A better write-only oblivious RAM

Daniel S. Roche Adam Aviv Seung Geol Choi Travis Mayberry

Computer Science Department
United States Naval Academy
Annapolis, Maryland, USA

LJK CASYS Seminar
Grenoble, France
21 September 2017

Problem: Can't trust your storage

- Want to store data on untrusted mediums
  e.g. Google cloud or an easily-stolen laptop
- Plain encryption is not enough
  Must protect access patterns as well as data
- Oblivious RAM is a solution
  Provides provable security, but is it fast enough?

Example: Cloud file storage

Storing data remotely is convenient and inexpensive.

Unfortunately, these services are not always the best for privacy.

- 2012: Google Drive terms of service states it can use your content in advertising
- 2014: Private photos from ≈ 500 Apple iCloud accounts published online
- 2016: Dropbox reveals a hack from 2012 actually revealed password hashes of millions of users

Example: Mobile devices

Most travellers carry a laptop and/or smartphone with sensitive data.

- 70 million smartphones stolen per year, only 7% recovered

  (source: Kensington)
- Border agents can legally clone your laptop hard drive

Encryption is not enough

End-to-end encryption of files is great, but does not protect metadata: which files are accessed, when, and by whom

- "*The public doesn't understand. [Metadata] is much more intrusive than content. . . If you can track that, you know exactly what is happening — you don't need the content.*"
  (mathematician and policy expert Susan Landau)
- "*We kill people based on metadata.*"
  (former NSA and CIA director Michael Hayden)

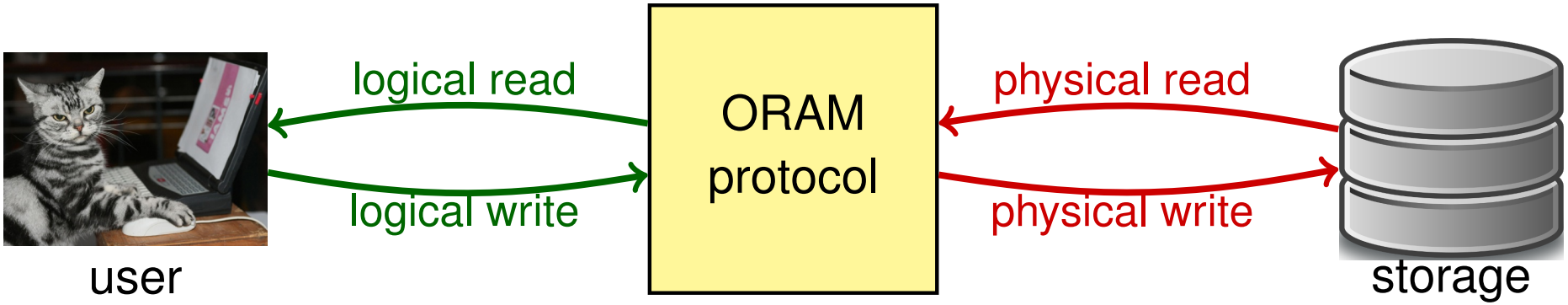
Talk outline: Four ORAMs

1. "Square Root" ORAM (Goldreich & Ostrovsky '96)
2. Path ORAM (Stefanov, Shi et al '12)
3. Write-Only ORAM (Blass, Mayberry et al '14)
4. Deterministic, stash-free write-only ORAM ('17)

Goal: Reducing overhead without compromising privacy (too much)

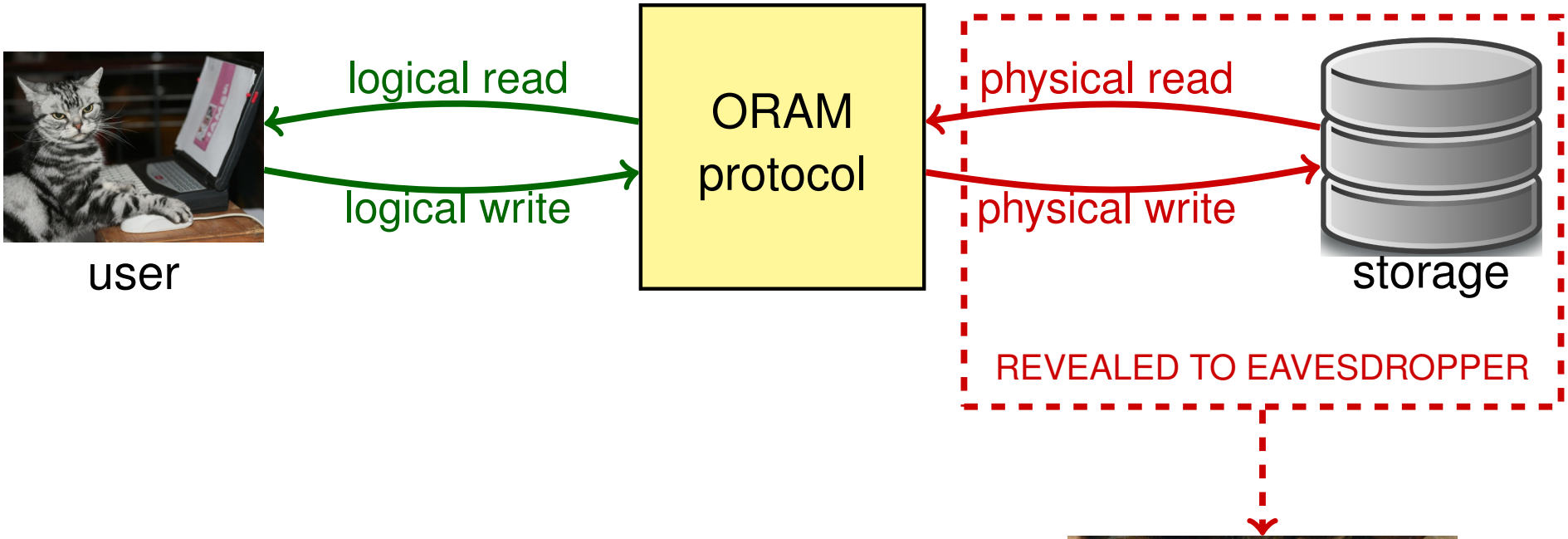
ORAM Setting

Oblivious RAM is a translator from logical to physical read/write operations.



ORAM Setting

Oblivious RAM is a translator from logical to physical read/write operations.



Goal: Eavesdropper should learn only the number of physical operations performed (not contents, op types, or addresses)

ORAM Security Definition

Formally, an ORAM is secure if any two sequences with the same number of operations are indistinguishable based on their physical reads/writes.

ORAM Security Game

1. Adversary chooses two logical op sequences of the same length
2. Execute one of those sequences with the ORAM
3. Give resulting "transcript" of physical reads and writes to adversary
4. Adversary has to guess which logical sequence was executed

Equivalently, anyone can generate a fake transcript at random, based only on the number of operations.

Square Root ORAM

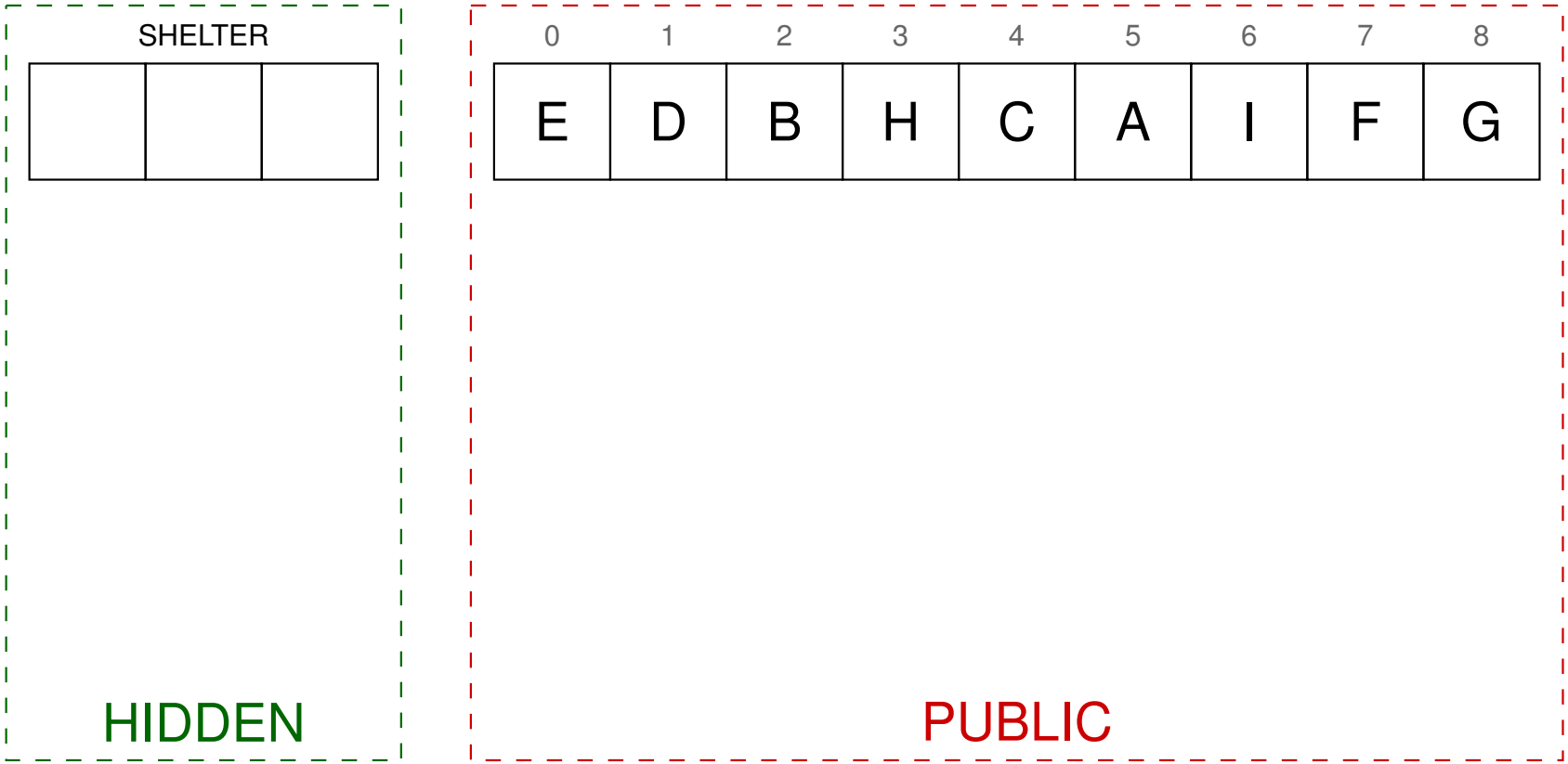
(Goldreich & Ostrovsky, JACM 1996)

Key ideas:

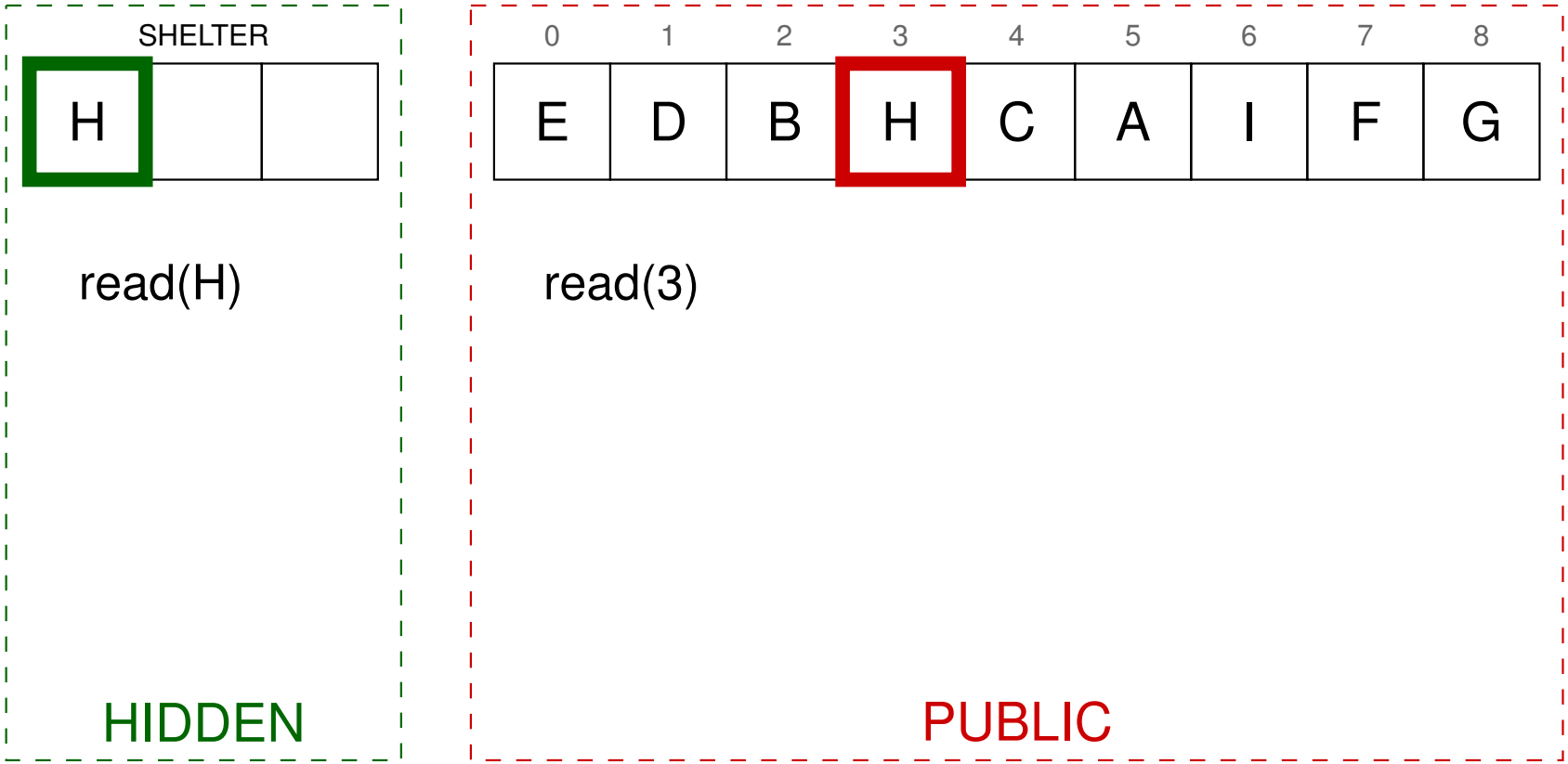
- Store blocks in random locations
- First time accessing a block, copy to a small local "shelter" area
- Repeated access, take from shelter and access something else ("dummy")
- When shelter is full, re-shuffle and incorporate changes

