L} \,||\, [1] \,||\, [0]^{2^{l/2}}$
  $M_2 = r \,||\, [0]^{X+0} \,||\, [1] \,||\, [0]^{2^{l/2}+L}$
  collide in tag, then the internal state value after proceeding P = r || $[0]^X$ is the root of the largest tree, X is the <u>height</u> of state after processing [r].

- Test tag collision between P || [M'] and [M_S] for one-block M' and M_S to **recover state for short message**, by testing enough M' and M_S pairs - unbalanced MITM.



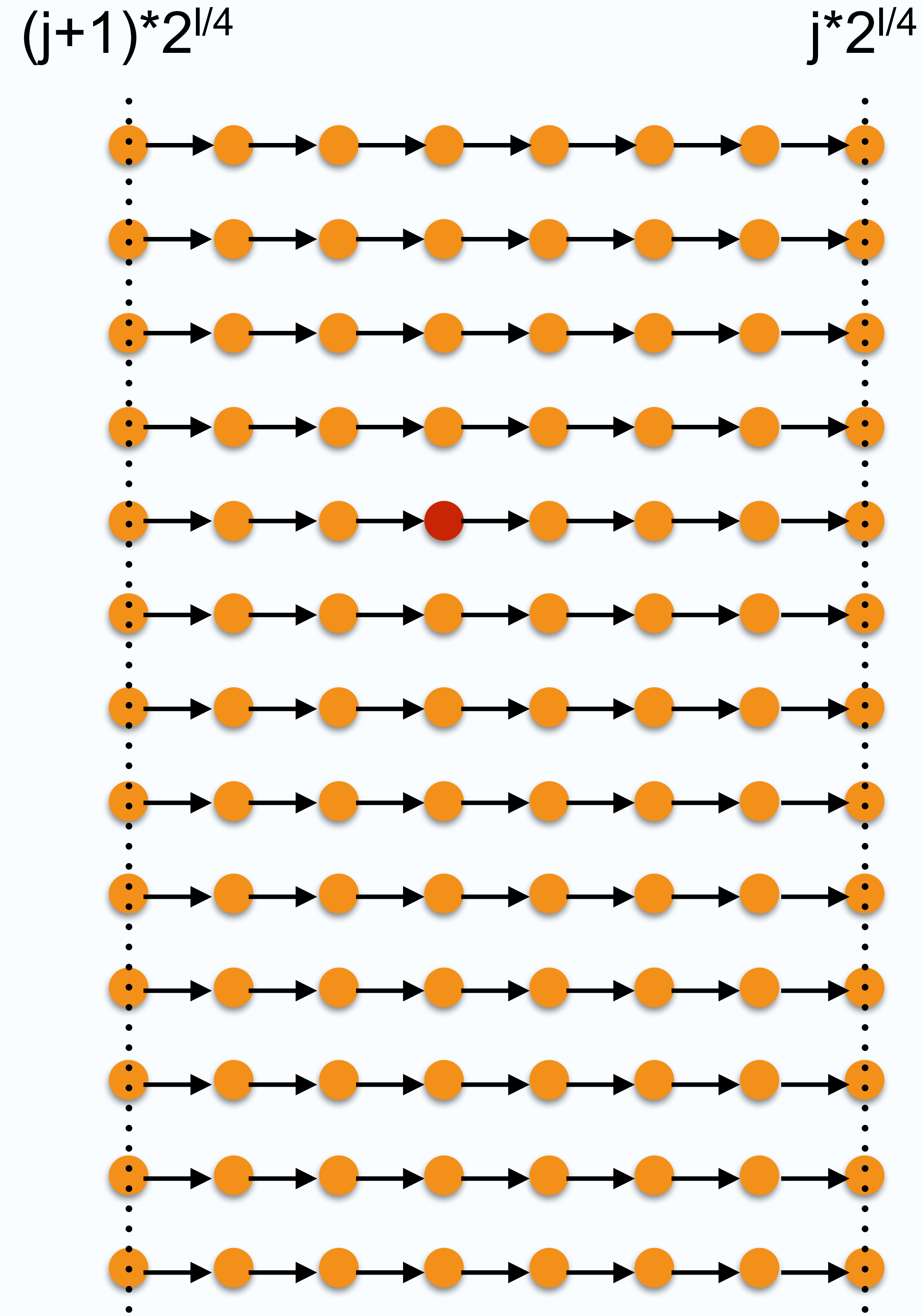
NANYANG TECHNOLOGICAL UNIVERSITY

HMAC: Universal Forgery

1. Offline phase: precompute nodes with heights multiple of $2^{l/4}$, and find the sets S_1, S_2, …, $S_{2^{l/4}}$ with each S_i containing at least $i*2^{l/4}$ nodes of height $2^{l/4}$.

