D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim  Minesh Patel
Hasan Hassan  Lois Orosa  Onur Mutlu

SAFARI

Carnegie Mellon  ETH Zürich
Executive Summary

• **Motivation:** High-throughput true random numbers enable system security and various randomized algorithms.
  • Many systems (e.g., IoT, mobile, embedded) do not have dedicated True Random Number Generator (TRNG) hardware but have DRAM devices

• **Problem:** Current DRAM-based TRNGs either
  1. do not sample a fundamentally non-deterministic entropy source
  2. are too slow for continuous high-throughput operation

• **Goal:** A novel and effective TRNG that uses existing commodity DRAM to provide random values with 1) high-throughput, 2) low latency and 3) no adverse effect on concurrently running applications

• **D-RaNGe:** Reduce DRAM access latency below reliable values and exploit DRAM cells’ failure probabilities to generate random values

• **Evaluation:**
  1. Experimentally characterize 282 real LPDDR4 DRAM devices
  2. D-RaNGe (717.4 Mb/s) has significantly higher throughput (211x)
  3. D-RaNGe (100ns) has significantly lower latency (180x)
**D-RaNGe Outline**

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
## D-RaNGe Outline

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Motivation</strong></td>
</tr>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td><strong>D-RaNGe</strong></td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td><strong>Methodology</strong></td>
</tr>
<tr>
<td><strong>Results</strong></td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
</tr>
</tbody>
</table>
Motivation and Goal

• High throughput **True Random Numbers** are required for many real-world applications
  
  - Importantly **cryptography** for securely encrypting file systems, network packets, data in standard protocols (TLS/SSL/RSA...)
  
  - Others include randomized algorithms, scientific simulation, statistical sampling, recreational entertainment

• **True random numbers** can only be generated via physical processes
  
  - e.g., radioactive decay, thermal noise, shot noise
  
  - Systems rely on **dedicated TRNG Hardware** that samples non-deterministic **various physical phenomena**
Motivation and Goal

• Smaller devices (e.g., IoT, mobile, embedded) require, but often lack, a high throughput True Random Number Generator (TRNG)

• DRAM devices are available on most systems

• Mechanism that generates TRN using DRAM enables:
  1. applications that require true random numbers to now run on most systems
  2. other use-cases, e.g., processing-in-memory applications to generate true random numbers within memory itself

• **Our Goal:** to provide a TRNG using DRAM devices that satisfies the characteristics of an effective TRNG
# D-RaNGe Outline

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Effective TRNG Characteristics

1. Low **implementation cost**

2. Fully **non-deterministic**
   - impossible to predict the next output given complete information about how the mechanism operates

3. Provide a continuous stream of true random numbers with **high throughput**

4. Provide true random numbers with **low latency**

5. Exhibit **low system interference**
   - not significantly slow down concurrently-running applications

6. Generate random values with **low energy overhead**
# D-RaNGe Outline

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td><strong>D-RaNGe</strong></td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
DRAM Organization

A DRAM bank is hierarchically organized into subarrays.

Columns of cells in subarrays share a local bitline.
Rows of cells in a subarray share a wordline.
DRAM Operation

![Diagram of DRAM operation with row decoder, cache line, and READ operations]

- **ACT R0**: Activation of Row 0
- **PRE R0**: Precharge Row 0
- **ACT R1**: Activation of Row 1
- **READ**: Read operation
Process variation during manufacturing results in cells having unique behavior.
Weaker cells have a higher probability to fail.
# D-RaNGe Outline

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
</tr>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td><strong>D-RaNGe</strong></td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
D-RaNGe Key Idea

• A cell’s latency failure probability is inherently related to random process variation from manufacturing.
• We can extract random values by observing DRAM cells’ latency failure probabilities.

High % chance to fail with reduced $t_{RCD}$

Low % chance to fail with reduced $t_{RCD}$
D-RaNGe Key Idea

- A cell’s latency failure probability is inherently related to random process variation from manufacturing.
- We can extract random values by observing DRAM cells.

The key idea is to extract random values by sampling DRAM cells that fail truly randomly.
D-RaNGe: Extracting Random Values

Identify all DRAM cells that fail randomly when accessed with a reduced $t_{RCD}$ (RNG Cell)

- When accessing an RNG Cell with a reduced $t_{RCD}$, the values read will be truly random values.

