

Two Birds with One Stone: Differential Privacy by Low-power SRAM Memory

Jianqing Liu*, Member, IEEE Na Gong*, Member, IEEE Hritom Das, Member, IEEE

Abstract—The software-based implementation of differential privacy mechanisms has been shown to be neither friendly for lightweight devices nor secure against side-channel attacks. In this work, we aim to develop a hardware-based technique to achieve differential privacy by design. In contrary to the conventional software-based noise generation and injection process, our design realizes local differential privacy (LDP) by harnessing the inherent hardware noise into controlled LDP noise when data is stored in the memory. Specifically, the noise is tamed through a novel memory design and power downscaling technique, which leads to double-faceted gains in privacy and power efficiency. A well-round study that consists of theoretical design and analysis and chip implementation and experiments is presented. The results confirm that the developed technique is differentially private, saves 88.58% system power, speeds up software-based DP mechanisms by more than 10^6 times, while only incurring 2.46% chip overhead and 7.81% estimation errors in data recovery.

Index Terms—local differential privacy, static random-access memory, low-power, hardware-software co-design.

1 INTRODUCTION

Nowadays, with the rapid development of Internet-of-Things (IoT) technologies, the collection of user's and environment's data becomes immense. Such a rich set of data are frequently aggregated and analyzed using advanced Artificial Intelligence (AI) algorithms to make faster and more informed decisions. Despite the increased convenience and intelligence, the collection and analysis of data could incur significant privacy risks which are evidenced by recent privacy breach incidents such as historical data collected from Netflix [1] leading to unwanted intrusive marketing. Preserving data privacy thus attracts unprecedented amount of attention in this era.

The line of work on data privacy has a groundbreaking advance with the seminal work by Dwork et al. [2] in differential privacy (DP), which now becomes the *de facto* standard for preserving data privacy. The essence of the DP notion is to ensure the rigorous privacy guarantee of individual's data without sacrificing its general statistics. Local differential privacy (LDP), in contrast to the traditional DP notion in the centralized setting [3], [4], is the state-of-the-art approach which perturbs data locally with guarantees of plausible deniability [5]. As such, LDP protects individual's data against the untrusted curator and still preserves useful statistics of the original data. One example of the LDP

realization is RAPPOR which was developed by Google and is currently implemented in its Chrome browser [6].

However, existing realization or implementation of DP (including LDP) have many issues. That is, existing DP mechanisms typically add (subtract) to (from) the accurate value a secret number (noise) sampled from a probability distribution, which unfortunately bears the following drawbacks. (i) The sampling/adding/subtracting process in DP mechanisms is converted to the floating-point arithmetic in the operating system (OS), which actually deviates markedly from the DP's mathematical abstraction, due to rounding rules, compounding errors and many more in the floating-point arithmetic, and could lead to side-channel attacks. The vulnerability was spotted by Mironov et al. [7] and later exploited by Andryenco et al. [8] to launch attacks to the Fuzz differentially private database [9]. (ii) The noise sampling process in DP mechanisms requires function calls in the high-layer protocol stack, which are easily supported by legacy OSs such as iOS, Linux and Windows but may not stand for "slim" OSs in lightweight IoT devices such as sensors, cameras or lightbulbs. These IoT OSs are mostly vendor- and application-specific, which makes the implementation of DP mechanisms difficult to scale. (iii) The computation processes of DP mechanisms involve a significant number of CPU calls and memory accesses, which could be resource-consuming and power hungry. Although the cost of one-time DP randomization could be negligible, it will be prohibitively high for real-time or streaming data collection services when computations repeat. The consumed resources and energy consumption could become prohibitive for lightweight IoT devices such as streaming cameras and wearable ECG sensors.

The above concerns thus call for a more secure, scalable, and lightweight implementation of DP mechanisms, but very limited number of works have set foot on this research problem. To fill this gap, this paper develops a novel hardware-based technique that perturbs data when

* co-primary authors.

J. Liu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27606 USA e-mail: jliu96@ncsu.edu

N. Gong is with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL 36688 USA email: nagong@southalabama.edu

H. Das is with the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, TN 37996 USA e-mail: hdas@utk.edu
The work by J. Liu was supported in part by the National Science Foundation under grants ECCS-2312738 and CNS-2247273. The work by N. Gong was supported in part by the National Science Foundation under grants CNS-2211215 and OIA-2218046.

it is stored in memory — an ubiquitous component in all electronic devices — thereby accomplishing the vision of (differential) *privacy by design* [10]. In contrast to existing hardware security primitives like physical unclonable function (PUF), we do not seek to generate and later use hardware noises but rather perturb the data in situ such that it carries DP noises naturally. Moreover, compared with other hardware components like CPU, achieving DP in memory avoids data passing between CPU and memories. This design idea of computation-storage integration is reminiscent of a new concept — “computing in memory” (CIM) which is inspired by the highly energy-efficient mammalian brain where memory and processing are deeply intertwined [11]. In addition, by minimizing system interactions, we can also avoid threat interfaces and reduce system overhead.

