Trapdoor Memory-Hard Functions

Benedikt Auerbach

Christoph Günther

Krzysztof Pietrzak

Eurocrypt 2024

Institute of
Science and
Technology
Austria

FWF Austrian Science Fund

SPyCoDe

This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/F85.

Memory-hard functions (MHFs)

- Need moderately-hard functions, e.g., password hashing, PoW, ...

Memory-hard functions (MHFs)

- Need moderately-hard functions, e.g., password hashing, PoW, ...
- Computationally hard functions are *not egalitarian* (easier on specialized hardware, e.g., ASICs)

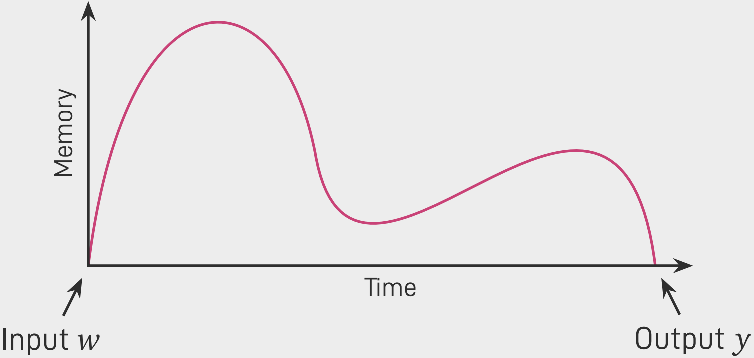
Memory-hard functions (MHFs)

- Need moderately-hard functions, e.g., password hashing, PoW, ...
- Computationally hard functions are *not egalitarian* (easier on specialized hardware, e.g., ASICs)
- **Memory-hardness:** Evaluation cost dominated by memory usage and not computation

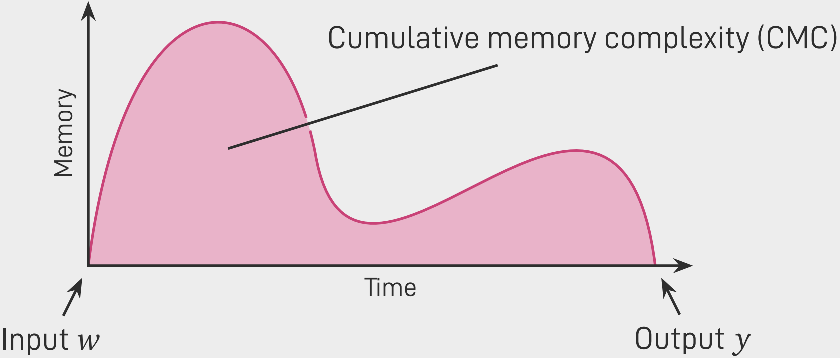
Memory-hard functions (MHFs)

- Need moderately-hard functions, e.g., password hashing, PoW, ...
- Computationally hard functions are *not egalitarian* (easier on specialized hardware, e.g., ASICs)
- **Memory-hardness:** Evaluation cost dominated by memory usage and not computation
- Scrypt, Argon2 family, DRSample, ...

Memory measure



Memory measure



Scrypt (Percival, BSDCan’09)

- Sequential algorithm $\mathrm{Eval}(w) \rightarrow y$

Scrypt (Percival, BSDCan'09)

- Sequential algorithm $\mathrm{Eval}(w) \to y$
- Mode of operation for a hash function $H\colon \{0,1\}^* \to \{0,1\}^\ell$

Scrypt (Percival, BSDCan'09)

- Sequential algorithm $\mathrm{Eval}(w) \longrightarrow y$
- Mode of operation for a hash function $H\colon \{0,1\}^* \longrightarrow \{0,1\}^\ell$
- Memory-hardness parameter n

Scrypt (Percival, BSDCan’09)

- Sequential algorithm $\mathrm{Eval}(w) \rightarrow y$
- Mode of operation for a hash function $H\colon \{0,1\}^* \rightarrow \{0,1\}^\ell$
- Memory-hardness parameter n
- Eval's CMC is $\Theta(n^2 \ell)$

Scrypt (Percival, BSDCan’09)

- Sequential algorithm $\mathrm{Eval}(w) \to y$
- Mode of operation for a hash function $H\colon \{0,1\}^* \to \{0,1\}^\ell$
- Memory-hardness parameter n
- Eval's CMC is $\Theta(n^2\ell)$

Theorem (Alwen et al., EC’17)

Any (parallel) algorithm evaluating Scrypt has a CMC of $\Omega(n^2\ell)$ in the random oracle model

Scrypt’s Eval

$w = w_0$

Scrypt’s Eval

$$w = w_0 \xrightarrow{H(w_0)} w_1$$

Scrypt's Eval

$$w = w_0 \xrightarrow{H(w_0)} w_1 \xrightarrow{H(w_1)} w_2$$

Scrypt’s Eval

$$\begin{tikzcd}
w = w_0 \arrow[r, "H(w_0)"] & w_1 \arrow[r, "H(w_1)"] & w_2 \arrow[r, dashed] & w_{n-1}
\end{tikzcd}$$