RNG Cell

Random values when accessed with $t_{RCD}$ reduced by 45%
D-RaNGe: Identifying RNG Cells

• To identify RNG Cells, extract 1M values (bitstream) from each DRAM cell
• An RNG Cell is a DRAM cell whose output passes the NIST statistical test suite for randomness
• NIST tests [Rukhin+, Tech report, 2001] include tests for:
  - Unbiased output of 1’s and 0’s across entire bitstream
  - Unbiased output within smaller segments of the bitstream
  - Limited number of uninterrupted sequence of identical bits
  - Peak heights in the discrete fourier transform of bitstream
  - Even distribution of short sequences within bitstream
  - Cumulative sum always stays close to zero
  - ...

SAFARI
D-RaNGe: Access Pattern

• To maximize the bits that are accessed immediately following activation, we alternate accesses to distinct rows in each bank:
  - quickly generate tRCD failures within cache lines in two rows
  - maximizes tRCD failures when using reduced tRCD
D-RaNGe: Access Pattern

- To maximize the bits that are accessed immediately following activation, we alternate accesses to distinct rows in each bank.

Accessing cache lines containing more RNG cells will result in more random values.
D-RaNGe: Exclusive Access

- To minimize system interference, D-RaNGe has exclusive access to RNG cells
- **In a bank**, find the **two cache lines** in distinct rows with the most number of RNG cells
D-RaNGe: Exclusive Access

- Reserve rows containing selected cache lines exclusively for D-RaNGe accesses to minimize interference

![Diagram showing cache lines and row decoder](image)
D-RaNGe: Exclusive Access

- Reserve neighboring rows to minimize DRAM data pattern/read interference

- Cache lines containing more RNG cells provide more random bits of data per access

- In a bank, find the two cache lines in distinct rows with the most number of RNG cells
D-RaNGe: Exclusive Access

- Cache lines containing more RNG cells provide more random bits of data per access.
- In a bank, find the two cache lines in distinct rows with the most number of RNG cells.

We can parallelize accesses across all available DRAM banks for higher throughput of random values.
**D-RaNGe: Example Implementation**

- Memory controller reserves rows containing selected RNG cells and neighboring rows

- When system not accessing a bank, memory controller runs D-RaNGe firmware to generate random values in the bank

- Memory controller has buffer of random data

- Stores random values in memory controller buffer

- Expose API for returning random values from the buffer when requested by the user
## D-RaNGe Outline

<table>
<thead>
<tr>
<th>Section</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td></td>
</tr>
<tr>
<td>Effective True Random Number Generators</td>
<td></td>
</tr>
<tr>
<td>D-RaNGe</td>
<td></td>
</tr>
<tr>
<td>DRAM Operation</td>
<td></td>
</tr>
<tr>
<td>Key Idea</td>
<td></td>
</tr>
<tr>
<td><strong>Methodology</strong></td>
<td></td>
</tr>
<tr>
<td>Results</td>
<td></td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
<td></td>
</tr>
<tr>
<td>Command scheduling</td>
<td></td>
</tr>
<tr>
<td>Cell charge retention</td>
<td></td>
</tr>
<tr>
<td>Start-up values</td>
<td></td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td></td>
</tr>
</tbody>
</table>
Methodology

• **282 2y-nm LPDDR4 DRAM devices**
  - 2GB device size from 3 major DRAM manufacturers

• **Thermally controlled testing chamber**
  - Ambient temperature range: \(40°C – 55°C\) ± 0.25°C
  - DRAM temperature is held at 15°C above ambient

• **Control over DRAM commands/timing parameters**
  - Test reduced latency effects by reducing \(t_{RCD}\) parameter

• **Cycle-level simulator: Ramulator** [Kim+, CAL’15]
  - [https://github.com/CMU-SAFAIR/ramulator](https://github.com/CMU-SAFAIR/ramulator)
  - SPEC CPU2006 workloads, 4-core

• **DRAM Energy: DRAMPower** [Chandrasekar+, ‘12]
  - [http://www.es.ele.tue.nl/drampower/](http://www.es.ele.tue.nl/drampower/)
  - Using output from Ramulator
# D-RaNGe Outline

- **Motivation**
- **Effective True Random Number Generators**
- **D-RaNGe**
  - DRAM Operation
  - Key Idea
- **Methodology**
- **Results**
- **Prior work on DRAM-based TRNGs**
  - Command scheduling
  - Cell charge retention
  - Start-up values
- **Summary**
### Results – NIST Randomness Tests

How do we know whether D-RaNGe is truly random?

