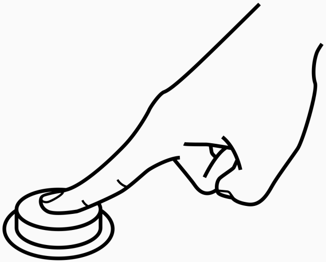
Voodoo: Memory Tagging, Authenticated Encryption, and Error Correction through MAGIC

Lukas Lamster Martin Unterguggenberger David Schrammel Stefan Mangard

August 16, 2024

IAIK – Graz University of Technology

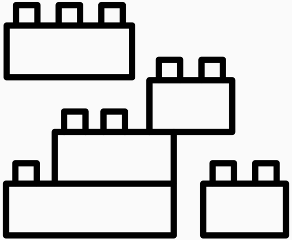
We want Security!



Activate Security!

That's not how it works!

Activate Security!



Security Building Blocks

Data Encryption

Tagged Memory

DRAM Error Correction

Data Encryption

≈ 12.5%

≈ 2.2%

Tagged Memory

≈ 3%

≈ 1.4%

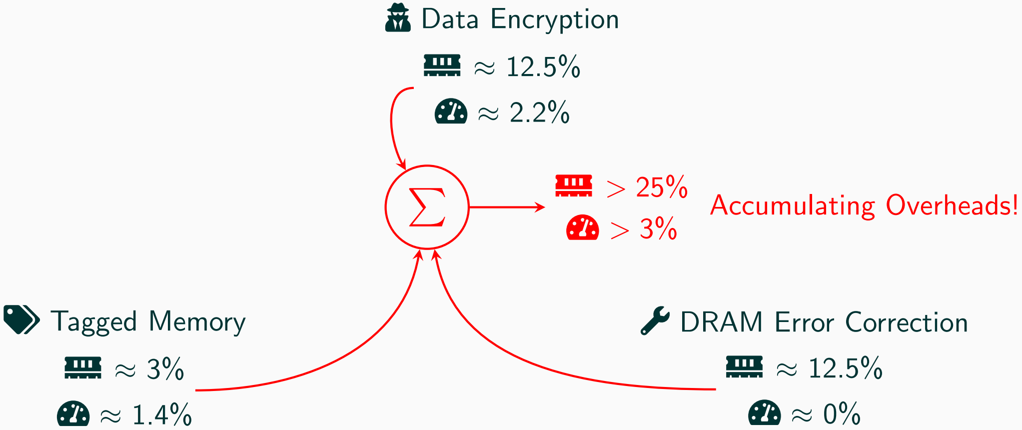
DRAM Error Correction

≈ 12.5%

≈ 0%

Achieving System Security

www.tugraz.at



We create a combined primitive!

Unify Security Features

- Auth. Encryption
- Error Correction
- Memory Tagging

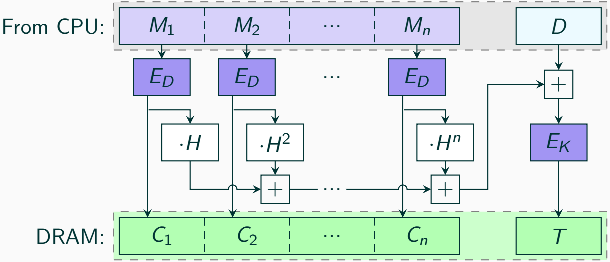
Low Overheads

- 12.5% add. Memory
- 1.4% Geomean

Strong Error Correction

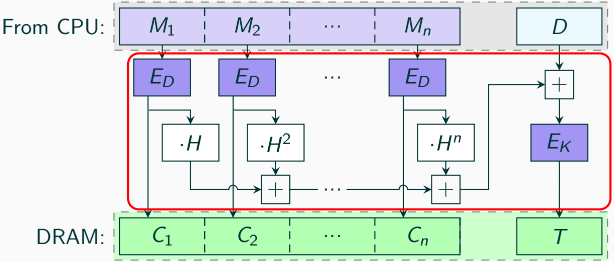
- 99% Single-Block Correction
- 99% Multi-Block Detection

- Combine AE and error correction



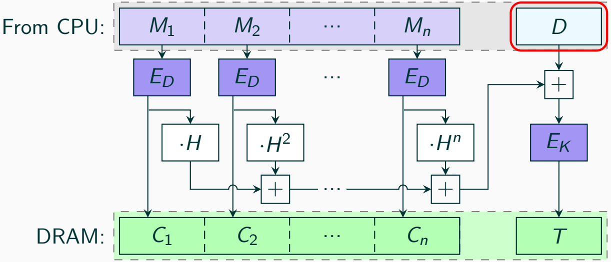
Background - The MAGIC mode [1]