To this end, we develop a novel static random access memory (SRAM) architecture, coined as `SRAM_DP`, to achieve *randomized response* (*RR*) [12] — the widely adopted approach to achieve LDP. The novelty of `SRAM_DP` lies in its integration of data storage and LDP and enhanced power efficiency. Specifically, by reducing the memory’s supply voltage, each custom memory cell becomes volatile and has a probability f to fail thus flipping the stored bit ($1 \rightarrow 0$ or $0 \rightarrow 1$); otherwise, following a probability $1 - f$ to maintain intact. In other words, a memory array — an array of memory cells — is randomized by independently applying the RR technique to each bit position through adjusting supply voltages and cell sizing. In so doing, the original data represented and stored in binary format in the memory is sanitized with LDP noises. There are a few challenges in developing `SRAM_DP`: (i) how to ensure that the perturbation on binary values preserve LDP on their original values (e.g., decimal, categorical values); (ii) how could the curator revert useful statistics from a large number of sanitized data with high utility; (iii) how to reduce system overhead such as latency in control signaling. This work addresses these challenges; and in summary, we push the frontier of the state-of-the-art as follows.

- 1) This is the first work that simultaneously realizes LDP in hardware (thereby achieving privacy-by-design) and improves power efficiency and system responsiveness, which promises a “win-win” outcome.
- 2) The data is transformed from its original type to the binary domain for LDP randomization. Theoretical analysis are provided for its rigorous bounds in privacy, utility, and side-channel leakage.
- 3) We design and implement a combination of hardware techniques — voltage down-scaling and custom memory design — to introduce controlled noise to the binary data. The developed hardware complies to LDP, preserves utility by protecting significant bits from failure, and reduces power consumption and latency.

To the best of the authors’ knowledge, the proposed hardware-based LDP realization has made the first and interdisciplinary attempt to exploit low-power memory design for LDP. Also, other than the presented SRAM-based scheme presented in this paper, the proposed technique can

be implemented using different memory technologies, such as DRAM and nonvolatile memories [13].

The rest of the paper is organized as follows. Section 2 provides preliminaries on LDP and SRAM failure characteristics. The details of `SRAM_DP` are described in Section 3. We present the theoretical study on privacy and utility in Section 4. The statistical recovery algorithms are then presented in Section 5. Section 6 demonstrates and discusses simulation and experiment results, which is followed by a discussion on `SRAM_DP` reliability issues in Section 7. Finally, Section 8 surveys related works and Section 9 concludes the paper.

2 PRELIMINARIES

2.1 Local Differential Privacy

LDP enriches the conventional differential privacy framework by ensuring the *indistinguishability* of two singleton databases (i.e., two arbitrary records) via a local randomization mechanism. The definition of LDP and its enabling technique are detailed as follows.

Definition 2.1 (Local Differential Privacy [6]). *LDP allows data contributor to locally perturb its own data using a randomization technique \mathcal{M} that satisfies ϵ -LDP, before sharing data to the data curator. Specifically, a randomized mechanism \mathcal{M} satisfies ϵ -LDP, if and only if for two data records v_1 and v_2 and for all output $S \subseteq \text{Range}(\mathcal{M})$, the following inequality holds:*

$$\Pr[\mathcal{M}(v_1) \in S] \leq e^\epsilon \Pr[\mathcal{M}(v_2) \in S]$$

To achieve LDP, it has been proven that the classic *random response* technique that is widely used in surveys of people’s “yes” or “no” answers for sensitive questions can be adapted to achieve LDP. Its definition is as follows.

Definition 2.2 (Randomized Response [12]). *For a private binary value $x \in \{0, 1\}$, a randomized response (RR) scheme follows a 2×2 design matrix to perturb x , that is $p_{sv} = p[y = s|x = v]$ ($s, v \in \{0, 1\}$) as the probability of the output being s when the input is v . The RR that satisfies ϵ -LDP follows that $p_{00} = p_{11} = \frac{e^\epsilon}{1+e^\epsilon}$ and $p_{01} = p_{10} = \frac{1}{1+e^\epsilon}$.*

To handle more general data types, Google developed RAPPOR [6] which applies RR to the Bloom filter encoded data.

2.2 SRAM Failure Characteristics

Memories are ubiquitous in today’s electronic systems. Among different memories, SRAM is mainly used as the cache and internal registers of a CPU thanks to its fast read/write speed. SRAM is volatile; its stored data is lost when the supply voltage is removed. This unique property has invited researchers to investigate the voltage down-scaling technique, a.k.a., low-power memory design, to balance power consumption and data integrity [14].

In addition to supply voltage, the SRAM failure characteristics also depend on its cell structure and transistor sizes. Specifically, among various SRAM designs, 6T and 8T are the two most widely-applied SRAM cells. Figure 1 shows 6T and 8T cells with the smallest silicon area in a 45 nm CMOS technology. 6T can enable compact memory design for its small silicon area while 8T can effectively

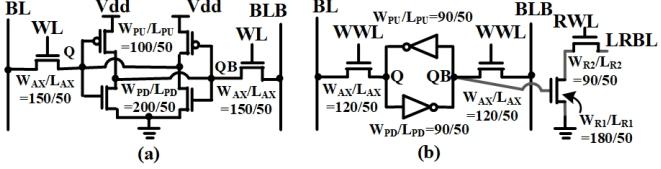


Fig. 1: 45nm SRAM cell schematics with the smallest silicon area: (a) 6T (C61) and (b) 8T (C81).