<table>
<thead>
<tr>
<th>NIST Test Name</th>
<th>P-value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>monobit</td>
<td>0.675</td>
<td>PASS</td>
</tr>
<tr>
<td>frequency_within_block</td>
<td>0.096</td>
<td>PASS</td>
</tr>
<tr>
<td>runs</td>
<td>0.501</td>
<td>PASS</td>
</tr>
<tr>
<td>longest_run_ones_in_a_block</td>
<td>0.256</td>
<td>PASS</td>
</tr>
<tr>
<td>binary_matrix_rank</td>
<td>0.914</td>
<td>PASS</td>
</tr>
<tr>
<td>dft</td>
<td>0.424</td>
<td>PASS</td>
</tr>
<tr>
<td>non_overlapping_template_matching</td>
<td>&gt;0.999</td>
<td>PASS</td>
</tr>
<tr>
<td>overlapping_template_matching</td>
<td>0.624</td>
<td>PASS</td>
</tr>
<tr>
<td>maurers_universal</td>
<td>0.999</td>
<td>PASS</td>
</tr>
<tr>
<td>linear_complexity</td>
<td>0.663</td>
<td>PASS</td>
</tr>
<tr>
<td>serial</td>
<td>0.405</td>
<td>PASS</td>
</tr>
<tr>
<td>approximate_entropy</td>
<td>0.735</td>
<td>PASS</td>
</tr>
<tr>
<td>cumulative_sums</td>
<td>0.588</td>
<td>PASS</td>
</tr>
<tr>
<td>random_excursion</td>
<td>0.200</td>
<td>PASS</td>
</tr>
<tr>
<td>random_excursion_variant</td>
<td>0.066</td>
<td>PASS</td>
</tr>
</tbody>
</table>

[Rukhin+, Tech report, 2001]

Passes all tests in NIST test suite for randomness!

More details in the paper
Results – 64-bit TRN Latency

Latency is related to density of available RNG cells per cache line

Across our devices, we analyze availability of RNG cells per cache line in a bank. Each point is the number of occurrences in a bank. We plot the distribution across many banks as box-and-whisker plot.
Results – 64-bit TRN Latency

Latency is related to density of available RNG cells per cache line

Maximum latency: 960 ns
assuming 1 RNG cell / cache line from a single bank

Minimum empirical latency: 100 ns
assuming 4 RNG cell / cache line in all 32 banks in 4-channels
Results – Single Channel Throughput

We determine **throughput** using the RNG cell densities found.

For each bank utilized (x-axis), select the two cache lines containing the **most** number of RNG cells.

\[
\text{Throughput} = \frac{\text{Accesses}}{\text{Second}} \times (\sum_{i} \text{selected cache lines} \times \text{RNG Cell Density}_i)
\]
Since there are only between 1 and 4 RNG cells per cache line, there are a limited number of possible throughputs

- At least **40 Mb/s** when using all **8 banks** in a single channel
- Maximum throughput for A/B/C: **179.4/179.4/134.5 Mb/s**
- 4-channel max (avg) throughput: **717.4 Mb/s (435.7 Mb/s)**
Results

• System Interference
  - Capacity overhead: 6 DRAM rows per DRAM bank (~0.018%)
  - D-RaNGe is flexible and can adjust its level of interference
  - D-RaNGe throughput with SPEC CPU2006 workloads in the pessimistic case where D-RaNGe only issues accesses to a DRAM bank when it is idle (no interference)
    • Average throughput of 83.1 Mb/s

• Energy Consumption
  - 4.4 nJ/bit
  - Determined by Ramulator + DRAMPower
    • [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)
    • [http://www.es.ele.tue.nl/drampower/](http://www.es.ele.tue.nl/drampower/)
Other Results in the Paper

• LPDDR4 DRAM Activation Failure Characterization
  - Spatial distribution, data pattern dependence, temperature effects, variation over time

• A detailed analysis on:
  - Devices of the three major DRAM manufacturers
  - D-RaNGe energy consumption, 64-bit latency, throughput

• Further discussion on:
  - Algorithm for D-RaNGe to effectively generate random values
  - Design considerations for D-RaNGe
  - D-RaNGe overhead analysis
  - Analysis of NIST statistical test suite results
  - Detailed comparison against prior work
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D-RaNGe Outline</strong></td>
</tr>
<tr>
<td>Motivation</td>
</tr>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Prior Work: Command Scheduling