Overhead: Amortized $O(\sqrt{N})$ per operation

Square Root ORAM Example

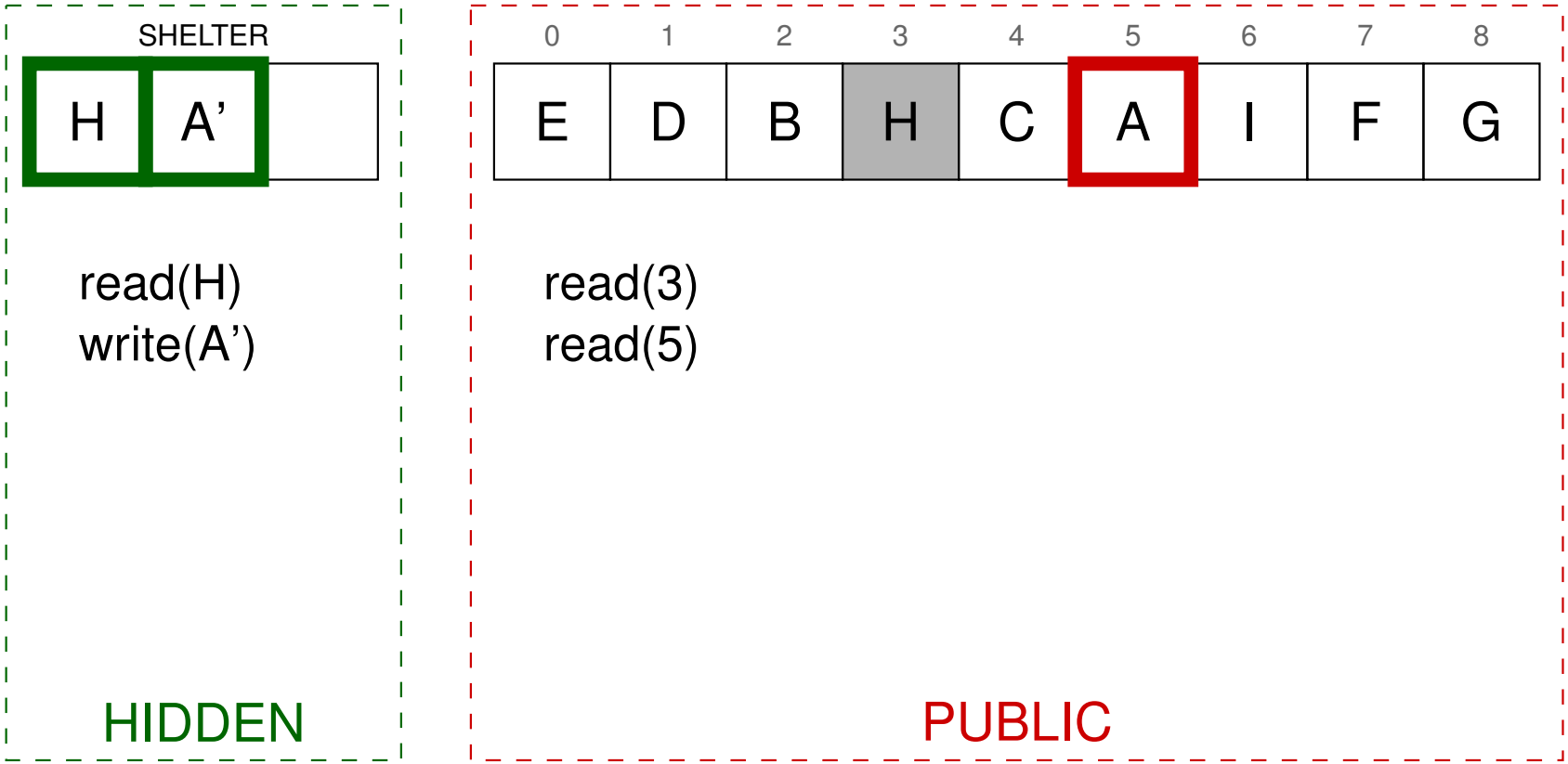


Square Root ORAM Example



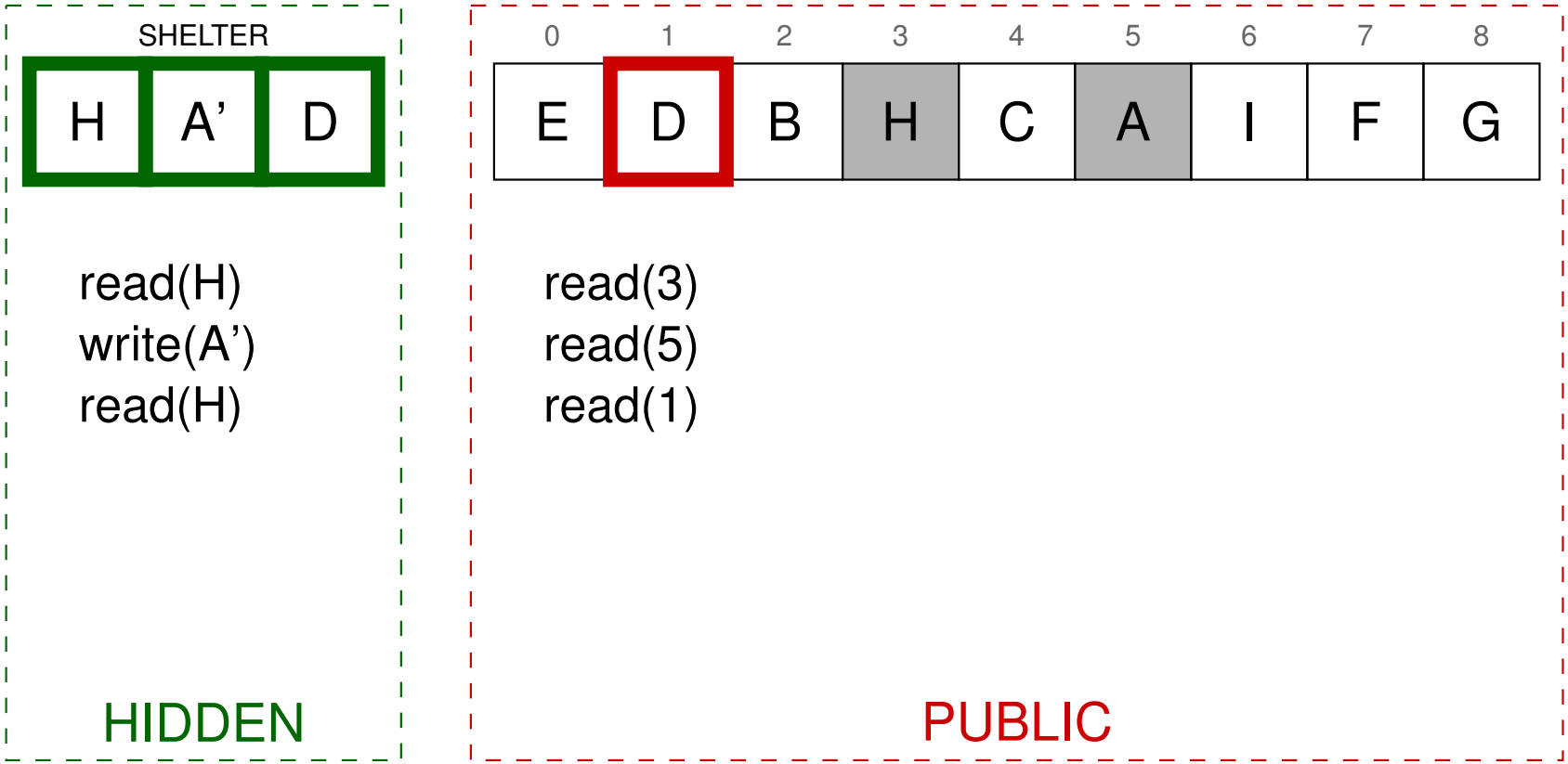
1. Normal read. Result saved to shelter.

Square Root ORAM Example



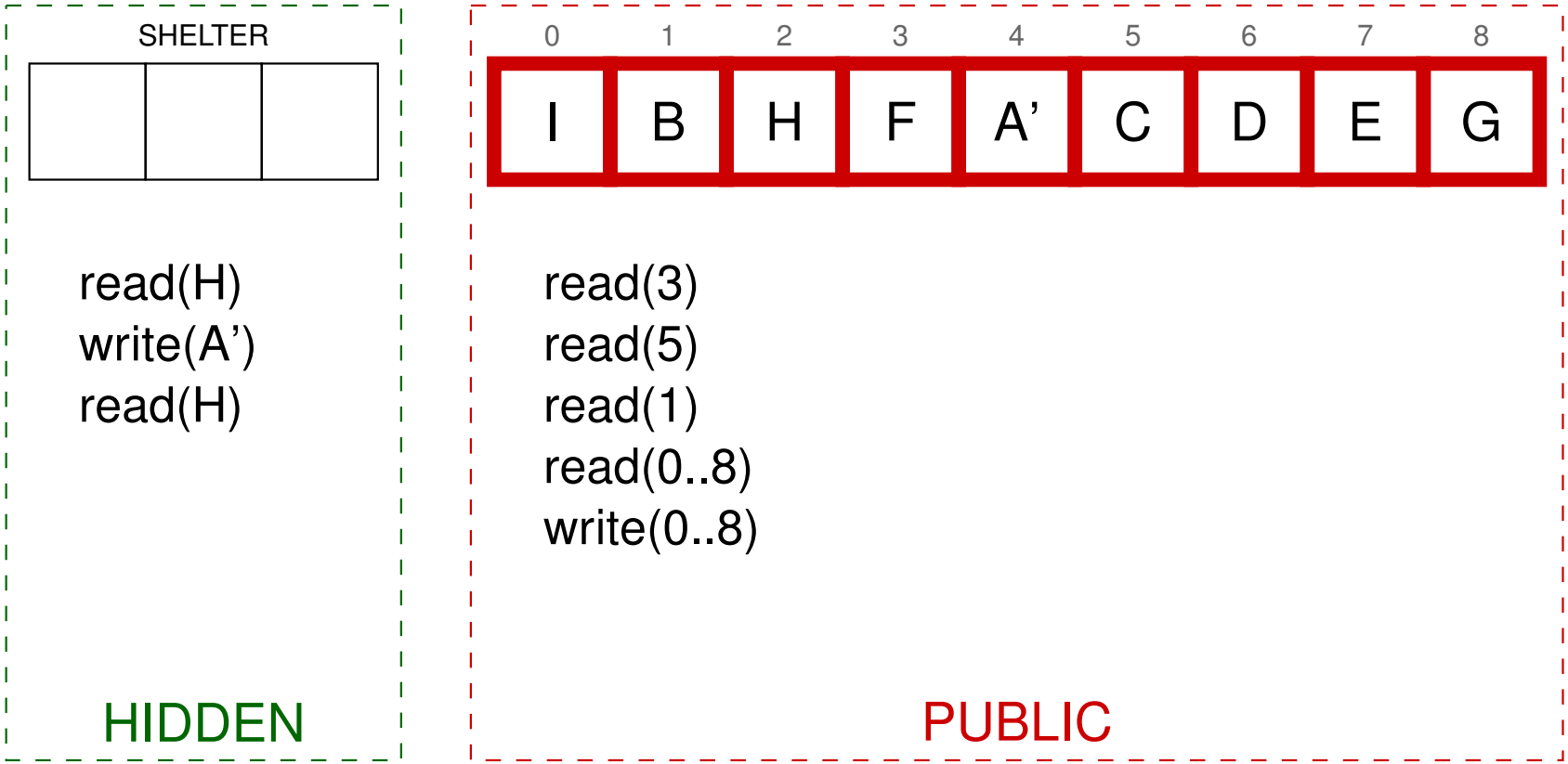
2. Normal write. New value stored in shelter.

Square Root ORAM Example



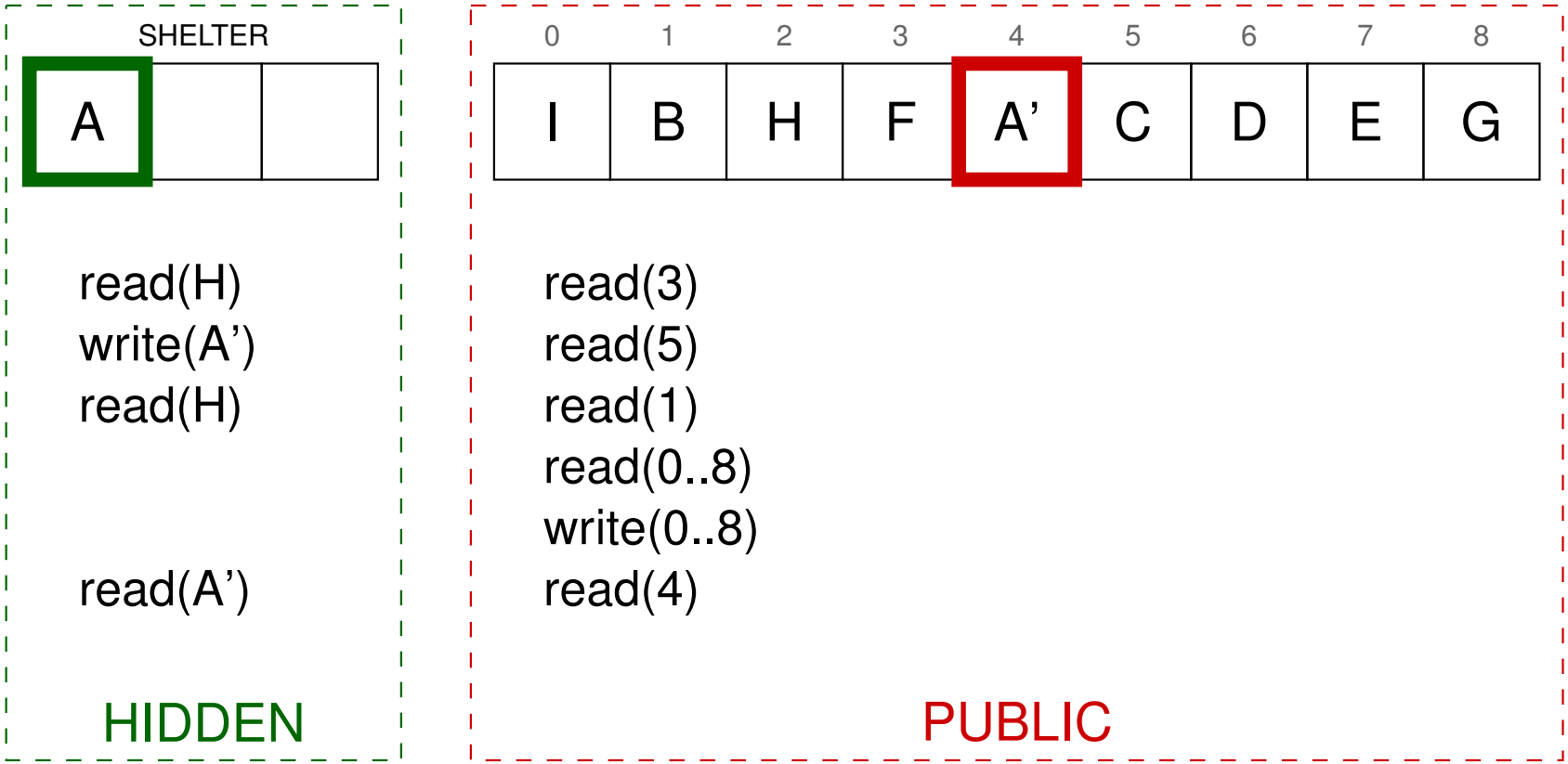
3. Repeat read. Perform dummy read, return value from shelter

Square Root ORAM Example



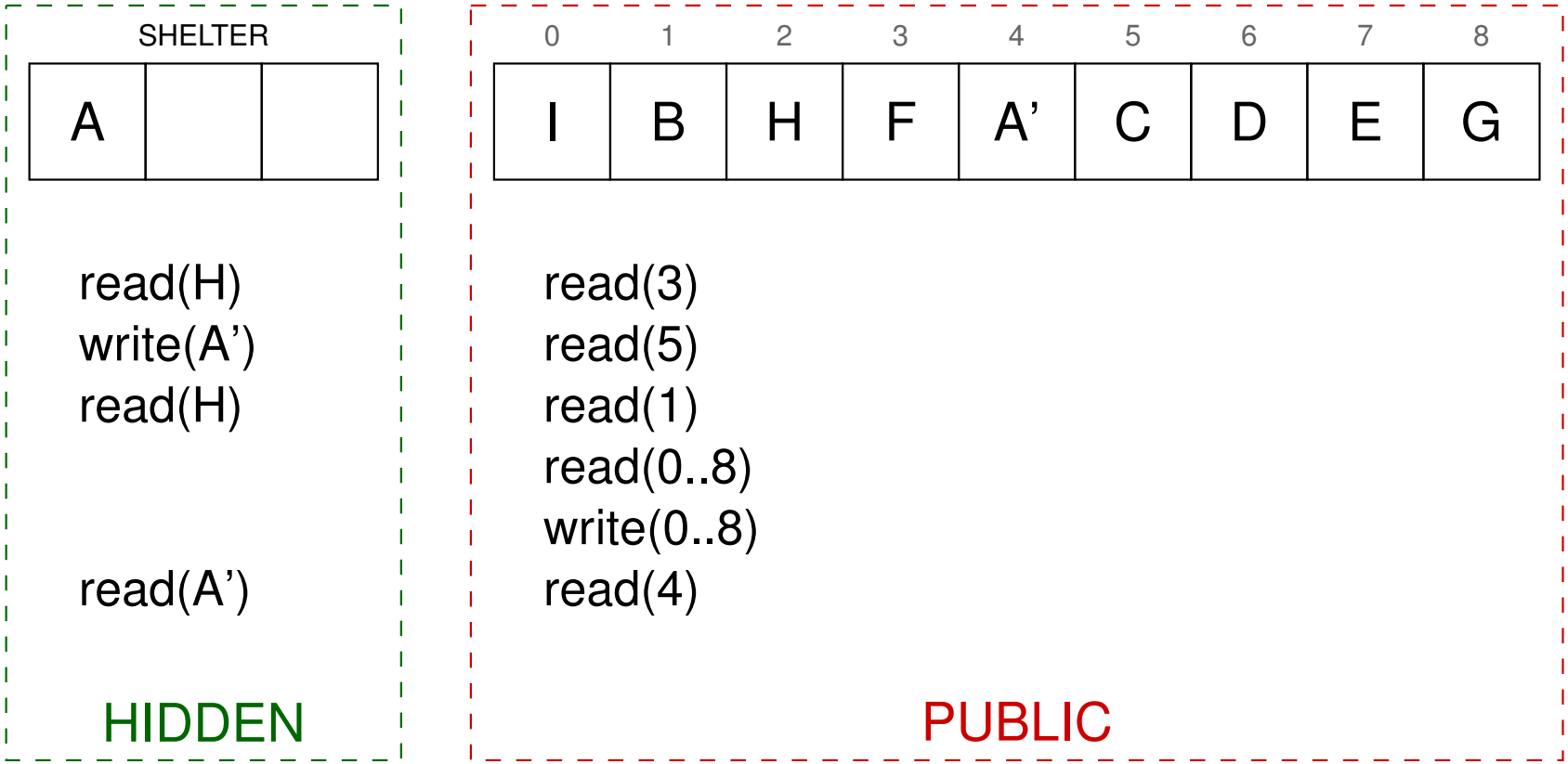
4. Re-shuffle memory and clear shelter.

Square Root ORAM Example



5 Normal read (not repeated; different round).

Square Root ORAM Example



Public transcript reveals nothing about the logical operations.

ORAM Comparison*

	Overhead		Storage	
	Read	Write	Hidden	Public
Trivial	N	N	1	N
Square Root ORAM	$\sqrt{N}$	$\sqrt{N}$	$O(\sqrt{N})$	N

*Not counting the position map

Path ORAM

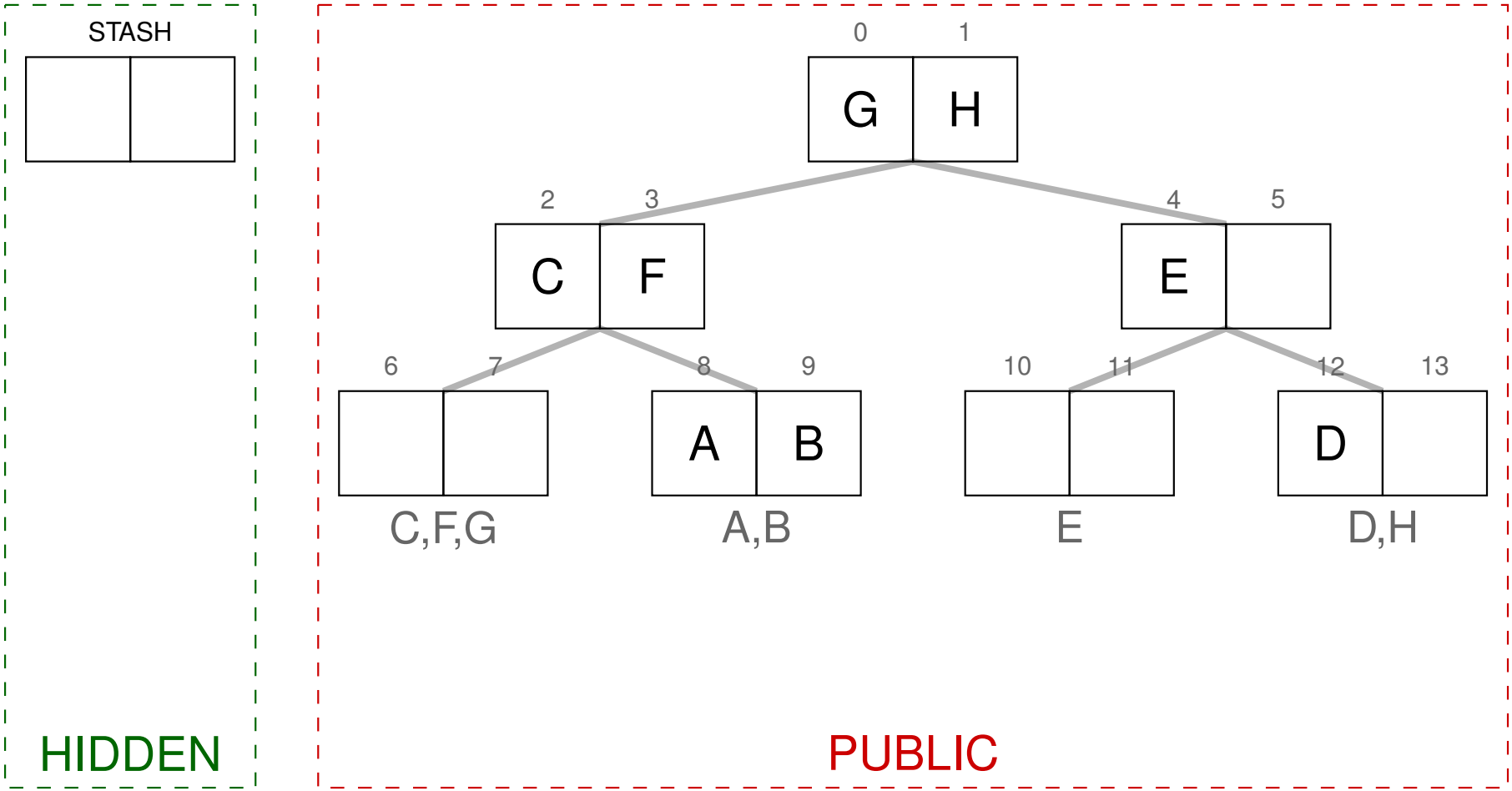
(Stefanov, van Dijk, Shi, Chan, Fletcher, Ren, Yu, and Devadas, CCS'13)