Scrypt’s Eval

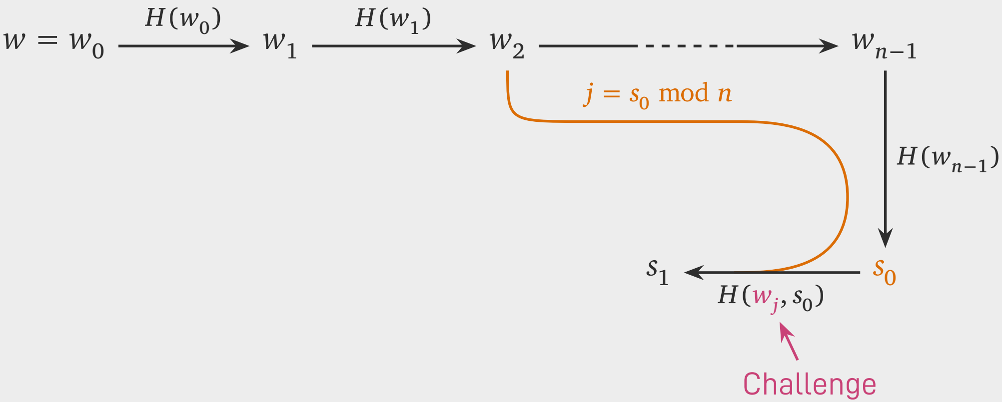
$$\begin{tikzcd}
w = w_0 \arrow[r, "H(w_0)"] & w_1 \arrow[r, "H(w_1)"] & w_2 \arrow[r, dashed] & w_{n-1} \arrow[d, "H(w_{n-1})"] \\
 & & & s_0
\end{tikzcd}$$

Scrypt's Eval

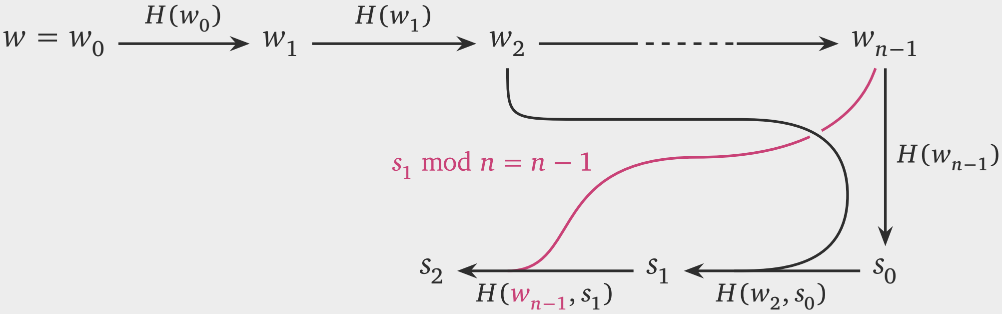
$$\begin{tikzcd}
w = w_0 \arrow[r, "H(w_0)"] & w_1 \arrow[r, "H(w_1)"] & w_2 \arrow[rr, dashed] & & w_{n-1} \arrow[d, "H(w_{n-1})"] \\
& & & s_1 & s_0 \arrow[l, "H(w_j, s_0)"']
\end{tikzcd}$$