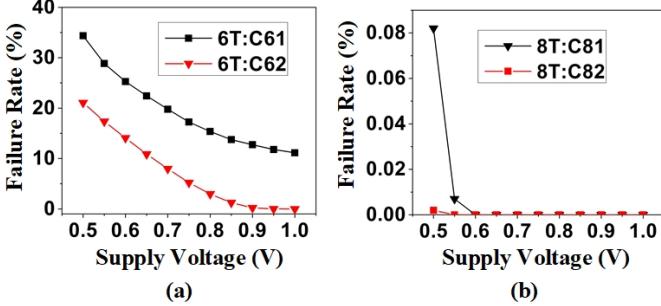


Fig. 2: Failure characteristics of 45 nm 6T and 8T cells.

reduce memory failures thanks to the decoupled read and write paths [15]. At low voltages, SRAM failures are mainly caused by process variations, in particular threshold voltage variations ($\sigma_{V_{th}}$), which can be expressed as $\sigma_{V_{th}} = \frac{A_{VT}}{\sqrt{WL}}$ [16]. A_{VT} is a technology dependent constant, and W, L respectively represent the width and length of the transistors. As the W and L of transistors increase, the threshold voltage variations are reduced which makes the SRAM more reliable in retaining its data at low voltages. In other words, upsizing SRAM cells avoids data losses under low voltages but comes with a tradeoff in silicon area.

In our experiments, we show in Figure 2 the failure characteristics of two 6T and two 8T cells with different transistor sizes. The detailed silicon area data of four cells is listed in Table I. As compared to the smallest 6T design, i.e., C61, the area overhead of the upsized 6T (C62), the smallest 8T (C81), and the upsized 8T (C82) is 15.4%, 9.6%, and 14.3%, respectively. In our analysis, 100,000 HSPICE Monte-Carlo simulations are performed in the worst process corner to obtain the failure rates of cells, i.e. “fs” (fast NMOS and slow PMOS) for 6T and “sf” (slow NMOS and fast PMOS) for 8T. As shown in Figure 2, the failure rate of a memory is monotone decreasing with respect to (w.r.t.) voltage and area. As also observed, with similar silicon area, 8T cell (C82) has a significant lower failure rate than upsized 6T cells (C62).

TABLE 1: Silicon areas of 45 nm memory cells.

Different Cells	Height (μm)	Width (μm)	Area (μm^2)	Area Ratio
6T:C61	0.45	1.523	0.685	1.000
6T:C62	0.45	1.758	0.791	1.154
8T:C81	0.45	1.663	0.751	1.096
8T:C82	0.45	1.740	0.783	1.143

3 PRIVACY BY DESIGN IN SRAM MEMORY

In this paper, we propose to utilize the SRAM failures at low voltages to inject LDP noises to the stored data, thereby achieving privacy by design while reducing system power consumption. This idea creates a “win-win” effect between privacy preservation and power efficiency — reminiscent of the old-saying: “kill two birds with one stone”. However, existing SRAM chips cannot accomplish the goal for two reasons. First, a failed memory cell cannot generate a random binary value. In theory, for any bit $x \in \{0, 1\}$ that is stored in a failed cell, it becomes ambiguous to determine implying that 0 and 1 are the possible readouts. However, in practice, for a specific SRAM chip after fabrication, its failed bits will always be readout as 0 or 1. For example, at low voltages, the failed bits of Cypress’s commercial memory chip (CY62146GN) consistently generate 1s [17]. This property, coined as “*fixed output upon cell failure*” is unfortunately unacceptable for achieving LDP. Second, SRAM cell failure exhibits a “*fault inclusion*” property; the cells that fail at voltage V_1 will certainly fail at a lower voltage V_2 where $V_2 < V_1$ [18]. This deterministic pattern will reveal side-information to adversaries who may conduct cell-failure profiling experiments and eliminate unlikely bit values. In this paper, we develop a new memory architecture to address the above two challenges thus accomplishing the vision of `SRAM_DP`.

3.1 Principle of `SRAM_DP`

Given a user’s value which could be in any format such as numbers, characters and categories, it is converted to binary bit strings $X = \{x_1, \dots, x_n\}$ and then stored at the user device’s SRAM cells $C = \{c_1, \dots, c_n\}$. Whenever the user needs to send her data to a curator, X is read out from the memory as bit strings $O = \{o_1, \dots, o_n\}$ with LDP noise. The overview of the proposed `SRAM_DP` mechanism is shown below, which consists of four steps.

- 1) **Bit Shift.** User’s memory system configures a finite set of permutation vectors as $\Pi = \{\pi_1, \dots, \pi_m\}$ offline. Before writing a value X to memory for storage, a permutation vector π_i is randomly selected from Π to re-organize the bit-string X which is then stored in the memory. The selected permutation vector is also stored in memory for data reserve-shift.
- 2) **Memory Storage.** The noise injection process is done by enabling low voltages for the custom memory. As discussed in Section 2.2, user’s low-voltage memory system generates cell failures across the memory. At a specific voltage, the failure positions can be determined by its built-in self test (BIST) process.
- 3) **Noise Injection.** Upon a memory read access, depending on the position of failed cells, the readout bits on non-failed cells are kept the same while the readout bits on failed cells are replaced by 0 or 1 randomly.
- 4) **Reverse Bit Shift.** User’s memory system re-applies the permutation vector π_i used in Step 1 to revert the generated bit string into O as the output.

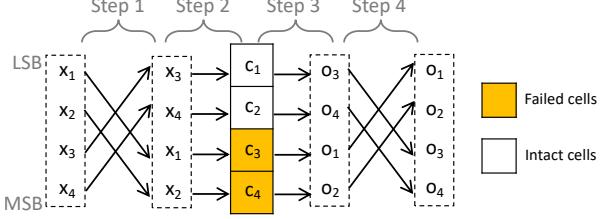


Fig. 3: A toy example showing the procedures of SRAM_DP.

To put the above steps into perspective, we present a toy example in Figure 3 that shows how a 4-bit data X is perturbed. The objective of each step is narrated as follows.

