Memory-Hard Functions

Joël Alwen – IST Austria

Theory: Quo Vadis?

Goal 1: Inform future research direction aiming it in a “useful” direction.

Goal 2: Raise awareness of potential implications of recent results for Password-hashing standardization.

Some example questions to keep in mind…

1. Computational Model: Too weak / strong for security statements / attacks? If so what is wrong?
2. Complexity Measures: Too weak / strong for security statements / attacks?
3. Statements: Are the type of statements being proven relevant to practice? What more would we like to know?

MHF a la Percival

- Observation: Computation is cheaper for custom hardware (e.g. ASICs) then general purpose CPUs.
- Goal: Functions which require as much memory as possible for a given number of computational steps even in parallel.

  $\Longrightarrow$ Decrease "evaluations/second per dollar" advantage of ASICs.
- Recall: Area x Time (AT) complexity of a circuit evaluating $f \approx$ dollar cost per unit of rate (rate = # of f evaluations / sec).
- Percival: Since high speed memory is expensive in circuits replace "area" with "space" (i.e. memory).

MHF a la Percival

Definition [Per09]:

An MHF is a function f with hardness parameter n such that f_n:

1. can be computed on a Random Access Machine (RAM) in T*(n) time.
2. can not be computed on a Parallel RAM (PRAM) with S(n) space and processors and T (n) time such that $T(n) \times S(n) = O(n^{2-c})$ for some $c > 0$.

Data-(in)dependence

- Is the honest evaluation algorithms memory access pattern input-dependent?
  - Yes: data-dependent MHF (dMHF). Example: scrypt, Argon2d.
  - No: data-independent MHF (iMHF). Example: Argon2i, Balloon Hashing.

iMHF Advantage: Implementations easier to secure against certain timing attacks.

Overview

1. Intuitive goals of an MHF.
2. Theory for proving security.
3. Attacking an MHF.

Computational Model

Problem: Proving complexity lower-bounds is hard.

Fortunately almost all proposed MHFs based on compression functions.

Idea: Use (Parallel) Random Oracle Model.

Parallel Random Oracle Model

- Computational Model: PROM
  - Algorithms **A** invoked iteratively.
  - At iteration *i* do:
    1. Get input state s_{i-1} (state = arbitrary bit-string).
    2. Perform arbitrary computation.
    3. Make one <u>batch</u> of queries to $\mathcal{H}$. (i.e. make parallel queries.)
    4. Perform arbitrary computation.
    5. Output new state s_i.
  - Set s_0 to be the input to the computation.
  - Repeat until **A** produces a special output state s_z = result of computation.

Parallel Random Oracle Model

Intuition: Good for proving security because…

1. Rather permissive $\Rightarrow$ security proofs carry more weight.
   - Arbitrary non-RO dependent computation allowed for free at each step.
   - Memory only measured between calls to RO.
   - Any PRAM algorithm is a PROM algorithm (at no added cost).

2. Proving <u>exact</u> lower-bounds with <u>reasonable constants</u> is tractable.

ST-Complexity

- Computational Model: PROM
  - Algorithms **A** invoked iteratively.
  - At iteration *i* do:
    1. Get input state s_{i-1} (state = arbitrary bit-string).
    2. Perform arbitrary computation.
    3. Make one <u>batch</u> of queries to $\mathcal{H}$. (i.e. make parallel queries.)
    4. Perform arbitrary computation.
    5. Output new state s_i.
  - Repeat until **A** produces a special output state s_z = result of computation.

- Cost(execution) := $\max_{i \in [z]} |s_i| \times z$

computation time

bit length largest state

Sanity check? "Cost(execution) is high ⇒ AT(execution) is high
⇒ expensive to implement in ASIC or FPGA."

ST-Complexity of a Function

- Complexity of an algorithm **A** on input x:

$$ST(\mathbf{A},x) \geq c \Leftrightarrow \Pr[\, ST(exec(\mathbf{A}^H(x)) \geq c] \geq 1\text{- negligible}$$

over the choice of RO.

Intuition: "On input x algorithm A almost always runs with ST-complexity at least c."