Challenge

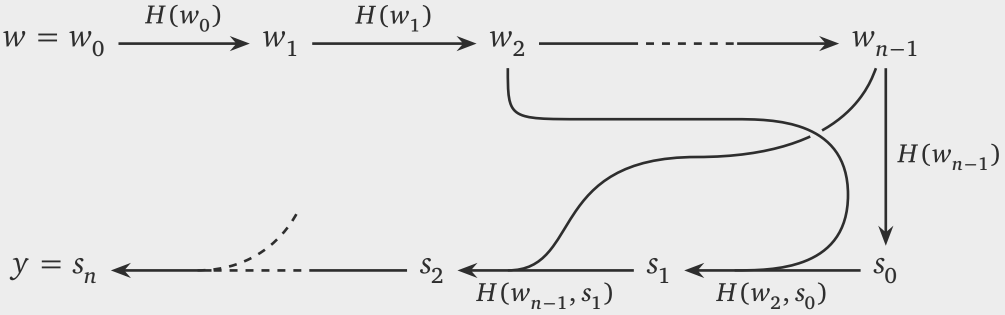
Scrypt's Eval



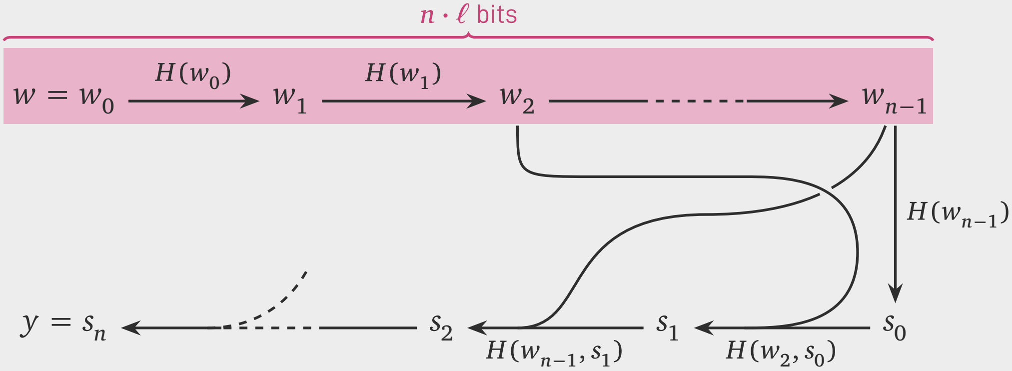
Scrypt's Eval



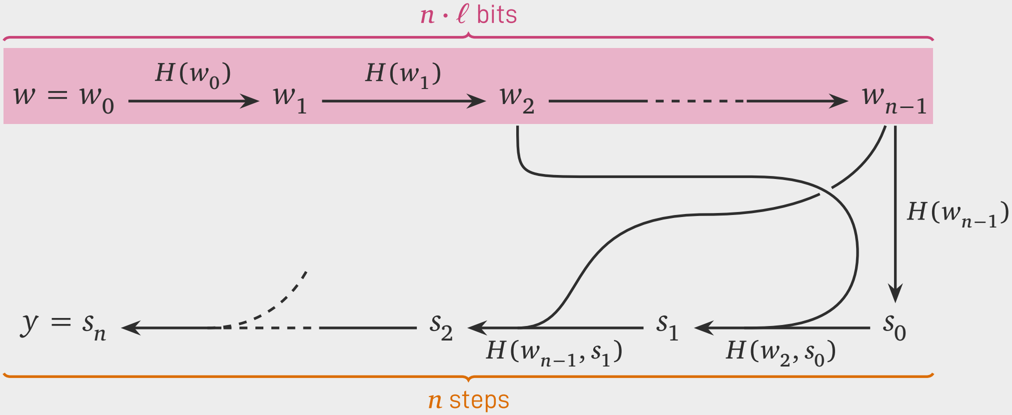
Scrypt's Eval



Scrypt's Eval



Scrypt's Eval



CMC $\approx n \cdot n \cdot \ell$

Trapdoor memory-hard functions

Algorithms

- $\text{Setup}() \rightarrow \text{pp}$
- $\text{Eval}(\text{pp}, w) \rightarrow y$

Trapdoor memory-hard functions

Algorithms

- $\mathrm{Setup}(\,) \to \mathrm{pp}, \mathrm{td}$
- $\mathrm{Eval}(\mathrm{pp}, w) \to y$
- $\mathrm{TDEval}(\mathrm{pp}, w, \mathrm{td}) \to y$

Trapdoor memory-hard functions

Algorithms

- $\mathrm{Setup}(\,) \rightarrow \mathrm{pp}, \mathrm{td}$
- $\mathrm{Eval}(\mathrm{pp}, w) \rightarrow y$
- $\mathrm{TDEval}(\mathrm{pp}, w, \mathrm{td}) \rightarrow y$

Correctness

$$\mathrm{Eval}(\mathrm{pp}, w) = \mathrm{TDEval}(\mathrm{pp}, w, \mathrm{td})$$

Trapdoor memory-hard functions

Algorithms

- $\mathsf{Setup}() \to \mathsf{pp}, \mathsf{td}$
- $\mathsf{Eval}(\mathsf{pp}, w) \to y$
- $\mathsf{TDEval}(\mathsf{pp}, w, \mathsf{td}) \to y$

Correctness

$\mathsf{Eval}(\mathsf{pp}, w) = \mathsf{TDEval}(\mathsf{pp}, w, \mathsf{td})$

Memory-hardness

Evaluation *without* td has high CMC

Trapdoor memory-hard functions

Algorithms

- $\mathsf{Setup}() \to \mathsf{pp}, \mathsf{td}$
- $\mathsf{Eval}(\mathsf{pp}, w) \to y$
- $\mathsf{TDEval}(\mathsf{pp}, w, \mathsf{td}) \to y$

Correctness

$\mathsf{Eval}(\mathsf{pp}, w) = \mathsf{TDEval}(\mathsf{pp}, w, \mathsf{td})$

Memory-hardness

Evaluation *without* td has high CMC

TD-Efficiency

CMC of TDEval $\ll$ CMC of Eval

Use case: spam prevention

Server

Client

Use case: spam prevention

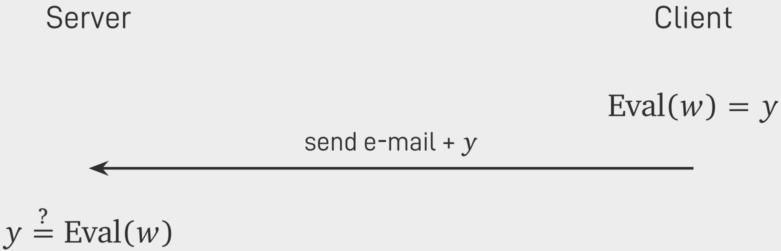
Server

Client

$\text{Eval}(w) = y$

send e-mail + y

Use case: spam prevention



Server

Client

$\mathrm{Eval}(w) = y$

send e-mail + y

$y \overset{?}{=} \mathrm{Eval}(w)$

Use case: spam prevention

Server

Client

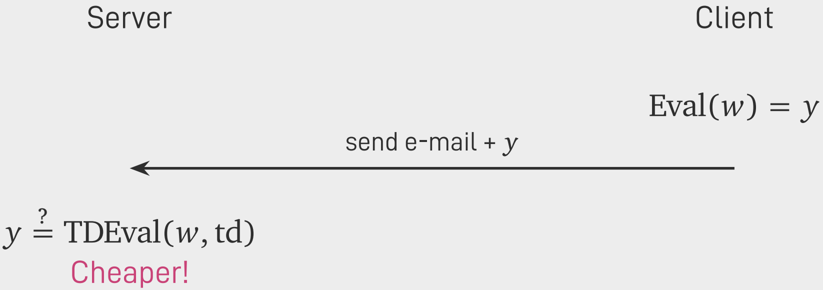
$\mathrm{Eval}(w) = y$

send e-mail + y

$y \overset{?}{=} \mathrm{Eval}(w)$

Expensive!