- Combine AE and error correction
- $T = E_K \left(D + \sum_{i=1}^{n} C_i \cdot H^i \right)$



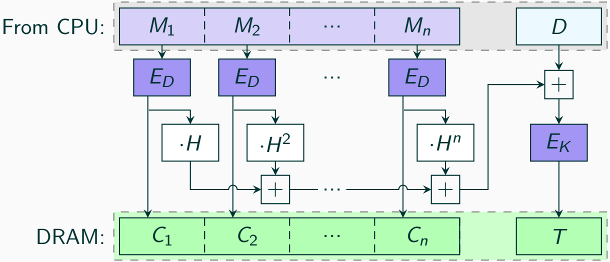
Background - The MAGIC mode [1]

- Combine AE and error correction
- $T = E_K\left(D + \sum_{i=1}^{n} C_i \cdot H^i\right)$
- Authenticate additional data D



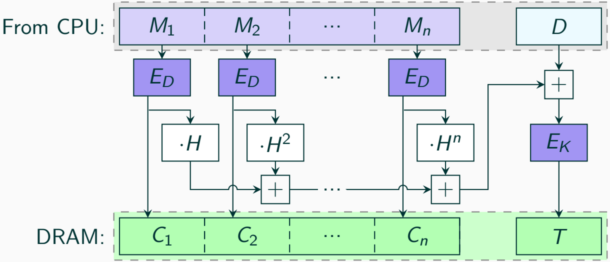
Background - The MAGIC mode [1]

- Combine AE and error correction
- $T = E_K \left(D + \sum_{i=1}^{n} C_i \cdot H^i \right)$
- Authenticate additional data D
- Correct up to T_{th} bits in one N-bit block

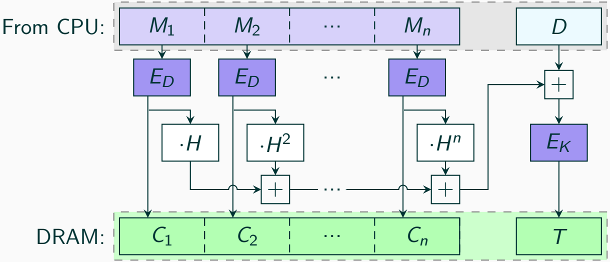


Background - The MAGIC mode [1]

- Combine AE and error correction
- $T = E_K \left(D + \sum_{i=1}^{n} C_i \cdot H^i \right)$
- Authenticate additional data D
- Correct up to T_{th} bits in one N-bit block
- Syndrome $S = e_j \cdot H^j$



- Combine AE and error correction
- $T = E_K\left(D + \sum_{i=1}^{n} C_i \cdot H^i\right)$
- Authenticate additional data D
- Correct up to T_{th} bits in one N-bit block
- Syndrome $S = e_j \cdot H^j$



The MAGIC is in the H

Impose conditions on Hamming weights of $HW(e \cdot H^i)$ for all errors e. The faulted block is found using the Hamming weight of $S \cdot H^{-i}$.

How can we include memory tagging into MAGIC?

? Can we just replace the authenticated data D with a memory tag M_T?

$$T = E_K\left(D + \sum_{i=1}^{n} C_i \cdot H^i\right) \rightarrow T = E_K\left(M_T + \sum_{i=1}^{n} C_i \cdot H^i\right)$$

How can we include memory tagging into `MAGIC`?

? Can we just replace the authenticated data D with a memory tag M_T?

$$T = E_K\left(D + \sum_{i=1}^{n} C_i \cdot H^i\right) \rightarrow T = E_K\left(M_T + \sum_{i=1}^{n} C_i \cdot H^i\right)$$

Can we ...

- Detect Mismatches?
- Correct Errors?
- Read Tags?