The **Bit Shift** in Step 1 serves for two purposes. First, permuting input X will address the “*fault inclusion*” problem and effectively defend against side-channel attacks. Second, by shifting bit positions of input X , we achieve fine-grained control over the failure rate of each bit position for high utility preservation. Specifically, the most significant bit (MSB) (resp. the least significant bit (LSB)) could be shifted to the cell with least (resp. highest) failure rate, as shown in Figure 3.

The **Memory Storage** in Step 2 is to let memory cells manipulate the bits stored therein. Specifically, by online adjusting the voltage V and offline designing the set of permutation vectors Π , we have control over a key parameter f in LDP; that is, a bit $x_i \in X$ will maintain its original value with probability $1 - f_i$ (i.e., by being stored in an intact memory cell) while staying in an ambiguous state with probability f_i (i.e., by being stored in a failed memory cell). Mathematically, the output of this step is

$$o_i = \begin{cases} 0 & \text{with probability } \frac{1}{2}f_i \\ 1 & \text{with probability } \frac{1}{2}f_i \\ x_i & \text{with probability } 1 - f_i \end{cases}$$

The **Noise Injection** in Step 3 is to address the “*fixed output upon cell failure*” issue. During the readout process, we inject random noises (0 or 1) to the failed cell positions.

The **Reverse Bit Shift** in Step 4 is to recover the original bit positions of X .

3.2 Hardware Architecture of SRAM_DP

To support the above procedures in our proposed SRAM_DP, we re-design the conventional SRAM memory architecture into a novel one which is shown in Figure 4.

To protect privacy, the failed bits in SRAM would be used to add random binary bits as noise. As shown in Figure 4, a data shuffler is added to implement the **Bit Shift** in Step 1, which is designed based on the configured permutation vectors. The random bit generation logic for the **Noise Injection** in Step 3 can be efficiently implemented by connecting multiplexers (MUX) to sense amplifier (SA) readout values of conventional SRAM. Each SA readout bit is connected to a MUX which is controlled by the received memory failure positions. If a memory failure is indicated, a random binary bit is enabled by selecting output of random bit generator. The failure position information will be used as the select signal of the MUX to control which bit to be the output. Similar to other existing fault position aware

mitigation techniques [19], the proposed SRAM_DP receives pre-determined locations of failed bits, which is usually executed either during post fabrication testing or during power-on self-test (POST). Finally, a data reshuffler is added to realize the **Reverse Bit Shift** in Step 4, based on the permutation pattern selection bits stored in the memory. The detailed hardware implementations and evaluation results of SRAM_DP will be discussed in Appendix B and Section 6, respectively.

4 PRIVACY AND UTILITY ANALYSIS

4.1 Compliance to LDP Notion

First of all, we attempt to draw a connection between the parameter f and the memory behaviors of SRAM_DP. It is clear that whether a bit $x_i \in X$ can maintain its original value, i.e., $o_i = x_i$, is determined by (i) the bit shift pattern and (ii) memory cell failure rate. For the former factor, suppose that a randomly selected permutation vector π could shift a bit x_i to any memory cell c_k following the probability vector $P_{\pi, x_i} = \{p_{x_i \rightarrow c_1}, \dots, p_{x_i \rightarrow c_n}\}$ where $\sum_{k=1}^n p_{x_i \rightarrow c_k} = 1$ and $x_i \rightarrow c_k$ denotes the event that bit x_i is shifted to (or stored at) the memory cell c_k . For the latter factor, when a specific voltage V is applied, a memory will exhibit the failure characteristics similar to Figure 2. Suppose that a memory cell c_k exhibits a failure probability p_{V, c_k} at voltage V . By taking these two factors into account, the probability that a bit retains its original value because of being stored in an intact cell is

$$P(o_i = x_i) = \sum_{k=1}^n p_{x_i \rightarrow c_k} (1 - p_{V, c_k}) = 1 - \sum_{k=1}^n p_{x_i \rightarrow c_k} p_{V, c_k}.$$

By relating the above equation to f , we can conclude that $f_i = \sum_{k=1}^n p_{x_i \rightarrow c_k} p_{V, c_k}, \forall x_i \in X$.

Next, we present the relationship between f and ϵ_∞ in the following theorem.

Theorem 4.1. *The SRAM_DP mechanism satisfies ϵ_∞ -differential privacy for any input bit strings X , where $\epsilon_\infty = \sum_{i=1}^n \ln(\frac{1 - \frac{1}{2}f_i}{\frac{1}{2}f_i})$.*

Proof. Without loss of generality, assume $X_1 = \{x_1 = 0, \dots, x_n = 0\}$ and $X_2 = \{x_1 = 1, \dots, x_n = 1\}$ meaning that X_1 and X_2 are complimentary and the l_1 norm is maximized. Besides, we know that for any bit $x_i \in X$, SRAM_DP generates the readout bit $o_i \in O$ with probability

$$\begin{aligned} P(o_i = 1 | x_i = 1) &= \frac{1}{2}f_i + 1 - f_i = 1 - \frac{1}{2}f_i, \\ P(o_i = 1 | x_i = 0) &= \frac{1}{2}f_i. \end{aligned} \tag{1}$$