Use case: spam prevention



Server

Client

$\mathrm{Eval}(w) = y$

send e-mail + y

$y \overset{?}{=} \mathrm{TDEval}(w, \mathrm{td})$

Cheaper!

Diodon (Biryukov & Perrin, AC’17)

- $\mathcal{S}$etup samples hidden-order RSA group
    - $\mathrm{pp} = N$ RSA modulus
    - $\mathrm{td} = \varphi(N)$ Group order

Diodon (Biryukov & Perrin, AC'17)

- $\mathcal{S}$etup samples hidden-order RSA group
  - $\text{pp} = N$ RSA modulus
  - $\text{td} = \varphi(N)$ Group order
- Input is group element $W \in \mathbb{Z}_N^*$

Diodon (Biryukov & Perrin, AC'17)

- $\mathcal{S}$etup samples hidden-order RSA group
  - $\mathrm{pp} = N$ RSA modulus
  - $\mathrm{td} = \varphi(N)$ Group order
- Input is group element $W \in \mathbb{Z}_N^*$
- Scrypt but hashes $w_{i+1} = H(w_i)$ replaced by squares

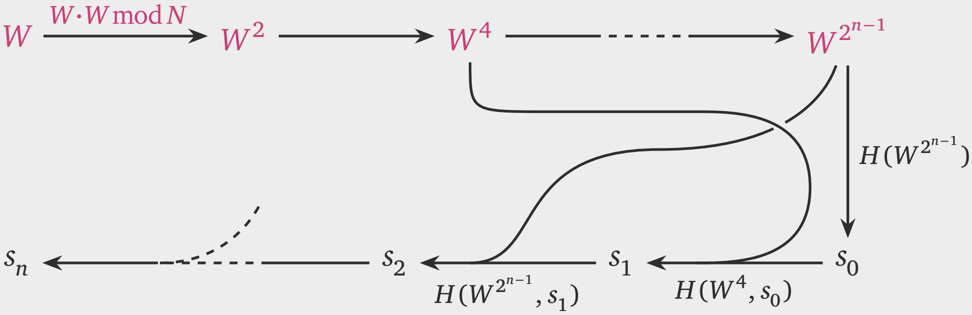
$$W_{i+1} = W_i^2 \bmod N$$

Diodon (Biryukov & Perrin, AC'17)

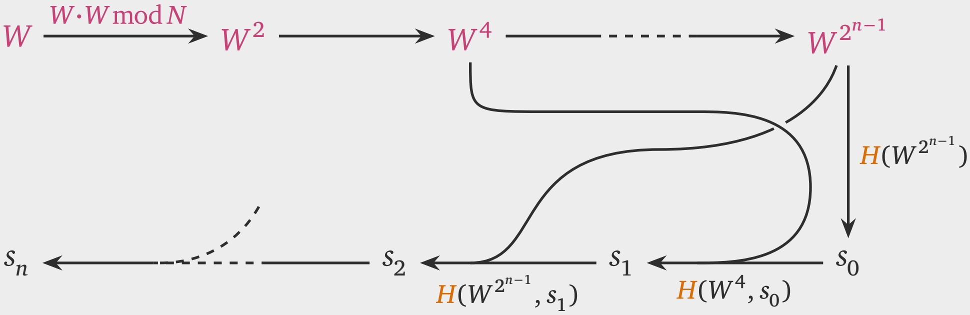
- Setup samples hidden-order RSA group
  - $\mathrm{pp} = N$ RSA modulus
  - $\mathrm{td} = \varphi(N)$ Group order
- Input is group element $W \in \mathbb{Z}_N^*$
- Scrypt but hashes $w_{i+1} = H(w_i)$ replaced by squares

$$\begin{aligned} W_{i+1} &= W_i^2 \bmod N \\ &= W^{2^{i+1}} \bmod N \end{aligned}$$

Diodon’s Eval



Diodon’s Eval



Diodon’s TDEval

$$\begin{tikzcd}
W \arrow[r, "\text{modular reduction \& exponentiation}"] & W^{2^{n-1} \bmod \varphi(N)} \arrow[d, "H(W^{2^{n-1}})"] \\
 & s_0
\end{tikzcd}$$

Diodon’s TDEval

$$\begin{tikzcd}
W \arrow[rr, "\text{mod. red. \& exp.}"] & & W^{2^j} \bmod \varphi(N) \arrow[dl, bend left] & & \\
\cdots & s_{i+1} & & s_i \arrow[ll, "H(W^{2^j}, s_i)"'] & \cdots
\end{tikzcd}$$

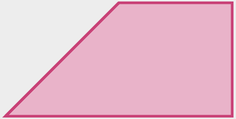
Diodon’s properties

- **Correctness:** By inspection

Diodon’s properties

- **Correctness:** By inspection
- **TD-Efficiency:**

$$\text{Eval} \approx n^2 \log(N)$$



Diodon’s properties

- **Correctness:** By inspection
- **TD-Efficiency:**

$$\text{Eval} \approx n^2 \log(N) \qquad \gg \qquad \text{TDEval} \approx n \log(n) \log(N)^2$$

Diodon’s properties

- **Correctness:** By inspection
- **TD-Efficiency:**

$$\text{Eval} \approx n^2 \log(N) \qquad \gg \qquad \text{TDEval} \approx n \log(n) \log(N)^2$$

- **Memory-hardness:** ???

