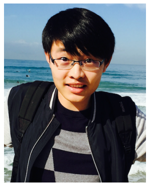
Scrypt is Maximally Memory Hard

Joël Alwen
IST Austria

Binyi Chen
UCSB

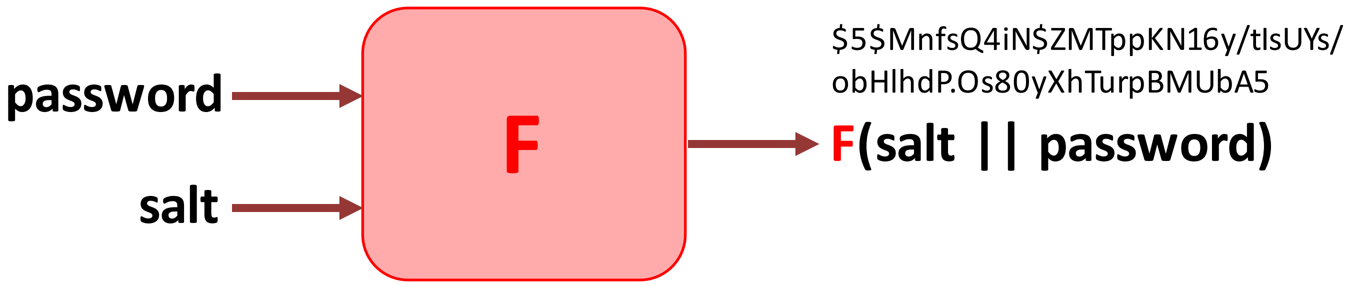


Krzysztof Pietrzak
IST Austria

Leonid Reyzin
Boston University

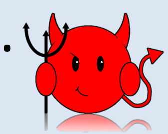
Stefano Tessaro
UCSB

Password hashing: Store a hash of a password + salt



F is **moderately hard**

- **Honest users** can still login **quickly**.
  $\approx$ 1 evaluation of **F**
- Brute-force attack is **infeasible**.
  Many evaluations of **F**



Traditional hardness metric: Time complexity (e.g., PKCS #5)

ASIC-resistance

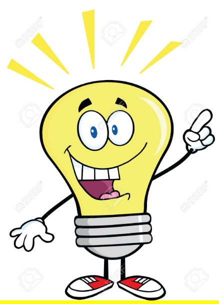
better parallelization, pipelining for speedup; lower energy costs …

Honest users
(General-purpose CPU)

cost per **F** eval = $\boldsymbol{C}$

Adversaries
(ASICs)

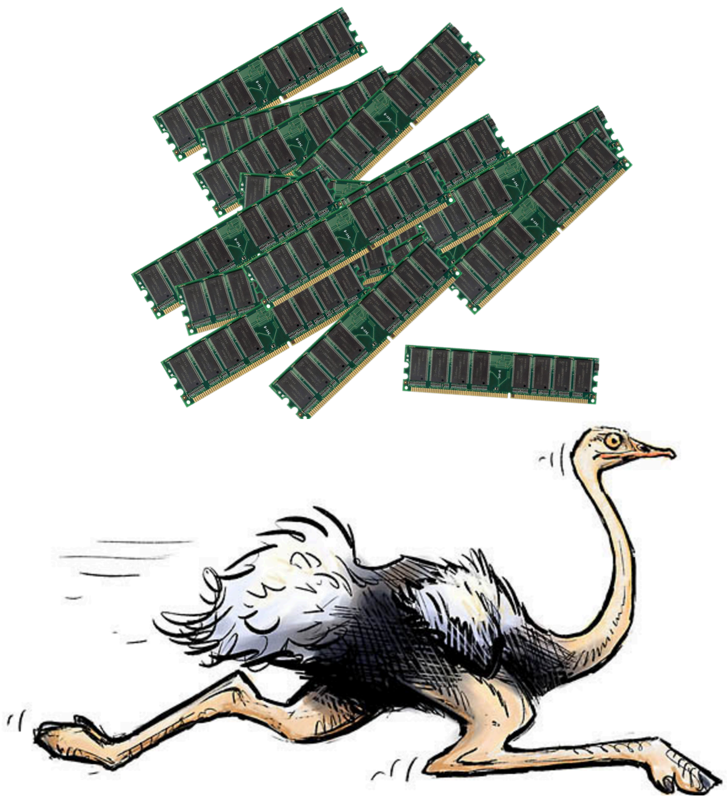
cost per **F** eval = $\boldsymbol{C'} \ll \boldsymbol{C}$

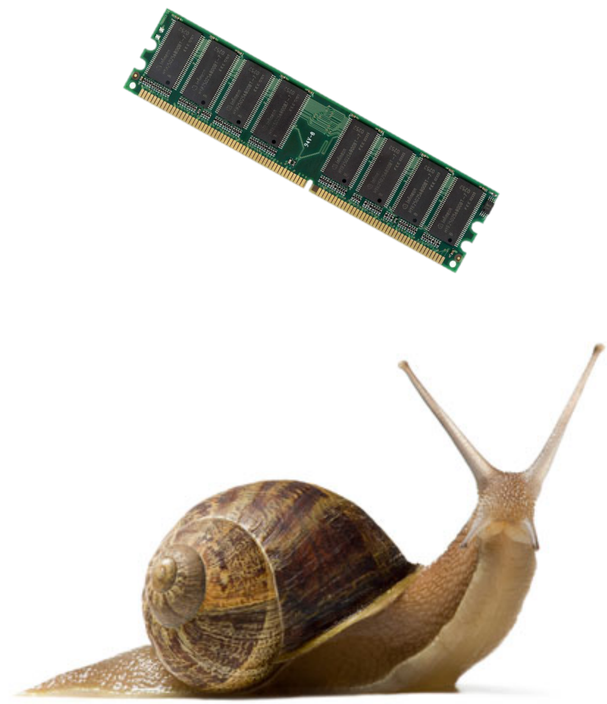


Can we enforce $\boldsymbol{C'} \approx \boldsymbol{C}$?

Idea: Memory costs (e.g., on-chip area, access time, $-cost) are platform independent

Memory-hard functions (MHFs)[Percival, 2009]