Key ideas:

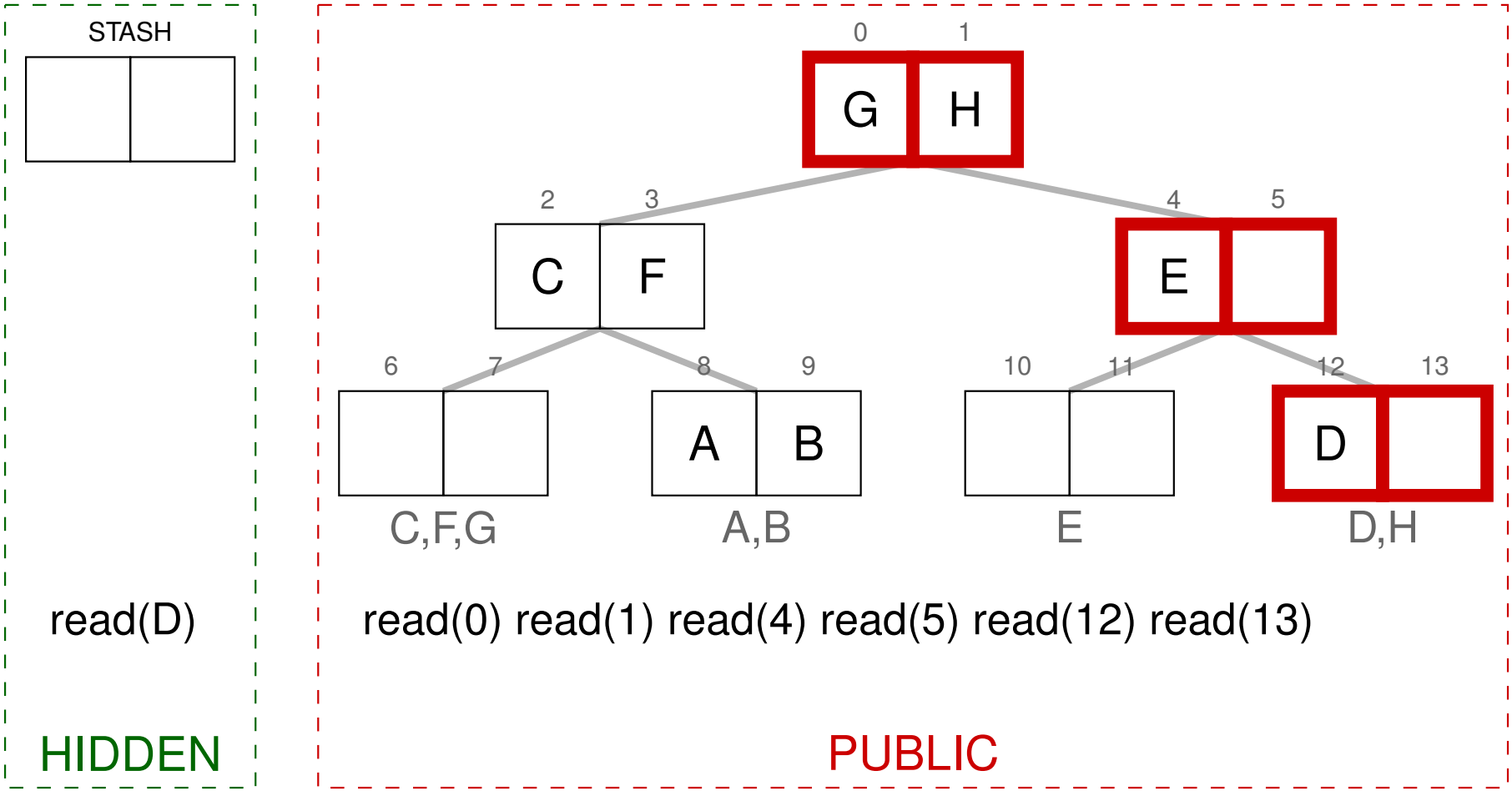
- Arrange storage "buckets" in a binary tree
- Every block is assigned a random tree path
- On every access, read and re-write the entire path

Overhead: $O(\log N)$ per operation

Path ORAM Example

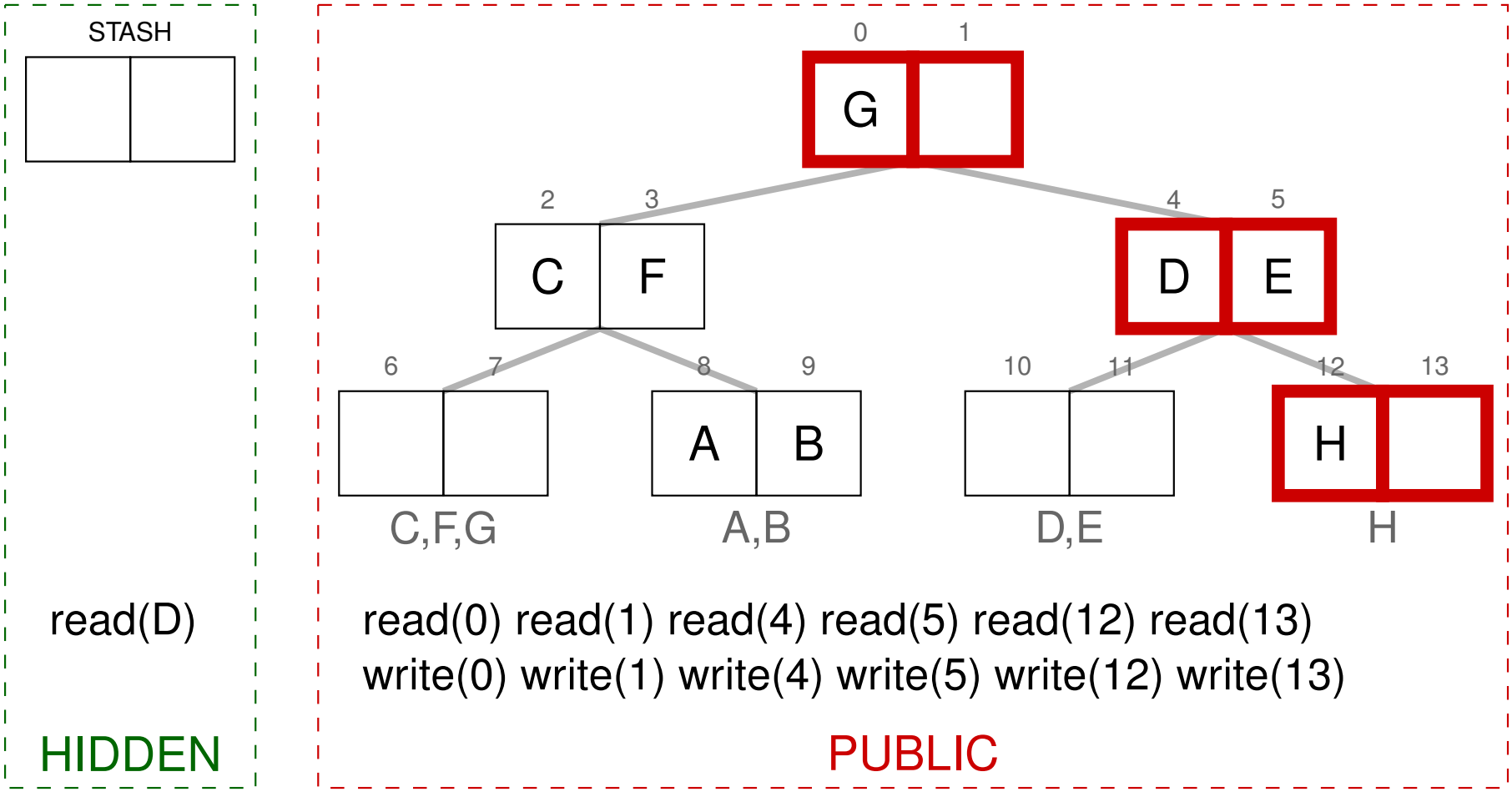


Path ORAM Example



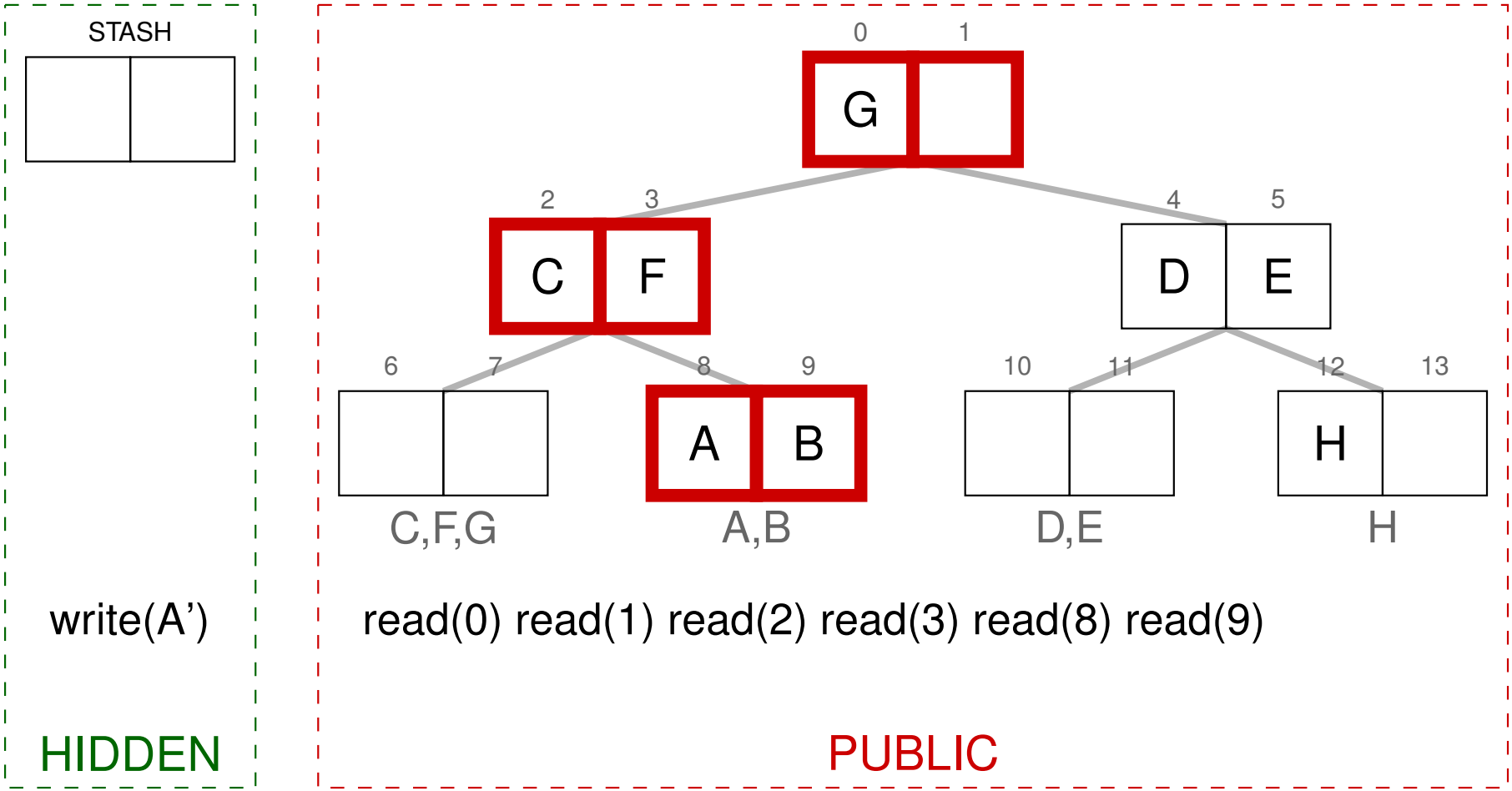
1. Read path to D

Path ORAM Example



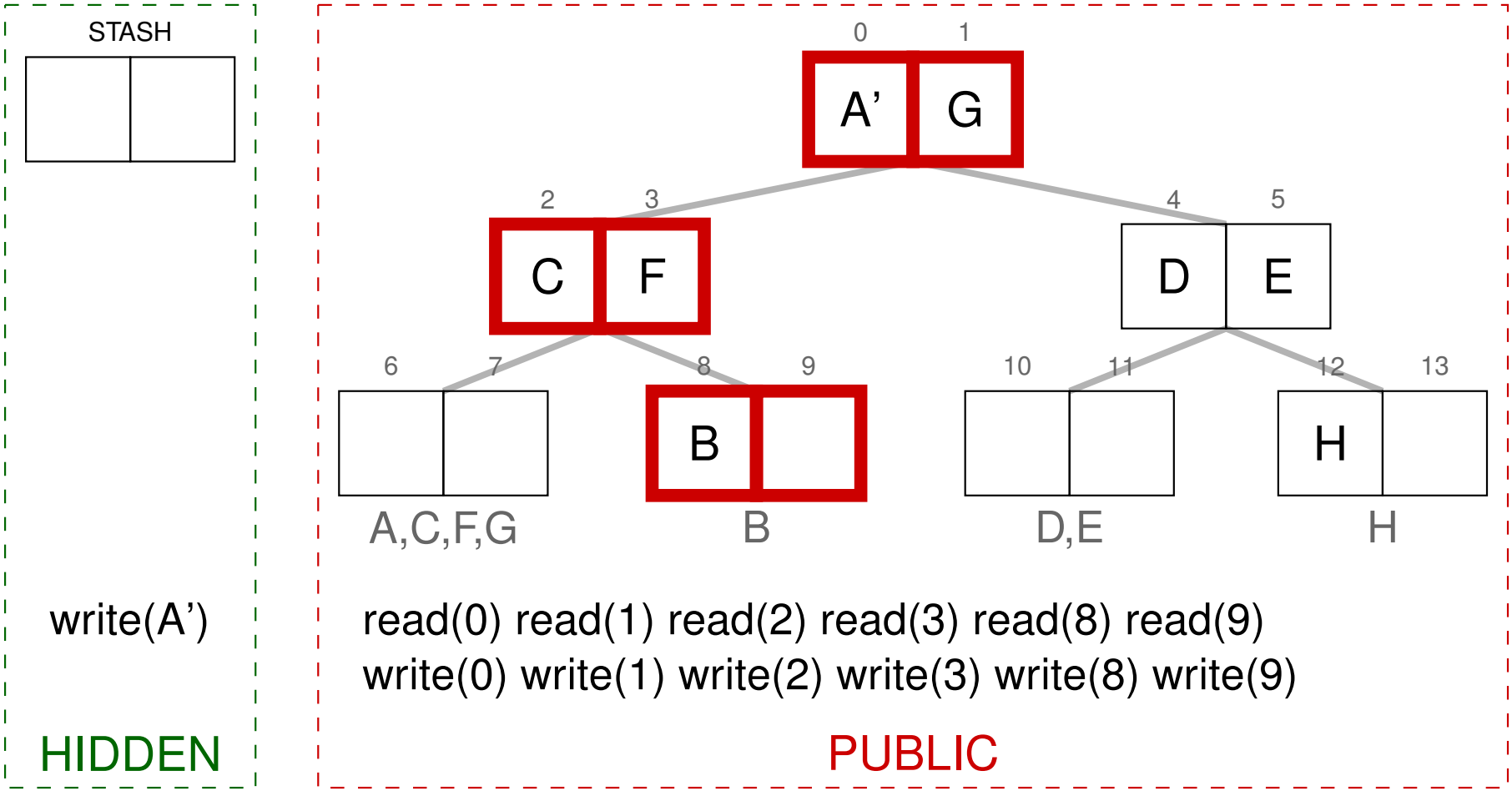
2 Reassign D's leaf and re-write old path