Diodon’s properties

- **Correctness:** By inspection
- **TD-Efficiency:**

$$\mathrm{Eval} \approx n^2 \log(N) \qquad \gg \qquad \mathrm{TDEval} \approx n \log(n) \log(N)^2$$

- **Memory-hardness:** **Yes (this work)**

Main result

Theorem

Assuming that factoring is hard, Diodon has a CMC lower bounded by

$$\Omega\left(n^2 \log(N) \cdot \frac{1}{\log n} \right)$$

in the (parallel) random oracle and generic group model

Main result

Theorem

Assuming that factoring is hard, Diodon has a CMC lower bounded by

$$\Omega\left(n^2 \log(N) \cdot \frac{1}{\log n}\right)$$

in the **(parallel) random oracle and generic group model**

Main result

Theorem

Assuming that factoring is hard, *Diodon has a CMC lower bounded by*

$$\Omega\left(n^2 \log(N) \cdot \frac{1}{\log n}\right)$$

in the (parallel) random oracle and generic group model

Main result

Theorem

Assuming that factoring is hard, Diodon has a CMC lower bounded by

$$\Omega\left(n^2 \log(N) \cdot \frac{1}{\log n}\right)$$

in the (parallel) random oracle and generic group model

Proof outline

- Scrypt's proof (Alwen et al., EC'17)
    1. Single-challenge time-memory trade-off
    2. Multi-challenge memory complexity lower bound

Proof outline

- Scrypt's proof (Alwen et al., EC'17)
    1. Single-challenge time-memory trade-off
    2. Multi-challenge memory complexity lower bound
- Re-use multi-challenge lower bound (thankfully…)

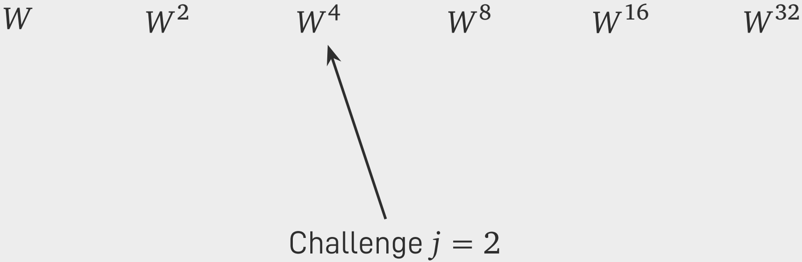
Proof outline

- Scrypt's proof (Alwen et al., EC'17)
  1. Single-challenge time-memory trade-off
  2. Multi-challenge memory complexity lower bound
- Re-use multi-challenge lower bound (thankfully…)
- **Primary hurdle:** Single-challenge trade-off

Single-challenge trade-off

W W^2 W^4 W^8 W^{16} W^{32}

Single-challenge trade-off



W W^2 W^4 W^8 W^{16} W^{32}

Challenge $j = 2$

Single-challenge trade-off

W W^2 W^4 W^8 W^{16} W^{32}

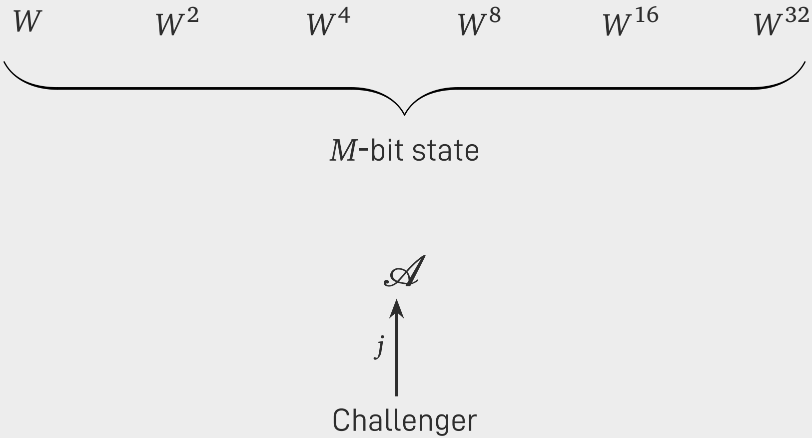
$\mathscr{A}$

Single-challenge trade-off

$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}_{M\text{-bit state}}$$

$$\mathscr{A}$$

Single-challenge trade-off



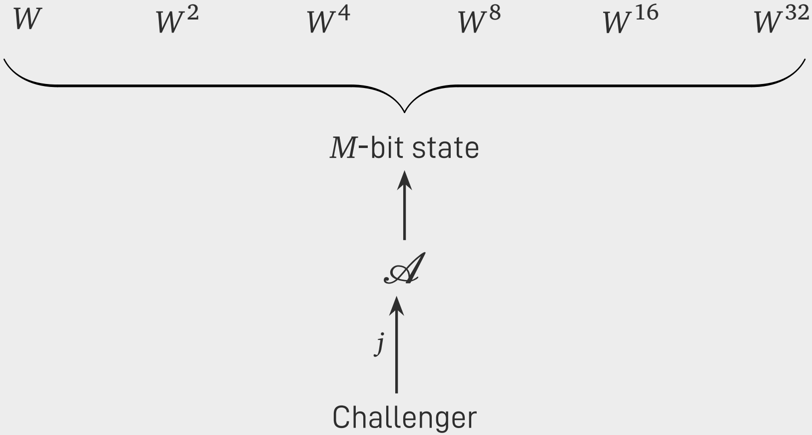
$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}_{M\text{-bit state}}$$