Fast evaluation of MHF F ⇒

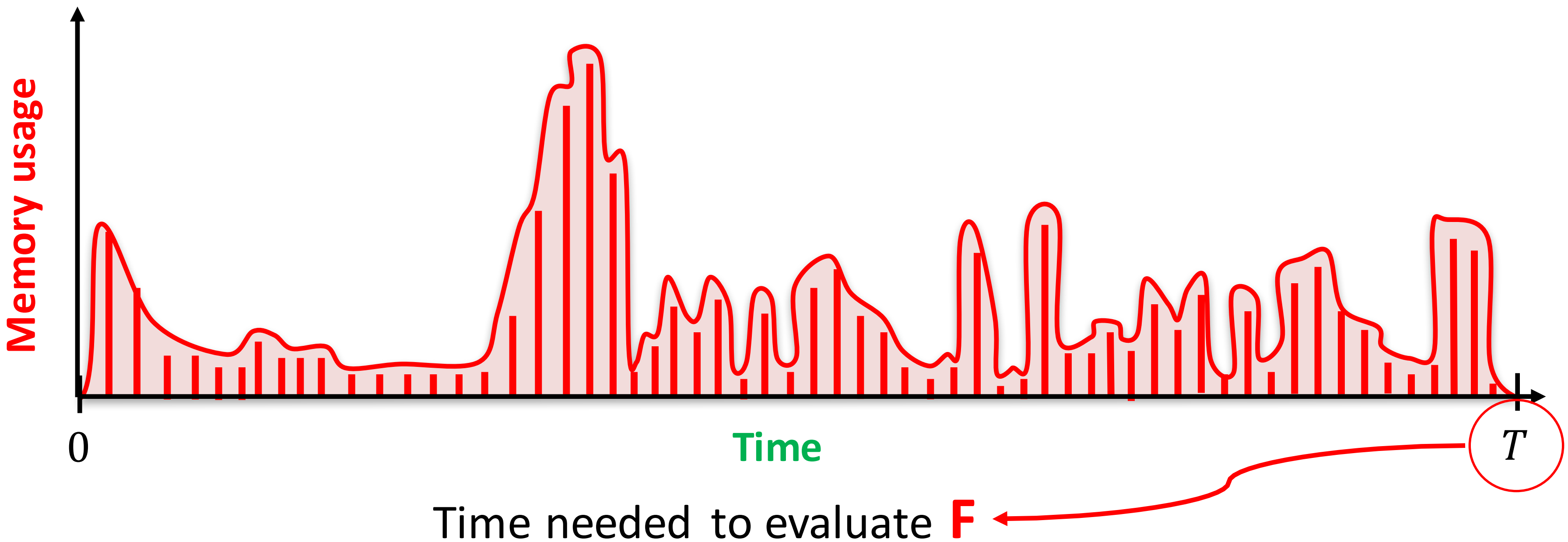
large memory

Small memory ⇒

slow evaluation of MHF F

Memory-hardness, more precisely

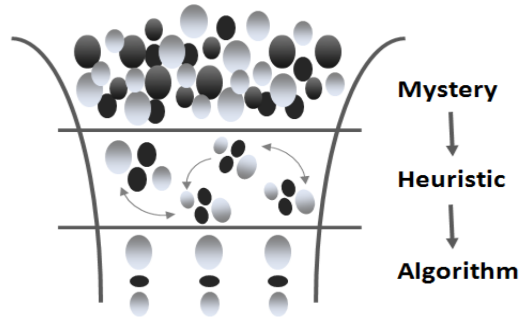
Goal: Maximize cumulative memory complexity (CMC) for any (possibly **parallelized**) strategy to evaluate **F.**



$$\text{CMC} = \sum_{t=0}^{T} \text{Memory}(t)$$

[Alwen and Serbinenko, STOC '15]

Memory-hardness



PHC call for submissions

The Password Hashing Competition (PHC) organizers solicit proposals from any interested party for candidate password hashing schemes, to be considered for inclusion in a portfolio of schemes suitable for widespread adoption, and covering a broad range of applications.

Memory-hardness was de-facto requirement for PHC

Security

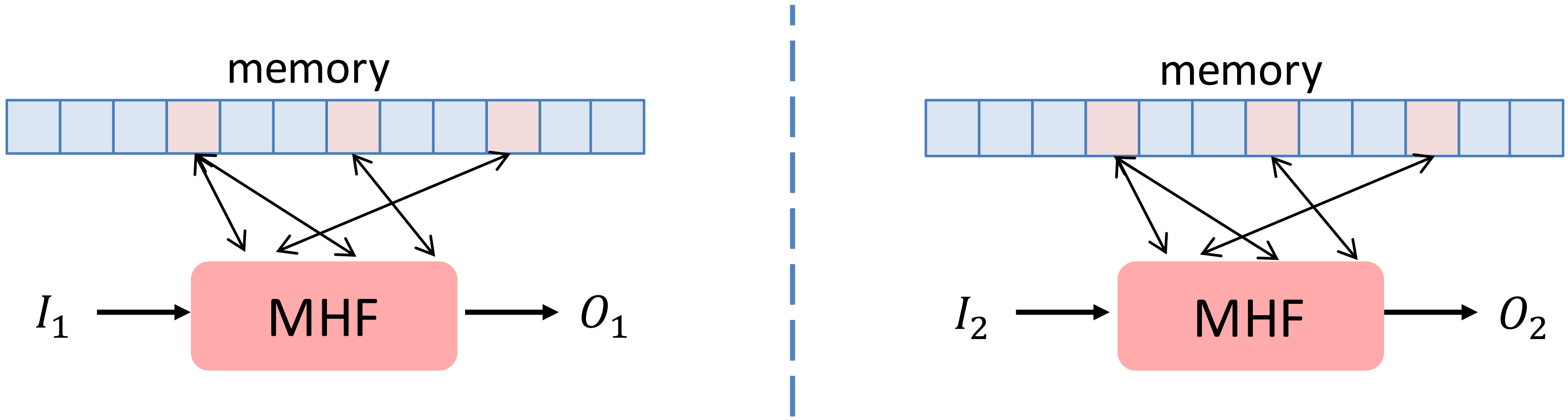
- Cryptographic security: the function should behave as a random function (random-looking output, one-way, collision resistant, immune to length extension, etc.).
- Speed-up or other efficiency improvement (e.g., in terms of memory usage per password tested) of cracking-optimized implementations (checking multiple sets of inputs in parallel, and doing so in a CPU's native code) compared to implementations intended for password validation should be minimal.
- Speed-up or other efficiency improvement (e.g., in terms of area-time product per password tested) of cracking-optimized ASIC, FPGA, and GPU implementations (checking multiple sets of inputs in parallel) compared to CPU implementations intended for password validation should be minimal.
- Resilience to side-channel attacks (timing attacks, leakages, etc.). In particular, information should not leak on a password's length.

Many memory-hard candidates: Argon2d, Argon2i, Scrypt, Lyra2, Balloon hashing, Catena, Yescrypt, ……

Can we build <u>provably</u> memory-hard functions?

Towards optimal memory hardness

Previous provably MHFs [AS15,BCS16,ABP17] are iMHFs: data-independent memory access patterns!



Two issues raised by Alwen and Blocki:

(1) No **iMHF** achieves optimal memory hardness.

(2) Practical iMHFs are even less memory hard for parallel evaluation strategies.

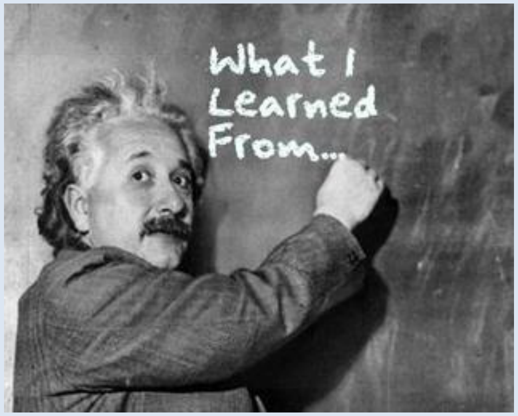
Can data-dependence help?

This paper: Scrypt is optimally memory hard

- Very first conjectured MHF: Proposed by **Colin Percival** in 2009
- Used within PoWs in **Litecoin**
- Inspired the design of **Argon2d** – one of the winners of Password Hashing Competition
- Covered by RFC 7914

Take home message:

Very first example of function with provably optimal memory hardness.