[Pyo+, IET, 2009]

- **Randomness source:** time it takes to run a code segment of many DRAM accesses
  - Since time to access DRAM is **unpredictable** due to memory conflicts, refresh operations, calibration, etc.
  - Lower bits of the cycle timer used as random values

- Can produce random numbers at **3.4 Mb/s**

- **D-RaNGe** can produce TRNs at **>700Mb/s (211x higher)**

- **Downsides of DRAM Command Scheduling based TRNGs**
  - Randomness source is **not truly random:** depends on memory controller implementation and concurrently running applications
  - Much lower TRN throughput than **D-RaNGe**
<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
DRAM Cell Leakage

DRAM encodes information in **leaky** capacitors

Stored data is **corrupted** if too much charge leaks (i.e., the capacitor voltage degrades too much)

[Patel et al., REAPER, ISCA’17]
DRAM Cell Retention

Retention failure – when leakage corrupts stored data
Retention time – how long a cell holds its value

[Patel et al., REAPER, ISCA’17]
Retention-based TRNGs

[Keller+, ISCAS, 2014] [Hashemian, DATE, 2015] [Sutar+, TECS, 2018]

Generate random values using data from cells that fail randomly with a refresh interval $N$

Can handle a longer refresh interval

Fails with refresh interval $N$

After time $N$, some cells leak close to $V_{min}$. These RNG cells fail randomly
Retention-based TRNGs

[Keller+, ISCAS, 2014] [Hashemian, DATE, 2015] [Sutar+, TECS, 2018]

Generate random values using data from cells that fail randomly with a refresh interval $N$

The **key idea** is to extract **random values** by aggregating values from RNG cells after every *increased* refresh interval $N$

After time $N$, some cells leak close to $V_{min}$.

These **RNG cells** fail randomly
DRAM Retention TRNG Weaknesses

High latency
- Prior work shows that 40 sec refresh interval results in 256 random bits of data per 4MiB DRAM block
- D-RaNGe’s latency is 100ns (>9 orders of magnitude faster)

Low Throughput / High DRAM capacity overhead
- Requires more capacity for higher throughput
  - Fully reserving a 32GB DRAM device results in 0.05 Mb/s
- D-RaNGe has 14,000x higher throughput with a fixed capacity overhead (384 KB)

High energy consumption
- 6.8mJ/bit mainly due to long idle periods
- D-RaNGe: 4.4 nJ/bit (>7 orders of magnitude lower)
# D-RaNGe Outline

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Start-up Values as Random Numbers

[Teheranipoor, HOST, 2016]

• When a device is powered up, some DRAM cells have random values due to interaction between
  - precharge logic
  - row decoder logic
  - column select lines

• Prior works propose power cycling DRAM to extract the random data resident in those cells

• Downsides of DRAM Start-up value based TRNGs
  - Must power cycle DRAM to generate random values:
    • High latency: based on power cycle time and data migration
    • High storage cost: all data must be migrated or will be lost
D-RaNGe Comparison against Prior Work

• Compared to Command Scheduling, D-RaNGe:
  - samples a truly random entropy source
  - $211x$ higher throughput
  - $180x$ lower latency

• Compared to Retention Time, D-RaNGe:
  - $>5$ orders of magnitude higher throughput
  - $>9$ orders of magnitude lower latency
  - $>7$ orders of magnitude more energy efficient

• Compared to Startup Values, D-RaNGe:
  - continuously produces random values
  - does not require a system restart
# D-RaNGe Outline

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective True Random Number Generators</td>
</tr>
<tr>
<td>D-RaNGe</td>
</tr>
<tr>
<td>DRAM Operation</td>
</tr>
<tr>
<td>Key Idea</td>
</tr>
<tr>
<td>Methodology</td>
</tr>
<tr>
<td>Results</td>
</tr>
<tr>
<td>Prior work on DRAM-based TRNGs</td>
</tr>
<tr>
<td>Command scheduling</td>
</tr>
<tr>
<td>Cell charge retention</td>
</tr>
<tr>
<td>Start-up values</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Summary and Conclusion

- **Motivation**: High-throughput true random numbers enable system security and various randomized algorithms.
  - Many systems (e.g., IoT, mobile, embedded) do not have dedicated True Random Number Generator (TRNG) hardware but have DRAM devices.