Detecting Tag Mismatches

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + M'_T + \sum_{i=1}^{n} C'_i \cdot H^i = M_T + M'_T = e_t$$

Detecting Tag Mismatches

- Assume $C_i = C_i'$ for all blocks

Cancels out

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + M_T' + \sum_{i=1}^{n} C_i' \cdot H^i = M_T + M_T' = e_t$$

Detecting Tag Mismatches

- Assume $C_i = C_i'$ for all blocks
- Syndrome only depends on $e_t = M_T + M_T'$

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + M_T' + \sum_{i=1}^{n} C_i' \cdot H^i = M_T + M_T' = e_t$$

Detecting Tag Mismatches

- Assume $C_i = C_i'$ for all blocks
- Syndrome only depends on $e_t = M_T + M_T'$
- What values can e_t assume?

$$
\begin{aligned}
S =& M_T + \sum_{i=1}^{n} C_i \cdot H^i + \\
& M_T' + \sum_{i=1}^{n} C_i' \cdot H^i = \\
& M_T + M_T' = e_t
\end{aligned}
$$

Detecting Tag Mismatches

- Assume $C_i = C'_i$ for all blocks
- Syndrome only depends on $e_t = M_T + M'_T$
- What values can e_t assume?
- Any N-bit value

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + M'_T + \sum_{i=1}^{n} C'_i \cdot H^i = M_T + M'_T = e_t$$

Detecting Tag Mismatches

- Assume $C_i = C_i'$ for all blocks
- Syndrome only depends on $e_t = M_T + M_T'$
- What values can e_t assume?
- Any N-bit value

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + M_T' + \sum_{i=1}^{n} C_i' \cdot H^i = M_T + M_T' = e_t$$

We cannot distinguish tag mismatches from errors

Tag errors can look like correctable errors → Miscorrection and Data Corruption.

Reading Tags

- Access with $M'_T = 0$

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + 0 + \sum_{i=1}^{n} C'_i \cdot H^i = M_T$$

Reading Tags

- Access with $M'_T = 0$
- Syndrome equals $e_t = M_T + 0$

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i + 0 + \sum_{i=1}^{n} C'_i \cdot H^i = M_T$$

Reading Tags

- Access with $M'_T = 0$
- Syndrome equals $e_t = M_T + 0$
- $e_t = M_T + e_j \cdot H^j$

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i +$$

$$0 + \sum_{i=1}^{n} C'_i \cdot H^i = M_T$$

Reading Tags

- Access with $M'_T = 0$
- Syndrome equals $e_t = M_T + 0$
- $e_t = M_T + e_j \cdot H^j$
- A faulty tag is read

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i +$$

$$0 + \sum_{i=1}^{n} C'_i \cdot H^i = M_T$$

Reading Tags

- Access with $M'_T = 0$
- Syndrome equals $e_t = M_T + 0$
- $e_t = M_T + e_j \cdot H^j$
- A faulty tag is read

$$S = M_T + \sum_{i=1}^{n} C_i \cdot H^i +$$
$$0 + \sum_{i=1}^{n} C'_i \cdot H^i = M_T$$

We cannot reliably read tags in the presence of errors

Data errors will not be detected and the read operation will yield a faulty tag.

Takeaway: A naïve encoding is not suitable!

- Tag errors are not clearly identifiable
- Miscorrection and data corruption is possible
- Reading tags only works without data errors

Takeaway: A naïve encoding is not suitable!

- Tag errors are not clearly identifiable
- Miscorrection and data corruption is possible
- Reading tags only works without data errors

We can solve these issues!

We present three encodings for M_T that circumvent these problems.

- Check Pattern Encoding
- Encrypted Tag Encoding
- Bounded Hamming Weight Encoding

Takeaway: A naïve encoding is not suitable!

- Tag errors are not clearly identifiable
- Miscorrection and data corruption is possible
- Reading tags only works without data errors

We can solve these issues!

We present three encodings for M_T that circumvent these problems.

- **Check Pattern Encoding**
- Encrypted Tag Encoding
- **Bounded Hamming Weight Encoding**

Design - Check Pattern Encoding

- Tags are small, we do not need $[0, 1]^N$ tag space.

Design - Check Pattern Encoding

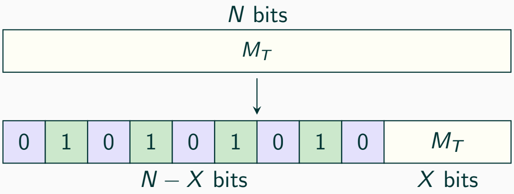
- Tags are small, we do not need $[0, 1]^N$ tag space.
- Fix upper bits of tag to certain pattern P.

Design - Check Pattern Encoding

💡 Tags are small, we do not need $[0,1]^N$ tag space.