+ it is practical, already in use, and relatively simple

Finding such proof has been a surprisingly hard problem:

- [Percival, 2009] is incorrect
- [ACKKPT16] only gave restricted result

No iMHF achieves optimal memory hardness

Roadmap

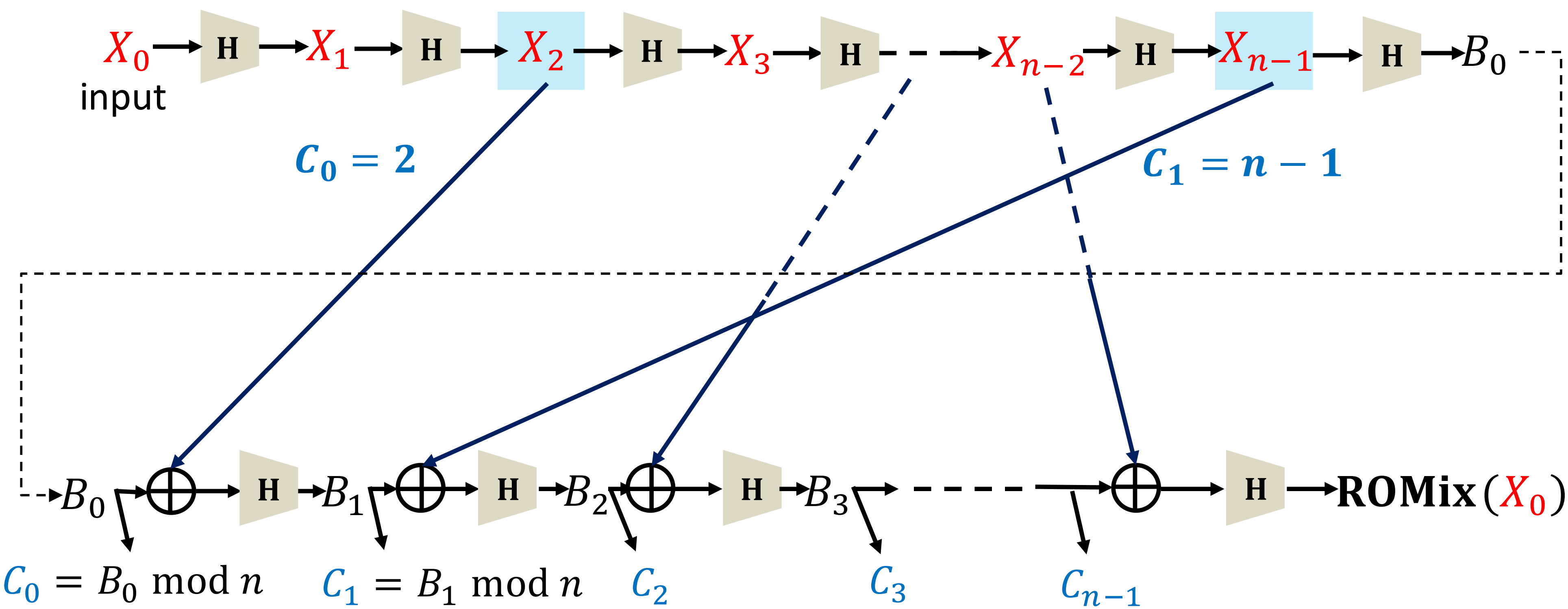
1. The Scrypt function
   Definition, memory-hardness intuition, and challenges

2. Optimal memory hardness of Scrypt

3. Conclusions

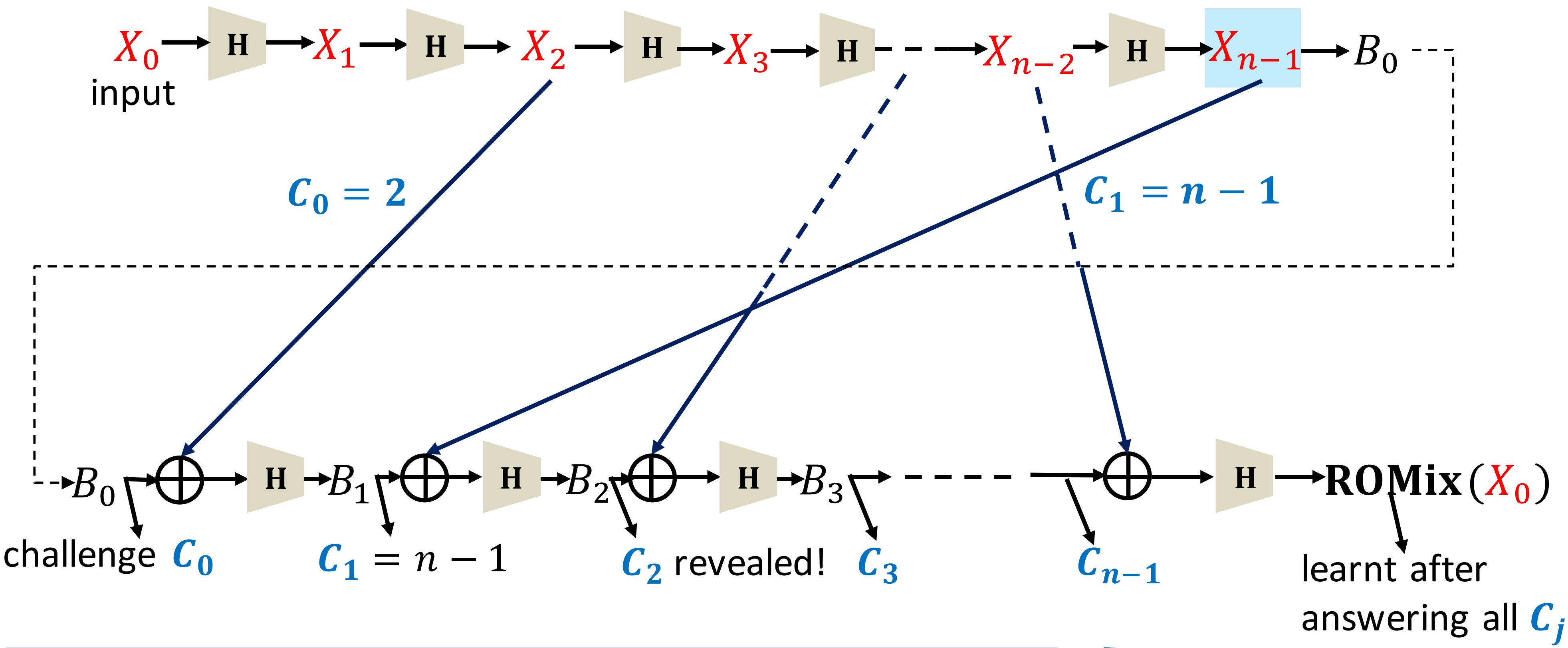
Core of Scrypt: ROMix

Modeled as a random oracle! → **H:** A Salsa20 based “hash function” with output length $\boldsymbol{w}$.



n: a tunable parameter.
e.g., $n = 2^{14}$, $w = 1$ KB

ROMix: Answering challenges



$C_0, C_1, \ldots C_{n-1}$ **unpredictable challenges**:

1. Need to know X_{C_j} to learn C_{j+1}
2. Need to answer all challenges to complete the evaluation

Useful to abstract this!

Abstraction: Round Game

C_j

$X_0 \rightarrow \mathbf{H} \rightarrow X_1 \rightarrow \mathbf{H} \rightarrow X_2 \rightarrow \mathbf{H} \rightarrow X_3 \rightarrow \mathbf{H} \rightarrow X_4 \rightarrow \cdots \rightarrow \boxed{X_{n-2}} \rightarrow \mathbf{H} \rightarrow X_{n-1}$

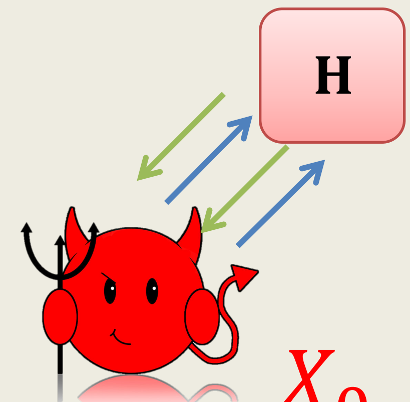
input

Abstract 2nd phase: challenges are ~~**H**-dependent~~ random!