Similarly, $P(o_i = 0 | x_i = 0) = 1 - \frac{1}{2}f_i$ and $P(o_i = 0 | x_i = 1) = \frac{1}{2}f_i$. Then, for any output $O^* = \{o_1, \dots, o_n\}$,

$$\begin{aligned} P(O = O^* | X = X_1) &= (\frac{1}{2}f_1)^{o_1} \cdot (1 - \frac{1}{2}f_1)^{1-o_1} \times \dots \\ &\quad \times (\frac{1}{2}f_n)^{o_n} \cdot (1 - \frac{1}{2}f_n)^{1-o_n} \end{aligned}$$

$$\begin{aligned} P(O = O^* | X = X_2) &= (1 - \frac{1}{2}f_1)^{o_1} \cdot (\frac{1}{2}f_1)^{1-o_1} \times \dots \\ &\quad \times (1 - \frac{1}{2}f_n)^{o_n} \cdot (\frac{1}{2}f_n)^{1-o_n} \end{aligned}$$

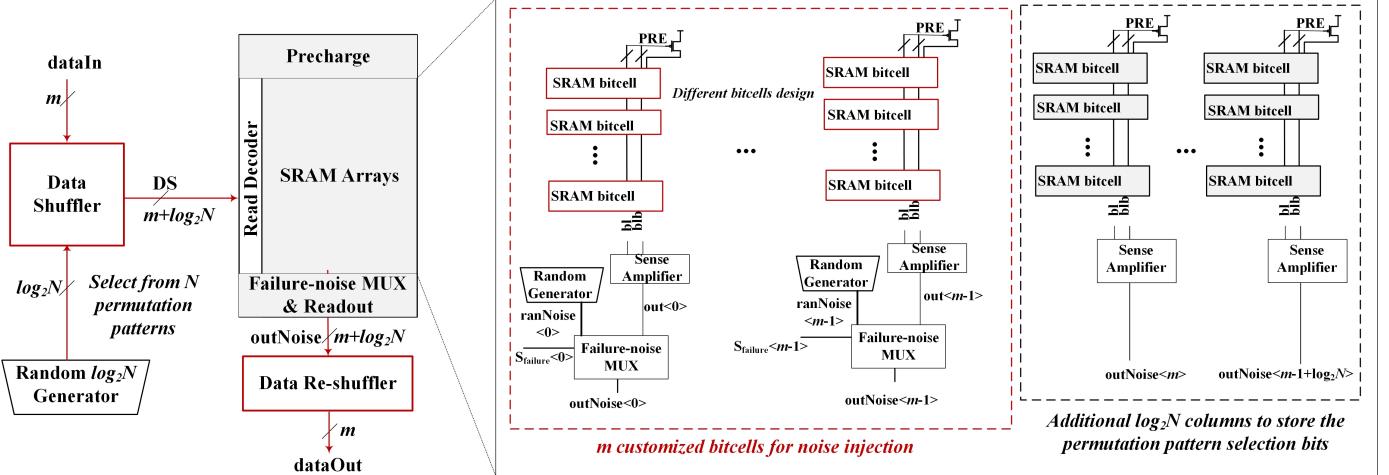


Fig. 4: Hardware Architecture of the proposed SRAM_DP.

Consider that the SRAM_DP design abstractly provides a randomization function M where output $M(X) \in S$. For the LDP condition to hold, we need to ensure the ratio of probabilities of getting the same output for any two arbitrarily “adjacent” inputs X_1 and X_2 to be bounded by $\exp(\epsilon_\infty)$. Mathematically, we have

$$\begin{aligned} \frac{P(M(X_1) \in S)}{P(M(X_2) \in S)} &= \frac{P(O = M(X_1) | X = X_1)}{P(O = M(X_2) | X = X_2)} \\ &= \frac{\sum_{O^* \in S} P(O = O^* | X = X_1)}{\sum_{O^* \in S} P(O = O^* | X = X_2)} \\ &\leq \max_{O^* \in S} \frac{P(O = O^* | X = X_1)}{P(O = O^* | X = X_2)} \\ &= \left(\frac{1}{2} f_1\right)^{2o_1-1} \cdot \left(1 - \frac{1}{2} f_1\right)^{1-2o_1} \times \dots \\ &\quad \times \left(\frac{1}{2} f_n\right)^{2o_n-1} \cdot \left(1 - \frac{1}{2} f_n\right)^{1-2o_n} \\ &= \prod_{i=1}^n \left(\frac{1}{2} f_i\right)^{2o_i-1} \\ &\leq \prod_{i=1}^n \left(\frac{1 - \frac{1}{2} f_i}{\frac{1}{2} f_i}\right) = \exp(\epsilon_\infty) \end{aligned}$$

where the second inequality holds when every $o_i = 0$. The reason is as follows. First of all, it is clear that $\frac{1}{2} f_i$ takes values from $[0, 1]$ because $f_i \in [0, 1]$. Then, provided that o_i is binary (0 or 1), the exponent $2o_i - 1$ is either -1 or 1, respectively. Therefore, the upper bound of the above equation occurs when $o_i = 0, \forall i$. ■

4.2 Side-channel Analysis

In addition to proving that SRAM_DP guarantees LDP, it is equally vital to examine if SRAM_DP leaks any side-channel information that may jeopardize data privacy. Amongst the designed components of SRAM_DP, the power control unit — a firmware executed by CPU — for voltage scaling is vulnerable to (remote) power profiling attacks [20], [21]. For instance, Hertzbleed threat allows attackers to steal full cryptographic keys by observing variations in CPU frequency enabled by dynamic voltage scaling [20]. Nonetheless, in our SRAM_DP design, the voltage profile is only

correlated with ϵ as shown in the above proof and Table 3. Knowing ϵ gives no benefits to attackers — ϵ by default is a public information, not to mention that extracting ϵ is almost impossible because attackers have no clues of victim devices’ SRAM specifications such as its transistor sizing, as illustrated earlier in Figure 2.