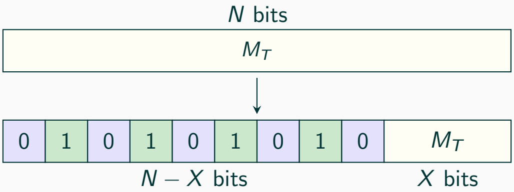
💡 Fix upper bits of tag to certain pattern P.

- Check if upper $N - X$ bits are 0 in S
- P always cancels out
- Upper bits '0' $\rightarrow$ (likely) tag mismatch



- Tags are small, we do not need $[0,1]^N$ tag space.
- Fix upper bits of tag to certain pattern P.

- Check if upper $N - X$ bits are 0 in S
- P always cancels out
- Upper bits '0' $\rightarrow$ (likely) tag mismatch



Check Pattern Encoding never miscorrects due to a tag mismatch.

Guaranteed to identify tag errors.

Some correctable errors may be treated as uncorrectable.

No data corruption from tag errors!

Design - Bounded Hamming Weight Encoding

- Select tags to have Hamming weight below T_{th}

Design - Bounded Hamming Weight Encoding

- Select tags to have Hamming weight below T_{th}
- No aliasing and no miscorrection possible

- Select tags to have Hamming weight below T_{th}
- No aliasing and no miscorrection possible

$$|\mathcal{M}_{\mathcal{T}}| = \sum_{t=0}^{\lfloor T_{th}/2 \rfloor} \binom{N}{t}$$

- Limit M_T s.t. $HW(e_t) < T_{th}$
- No S will have $HW(S) \leq T_{th}$
- Tag errors are uniquely identifiable

$$HW(M_T) \leq \lfloor T_{th}/2 \rfloor$$

$$HW(M_T + e_j H^j) > \lfloor T_{th}/2 \rfloor + 1$$

- Select tags to have Hamming weight below T_{th}
- No aliasing and no miscorrection possible

$$|\mathcal{M}_{\mathcal{T}}| = \sum_{t=0}^{\lfloor T_{th}/2 \rfloor} \binom{N}{t}$$

- Limit M_T s.t. $HW(e_t) < T_{th}$
- No S will have $HW(S) \leq T_{th}$
- Tag errors are uniquely identifiable

$$HW(M_T) \leq \lfloor T_{th}/2 \rfloor$$

$$HW(M_T + e_j H^j) > \lfloor T_{th}/2 \rfloor + 1$$

Bounded Hamming Weight Encoding allows for fully deterministic tagging.

No aliasing possible.

Smaller tag space than other encodings.

Table 1: ● deterministic tagging ○ impossible configuration

Architecture	Tag Bits per 64 B	f-Bounded N=64	f-Bounded N=128	f-Pattern N=64	f-Pattern N=128	f-Encrypt N=64	f-Encrypt N=128
CHERI ISA (256)	2-bit	●	●	0	0	2^{-57}	2^{-120}
CHERI ISA (128)	4-bit	●	●	0	0	2^{-57}	2^{-120}
SPARC ADI	4-bit	●	●	0	0	2^{-57}	2^{-120}
DIFT, M-Machine, HDFI, Shakti-T	8-bit	●	●	2^{-63}	0	2^{-52}	2^{-119}
Model 1 A / B	15-bit	●	●	2^{-49}	2^{-128}	2^{-48}	2^{-109}
MTE, Mondrian	16-bit	○	●	2^{-48}	2^{-123}	2^{-47}	2^{-109}
SPEAR-V	24-bit	○	●	2^{-40}	2^{-106}	2^{-37}	2^{-103}
lowRISC ,	32-bit	○	●	2^{-32}	2^{-96}	2^{-31}	2^{-95}
Model 2 A / B	36-bit	○	●	2^{-28}	2^{-92}	2^{-26}	2^{-91}

Case Study

Table 1: ● deterministic tagging ○ impossible configuration

Architecture	Tag Bits per 64 B	f-Bounded N=64	f-Bounded N=128	f-Pattern N=64	f-Pattern N=128	f-Encrypt N=64	f-Encrypt N=128
CHERI ISA (256)	2-bit	●	●	0	0	2^{-57}	2^{-120}
CHERI ISA (128)	4-bit	●	●	0	0	2^{-57}	2^{-120}
SPARC ADI	4-bit	●	●	0	0	2^{-57}	2^{-120}
DIFT, M-Machine, HDFI, Shakti-T	8-bit	●	●	2^{-63}	0	2^{-52}	2^{-119}
Model 1 A / B	15-bit	●	●	2^{-49}	2^{-128}	2^{-48}	2^{-109}
MTE, Mondrian	16-bit	○	●	2^{-48}	2^{-123}	2^{-47}	2^{-109}
SPEAR-V	24-bit	○	●	2^{-40}	2^{-106}	2^{-37}	2^{-103}
lowRISC,	32-bit	○	●	2^{-32}	2^{-96}	2^{-31}	2^{-95}
Model 2 A / B	36-bit	○	●	2^{-28}	2^{-92}	2^{-26}	2^{-91}