For all round $j = 0, \dots, n-1$:

$C_j \leftarrow \{0,1, \dots, n-1\}$

X_{C_j}



X_0

Challenger

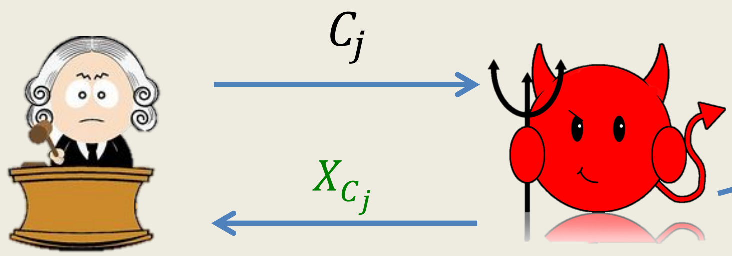
Adversary

<u>Adversary's goal:</u> Reduce its own CMC for answering all challenges!

CMC = Cumulative Memory Complexity = $\sum_{t=0}^{T} \mathrm{Memory}(t)$

Round game – Naïve strategy

For round $j = 0$ to $n-1$:



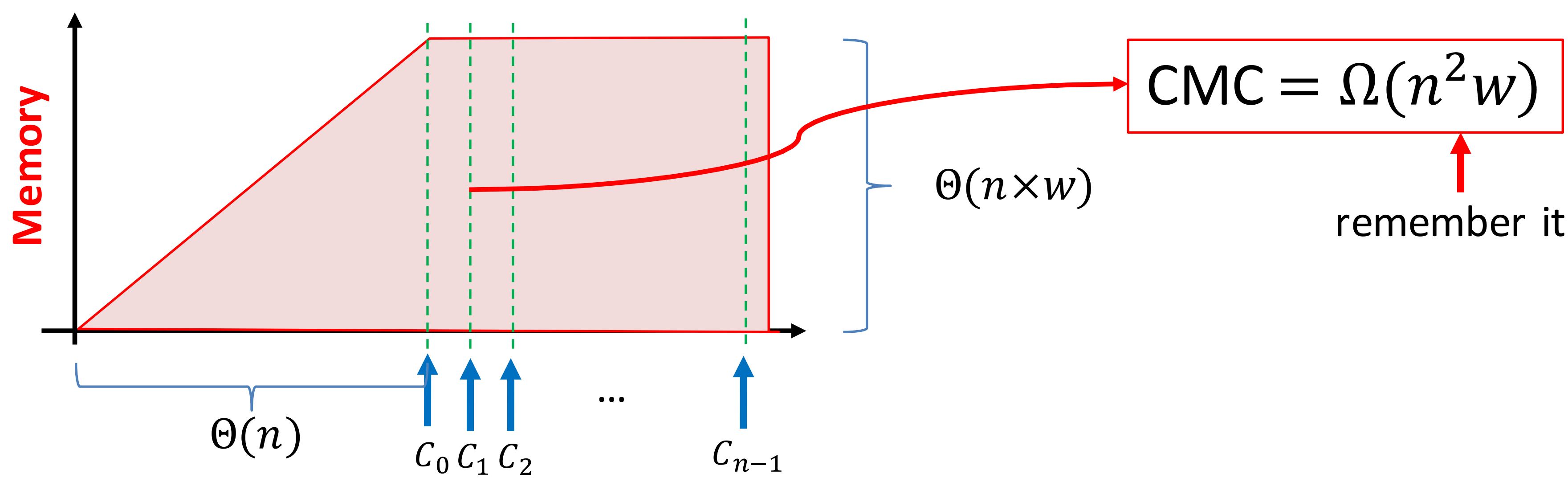
Init:

$\text{Mem}[0] \leftarrow X_0$

for $i = 1, \ldots, n$ **do**

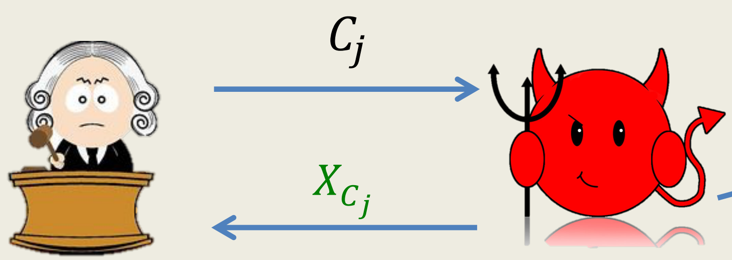
$\quad \text{Mem}[i] \leftarrow \mathbf{H}(\text{Mem}[i-1])$

Upon challenge C_j: **return** $\text{Mem}[C_j]$



Round game – Memory-less strategy

For round $j = 0$ to $n - 1$:



Upon challenge C_j:

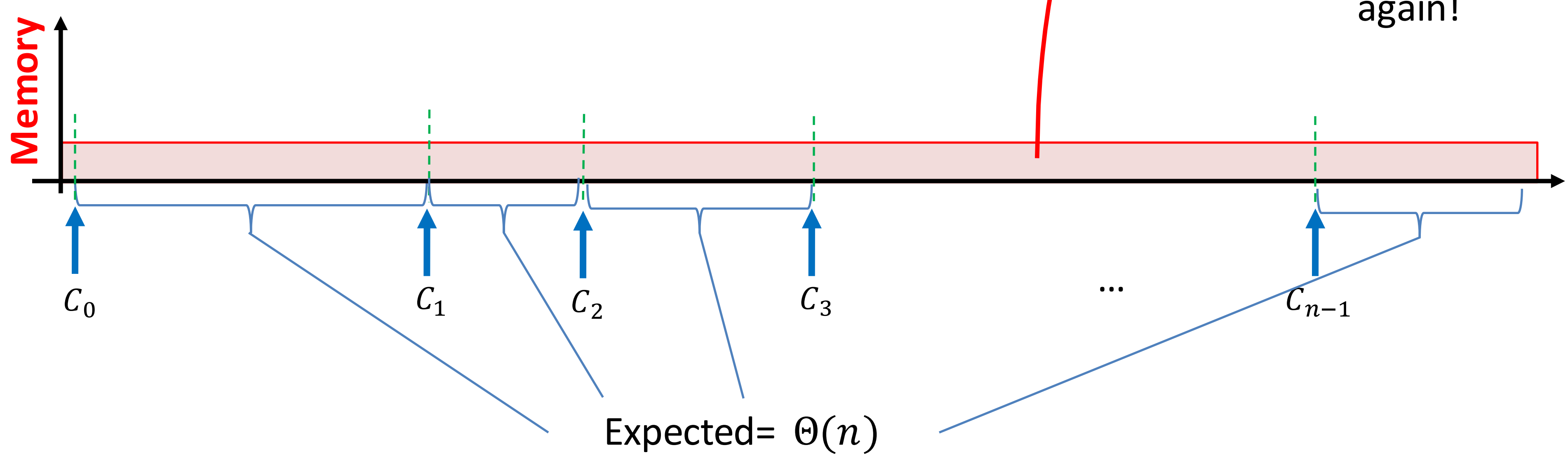
$X = X_0$

for $i = 1, \dots, C_j$ **do** $X \leftarrow \mathbf{H}(X)$

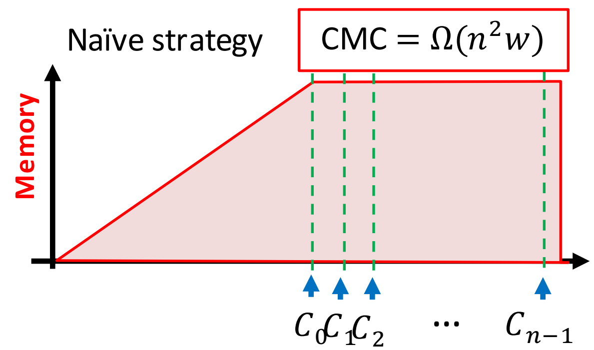
return X

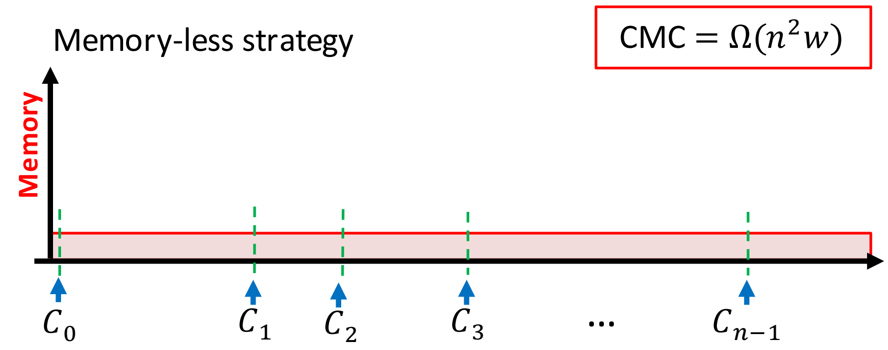
$\boxed{\mathrm{CMC} = \Omega(n^2 w)}$

again!