$\mathscr{A}$

j

Challenger

Single-challenge trade-off



W W^2 W^4 W^8 W^{16} W^{32}

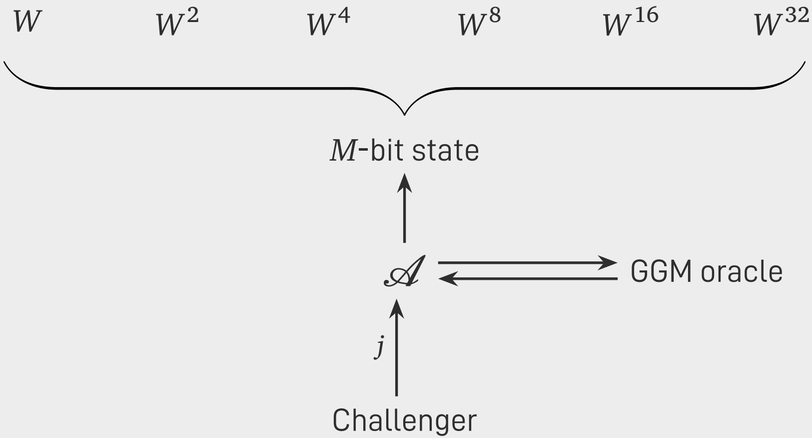
M-bit state

$\mathscr{A}$

j

Challenger

Single-challenge trade-off



W W^2 W^4 W^8 W^{16} W^{32}

M-bit state

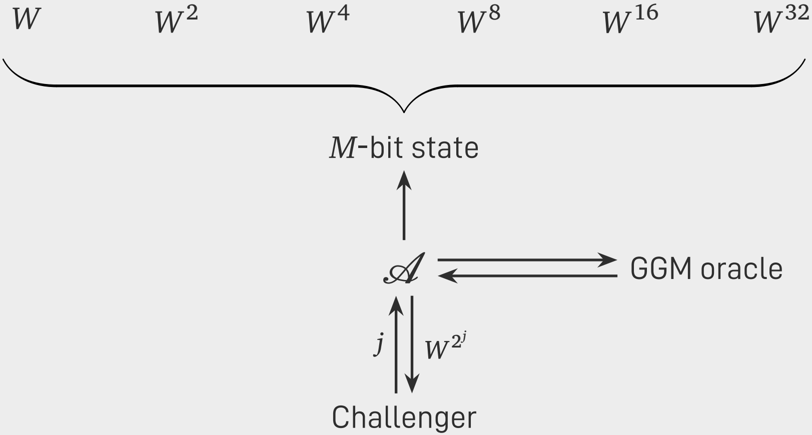
$\mathscr{A}$

GGM oracle

j

Challenger

Single-challenge trade-off



W W^2 W^4 W^8 W^{16} W^{32}

M-bit state

$\mathscr{A}$

GGM oracle

j

W^{2^j}

Challenger

Single-challenge trade-off

$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}$$

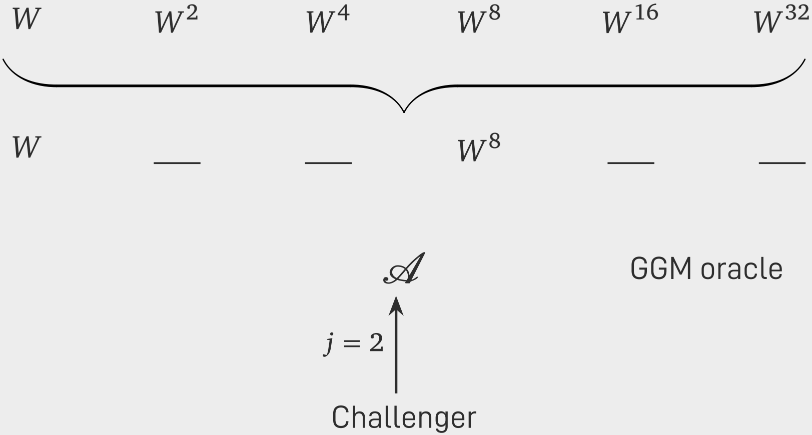
$$W \qquad \underline{\qquad} \qquad \underline{\qquad} \qquad W^8 \qquad \underline{\qquad} \qquad \underline{\qquad}$$

$\mathscr{A}$

GGM oracle

Challenger

Single-challenge trade-off



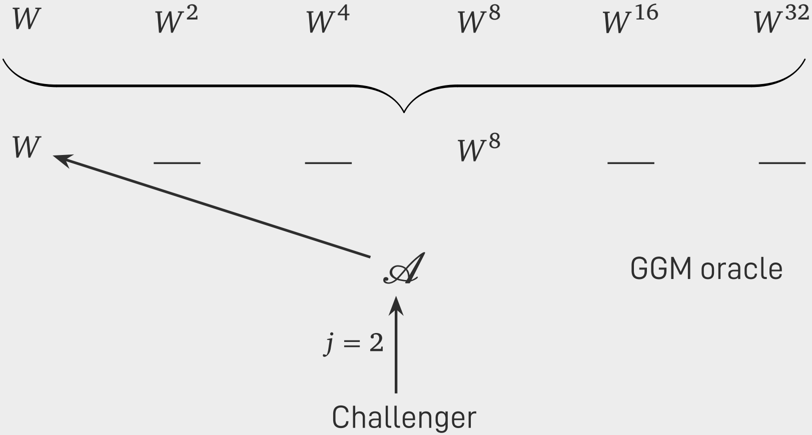
$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}$$

$$W \qquad \_\_\_ \qquad \_\_\_ \qquad W^8 \qquad \_\_\_ \qquad \_\_\_$$

$$\begin{tikzcd} \mathscr{A} \\ \text{Challenger} \arrow[u, "j = 2"] \end{tikzcd}$$