Even for the most adversarial case where attackers claim root access to CPU and gain full knowledge of SRAM specifications, LDP noises cannot be stolen or in any way inferred because they are generated and utilized on-the-fly during the SRAM read/write process (achieved by the sense amplifier and noise generator in Figure 4) and they are never stored. Moreover, the sensitive data are naturally perturbed when they are written into memory. Therefore, any attempt in reading data from SRAM will not be able to reverse their true values.

4.3 Utility Analysis

Theorem 4.2. *The expected l_1 utility loss of SRAM_DP, denoted as $\mathbb{E}(|O - X|)$, has a $O(c^n)$ upper bound where c is a constant relevant to $\{f_1, \dots, f_n\}$.*

Proof. Denote the change of bit value as $\Delta a_i = o_i - x_i \in \{-1, 0, 1\}$. Without any prior assumption on x_i , i.e., it can take value 0 or 1 with equal probability, the probability of Δa_i is given as

$$P(\Delta a_i) = \begin{cases} \frac{1}{4} f_i, & \Delta a_i = -1 \\ 1 - \frac{1}{2} f_i, & \Delta a_i = 0 \\ \frac{1}{4} f_i, & \Delta a_i = 1 \end{cases} \quad (2)$$

according to Eq.(1). Upon mapping randomization in binary domain back to its decimal format, we can obtain the change of data’s original value $\Delta A \stackrel{\text{def}}{=} O - X = \Delta a_{n-1} 2^{n-1} + \Delta a_{n-2} 2^{n-2} + \dots + \Delta a_0 2^0$. ΔA is a random variable whose value is an integer from $\{-2^n + 1, \dots, 0, \dots, 2^n - 1\}$. Then, to derive the expected value of $|O - X|$ (or $|\Delta A|$) which is l_1 loss of SRAM_DP, we have the following Lemma for the probability mass function (PMF) of ΔA .

Lemma 4.3. *The PMF of ΔA is zero-mean and symmetric w.r.t. $\Delta A = 0$. The proof is shown in the Appendix A.*

With Lemma 4.3, we can derive $\mathbb{E}(\Delta A)$ as follows:

$$\begin{aligned}\mathbb{E}(|O - X|) &= \sum_{|O-X|=0}^{2^n-1} |O - X| P(|O - X|) \\ &\stackrel{(1)}{=} 2 \sum_{\Delta A=0}^{2^n-1} \Delta A P(\Delta A) \\ &\stackrel{(2)}{=} 2 \sum_{\Delta A=0}^{2^n-1} \Delta A \sum_{K \in \mathcal{K}_{\Delta A}} \left[\prod_{i \in I} \left(1 - \frac{1}{2} f_i\right) \prod_{i \in I'} \left(\frac{1}{4} f_i\right) \right] \\ &\stackrel{(3)}{\leq} 2 \sum_{\Delta A=0}^{2^n-1} \Delta A \sum_{k=0}^n \left[\prod_{i=0}^k \left(1 - \frac{1}{2} f_i\right) \prod_{j=k+1}^n \left(\frac{1}{4} f_j\right) \right] \\ &\stackrel{(4)}{=} 2^{2n-1} \sum_{k=0}^n \left[\prod_{i=0}^k \left(1 - \frac{1}{2} f_i\right) \prod_{j=k+1}^n \left(\frac{1}{4} f_j\right) \right]\end{aligned}$$

We have equality (1) because of Lemma 4.3. The rationale of equality (2) is that any value of ΔA is due to a combination of $\{\Delta a_{n-1}, \dots, \Delta a_0\}$. For example, for a three-bit ($n=3$) decimal number, there are two combinatorics that can result in $\Delta A = 5$, which are $\mathcal{K}_{\Delta A=5} = \{\{1, 1, -1\}, \{1, 0, 1\}\}$. Here, $\mathcal{K}_{\Delta A}$ represents the set of combinatorics that lead to ΔA . Any combinatoric K in $\mathcal{K}_{\Delta A}$ is a specific realization of ΔA . For any K , it contains n corresponding values (e.g., $K(j)$ is the j^{th} element) in $\{\Delta a_{n-1}, \dots, \Delta a_0\}$. Then, the probability of $\Delta A = a$, $\forall a \in \{-2^n + 1, \dots, 0, \dots, 2^n - 1\}$, is simply the addition of probabilities for having every combinatorics in $\mathcal{K}_{\Delta A=a}$. More specifically, the probability of any combinatoric K in $\mathcal{K}_{\Delta A=a}$ is the product of probabilities in Eq.(2) for all $\{\Delta a_{n-1}, \dots, \Delta a_0\}$. For notation simplicity, denote $I = \{j | K(j) = 0, 1 \leq j \leq n\}$ and $I' = \{j | K(j) \neq 0, 1 \leq j \leq n\}$. For instance, in the above three-bit example, $P(\Delta A = 5) = \left(\frac{1}{4} f_1\right) \left(\frac{1}{4} f_2\right) \left(\frac{1}{4} f_3\right) + \left(\frac{1}{4} f_1\right) \left(1 - \frac{1}{2} f_2\right) \left(\frac{1}{4} f_3\right)$.

Note that deriving $\mathcal{K}_{\Delta A}$ for all possible ΔA requires exhaustive search which is significantly time-consuming as the search space increases exponentially w.r.t. n . Nonetheless, it is intuitive to assert that for any specific value of ΔA , as long as the number of zero-valued elements in $\{\Delta a_{n-1}, \dots, \Delta a_0\}$ are fixed, the remaining non-zero elements will be deterministic. In light of it, we have inequality (3) by relaxing the carnality of $\mathcal{K}_{\Delta A}$ to n — the maximum number of zero-valued elements. Moreover, without loss of generality, we can neglect the order of cell failure positions $\{f_1, \dots, f_n\}$ and simply consider the number of failed cells that lead to $\Delta a_* = 0$ (with probability $1 - \frac{1}{2} f_*$). Then, we can easily conclude with the result by calculating the sum of arithmetic series in equality (4).