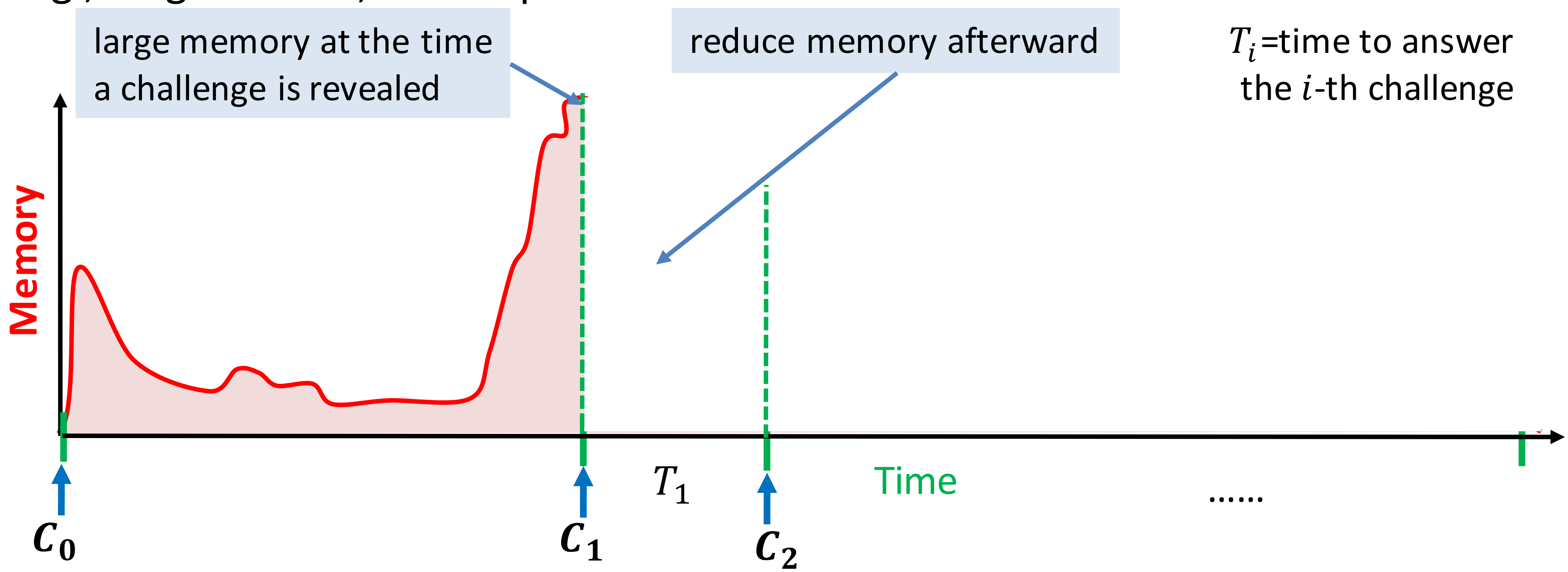
Previous two strategies are special cases: consistent memory size





More general strategy: memory consumption can vary a lot

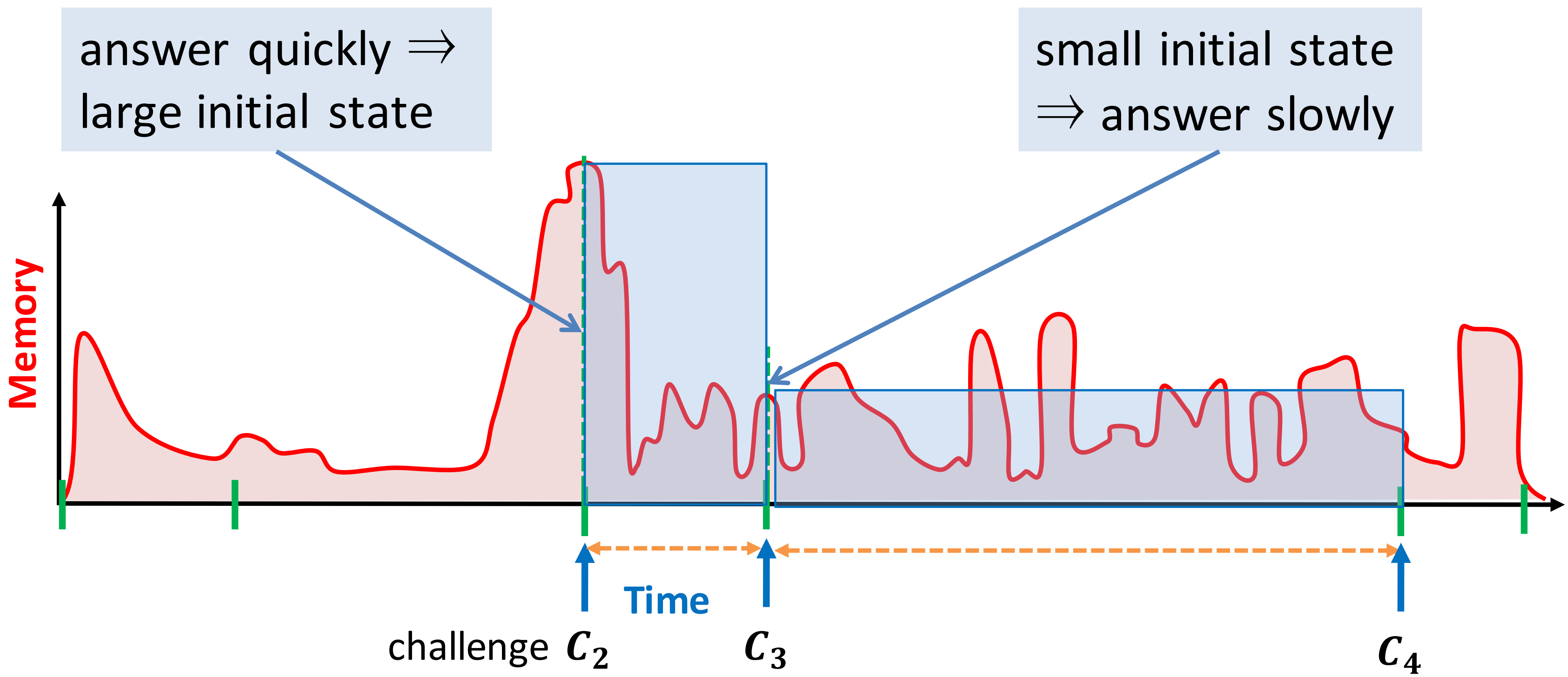
e.g., forget values, re-compute afterward



Goal: prove CMC $= \Omega(n^2 w)$!

Memory hardness: intuition

Intuition: Answering challenge <u>fast</u> requires large state!