- **Problem**: Current DRAM-based TRNGs either
  1. do not sample a fundamentally non-deterministic entropy source
  2. are too slow for continuous high-throughput operation

- **Goal**: A novel and effective TRNG that uses existing commodity DRAM to provide random values with 1) high-throughput, 2) low latency and 3) no adverse effect on concurrently running applications

- **D-RaNGe**: Reduce DRAM access latency below reliable values and exploit DRAM cells’ failure probabilities to generate random values

- **Evaluation**:
  1. Experimentally characterize 282 real LPDDR4 DRAM devices
  2. D-RaNGe (717.4 Mb/s) has significantly higher throughput (211x)
  3. D-RaNGe (100ns) has significantly lower latency (180x)
D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim       Minesh Patel
Hasan Hassan        Lois Orosa
Onur Mutlu

SAFARI

Carnegie Mellon      ETH Zürich
DRAM Organization + Operation

- CPU
  - core
  - memory controller

- 64-bit channel

- I/O pins

- DRAM Module
  - DRAM rank
  - DRAM Chip
    - DRAM bank (0)
    - DRAM bank (B - 1)

- Internal data/command bus

- Timing Diagram:
  - precharged
  - charge-sharing
  - sensing & restoration
  - restored
  - precharged

- Vdd/2
- tRCD
- tRAS
- tRP
Algorithm 1: DRAM Activation Failure Testing

1. DRAM_ACT_failure_testing(data_pattern, DRAM_region):
2. write data_pattern (e.g., solid 1s) into all cells in DRAM_region
3. set low $t_{RCD}$ for ranks containing DRAM_region
4. foreach col in DRAM_region:
5.  foreach row in DRAM_region:
6.       activate(row)  // fully refresh cells
7.       precharge(row) // ensure next access activates the row
8.       activate(row)
9.       read(col)       // induce activation failure on col
10.      precharge(row)
11.      record activation failures to storage
12.      set default $t_{RCD}$ for DRAM ranks containing DRAM_region
Figure 4: Activation failure bitmap in $1024 \times 1024$ cell array.
Activation Failure Temperature Dependence

Figure 6: Effect of temperature variation on failure probability
Full D-RaNGe Algorithm

Algorithm 2: D-RaNGe: A DRAM-based TRNG

1. D-RaNGe(num_bits):  // num_bits: number of random bits requested
2.  
3.      DP: a known data pattern that results in high entropy
4.      select 2 DRAM words with RNG cells in distinct rows in each bank
5.      write DP to chosen DRAM words and their neighboring cells
6.      get exclusive access to rows of chosen DRAM words and nearby cells
7.      set low $t_{RCD}$ for DRAM ranks containing chosen DRAM words
8.      for each bank:
9.          read data in $DW_1$  // induce activation failure
10.         write the read value of $DW_1$’s RNG cells to bitstream
11.         write original data value back into $DW_1$
12.         memory barrier  // ensure completion of write to $DW_1$
13.         read data in $DW_2$  // induce activation failure
14.         write the read value of $DW_2$’s RNG cells to bitstream
15.         write original data value back into $DW_2$
16.         memory barrier  // ensure completion of write to $DW_2$
17.         if $bitstream_size \geq num_bits$:
18.             break
19.         set default $t_{RCD}$ for DRAM ranks of the chosen DRAM words
20.         release exclusive access to rows of chosen words and nearby cells
# Summary Comparison Table

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Year</th>
<th>Entropy Source</th>
<th>True Random</th>
<th>Streaming Capable</th>
<th>64-bit TRNG Latency</th>
<th>Energy Consumption</th>
<th>Peak Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pyo+ [116]</td>
<td>2009</td>
<td>Command Schedule</td>
<td>✗</td>
<td>✓</td>
<td>18µs</td>
<td>N/A</td>
<td>3.40Mb/s</td>
</tr>
<tr>
<td>Keller+ [65]</td>
<td>2014</td>
<td>Data Retention</td>
<td>✓</td>
<td>✓</td>
<td>40s</td>
<td>6.8mJ/bit</td>
<td>0.05Mb/s</td>
</tr>
<tr>
<td>Tehranipoor+ [144]</td>
<td>2016</td>
<td>Startup Values</td>
<td>✓</td>
<td>✗</td>
<td>&gt; 60ns (optimistic)</td>
<td>&gt; 245.9pJ/bit (optimistic)</td>
<td>N/A</td>
</tr>
<tr>
<td>Sutar+ [141]</td>
<td>2018</td>
<td>Data Retention</td>
<td>✓</td>
<td>✓</td>
<td>40s</td>
<td>6.8mJ/bit</td>
<td>0.05Mb/s</td>
</tr>
<tr>
<td>D-RaNGe</td>
<td>2018</td>
<td>Activation Failures</td>
<td>✓</td>
<td>✓</td>
<td>100ns &lt; x &lt; 960ns</td>
<td>4.4nJ/bit</td>
<td>717.4Mb/s</td>
</tr>
</tbody>
</table>