Obviously, the expected l_1 utility loss of SRAM_DP has an $O(c^n)$ upper bound, in which c is a constant relevant to $\{f_1, \dots, f_n\}$. In fact, this claim will become more intuitive at a special case of homogeneous failure rate $f_1 = \dots = f_n = f$, for which the above derivation can be further simplified into

$$\mathbb{E}(|O - X|) \leq (4^n - 2^n) \cdot \frac{\left(1 - \frac{1}{2} f\right)^{n+1} - \left(\frac{1}{4} f\right)^{n+1}}{1 - \frac{3}{4} f}.$$

All the above analytical results of Lemma 4.3 will later be validated numerically in Figure 7. ■

5 STATISTICS RECOVERY ALGORITHMS

In this work, we assume each user's data is independently and identically distributed (i.i.d.) with a specific distribution \mathcal{P} . After applying SRAM_DP, a data curator needs to recover certain statistics of \mathcal{P} . In this section, we adopt two statistics recovery algorithms, namely the expectation-maximization (EM) algorithm and the constrained linear regression (CLR) algorithm, to achieve the goal. Note that EM and CLR are well-established methods and the purpose of selecting them is to convey the idea that the hardware-perturbed data by SRAM_DP can be easily analyzed by classical recovery algorithms.

Before running recovery algorithms, the data curator should (i) be aware and apply the same encoding algorithm that is used at users' SRAM system to convert data of arbitrary format into bit strings, (ii) obtain every user's failure rate vector $F = \{f_1, \dots, f_n\}$ thus the LDP parameter ϵ , and (iii) construct a set of data candidates Ω that cover all the possible values of users' choices. For instance, for heart beat monitoring, a reasonable heart beat range is between 30bpm to 150bpm so $\Omega = \{X | 30 \leq X \leq 150, X \in \mathbb{Z}^+\}$ and a user's specific input $X \in \Omega$. These requirements could be easily fulfilled by relying on users to piggyback them with the sanitized data or the curator retrieving them from side channels without compromising the differential privacy condition.

5.1 EM Algorithm

The essence of EM algorithm is to find the maximum-likelihood estimates (MLEs) through iteration. In the context of this work, the data curator is interested to search through the candidate set Ω for the most likely X^* that contributes to the observed sanitized data O . Specifically, our EM algorithm consists of the following steps.

- 1) **Initialization.** Assume the data curator has no additional prior information about users' data, it will initialize the prior probability by setting a uniform distribution as $P(X) = \frac{1}{|\Omega|}, \forall X \in \Omega$.

- 2) **Likelihood Calculation.** When the data curator observes a sanitized bit strings $O = \{o_1, \dots, o_n\}$ from a user, it calculates the likelihood of generating O by any candidate from Ω . Specifically, for the SRAM_DP mechanism, an output bit $o_i \in O$ maintains its original value with probability $1 - \frac{1}{2} f_i$ whereas flips with probability $\frac{1}{2} f_i$ as derived in Eq.(1). Then, for a candidate represented in bit string $X = \{x_1, \dots, x_n\}$, each of its bit has the likelihood of $P(o_i | x_i) = (\frac{1}{2} f_i)^{\beta_i} \cdot (1 - \frac{1}{2} f_i)^{1-\beta_i}$ of generating o_i for $\forall o_i \in O$, where $\beta_i = \text{XOR}(o_i, x_i)$ and $\text{XOR}()$ is the boolean exclusive OR operator. Then, for any user's submitted data, we have

$$P(O | X) = \prod_{i=1}^n (\frac{1}{2} f_i)^{\beta_i} \cdot (1 - \frac{1}{2} f_i)^{1-\beta_i}, \quad \forall X \in \Omega. \quad (3)$$

- 3) **Posterior Calculation.** Given all the conditional distributions of one particular observation O as above, the corresponding posterior probability can be calculated according to the Bayesian theorem, which is given below.

$$P(X|O) = \frac{P(X) \cdot P(O|X)}{\sum_{X \in \Omega} P(X) \cdot P(O|X)}, \quad \forall X \in \Omega. \quad (4)$$

- 4) **Update and Iteration.** After calculating the posterior for every user, we update the prior probability for the calculation of next round. Specifically, we take the average of posterior probabilities of all U users in the current iteration and assign it to the prior probability used for the next iteration, mathematically,

$$P(X) = \frac{1}{U} \sum_{m=1}^U P(X|O_m), \quad \forall X \in \Omega$$

The aforementioned steps continue till convergence. We can set a stopping criteria as $\max_{X \in \Omega} |P_t(X) - P_{t-1}(X)| \leq \delta$.

5.2 CLR Algorithm

Another popular algorithm to recover statistics is the empirical estimation method [22], which computes an empirical estimate $\hat{\mathcal{P}}$ of \mathcal{P} using the empirical distribution \hat{Q} of the sanitized data O given the LDP randomization matrix (or conditional probability matrix) \mathcal{M} . Mathematically, we have $\hat{\mathcal{P}}\mathcal{M} = \hat{Q}$, whose closed-form solutions can be easily computed and obtained. As the number of users increases, the empirical distribution \hat{Q} gets asymptotically close to the true distribution of sanitized data Q , thus $\hat{\mathcal{P}}$ also converges to \mathcal{P} given invariant \mathcal{M} . However, when the number of users is small, we cannot ensure the positiveness of $\hat{\mathcal{P}}$. One method is to use Bonferroni correction [23] to eliminate estimates below a significance level. Here in this work, we will leverage a constrained linear regression model to obtain $\hat{\mathcal{P}}$. The detailed steps are shown as follows.