Single-shot memory-time trade-off

Simplifying assumption: upon learning challenge C_j, adversary only stores p of the values $X_0, \dots, X_{n-1}$

$n = 10, p = 3$

stored

C_j

$$\boxed{X_0} \xrightarrow{\mathbf{H}} X_1 \longrightarrow X_2 \longrightarrow \boxed{X_3} \longrightarrow X_4 \longrightarrow X_5 \longrightarrow X_6 \longrightarrow \boxed{X_7} \longrightarrow X_8 \longrightarrow X_9$$

$\text{dist} = 3$

Fact: Avg-distance from X_{C_j} to closest stored X_i preceding X_{C_j} is $n/2p$

Regardless of parallelism, as computation of X-values is inherently sequential!

Expected <u>time</u> to answer the challenge is $n/2p$

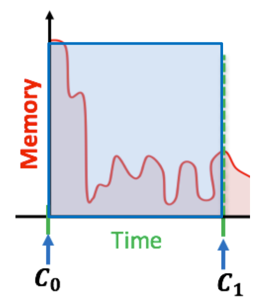
$\approx |\text{memory}|$

How to translate this intuition into a memory-hardness proof for ROMix?

Three technical barriers:

1. Adversary stores <u>arbitrary information</u>
   e.g., XOR of X_i values, halves of X_i, reconstruct information adaptively on challenges, etc.
   [ACKKPT16] considered restricted strategies and exhibited round games where general storing strategies can help!
2. Memory variation during computation
   single-shot memory-time trade-off not enough!
   [ACKKPT16] only shows CMC $= \Omega(\frac{n^2 w}{\log^2(n)})$
3. $\mathbf{H}$-dependent challenges, as opposed to truly random **see the paper!**

Focus on 1 and 2



Roadmap

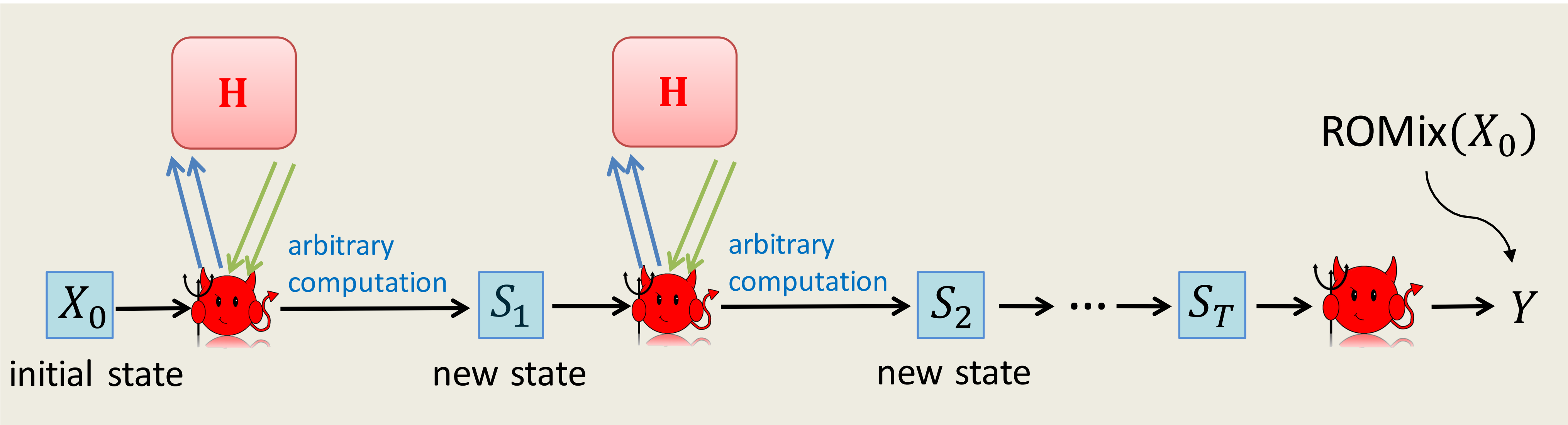
1. The Scrypt function

2. Optimal memory hardness of Scrypt
   Model, theorem, and proof approach

3. Conclusions

The parallel random oracle model

[Alwen and Serbinenko, STOC '15]



At each step: Adv asks one batch of parallel **H** queries + performs unbounded computation

Goal of adv: minimize CMC $= \sum_{i=1}^{T} |S_i|$

Main Theorem.

For any adversary **A** evaluating **ROMix**,

$$\mathbf{CMC}(\mathbf{A}) \geq \frac{1}{25} \cdot n^2 \cdot (w - 4 \cdot \log(n))$$

w/ overwhelming probability over the choice of $\mathbf{H}$.

The $4\log(n)$ loss is inherent in the proof.

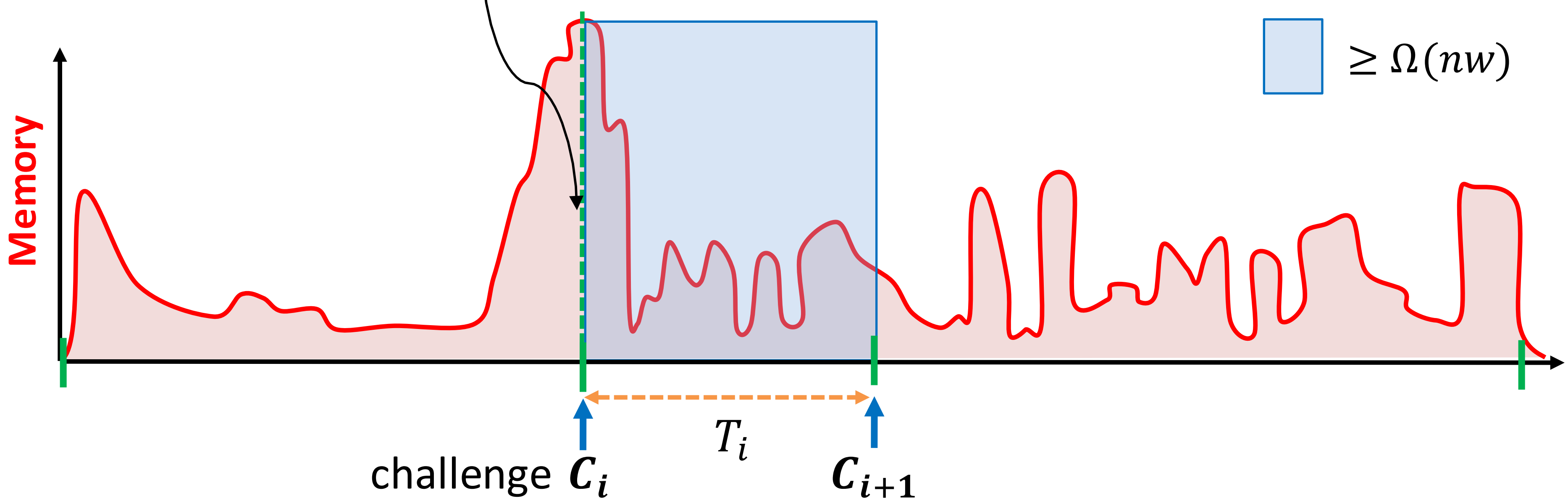
$\Omega(n^2 w)$ clearly best possible for any construction making n queries to $\mathbf{H}$.