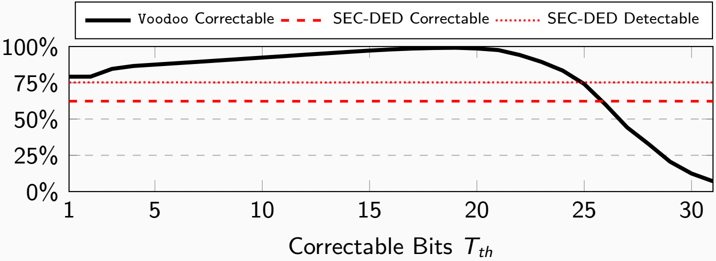
Evaluation - Single-Block Error Correction

- We cannot test `Voodoo` on real hardware.
- Model DRAM faults in Monte Carlo simulation [2].

Evaluation - Single-Block Error Correction

- We cannot test `Voodoo` on real hardware.
- Model DRAM faults in Monte Carlo simulation [2].

Class	Fault	Bits	Rate
Single-bit	Single-bit	1	55.06%
Multi-bit	Single-word	4	0.325%
	Single-column	4	3.85%
Subsequent	Two-column	8	2.84%
	Single-pin	8	0.67%
Large-scale	Partial row	32	24.345%
	Single row	32	0.26%
	Single row + single bit	32	0.975%
	Two row	32	4.125%
	Consecutive row	32	0.555%
	Cluster row	32	5.7%
	Single bank	32	0.065%
	Quarter device	32	0.135%
	Half device	32	0.09%
	Full device	32	0.605%
	Single lane	32	0.4%

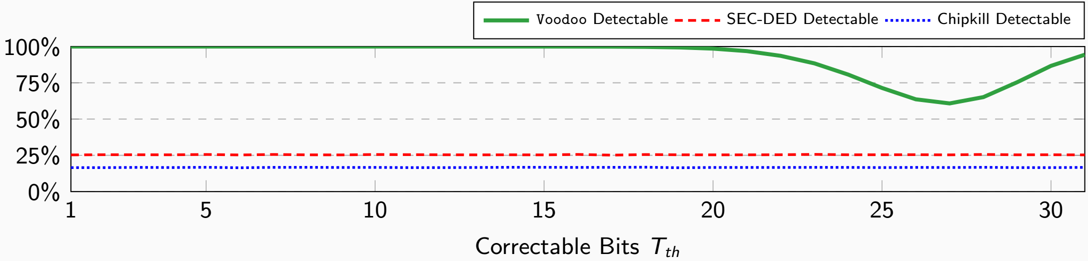


`Voodoo` offers strong single-block error correction

We can correct up to 99% of single-block faults. We find an optimum at $T_{th} = 19$.

- How well do we perform with multi-block faults?
- Model multi-block faults in Monte Carlo simulation.

- How well do we perform with multi-block faults?
- Model multi-block faults in Monte Carlo simulation.



Voodoo has a low miscorrection rate

Multi-block faults are most likely correctly identified as uncorrectable. Miscorrection happens if exactly one wrong error location indicator is found.

Conclusion

- Combine auth. encryption, ECC, and memory tagging
- Up to 36 tag bits per cache line
- Strong single-block error correction
- Strong multi-block error detection

Voodoo: Memory Tagging, Authenticated Encryption, and Error Correction through MAGIC

Lukas Lamster Martin Unterguggenberger David Schrammel Stefan Mangard

August 16, 2024

IAIK – Graz University of Technology

References

[1] Kounavis et al. **The MAGIC Mode for Simultaneously Supporting Encryption, Message Authentication and Error Correction.** In: IACR Cryptol. ePrint Arch. (2020).

[2] Beigi et al. **A Systematic Study of DDR4 DRAM Faults in the Field.** In: HPCA'23. 2023.