Path ORAM Example



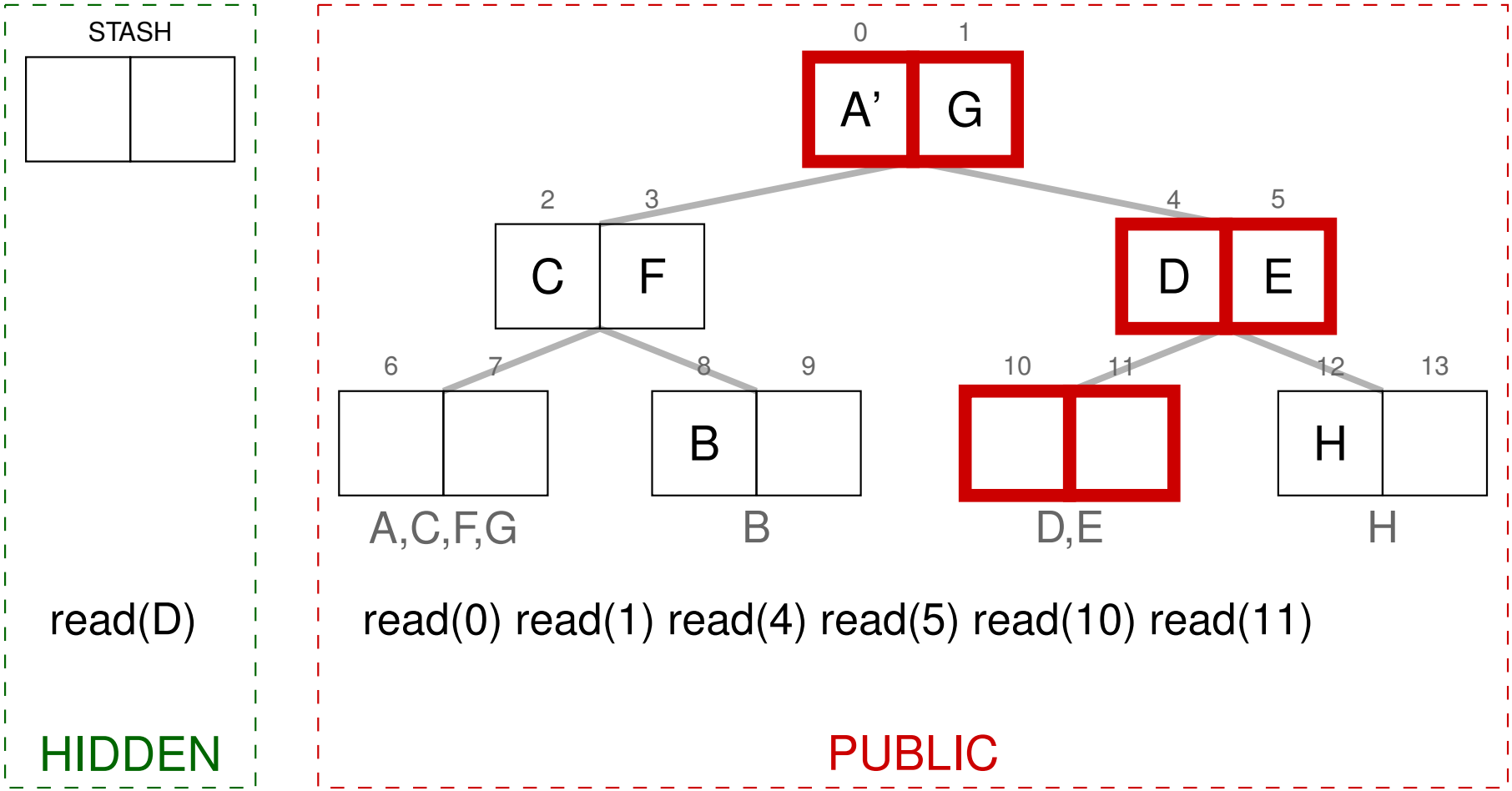
3 Read path to A

Path ORAM Example



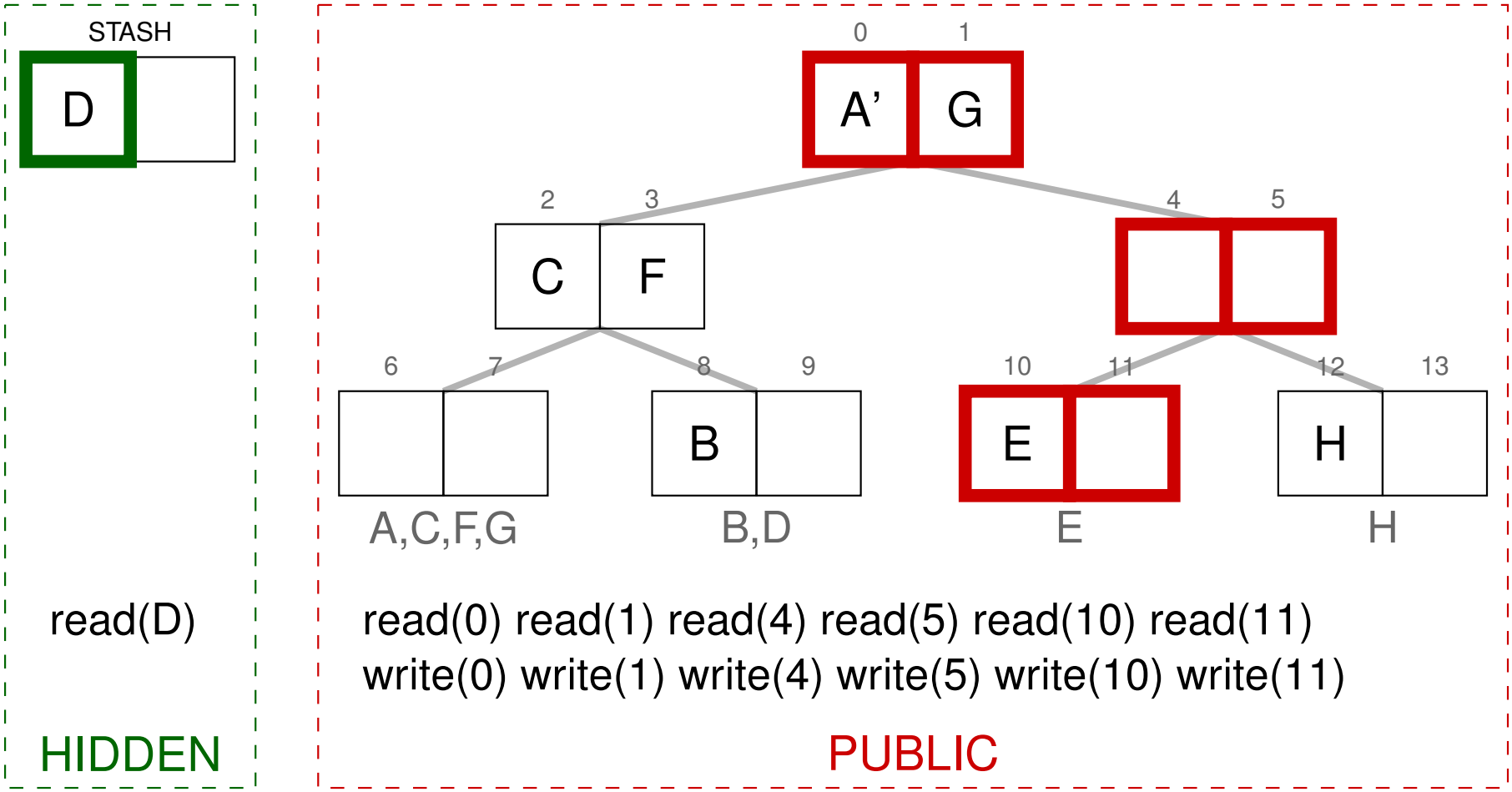
4 Reassign A’s leaf and re-write old path

Path ORAM Example



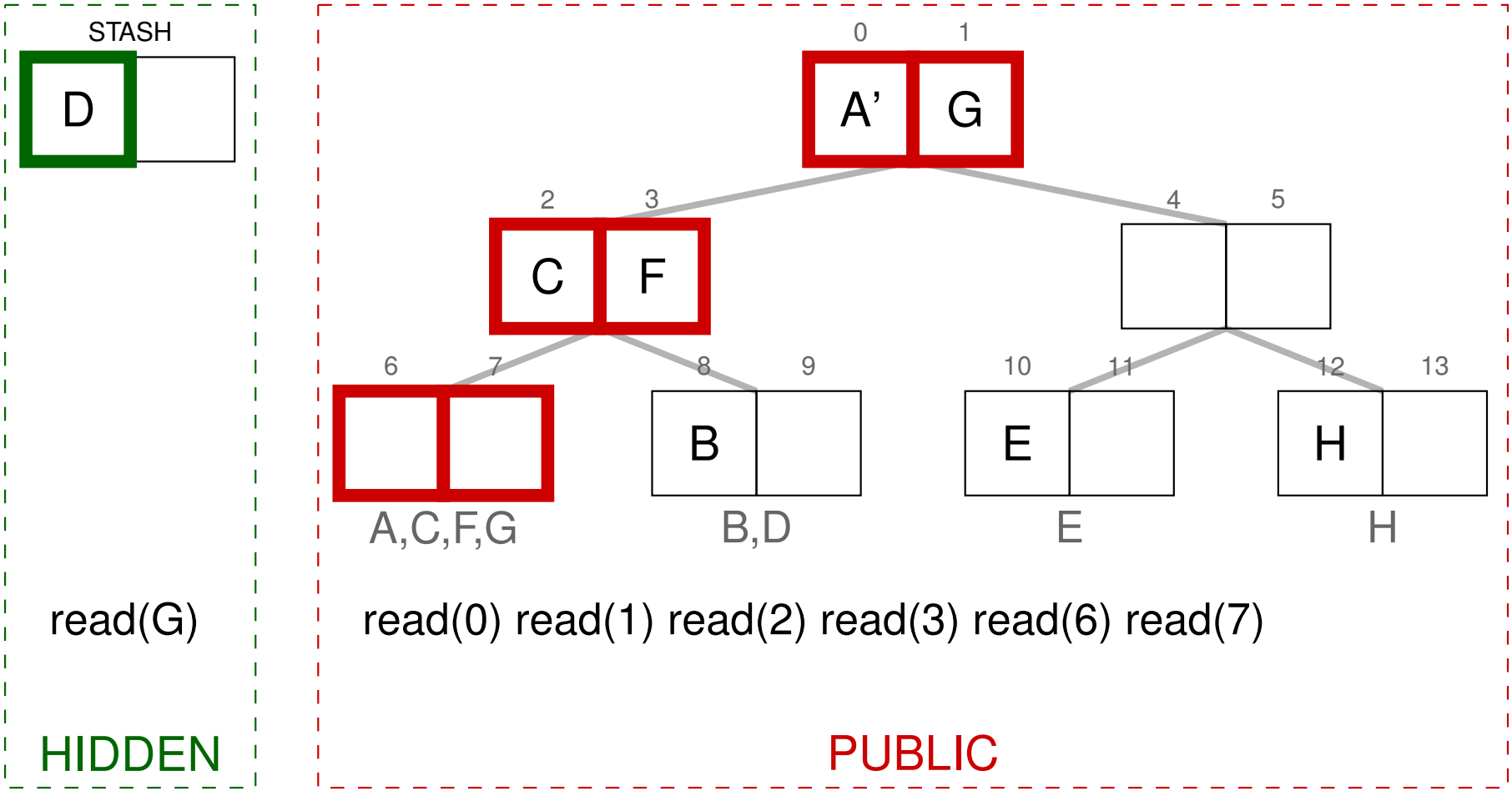
5 Read path to D

Path ORAM Example



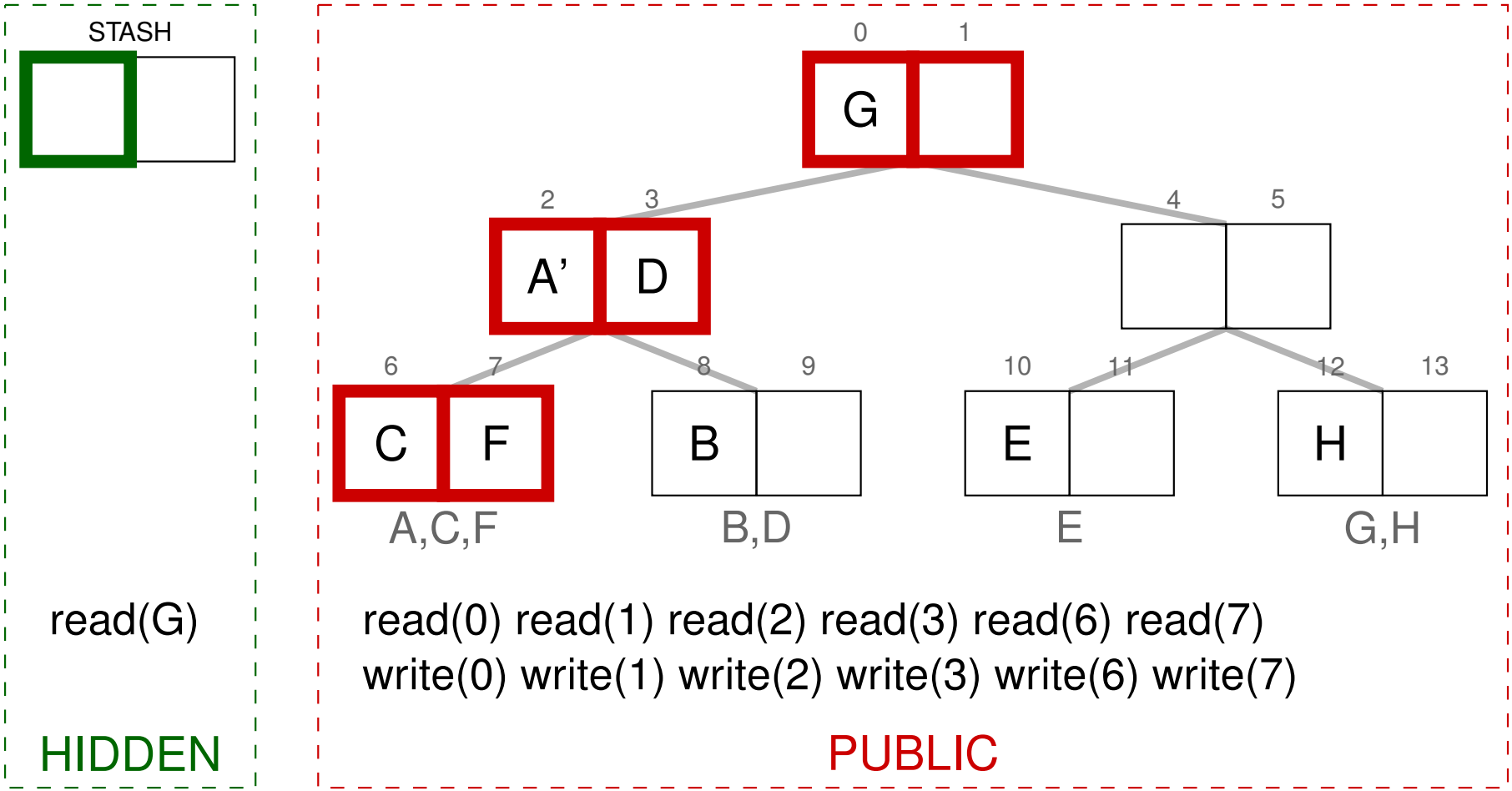
6 D can't fit in old path; must go into stash

Path ORAM Example



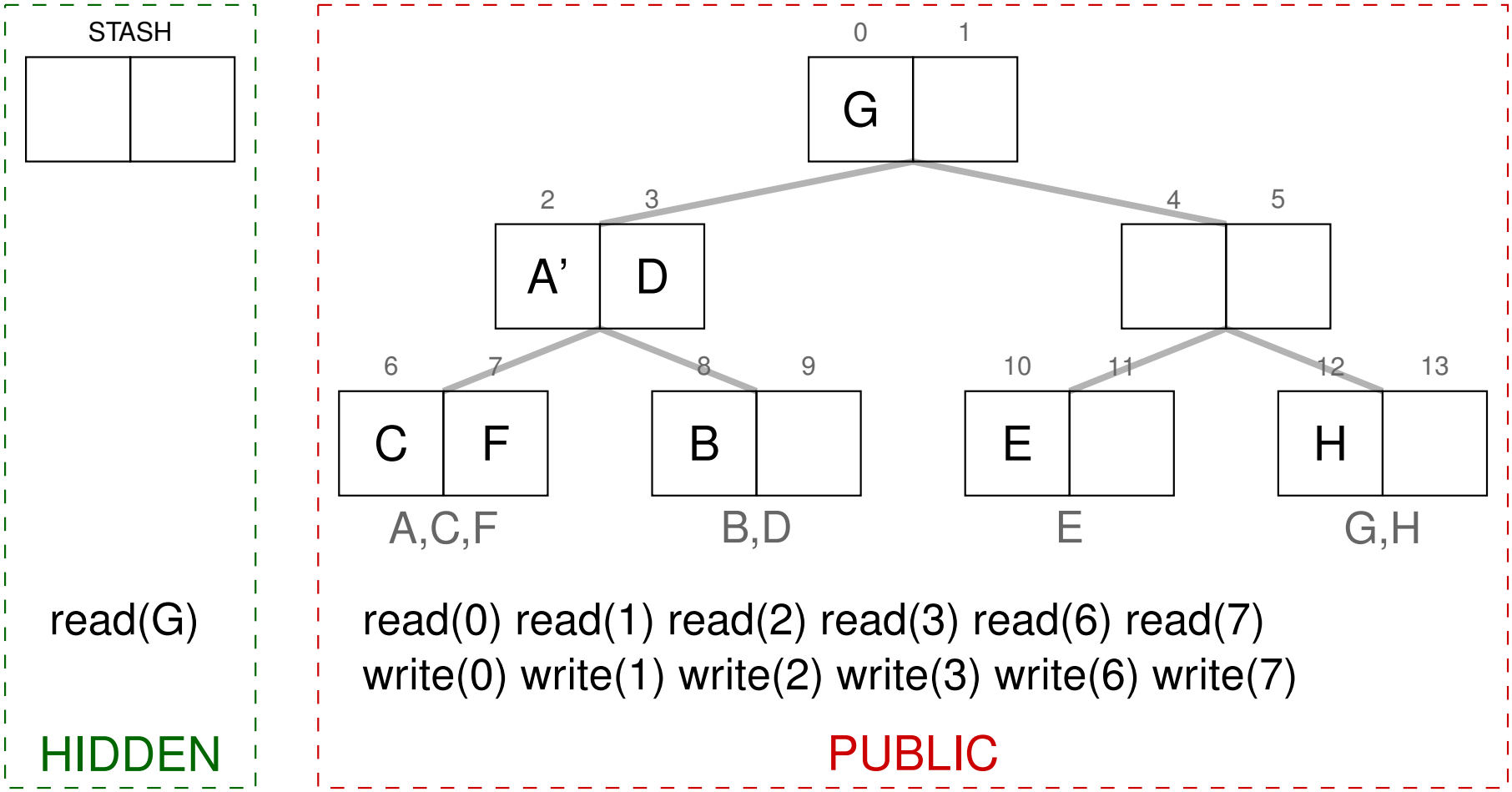
7 Read path to G

Path ORAM Example



8 Re-writing path allows to clear the stash

Path ORAM Example



Public transcript reveals nothing about the logical operations.

ORAM Comparison*

	Overhead		Storage	
	Read	Write	Hidden	Public
Trivial	N	N	1	N
Square Root ORAM	$\sqrt{N}$	$\sqrt{N}$	$O(\sqrt{N})$	N
Path ORAM	$5 \lg N$	$5 \lg N$	$O(\log N)$	$10N$

*Not counting the position map

What about the hidden storage?

Square root ORAM needs $O(\sqrt{N})$ hidden blocks for the “shelter”;
Path ORAM needs $O(\log N)$ for the “stash”.

This creates problems with multiple clients:
How to share the local (hidden) storage between different devices?

What about the hidden storage?

Square root ORAM needs $O(\sqrt{N})$ hidden blocks for the "shelter";
Path ORAM needs $O(\log N)$ for the "stash".

This creates problems with multiple clients:
How to share the local (hidden) storage between different devices?

Definition

An ORAM is **stateless** if it requires only $O(1)$ hidden storage.

Can be accomplished by transferring entire local state on each operation.

For Square Root ORAM and Path ORAM, this does not change the asymptotic overhead.

What about the position map?

Square root ORAM and Path ORAM require a position map to remember where blocks are stored.

Solution: Store the positions in $O(\log N)$ recursively smaller ORAMs of the same type.

(Assumes that a single block can hold many addresses.)