Naïve strategy: Make n calls, remember all outputs

Proof strategy: step 1

Green (memory usage at this step) is inversely proportional to orange (T_i)

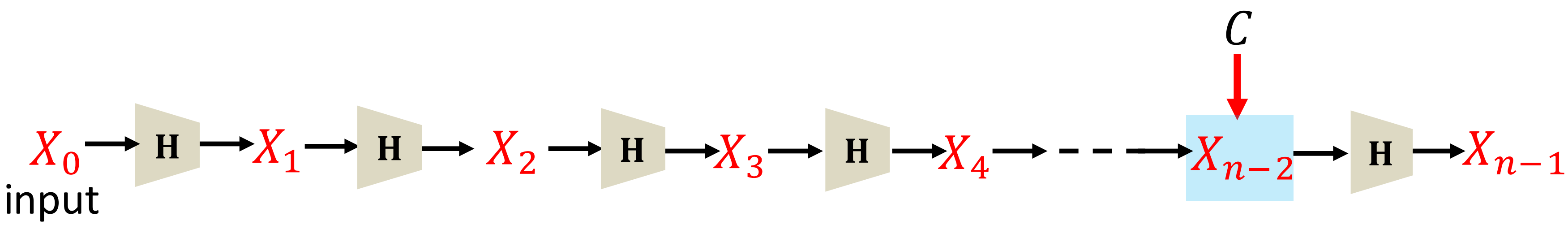
T_i=time to answer the i-th challenge



Memory-time trade-off $\Rightarrow$ lower bound on memory

The memory-time trade-off holds true for adv ~~storing X-values~~ even if the adv stores arbitrary information!

Single-shot memory-time trade-off



state

$A^{\mathbf{H}}(\ Z\ , C) \xrightarrow{T \text{ steps}}$ recover X_C

Z: arbitrary computation on $\mathbf{H}$-outputs

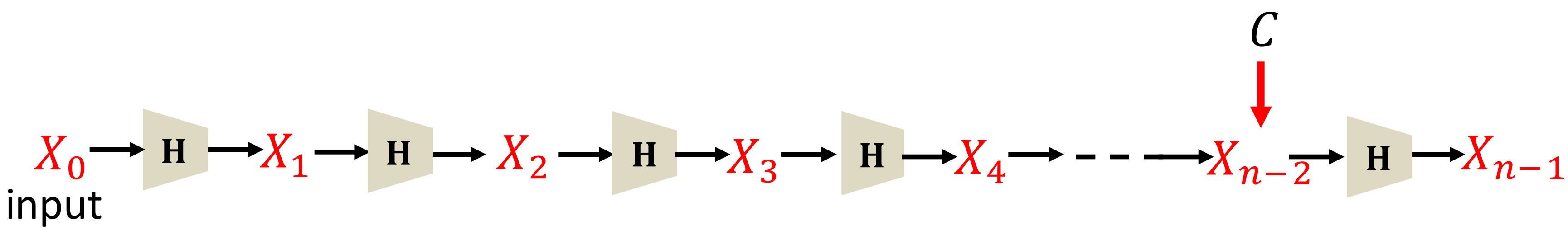
- E.g., pre-computation of $\mathbf{H}$'s entries, XOR of $\{X_i\}$ values, halves of X_i

Goal: Lower bound $|Z|$ as function of T and n

[ACKKPT16]: computation on $\mathbf{H}$-outputs can help in some round games

[This result]: computation on $\mathbf{H}$-outputs cannot help for Scrypt!

Single-shot memory-time trade-off



$$A^{\mathbf{H}}(\boxed{\;Z\;}, C) \xrightarrow{T \text{ steps}} \text{recover } X_C$$

Lemma. For all A, for most $\mathbf{H}$, if $|Z| \approx pw$ bits

$$\Pr_C\left[T > \frac{n}{2p}\right] > \frac{1}{2}$$

Lemma. For all A, for most $\mathbf{H}$, if $|Z| \approx pw$ bits

$$\Pr_{C}\left[T > \frac{n}{2p}\right] > \frac{1}{2}$$

Proof idea:

If adversary $A^{\mathbf{H}}(Z, C)$ answers too fast for most challenges C

$A^{\mathbf{H}}(Z, C)$ can output or query many X_i values w/o querying $\mathbf{H}$ first

Can compress the oracle $\mathbf{H}$ using state Z

Cannot be true for too many **H**: random oracle is incompressible

[Dwork, Naor and Wee, Crypto'05], [Alwen and Serbinenko, STOC '15]

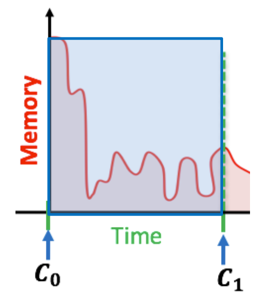
Proof strategy: step 2

Technical barriers:

1. Adversary stores arbitrary information

   Single-shot memory-time trade-off for arbitrary adv

2. Memory variation during computation



Single-shot memory-time trade-off

Optimal CMC lower bound for the round game

Generalize

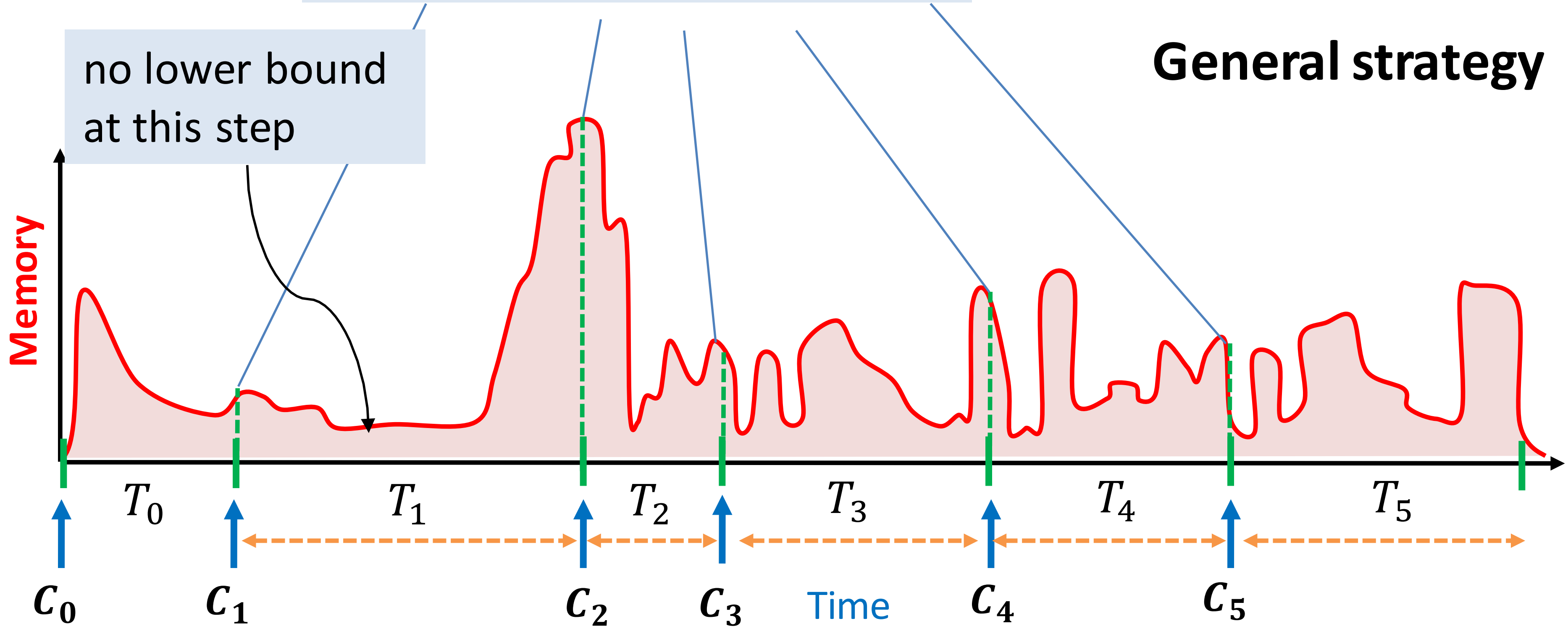
CMC lower bound

Lemma. $\Pr_C\left[p > \frac{n}{2T}\right] > \frac{1}{2}$

when learning the i-th challenge
$|\text{memory}| \geq \frac{nw}{2T_i}$

T_i=time to answer the i-th challenge

General strategy



memory-time trade-off $\Rightarrow$ memory lower bound for the step <u>right before</u> the challenge is revealed

CMC = ???