GGM oracle

Single-challenge trade-off



W W^2 W^4 W^8 W^{16} W^{32}

W ___ ___ W^8 ___ ___

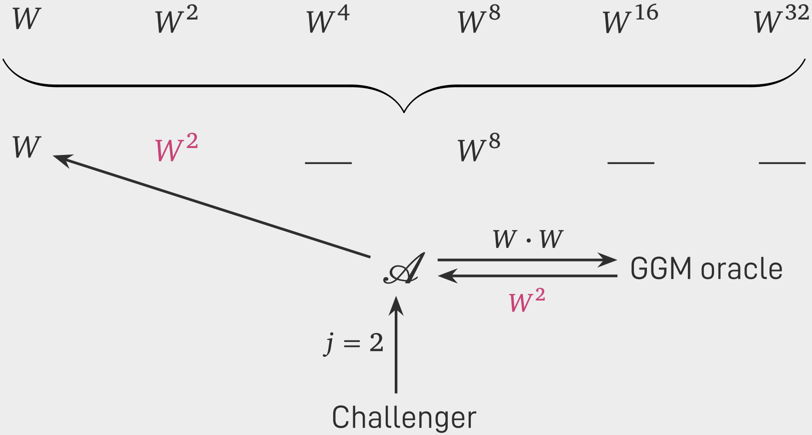
$\mathscr{A}$

GGM oracle

$j = 2$

Challenger

Single-challenge trade-off



$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}$$

$$W \qquad W^2 \qquad \_\_\_ \qquad W^8 \qquad \_\_\_ \qquad \_\_\_$$

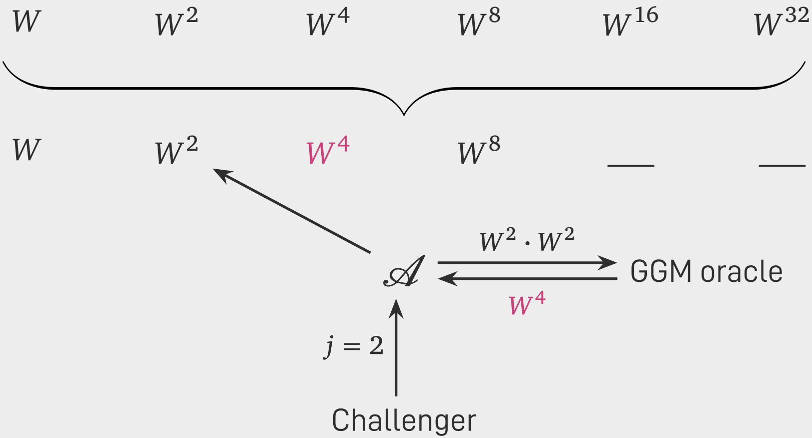
$\mathscr{A}$ $\xrightarrow{W \cdot W}$ GGM oracle

$\mathscr{A}$ $\xleftarrow[W^2]{}$ GGM oracle

$j = 2$

Challenger

Single-challenge trade-off



$$\underbrace{W \qquad W^2 \qquad W^4 \qquad W^8 \qquad W^{16} \qquad W^{32}}$$

$$W \qquad W^2 \qquad W^4 \qquad W^8 \qquad \_\_ \qquad \_\_$$

$\mathscr{A}$

$W^2 \cdot W^2$

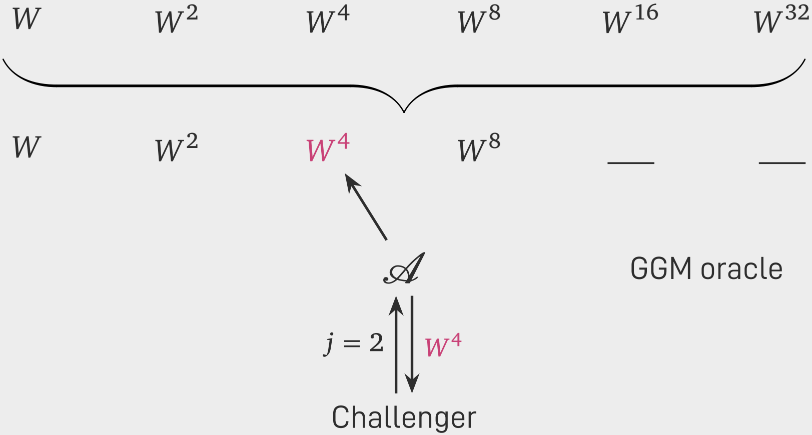
GGM oracle

W^4

$j = 2$

Challenger

Single-challenge trade-off



$W \quad W^2 \quad W^4 \quad W^8 \quad W^{16} \quad W^{32}$

$W \quad W^2 \quad W^4 \quad W^8 \quad \_\_\_ \quad \_\_\_$

$\mathscr{A}$

GGM oracle

$j = 2$

W^4

Challenger

Single-challenge intuition

- Memory reduced by 2/3...