- Complexity of a function f^H:

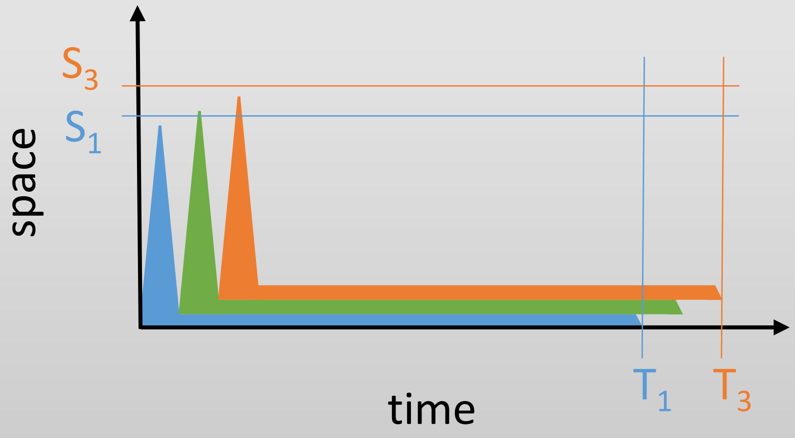
$$ST(f) = \min_{\mathbf{A},x} \{\, ST(\mathbf{A},x) \,\}$$

minimum over all alg. **A** and inputs x computing f(x).

Intuition: ST complexity of the best algorithm computing *f* on its favorite input x.

Amortized and Parallelism

- Problem: for parallel computation ST-complexity can scale badly in the number of evaluations of a function.



$$ST_1 = S_1 \times T_1 \approx S_3 \times T_3 = ST_3$$

cost of computing *f* once

cost of computing *f* three times

In fact ∃ function *f* (consisting of n RO calls) such that: $ST\left(f^{\times\sqrt{n}}\right) = O(ST(f))$

Amortized ST-Complexity of a Function

- Amortized ST-complexity of a function f

Intuition: "The ST-complexity per I/O pair of the best evaluation algorithm for f running on its favorite set of inputs."

$$\mathrm{aST}(f) = \min_{m\in\mathbb{N}} \frac{ST(f^{\times m})}{m}$$

- Sanity check? "If aST(f) is large $\Longrightarrow$ Implementing brute-force attack in an ASIC is expensive."

Examples of Results

- Argon2i (and Balloon Hashing) security proofs:
  - For any choice of mem-cost σ and time-cost τ = 1

$$aST(\text{Argon2i}_{\sigma,\tau}) \geq \Omega(\sigma^{1.666})$$

with probability at least $1-o(\sigma^{-3})$ over choice of RO and salt.

Note: larger τ can only give worse complexity because

Recall: In practice $\sigma \approx 2^{24}$ for 1GB of memory $\Longrightarrow \sigma^{-3} \approx 2^{-68}$

- Construct an iMHF f_n with:
  1. f_n computable in n Time and n Space in (sequential) ROM.
  2. $aST(f_n) = \Omega\left(\frac{n^2}{\log n}\right)$ in the PROM for all "reasonable" adversaries.

"completeness"

"security proof"

Overview

1. Intuitive goals of an MHF.
2. Theory for proving security.
3. Attacking an MHF.

When is an Evaluation Algorithm an “Attack”?

Intuitive Answer: An evaluation algorithm **A** is an “attack” if it has lower complexity then the honest algorithm **N**.

More fine grained: Quality(**A**) = complexity(**A**) / complexity(**N**).

But which “complexity”?

- aST considers only memory. What about cost of implementing RO?
- aST ≈ cost of building ASIC. What about cost of <u>running</u> device?

Two Stricter Complexity Measures

1) Amortized-Area/Time Complexity (a-AT) ≈ cost of building ASIC.

- Area: accounts for memory needed on chip <u>and RO cores</u>.

2) Amortized-Energy (aE) Complexity ≈ cost of running ASIC.

- Accounts for electricity consumed while storing values and RO evaluations.

amortized-AT Complexity

- Recall PROM: At iteration i make batch of queries q_i and store state s_i.
- Initial Idea: $\quad aAT(execution) := \max_i(|s_i|) + \max_j(q_j)$.