2. Online phase: given a message [M], recover its height h in functional graph $[j*2^{l/4}, (j+1)\, 2^{l/4})$, compute the state value for message $x \,||\, [0]^{h-j*2^{l/4}}$ for all x from S_{j+1}, check if it is indeed the state for [M].

3. Time complexity $2^{3l/4}$ for a given message of $2^{l/4}$ blocks.



HMAC: Key Recovery

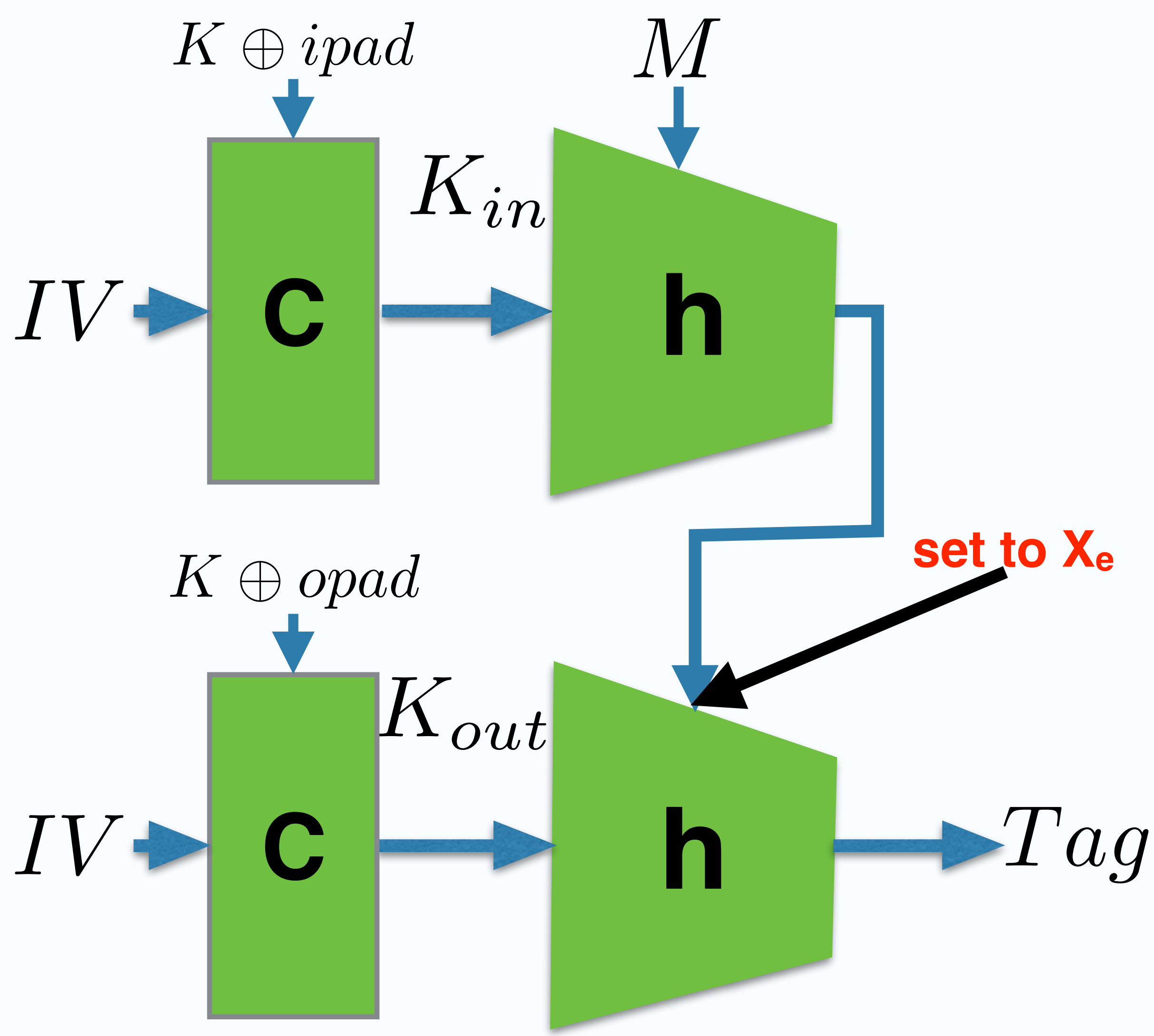
- The key recovery attack complexity is no longer bounded by the key size, but the internal state size. Note HMAC accepts key size of arbitrary long.
- With 2^l pre-computation, K_{in} and K_{out} can be recovered in $2^{3l/4}$.