Some recent work on ORAMs

- Automating secure computation of RAM-model programs (Liu, Huang, Shi, Katz '14)
- Burst ORAM (Dautrich, Stefanov, Shi '14)
- Constant communication ORAM (Moataz, Mayberry, Blass '15)
- Oblivious data structures (Wang, Nayak, Liu, Chan, Shi '14)
- Oblivious PRAM (Boyle, Chung, Pass '16)
- Onion ORAM (Devadas, van Dijk, Fletcher, Ren '16)
- Ring ORAM (Ren, Fletcher, Kwon '14)
- S3ORAM (Hoang, Ozkaptan, Yavuz, Guajard, Nguyen '17
- TWORAM (Garg, Mohassel, Papamanthou '16)
- Variable-size ORAM (Roche, Aviv, Choi '16)

ORAM Setting

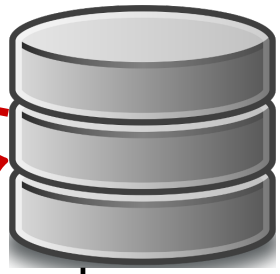
logical read

logical write

ORAM protocol

physical read

physical write

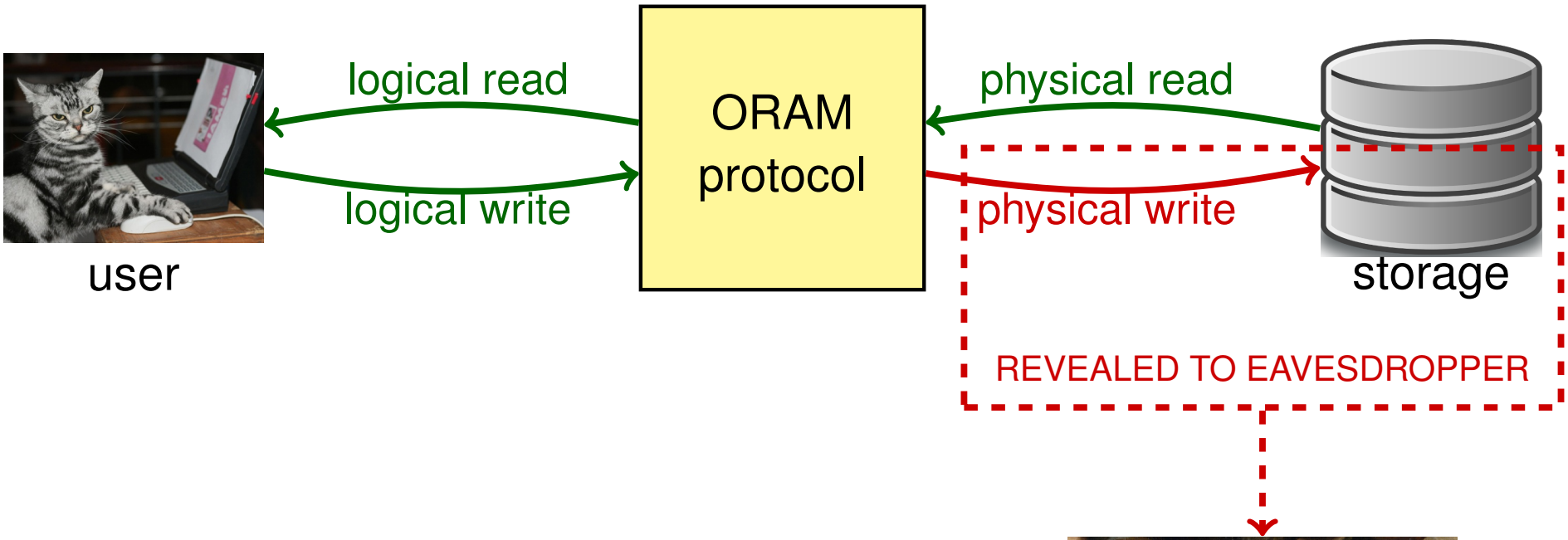


REVEALED TO EAVESDROPPER

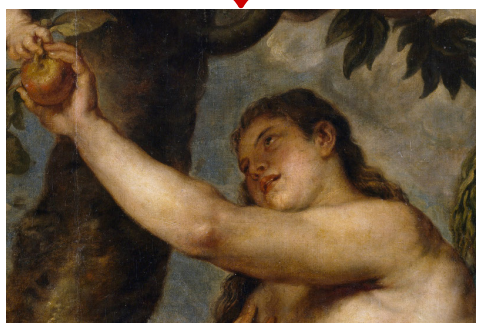
Goal: Eavesdropper should learn only the number of physical operations performed (not contents, op types, or addresses)

Write-Only ORAM Setting

In WoORAM, the attacker does not see physical read operations.



Goal: Eavesdropper should learn only the number of physical operations performed (not contents, op types, or addresses)



Applications of Write-Only ORAM

- Encrypted hidden volumes
  - Attacker must not learn whether disk is actually being used
  - Implemented with HiVE (Blass et al '14)
- Secure computation with remote attacker
  - Computing on trusted CPU (e.g. Intel SGX) with untrusted RAM
  - Adversary can only see snapshots of memory
  - Demonstrated attack on Montgomery's ladder (John et al '17)
- Encrypted backup and file synchronization *a la* Dropbox
  - User stores a local copy; only updates (writes) are sent to server
  - Implemented with ObliviSync (Aviv, Choi, Mayberry, Roche '17)

Write-Only ORAM Comparison*

	Overhead		Storage	
	Read	Write	Hidden	Public
Trivial	1	N	1	N
Square Root ORAM	1	$\sqrt{N}$	$O(\sqrt{N})$	N
Path ORAM	$5 \lg N$	$5 \lg N$	$O(\log N)$	$10N$

*Not counting the position map

BMNO-WoORAM

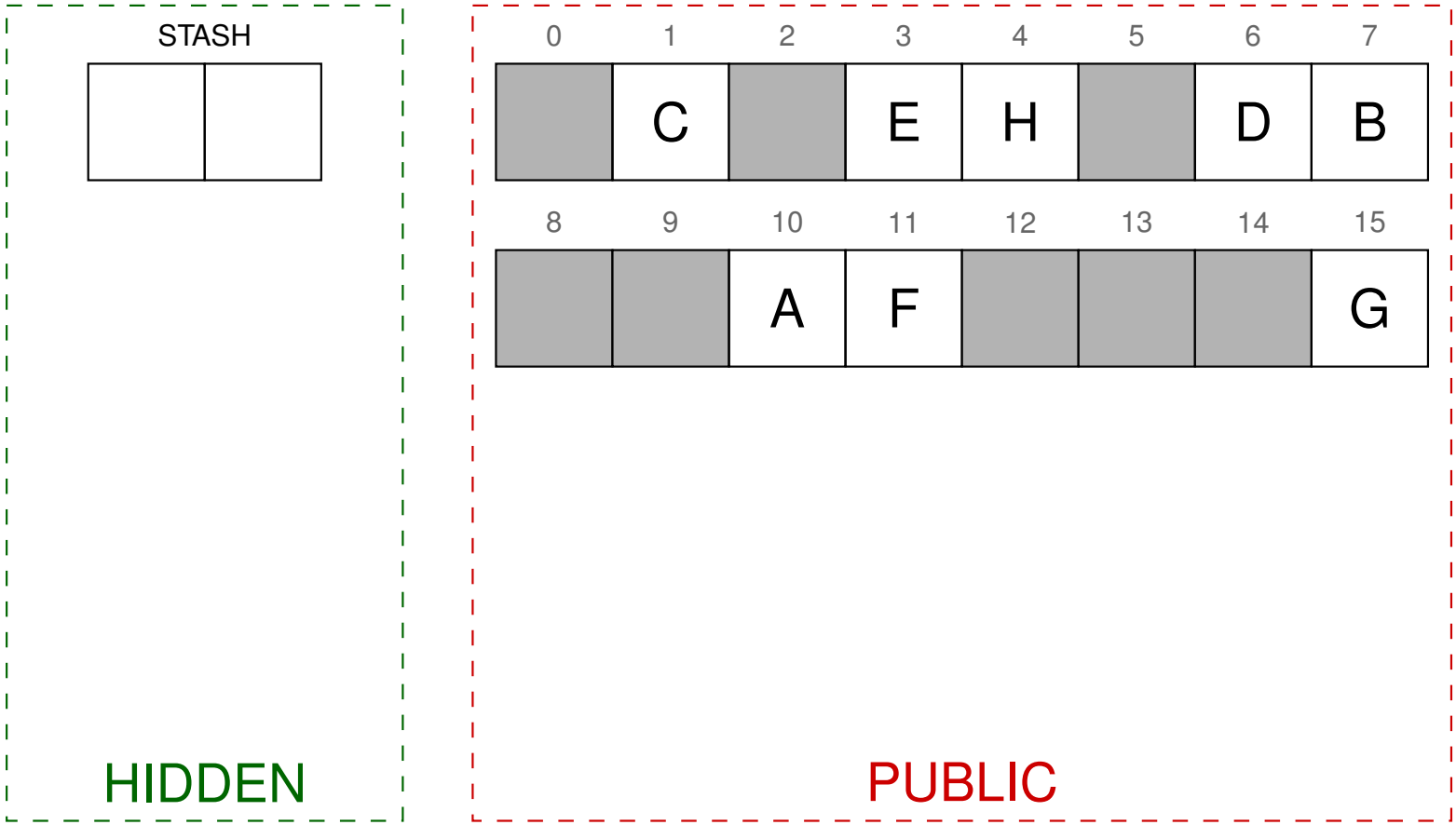
(Blass, Mayberry, Noubir, and Onarlioglu, CCS'14)

Key ideas:

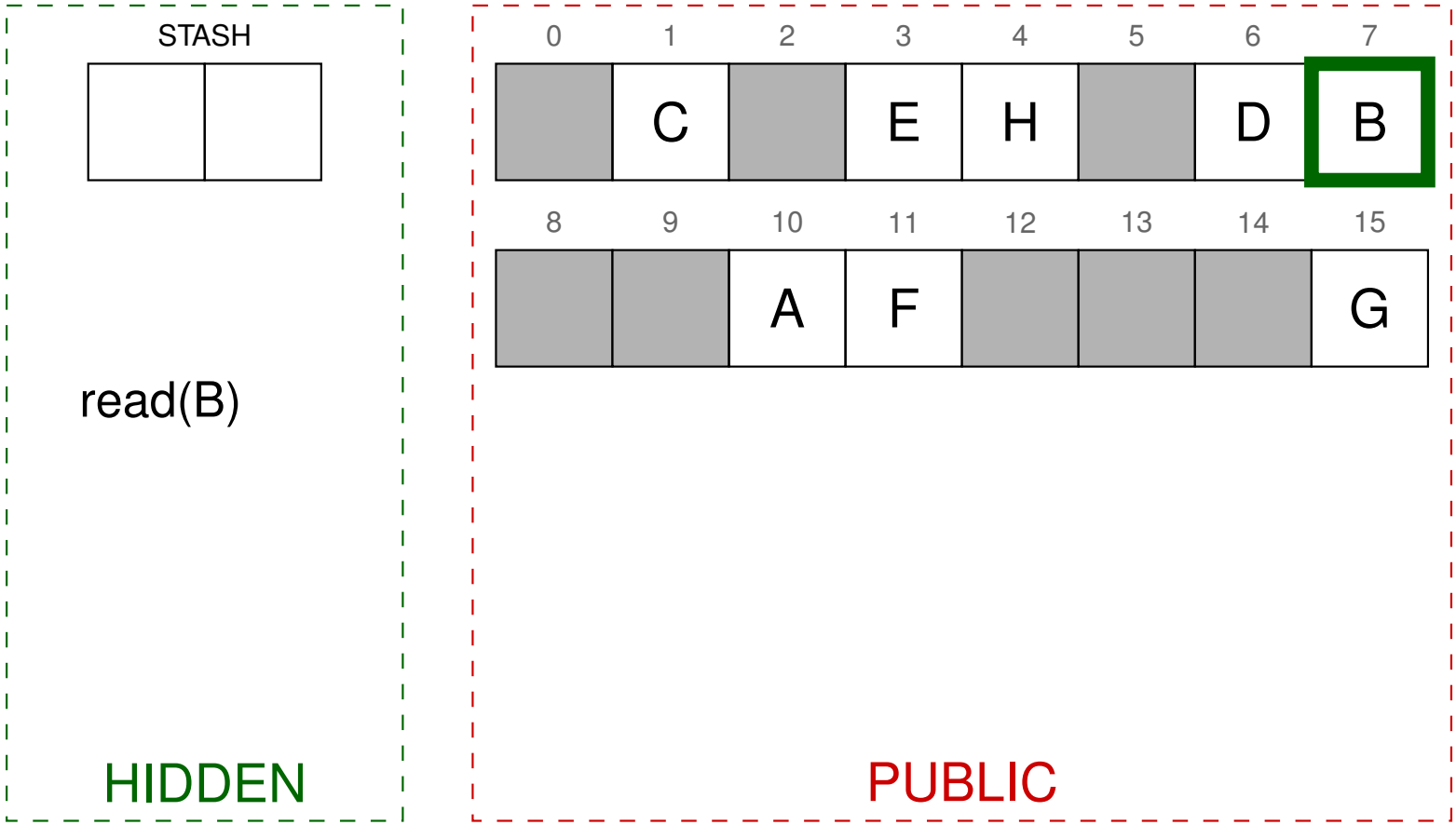
- No need to shuffle, just write to random locations
- "Stale" blocks identified by position map lookups can be overwritten
- Requires stash for when random locations are all non-stale

Overhead: $O(1)$ per write but $O(\log N)$ stateless

BMNO-WoORAM Example

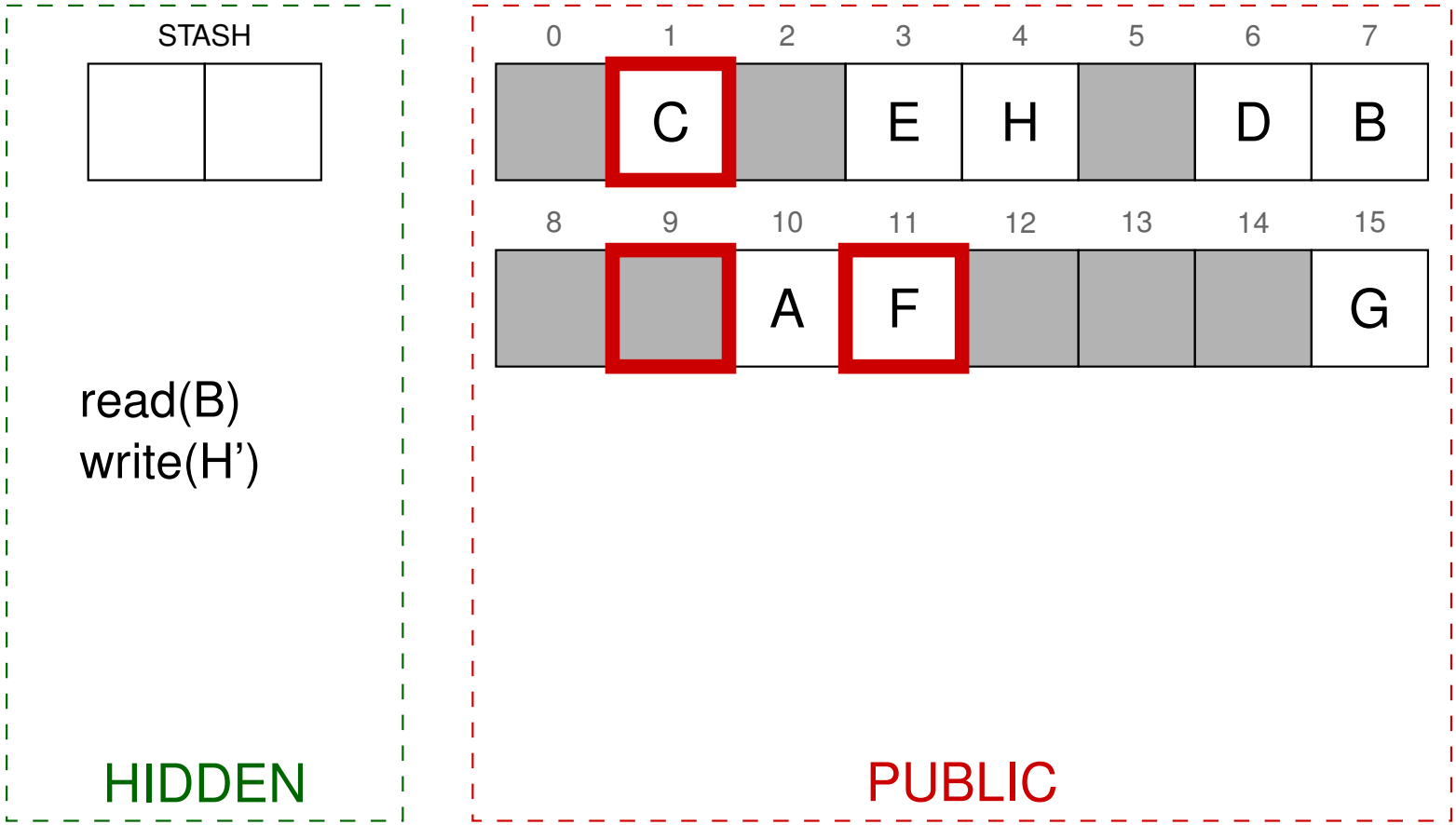


BMNO-WoORAM Example



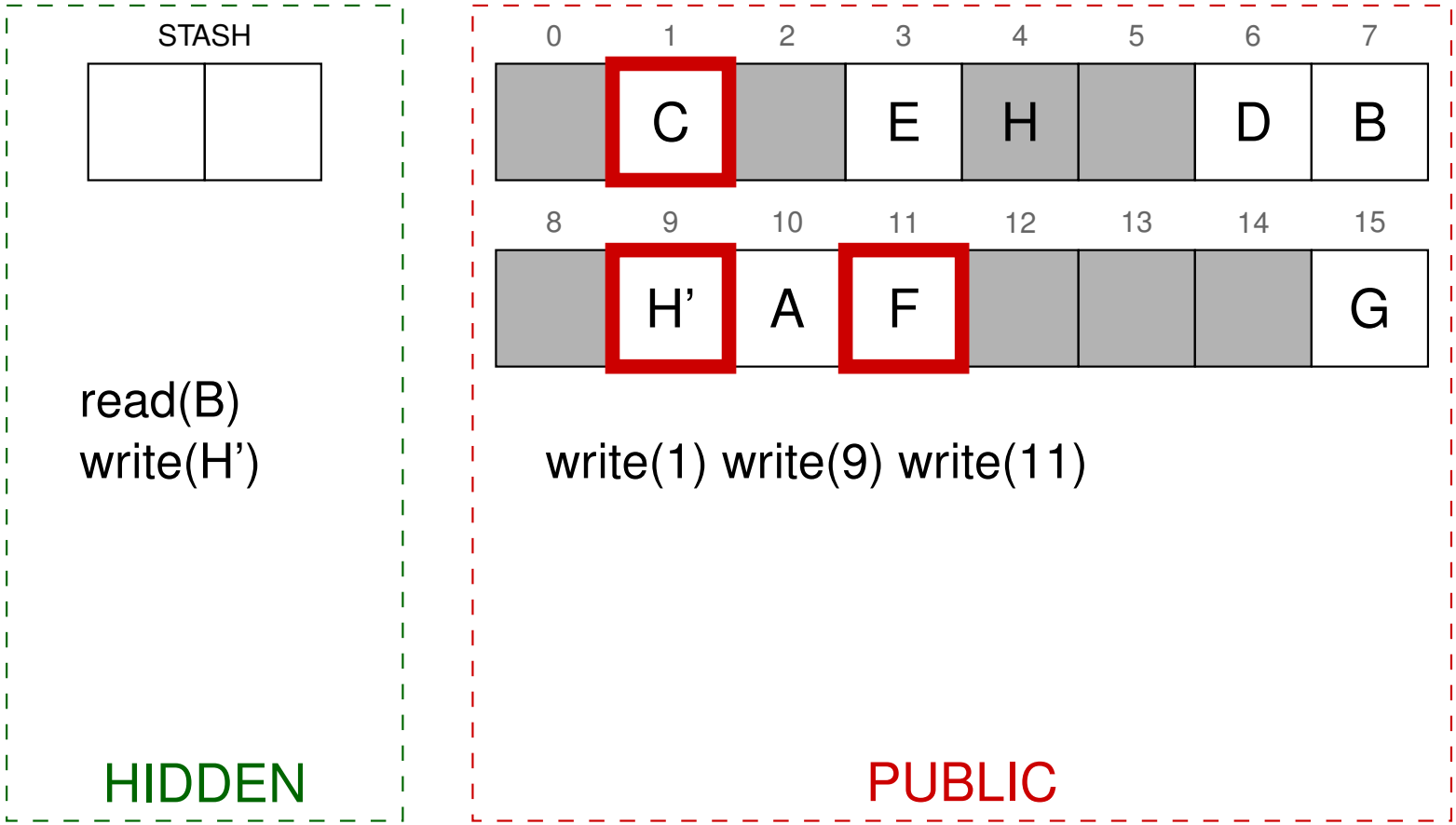
1. Normal read. Nothing is revealed.