# of memory cells needed to run execution.

# of RO cores needed to run execution.

amortized-AT Complexity

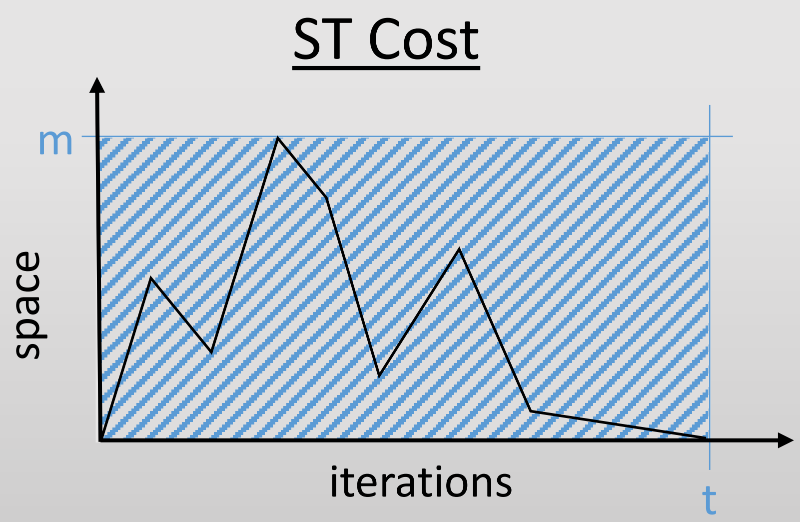
- Recall PROM: At iteration i make batch of queries q_i and store state s_i.
- Initial Idea: aAT(execution) := $\max_i(|s_i|) + \max_j(q_j)$.
- Problem: Storing 1-bit requires much less area then implementing, say, SHA1.
- Solution:

  "Core-memory area ratio" R := area(1-bit-storage) / area(RO)
- Parametrized Complexity:

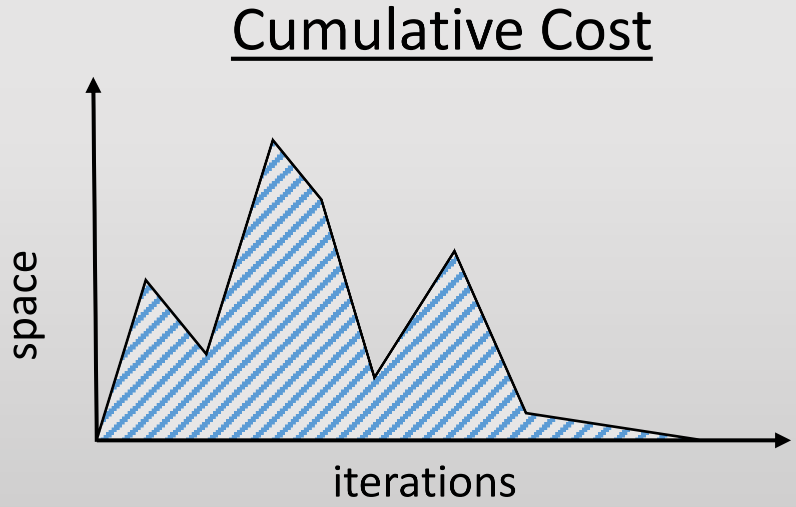
$$aAT_R(\text{execution}) := \max_i(|s_i|) + R*\max_j(q_j)$$

Energy Complexity

- Intuition: Only pay for memory that is being actively used.
- Idea: Define the complexity to be area under the “memory curve”.



$\longmapsto$



Energy Complexity

- Similarly for RO calls: Only pay for actually making a call.
- Unit of time: “tock” = time it takes to evaluate the RO.
- Unit of measure: milli-Watt-tock (mWt) = Electricity required to store 1-bit for one tock.
- “Core-memory energy ratio” R’ = mWt requires to evaluate the RO on one input.

$$aE_{R'}(\text{execution}) := \sum |s_i| + R' \times |qi|$$

Asymptotic Example: Argon2i

- [AB16] For mem-cost σ and time-cost τ such that $\sigma \times \tau = n$

$$\mathrm{aAT_R(Argon2i)} = O(n^{1.75} \log n + Rn^{1.25})$$

$$\mathrm{aAT_R(Honest\text{-}Alg)} = \Omega\left(\frac{n^2}{\tau} + Rn\right)$$

on expectation over the choice of salt and RO.

- Same for energy complexity.
- Similar (or stronger) asymptotic attacks for Catena-BRG, Catena-DBG, Balloon Hashing 1, 2 & 3, Lyra2, Gambit, Rigv2.

Asymptotic Example: General Upper-Bound

- Any MHF making n calls to a RO has complexity

$$\mathrm{aAT}_R(\mathrm{Argon2i}) = O\left(\frac{n^2}{\log n} + R \times n\right)$$

$\Longrightarrow$ At least in principle Percival's goal of n^2 is impossible for an iMHF.