Single-challenge intuition

- Memory reduced by $2/3$...
- ...but $1/3$ of the challenges require 2 queries!

Single-challenge intuition

- Memory reduced by $2/3$…
- …but $1/3$ of the challenges require 2 queries!
- Intuitively: $M / \log N$ equidistant group elements offers good trade-off

Single-challenge intuition

- Memory reduced by $2/3$...
- ...but $1/3$ of the challenges require 2 queries!
- Intuitively: $M / \log N$ equidistant group elements offers good trade-off
- **We prove that one cannot do much better**

Single-challenge bound

- M-bit state
- Challenge $j \in \{0, \ldots, n-1\}$ requires t_j GGM queries

Time-memory trade-off

$$\Pr_j\left[t_j \gtrsim \frac{n}{2 \cdot M / \log N} \cdot \frac{1}{\log n}\right] \geq \frac{1}{2}$$

Single-challenge bound

- M-bit state
- Challenge $j \in \{0, \ldots, n-1\}$ requires t_j GGM queries

Time-memory trade-off

$$\Pr_j\left[t_j \gtrsim \frac{n}{2 \cdot M / \log N} \cdot \frac{1}{\log n} \right] \geq \frac{1}{2}$$

1/2 of challenges

Single-challenge bound

- M-bit state
- Challenge $j \in \{0, \ldots, n-1\}$ requires t_j GGM queries

Time-memory trade-off

$$\Pr_j\left[\overbrace{t_j \gtrsim \frac{n}{2 \cdot M/\log N} \cdot \frac{1}{\log n}}^{M \text{ small} \implies t_j \text{ large}}\right] \geq \frac{1}{2}$$

1/2 of challenges

Single-challenge bound

- M-bit state
- Challenge $j \in \{0, \ldots, n-1\}$ requires t_j GGM queries

Time-memory trade-off

$$\Pr_j\left[\overbrace{t_j \gtrsim \frac{n}{2 \cdot M/\log N} \cdot \frac{1}{\log n}}^{M \text{ small} \implies t_j \text{ large}}\right] \geq \frac{1}{2}$$

1/2 of challenges

Equidistant strategy

Single-challenge bound

- M-bit state
- Challenge $j \in \{0, \ldots, n-1\}$ requires t_j GGM queries

Time-memory trade-off

$$\Pr_j\left[\overbrace{t_j \gtrsim \frac{n}{2 \cdot M/\log N} \cdot \frac{1}{\log n}}^{M \text{ small} \implies t_j \text{ large}}\right] \geq \frac{1}{2}$$

1/2 of challenges

Equidistant strategy

Not tight?

How to prove the bound?

Contradiction: $\mathscr{A}$ answers

- quickly (most t_j small)
- given small M-bit state
- for most RSA moduli N

How to prove the bound?

Contradiction: $\mathscr{A}$ answers

- quickly (most t_j small)
- given small M-bit state
- for most RSA moduli N

1. Run $\mathscr{A}$ on all $j = 0, \ldots, n-1$ given small M-bit state and extract a system $A\vec{x} = \vec{b}$ using the GGM oracle

How to prove the bound?

Contradiction: $\mathscr{A}$ answers

- quickly (most t_j small)
- given small M-bit state
- for most RSA moduli N

1. Run $\mathscr{A}$ on all $j = 0, \ldots, n-1$ given small M-bit state and extract a system $A\vec{x} = \vec{b}$ using the GGM oracle
2. Most t_j small and $\varphi(N)$ hidden $\implies$ $\vec{x}$ has many entries

How to prove the bound?

Contradiction: $\mathscr{A}$ answers

- quickly (most t_j small)
- given small M-bit state
- for most RSA moduli N

1. Run $\mathscr{A}$ on all $j = 0, \ldots, n-1$ given small M-bit state and extract a system $A\vec{x} = \vec{b}$ using the GGM oracle
2. Most t_j small and $\varphi(N)$ hidden

   $\implies$ $\vec{x}$ has many entries
3. **Case 1:** $\vec{x}$ has few entries

   $\implies$ $\mathscr{A}$ knows $\varphi(N)$

   $\implies$ Factor N ↯

How to prove the bound?

Contradiction: $\mathscr{A}$ answers

- quickly (most t_j small)
- given small M-bit state
- for most RSA moduli N

1. Run $\mathscr{A}$ on all $j = 0, \ldots, n-1$ given small M-bit state and extract a system $A\vec{x} = \vec{b}$ using the GGM oracle
2. Most t_j small and $\varphi(N)$ hidden

   $\Longrightarrow$ $\vec{x}$ has many entries
3. **Case 1:** $\vec{x}$ has few entries

   $\Longrightarrow$ $\mathscr{A}$ knows $\varphi(N)$

   $\Longrightarrow$ Factor N ↯
4. **Case 2:** $\vec{x}$ has many entries

   $\Longrightarrow$ $\vec{x}$ contains a lot of info about the GGM oracle

   $\Longrightarrow$ Compress to M bits ↯

Conclusion

Contribution

Diodon's CMC lower bounded by

$$\Omega\left(n^2 \log(N) \cdot \frac{1}{\log n}\right)$$

proving it memory-hard

Open questions

- Tight bound (no $1/\log n$)
- TMHF saving on time and memory
- TMHF for other MHF flavors

Trapdoor Memory-Hard Functions

B. Auerbach, C. U. Günther, and K. Pietrzak

https://eprint.iacr.org/2024/312