BMNO-WoORAM Example



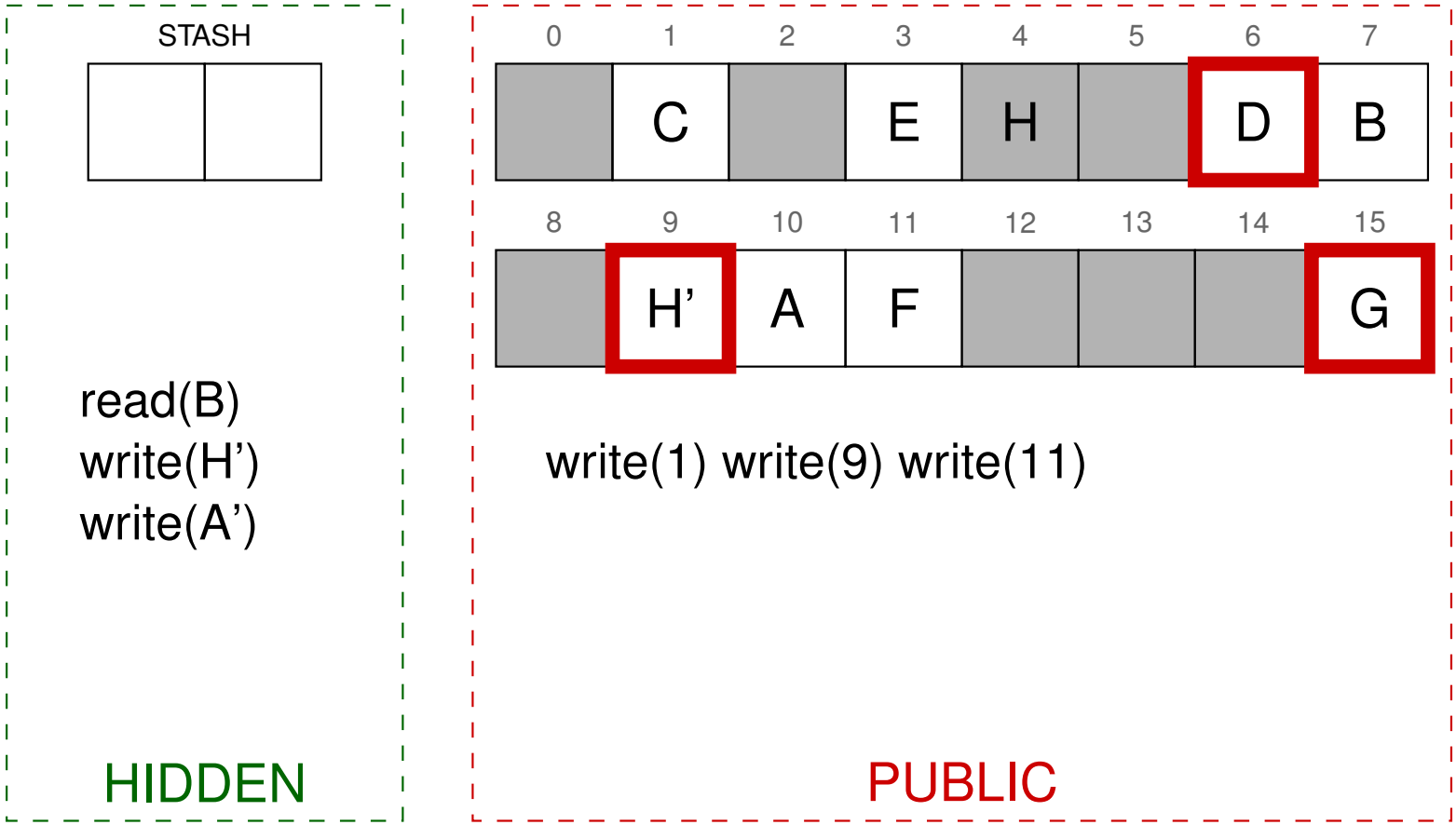
2 Normal write. Old value is not written but becomes stale.

BMNO-WoORAM Example



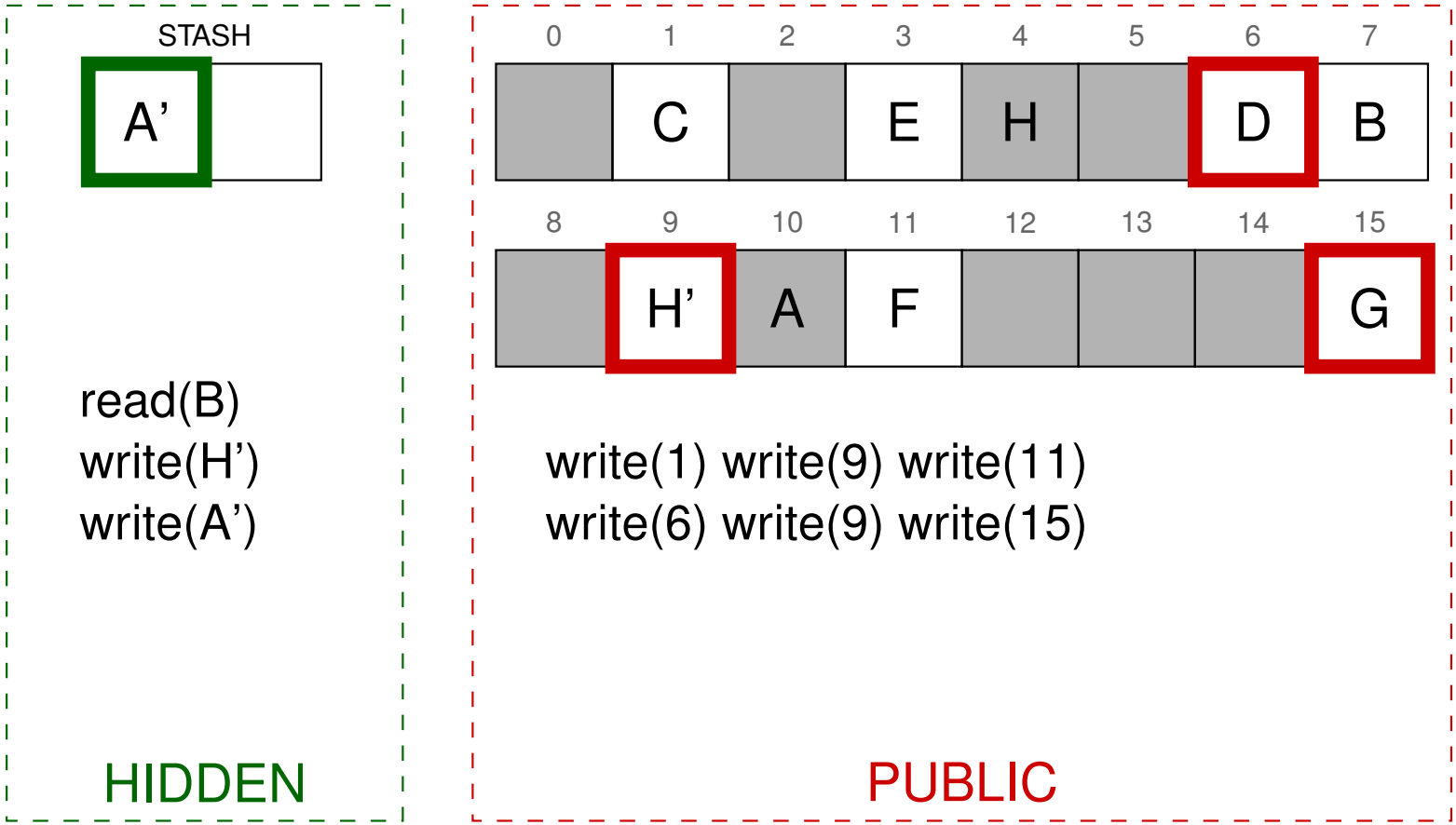
3. Normal write. Old value is not written but becomes stale.

BMNO-WoORAM Example



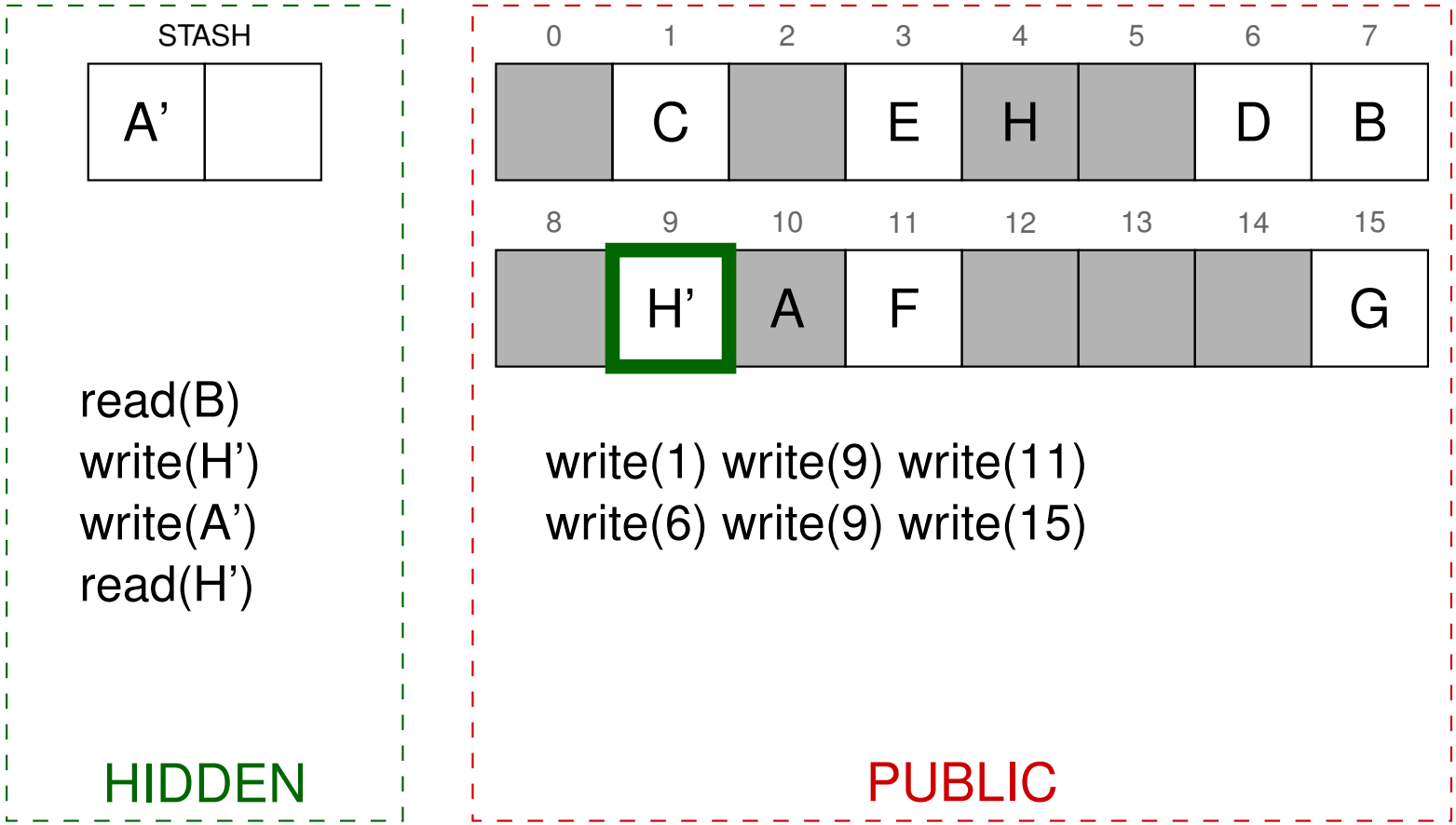
4 All selected positions are non-stale; add to stash.

BMNO-WoORAM Example



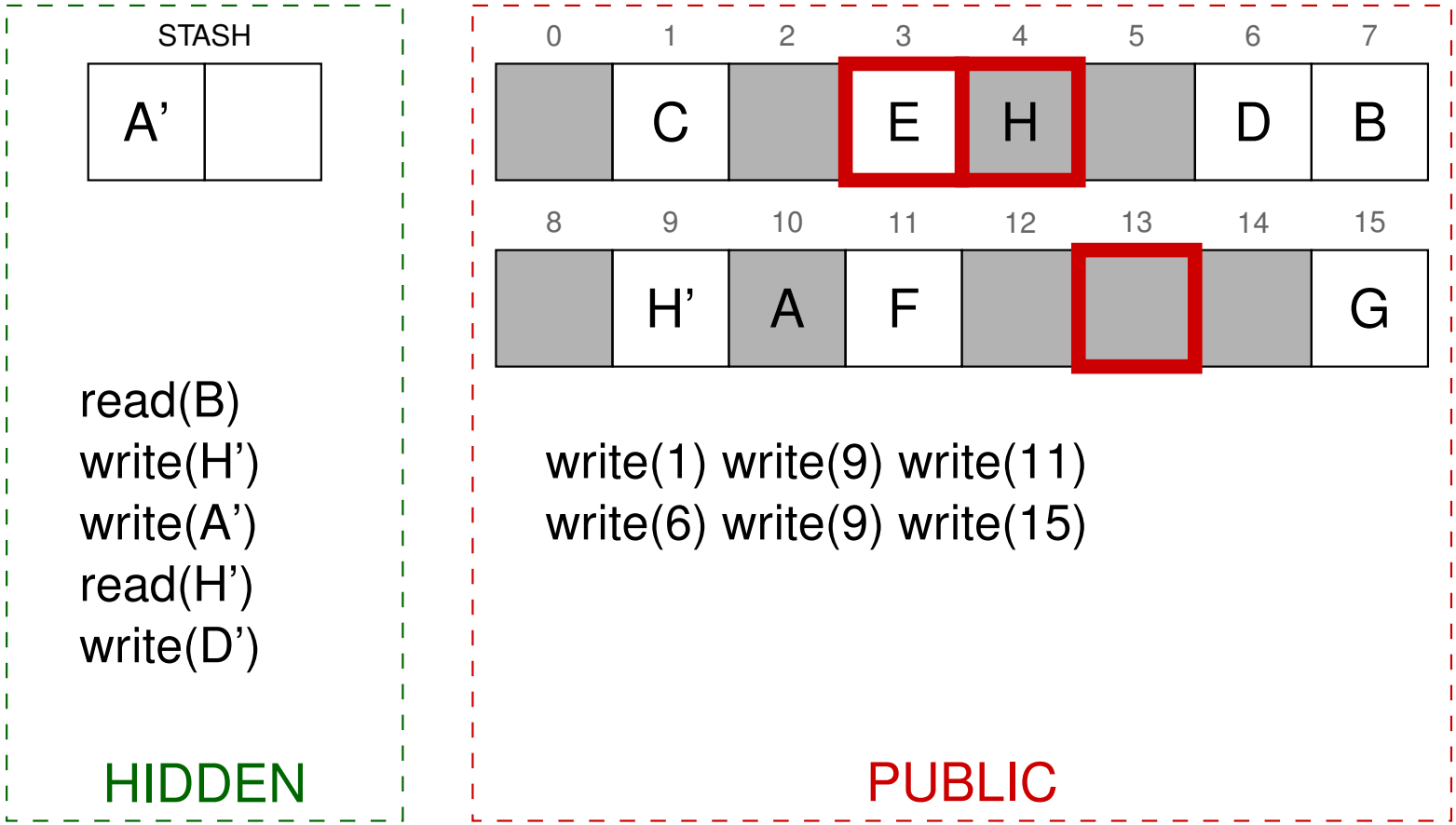
5 All selected positions are non-stale; add to stash.

BMNO-WoORAM Example



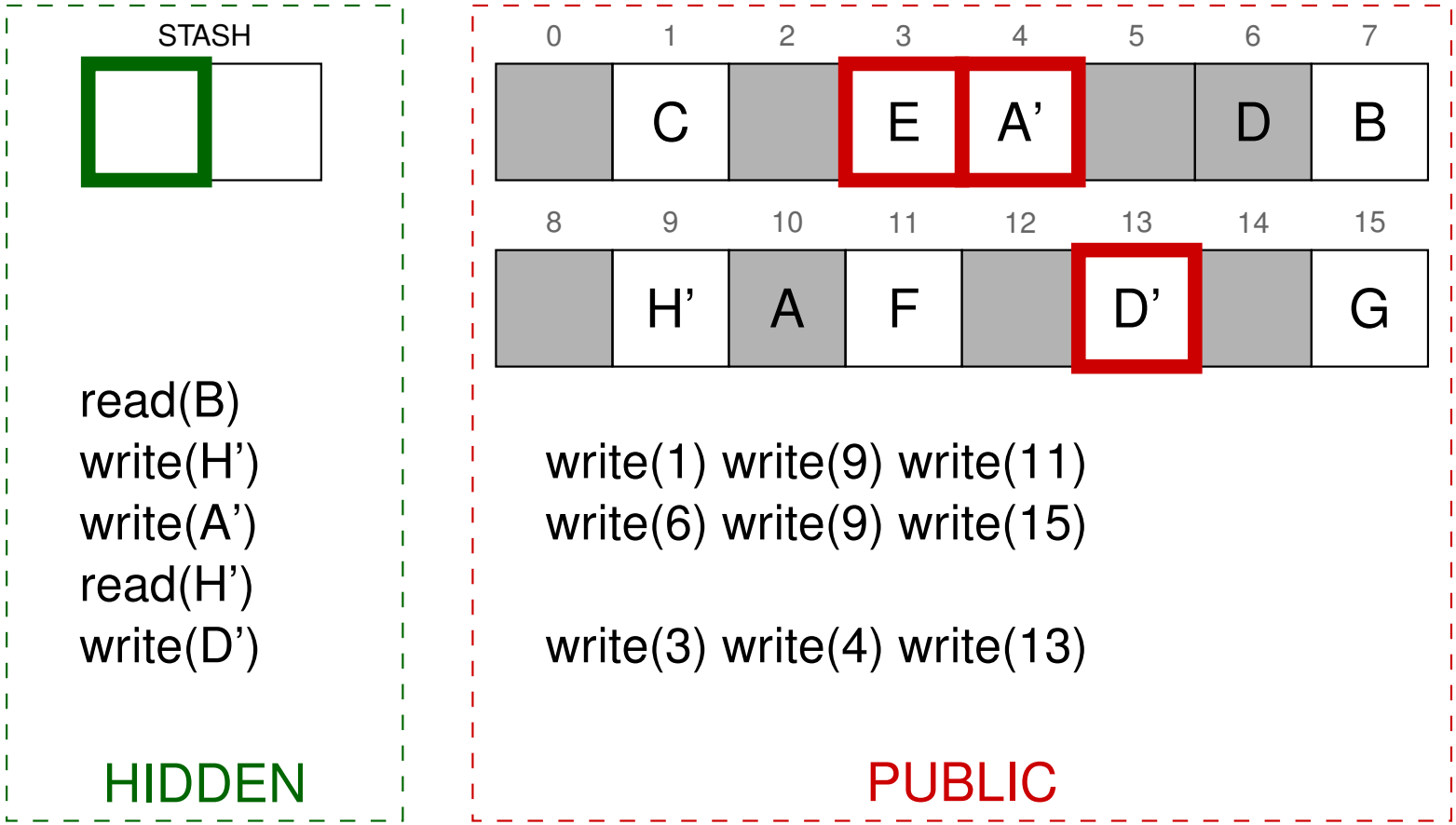
6 Read fetches most recent (non-stale) version.