Exact Example: Argon2i

- For mem-cost σ and time-cost τ such that σ × τ = n

$$\mathrm{aAT}_{\mathrm{R}}(\mathrm{Argon2i}) \leq 2n^{1.75}\left(5 + \frac{\log n}{2} + \tau + \frac{R}{n^{.75}} + \frac{R}{n^{.5}} + \frac{2R}{n}\right)$$

- Similar for $\mathrm{aE}_{\mathrm{R'}}$(Argon2i)

Exact Example: Argon2i

- What does this mean for standardizing Argon2i?

- Some arguments for “This is only a theoretical attack.”
    1. aAT complexity doesn’t charge for computation not involving a call to the RO so real complexity may be far bigger.
    2. Setting $n=2^{24}$, $R=3000$ and $\tau \geq 2$ gives worse complexity than honest alg.
    3. It needs unrealistic amounts of parallelism.

- First: besides calling RO practically no further computation done (In fact: potentially less than honest algorithm…)

Exact Example: Argon2i

- Second: Set $n=2^{24}$, $R=3000$ and $\tau \geq 2$ then this is <u>not</u> an attack.
- Conceptually: By increasing τ we increase computation while keeping memory the same. Intuitively it becomes “less memory-hard”.
- No attempt … n made … e:
  - for specific parameter ranges
  - minimizing exact security (vs. asymptotic)