HMAC: Key Recovery

1. set input to outer layer to constant X_e, apply Hellman's trade-off to recover K_{out}

2. recover the height of K_{in}, the value as before.

3. X_e can be reached by herding techniques.



HMAC: Other Results

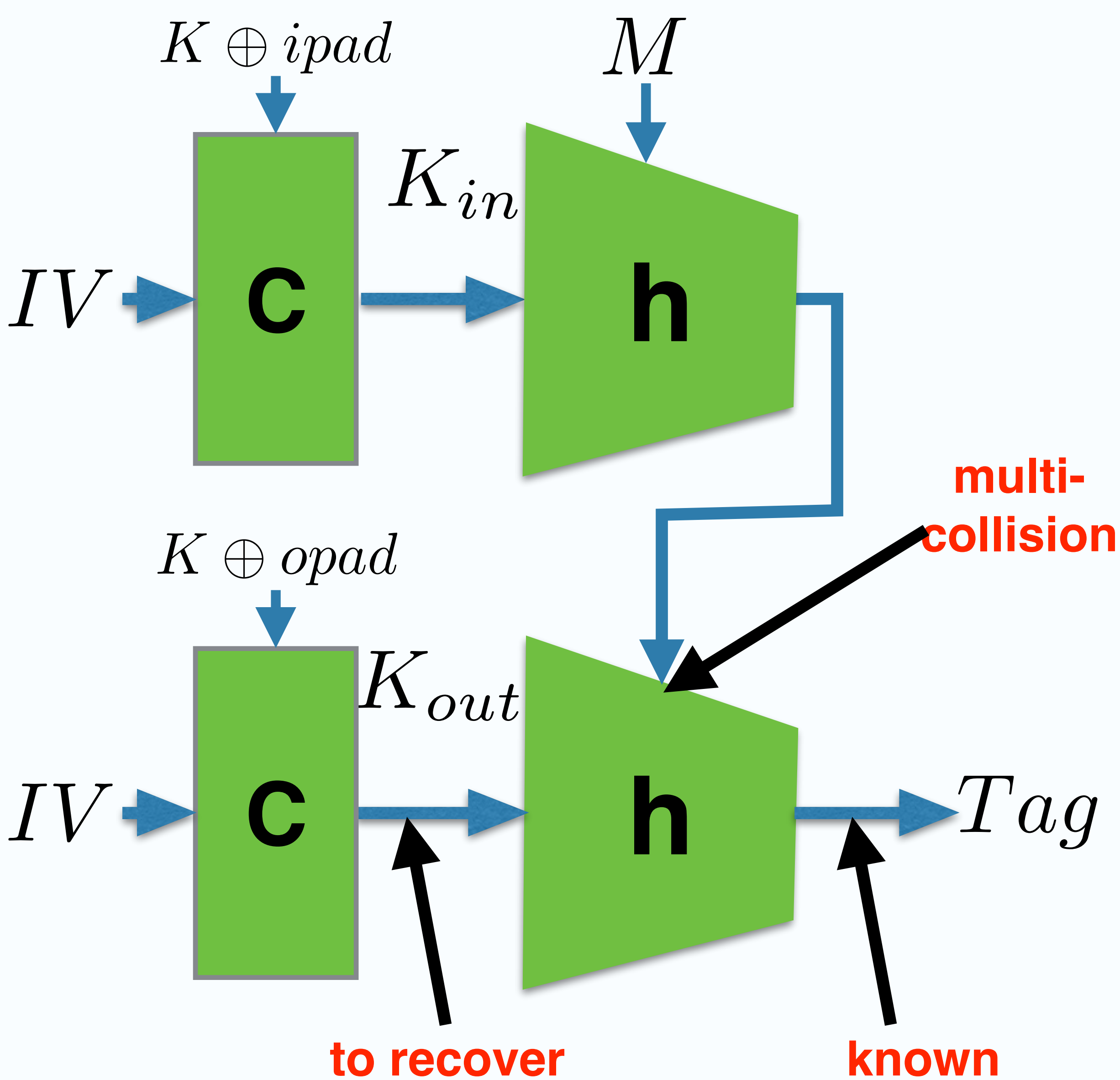
1. State recovery and universal forgery for short messages