BMNO-WoORAM Example



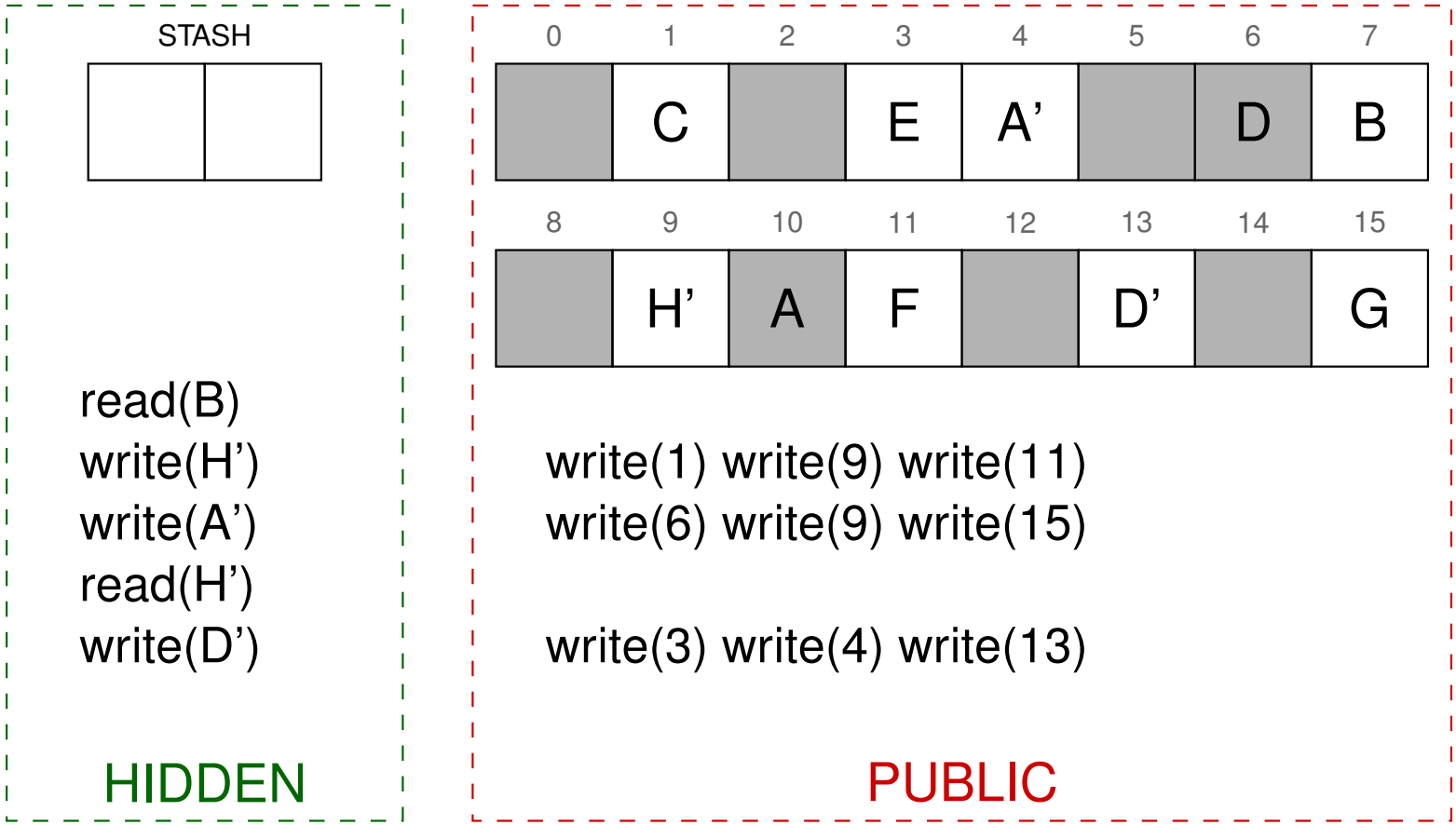
7 "Lucky" selection allows to write and clear stash.

BMNO-WoORAM Example



8 “Lucky” selection allows to write and clear stash.

BMNO-WoORAM Example



Public transcript reveals nothing about the logical operations.

Write-Only ORAM Comparison*

	Overhead		Storage	
	Read	Write	Hidden	Public
Trivial	1	N	1	N
Square Root ORAM	1	$\sqrt{N}$	$O(\sqrt{N})$	N
Path ORAM	$5 \lg N$	$5 \lg N$	$O(\log N)$	$10N$
BMNO-WoORAM	1	3	$O(\log N)$	$2N$

*Not counting the position map

Our WoORAM

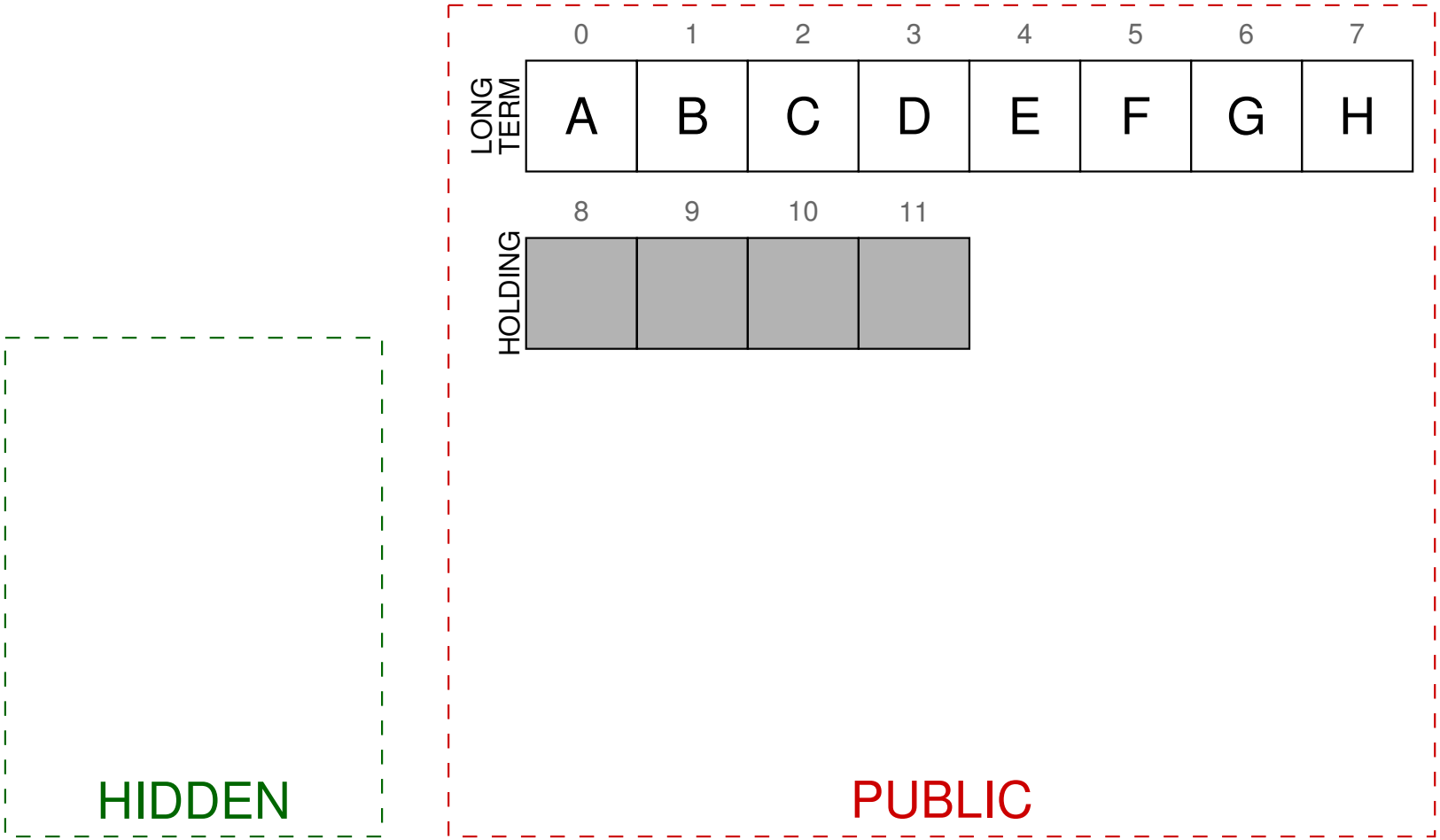
(Roche, Aviv, Choi, Mayberry, CCS'17)

Storage consists of two parts:

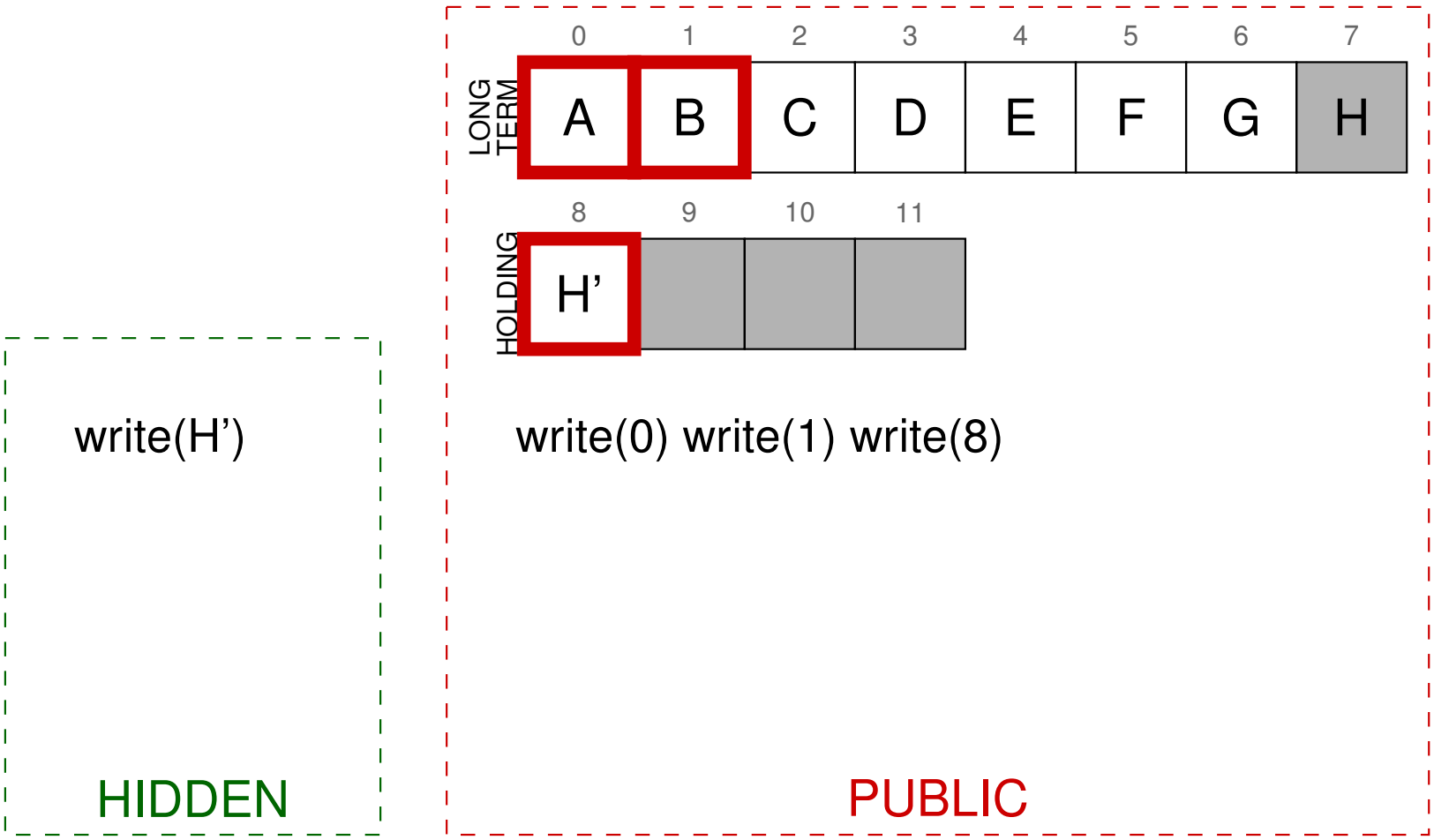
- **Holding area**:
  Circular buffer where new blocks are written sequentially
- **Long-term storage**:
  Size-N array where blocks go in their "true" locations
  Also written sequentially as items are copied from holding area

Key property: Every holding area block will definitely be copied to long-term storage before being overwritten.
(No randomization or stash needed!)

Deterministic WoORAM Example

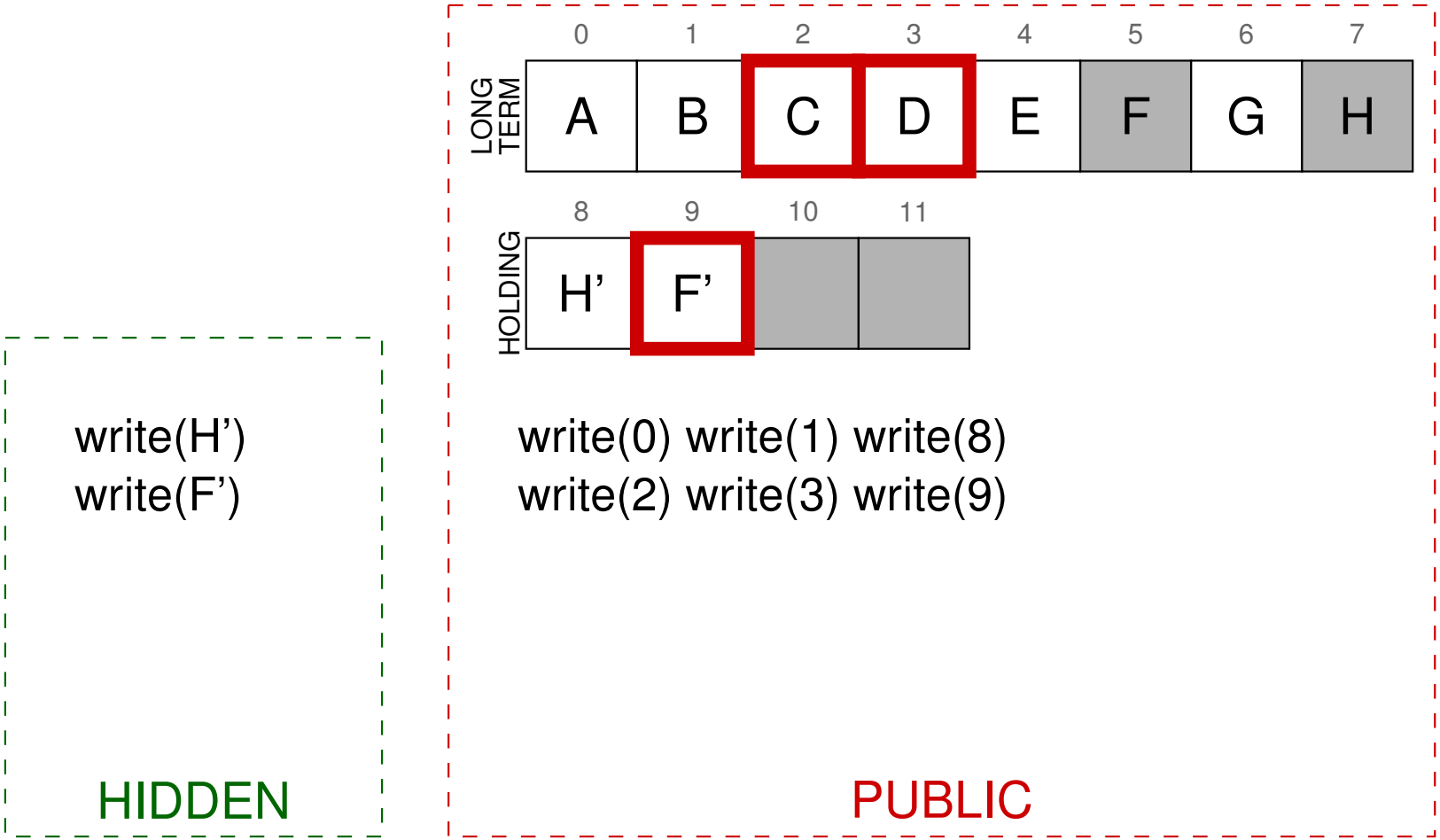


Deterministic WoORAM Example



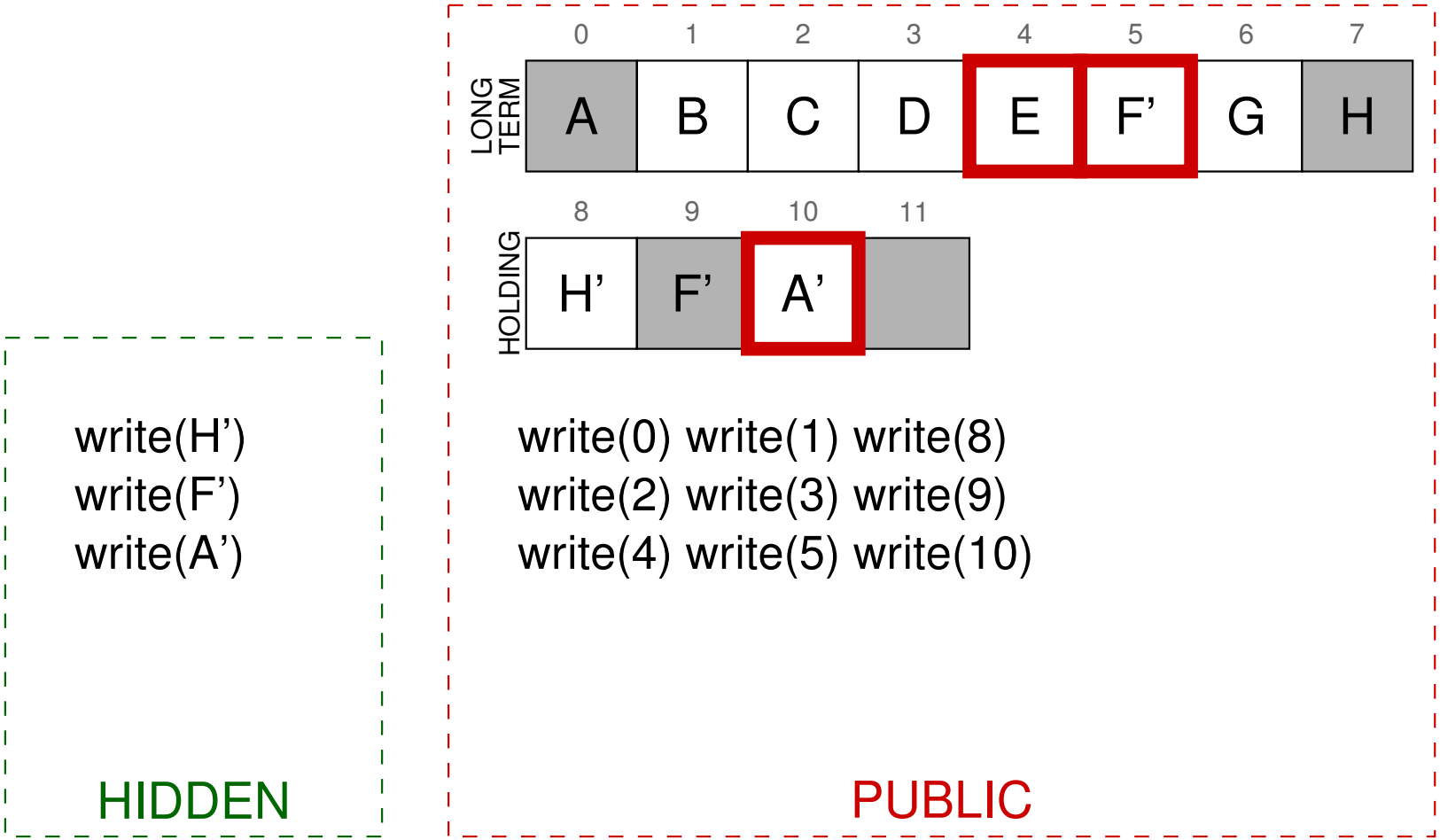
1. Normal write. Old value is not written but becomes stale.