1GB Mem

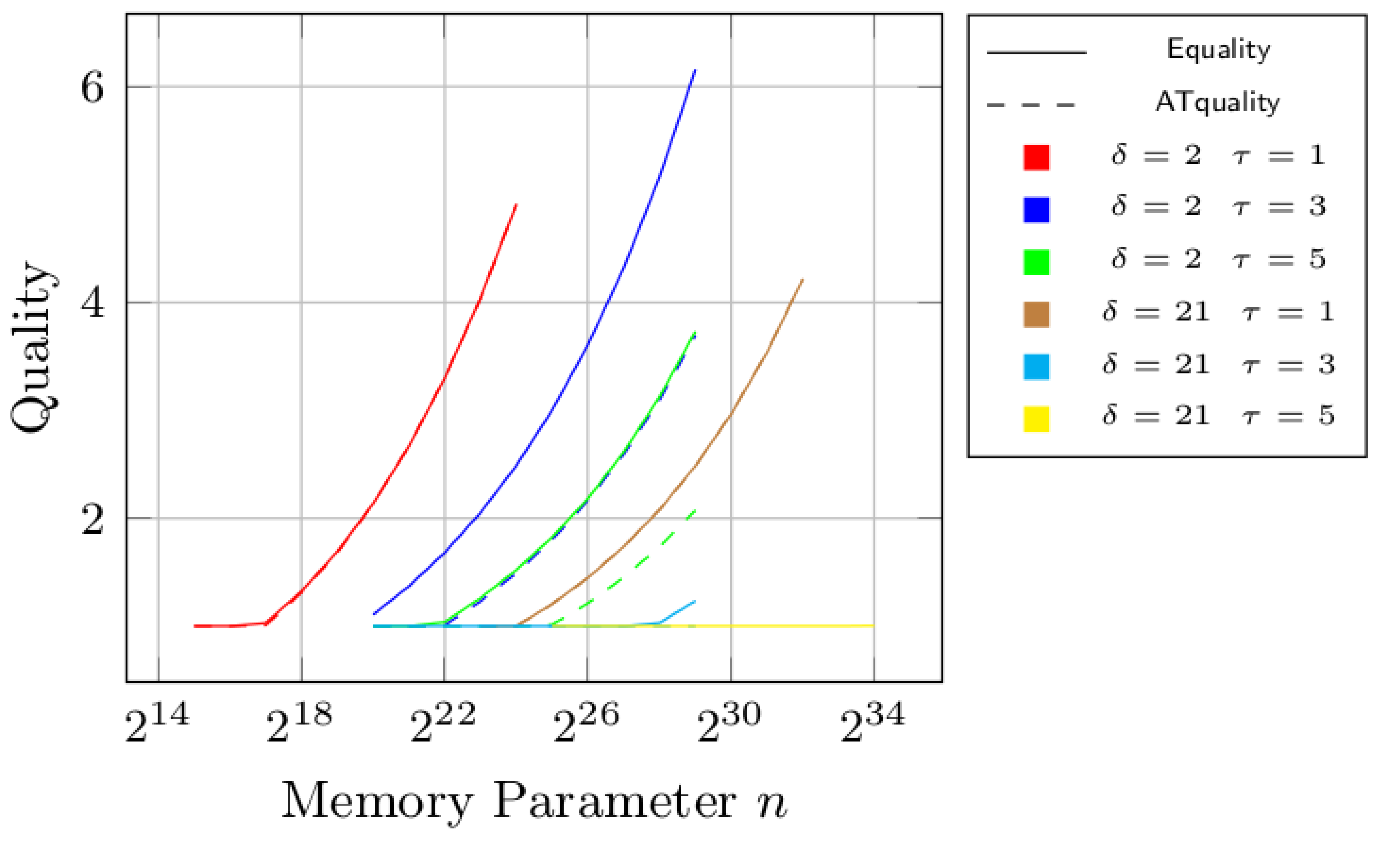
Passes over memory

Optimizing Analysis for Concrete Parameters

Argon2: indegree δ = 2

- For 1GB memory ($n=2^{24}$) actually need τ ≥ 6.
- For each quadrupling of memory need 1 more pass on memory.

Further optimizations of the <u>analysis</u> possible? Most likely…



(a) Argon2i and SB

Third: Can Actually Build This Attack?

- Example: Compute 2^{12} instances in time 2^{25}.
- Recall: In Argon2i RO = Blake-512 $\approx .1\ \mathrm{mm}^2$.
- Layout: 1 “big” ASIC + 256 “light” ASICs.
- Big ASIC: 2^{12} Blake-512 Cores $\approx 410\ \mathrm{mm}^2$.
- Total memory on device $\approx$ 50 GB.
- These aren’t unrealistic requirements for an attacker with decent budget...

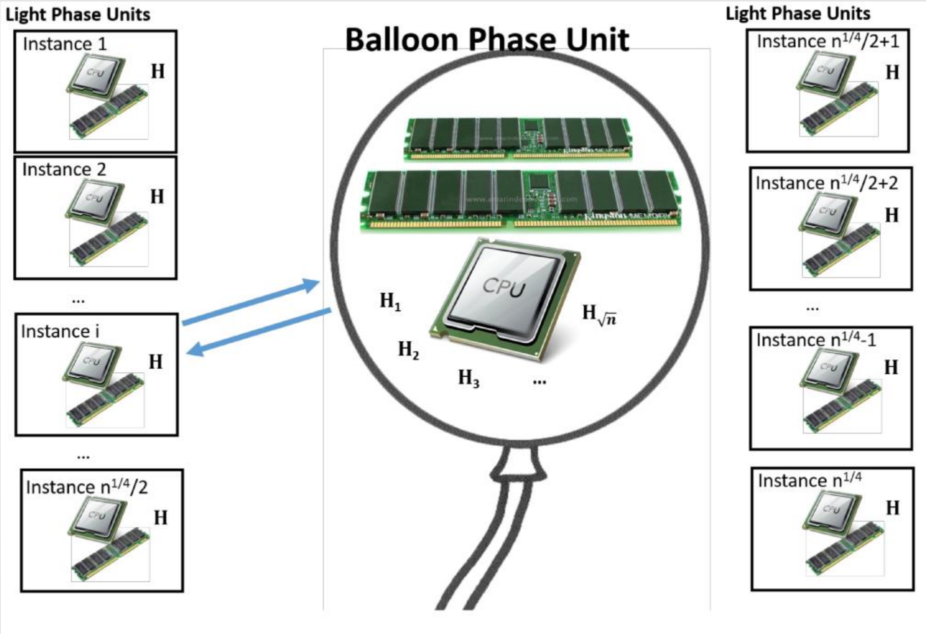


Figure 2: Argon2i: Attack Architecture

Conclusions

Argon2i

- In it’s current form attack is neither “apocalyptic” nor “only theoretical”.
- Could it improve: my opinion is “very likely yes” both asymptotically and exact.
    - See history of block ciphers and hash functions. Attacks tend to improve…
- What else could we even use?
    - Balloon Hashing?
    - Something new?

Theory: Quo Vadis?

- You tell me!
    - What do you think of the PROM?
    - How about aAT and Energy complexity?
    - Are the statements being proven somewhat meaningful?
    - What else could theory try to consider?

Questions? Comments?