2. Selective forgery applicable to HMAC based on many hash function standards

3. Improved applications to HMAC-Whirlpool from key recovery for 6 rounds to 7-round equivalent-keys recovery.

6-round HMAC-Whirlpool

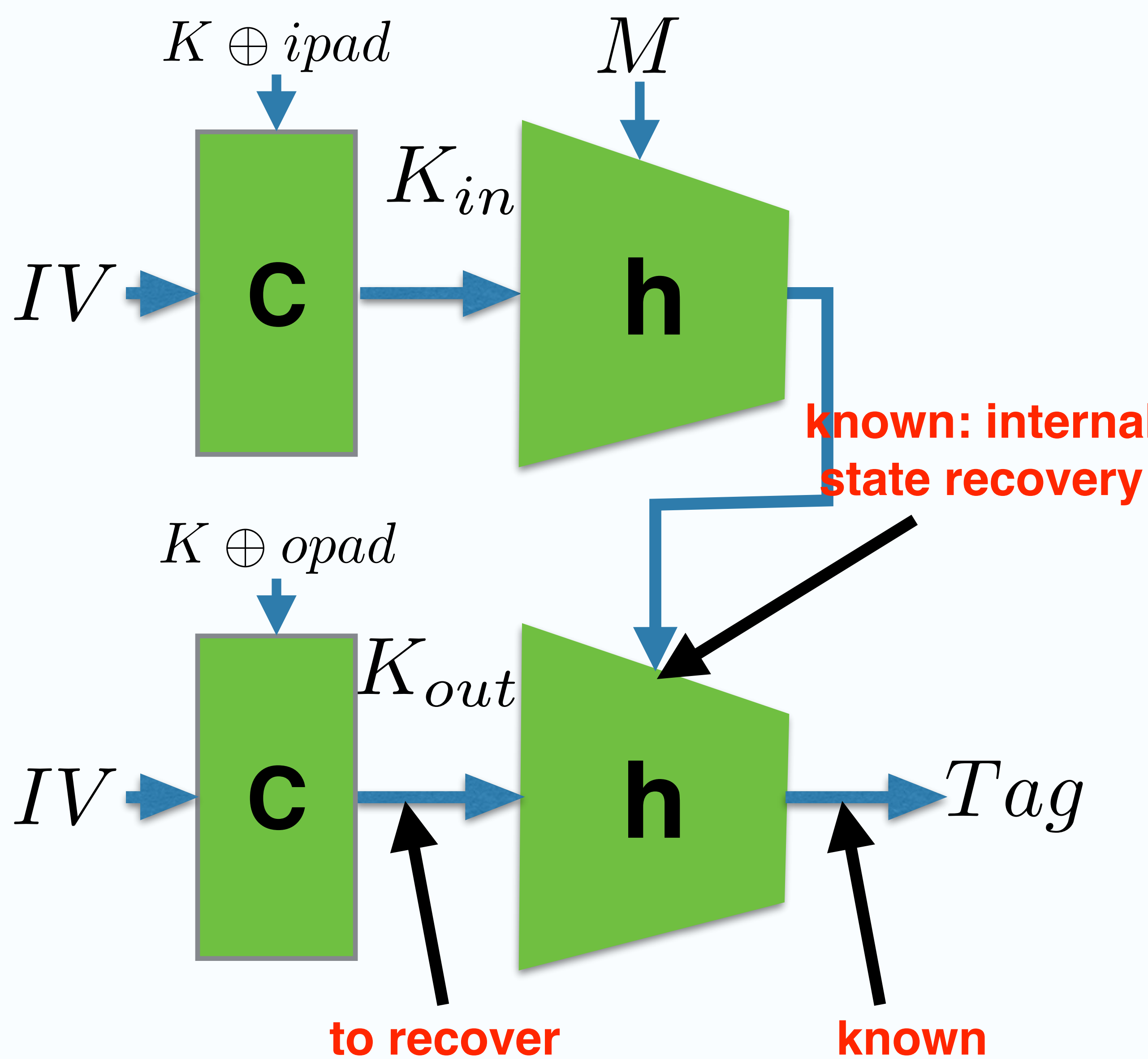
- (multi-)collision in inner layer
- recover K_{out},
- recover K from K_{out} using preimage attack techniques



NANYANG TECHNOLOGICAL UNIVERSITY

7-round HMAC-Whirlpool

- known message block to outer layer
- output is known as before
- recover K_{out}
- failed to recover K itself because there is no 7-round preimage attack in this setting yet.



NANYANG TECHNOLOGICAL UNIVERSITY

Open Problems

1. How to tweak HMAC to achieve n-bit security ? Or is it even possible to have n-bit security ?

2. Is the birthday-bound tight for HMAC? I.e., Are there generic forgery and key recovery attacks with birthday complexities ?

3. Are these techniques useful for block-cipher based and dedicated MAC designs ?

Thank you !