Deterministic WoORAM Example



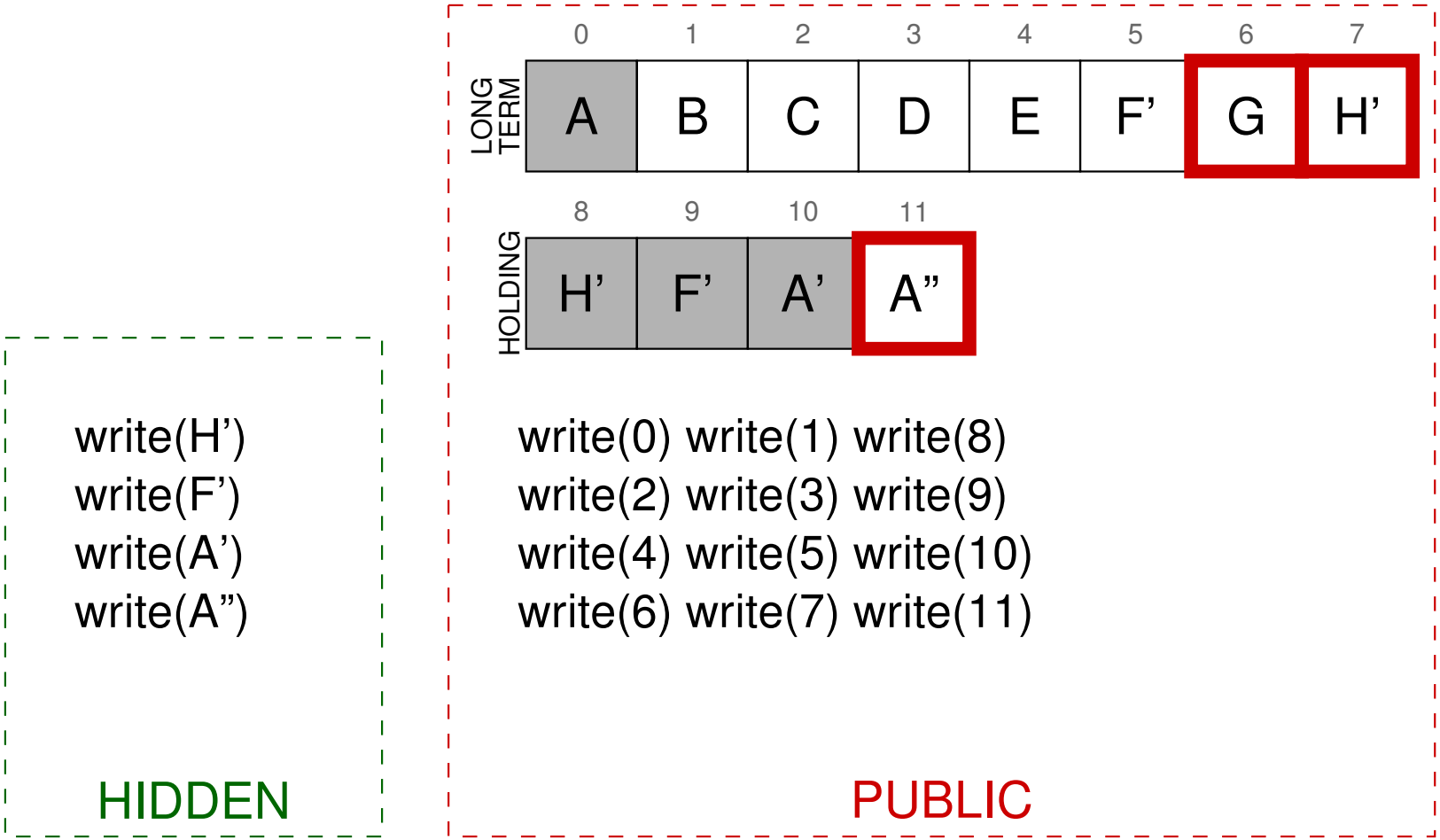
2 Normal write. Long-term blocks are “refreshed” even if not stale.

Deterministic WoORAM Example



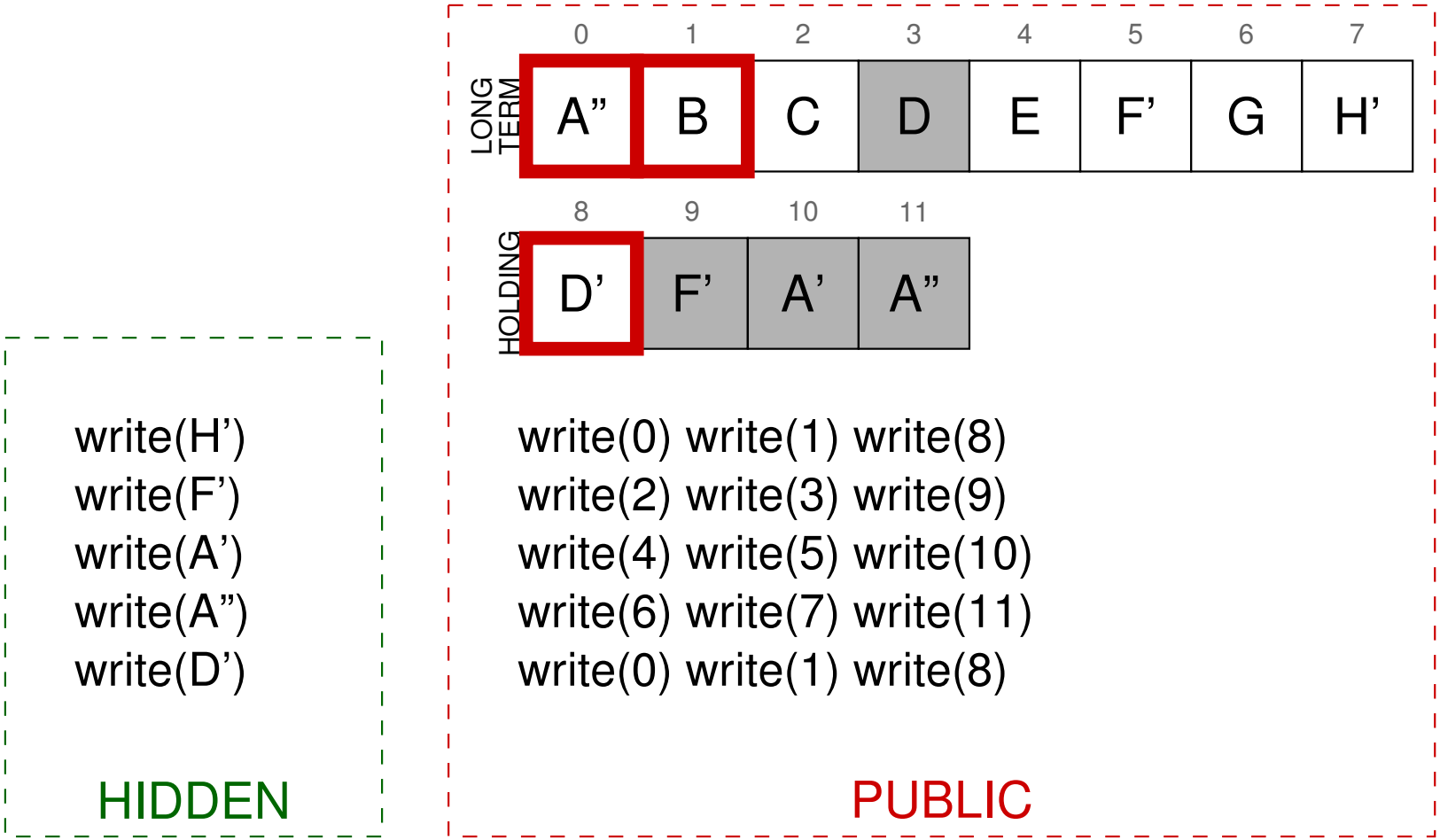
3 Just-written value is copied to long-term block.

Deterministic WoORAM Example



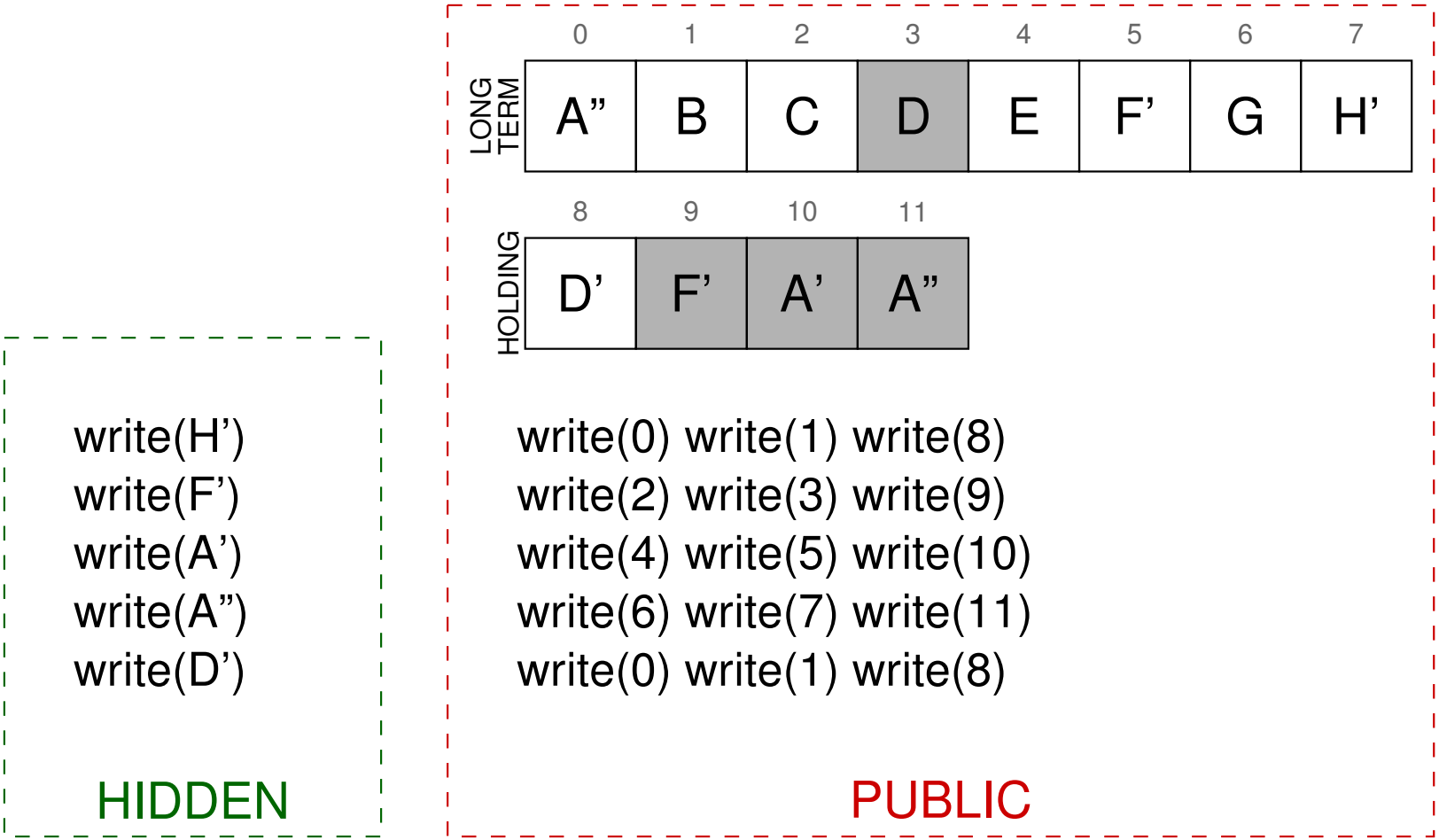
4 Holding area values can become stale before being refreshed.

Deterministic WoORAM Example



5 Both areas complete one “round” at the same time.

Deterministic WoORAM Example



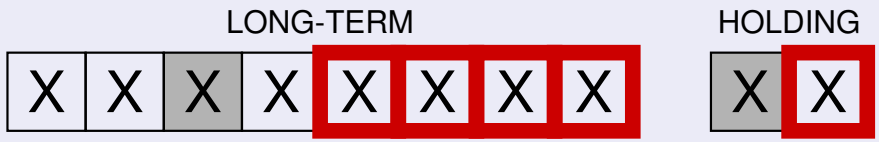
Public transcript reveals nothing about the logical operations.

Overhead-Storage Tradeoff

The scheme can be adjusted to write fewer blocks or use less storage.

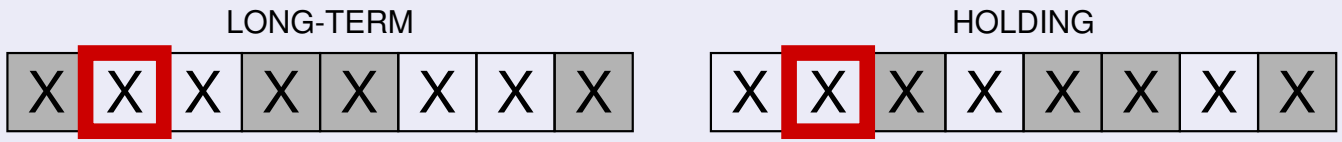
Example: Low storage

5 writes per op
Storage size $1.25N$



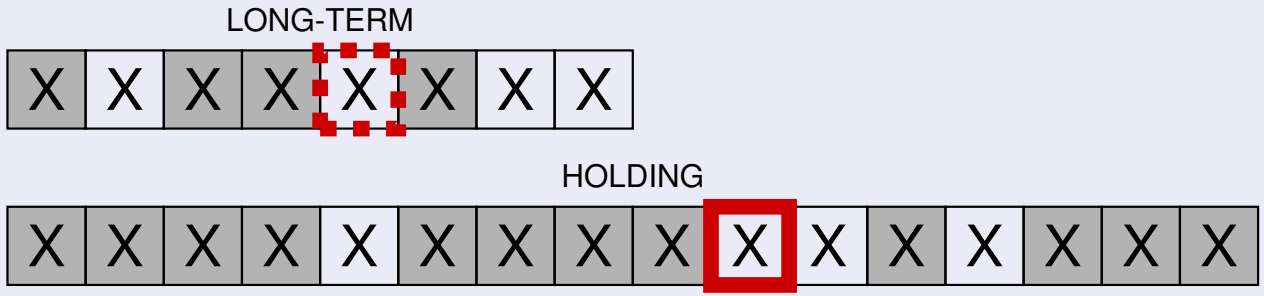
Example: Balanced

2 writes per op
Storage size $2N$



Example: Fast writes

1 or 2 writes per op (alternating)
Storage size $3N$



Advantages

- **Performance**: Asymptotically optimal.
  Can achieve (nearly) absolute optimal overhead or storage
- **Security**: Simplified security proof, relies only on symmetric cipher
- **Stateless**: Client only needs cipher key and a counter
- **Locality**: Long-term storage has spatial locality; holding area has temporal locality. Reduces cache misses during reads.
- **Sequential write pattern**: Ideal for SSD devices

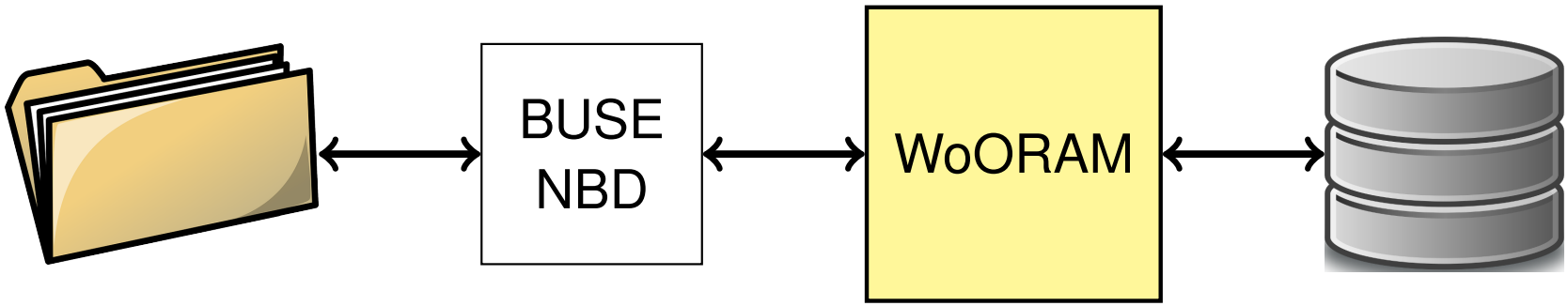
Write-Only ORAM Comparison*

	Overhead		Storage	
	Read	Write	Hidden	Public
Trivial	1	N	1	N
Square Root ORAM	1	$\sqrt{N}$	$O(\sqrt{N})$	N
Path ORAM	$5 \lg N$	$5 \lg N$	$O(\log N)$	$10N$
BMNO-WoORAM	1	3	$O(\log N)$	$2N$
Ours	1	$1+\epsilon$	2	$(1+\frac{1}{\epsilon})N$

*Not counting the position map

Implementation

- Both WoORAM implemented as C++ template classes
- BUSE (Block Device in Userspace) connects code to Linux kernel



Benchmark Results using `fio` with random reads&writes

- Hard disk: 10.2 MB/sec
  (200x faster than BMNO, 1.6x slower than baseline)
- Solid state: 34.4 MB/sec
  (4.1x faster than BMNO, 4.5x slower than baseline)

More details

See the paper for details on:

- **Ciphers**
  Using CTR mode cuts down on the storage of IVs.
- **Position map**
  We design an oblivious trie data structure to store the positions in just one recursive WoORAM.
  This saves a $O(\log N)$ factor compared to BMNO-WoORAM.
- **Block packing and striping**
  Everything (including position map) can be written in two sequential physical blocks under reasonable parameters.


Thank you for having me!