*Table 2: Comparison to previous DRAM-based TRNG proposals.*
Figure 5: Data pattern dependence of DRAM cells prone to activation failure over 100 iterations
writes consist of three major sequential steps: 1) activation, 2) read/write, and 3) precharge, each of which determines the set of cells that fail at each DRAM operation, and it is up to the memory controller to obey them. If the memory controller issues the next command (Section 2.3.1), the DRAM latency parameters make the process to complete, and thus result in incorrect operation. A longer refresh interval results in failures, with increasing error properties of DRAM cells to devise PUFs across a chip. [34, 51, 54, 58, 74, 78, 79, 101].

Prior work shows that 1) process variations become large enough to satisfy the characteristics of an effective PUF. Section 5 presents our experimental evaluation of DRAM retention PUFs on modern LPDDR4 DRAM devices.

As detailed in Section 2.1, the ACT command opens a row and prepares it for accesses. The timing parameter eras the amount of time required for the activation process. This means that after issuing an ACT command to a row, the memory controller must wait for a delay of 

errors becomes large enough to satisfy the characteristics of an effective PUF. Section 5 presents our experimental evaluation of DRAM retention PUFs on modern LPDDR4 DRAM devices.
Sources of Retention Time Variation

• Process/voltage/temperature

• Data pattern dependence (DPD)
  - Retention times change with data in cells/neighbors
  - e.g., all 1’s vs. all 0’s

• Variable retention time (VRT)
  - Retention time changes randomly (unpredictably)
  - Due to a combination of various circuit effects
New failing cells continue to appear over time
- Attributed to variable retention time (VRT)

The set of failing cells changes over time

Error correction codes (ECC) and online profiling are necessary to manage new failing cells

Reprensentative chip from Vendor B, 2048ms, 45°C
Single-cell Failure Probability (Cartoon)

Probability of Read Failure vs. Refresh Interval (s)

- Idealized cell (retention time = 3s)
- Actual cell $N(\mu, \sigma) \mid \mu = 3s$
Any cell is more likely to fail
at a *longer* refresh interval
OR a *higher* temperature
Temperature Relationship

• Well-fitting exponential relationship:

\[ R_A \propto e^{0.22\Delta T} \quad R_B \propto e^{0.20\Delta T} \quad R_C \propto e^{0.26\Delta T} \]

• E.g., 10°C \(\sim\) 10x more failures
Retention Failures @ 45°C

**Unique**: failures not observed at lower refresh intervals

**Non-repeat**: failures observed at lower refresh intervals, but not at current

**Repeat**: failures observed at both current and lower refresh intervals

---

**Vendor A**

---

**Vendor B**

---

**Vendor C**

---

**SAFARI**
VRT Failure Accumulation Rate

- Vendor A (fit: $y = 3.3 \times 10^{-10} \cdot x^{3.6}$)
- Vendor B (fit: $y = 2.9 \times 10^{-10} \cdot x^{3.6}$)
- Vendor C (fit: $y = 6.6 \times 10^{-11} \cdot x^{3.8}$)
800 Rounds of Profiling @ 2048ms, 45°C
800 Rounds of Profiling @ 2048ms, 45°C
Individual Cell Failure Probabilities

- Single representative chip of Vendor B at 40° C
- Refresh intervals ranging from 64ms to 4096ms
Individual Cell Failure Distributions

(a) 
PDF probability

μ (s)

(b) 
PDF probability

σ (ms)

40°C
45°C
50°C
55°C
Single-cell Failures With Temperature

- Single representative chip of Vendor B
- \{mean, std\} for cells between 64ms and 4096ms