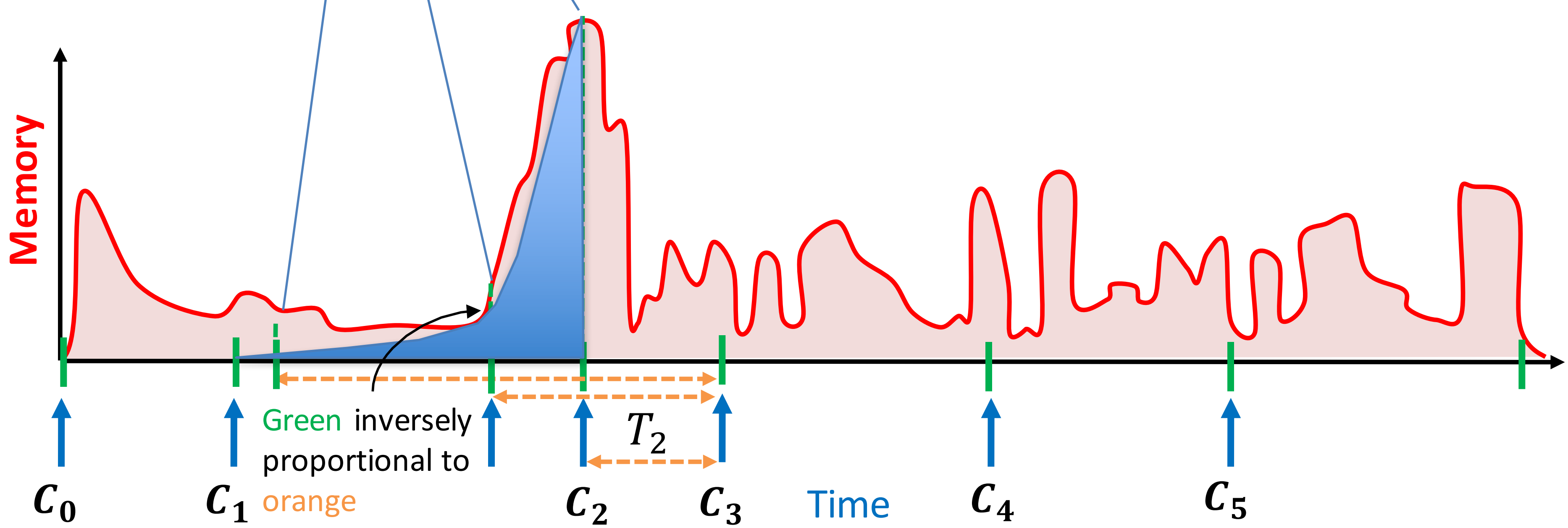
CMC lower bound

Lemma. $\Pr_{C}\left[p > \frac{n}{2T}\right] > \frac{1}{2}$

mem at every step $\geq$ funcs of $\boldsymbol{n}$ and **time** to answer the <u>next</u> challenge

T_i=time to answer the i-th challenge

General strategy



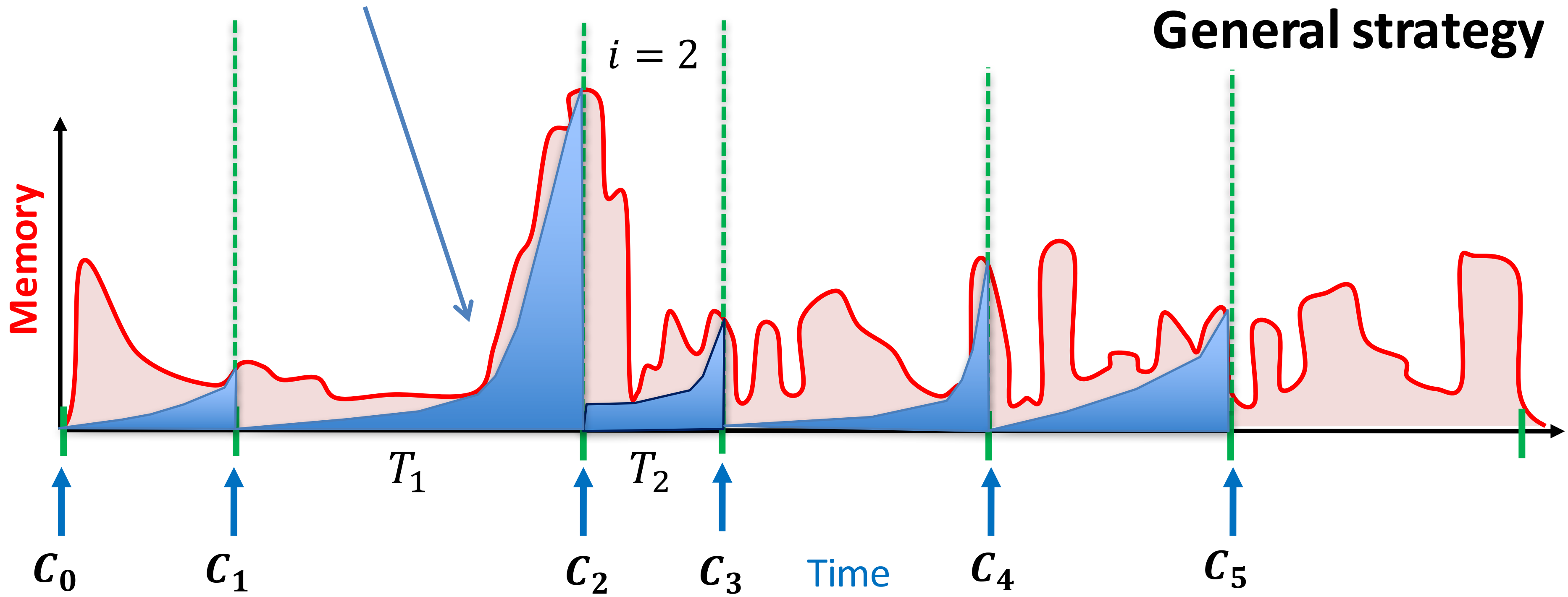
Similar trade-off holds for <u>every step</u> before challenge is revealed

CMC lower bound

Lemma. $\Pr_C \left[p > \frac{n}{2T} \right] > \frac{1}{2}$

During round $i-1$:
Sum of memory $\geq \frac{nw}{2} \ln\left(1 + \frac{T_{i-1}}{T_i}\right)$

T_i=time to answer the i-th challenge



By adding lower bounds over rounds from 0 to $n-1$, we have CMC $= \Omega(n^2 w)$

 $= \Omega(n^2 w)$

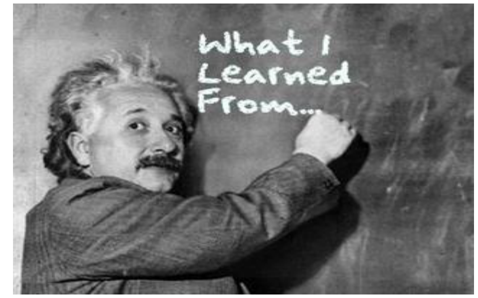
Roadmap

1. The Scrypt function

2. Optimal memory hardness of Scrypt

3. Conclusions

Summary



- Scrypt is maximally memory hard
  - First optimal memory-hardness proof.
  - Validates a practical MHF design.

- Open problem
  - Optimal memory hardness proof for Argon2d?

Thank you! – Merci!

https://eprint.iacr.org/2016/989