- 1) **Construct \mathcal{M} .** SRAM_DP applies a randomization matrix $\mathcal{M} \in \mathbb{R}_{\geq 0}^{|\Omega| \times |\Omega|}$, which characterizes the conditional probability of mapping an input X to a sanitized output O . The calculation of \mathcal{M} follows Eq.(3).
- 2) **Calculate $\hat{\mathcal{P}}$.** We first build the empirical distribution \hat{Q} by normalizing the frequencies of each record. Then, we solve the following constrained least-square problem:

$$\begin{aligned} \min_{\hat{\mathcal{P}}} \quad & \frac{1}{2} \|\hat{\mathcal{P}}\mathcal{M} - \hat{Q}\|_2^2 \\ \text{s.t.} \quad & \mathbb{E}[X^j] = m_j, \quad j \geq 1; \\ & 0 \leq \hat{\mathcal{P}}(X) \leq 1, \quad \forall X \in \Omega. \end{aligned} \quad (5)$$

to obtain the optimizer $\hat{\mathcal{P}}$ as the estimation for the input distribution \mathcal{P} . Note that $\mathbb{E}[X^j]$ is the j^{th} moment of the probability distribution \mathcal{P} , which is the prior knowledge known to the reconstruction algorithm.

6 PERFORMANCE EVALUATION

In this section, we give a thorough evaluation of our design in both simulations and experiments. We first present several metrics for quantitative evaluation. We consider two specific adversarial side information, one of which strongly pertains to our SRAM_DP mechanism, to examine the absolute privacy level besides the above ϵ_∞ -DP theoretical

analysis. We then assess how accurately the EM-based and CLR-based algorithms can recover original statistics. Furthermore, based on the simulation results, we implement a proof-of-concept SRAM to achieve SRAM_DP. We carry experiments to validate its performance w.r.t. the proposed metrics.

6.1 Evaluation Metrics

Privacy Meter: The SRAM_DP design conforms to the differential privacy notion given the proof in Theorem 4.1. In other words, an adversary is unable to distinguish any datum X from its “neighbouring” data. Without loss of generality, we denote that such “neighbouring” data collectively form an *indistinguishable set*, whose size is determined by the number of failed cells in SRAM — denoted as z . Obviously, an adversary will have to guess the true datum from a larger *indistinguishable set* when there are more failed cells, leading to a higher uncertainty for the adversary. Since different privacy only poses privacy guarantee within the *indistinguishable set*, the size of the *indistinguishable set* that is 2^z has a strong implication on the adversarial inference accuracy. In this work, we assume that an adversary follows the maximum likelihood estimation (MLE) to obtain the inferred data \hat{X} from its observation O . Then, we follow the privacy meter to measure the *in-accurateness* of adversarial inference as $\text{IA} = \sum_X P(X|O)|\hat{X} - X|$ which is equivalent to the privacy level.

Utility Meter: Privacy always comes with the cost of utility, and our design is no exception. Intuitively, more failed cells or higher cell failure rates lead to more severe data corruption thus lower utility. Moreover, utility is also affected by the positions of failed cells, in the sense that a higher utility is retained when failed cells occur at less significant bits. To characterize utility level of our design, we use l_1 loss (i.e., absolute error) to measure the utility loss as $\text{UL} = \sum_O P(O|X)|O - X|$.

6.2 Adversarial Side Information

We assume two specific adversarial knowledge. The simplest one is to assume an oblivious adversary, which possesses no additional information about the original data and has to guess randomly. That is to say, the prior probability in the MLE inference $P(X) = \frac{1}{2^z}$. We denote such adversarial knowledge as **K1**. Furthermore, we consider a more capable adversary that knows the statistics (e.g., range, mean and variance) of the original data set but is unaware of any specific individual’s data. This adversarial knowledge is denoted as **K2**. Then, an adversary with **K2** can easily derive a more accurate $P(X)$ by calculating the frequencies of specific elements, for instance, the frequency of data that is in binary form of “111****” (i.e., a 8-bit value with three leading 1s in MSBs followed by five wild cards). Note that both **K1** and **K2** are aware of the details of SRAM_DP such as cell failure rates and ϵ .

6.3 Input Data

We generate 1,000 8-bit random binary numbers (i.e., of values 0-255) as input data by sampling a Gaussian distribution with $\mu = 125$ and $\sigma = 20$. For notional convenience, we

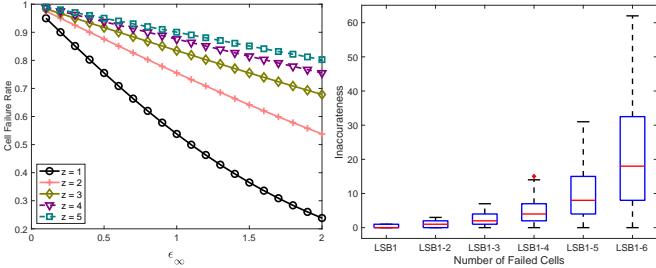
(a) Cell failure rates w.r.t. ϵ_∞ (b) IA w.r.t. z

Fig. 5: Privacy analysis.

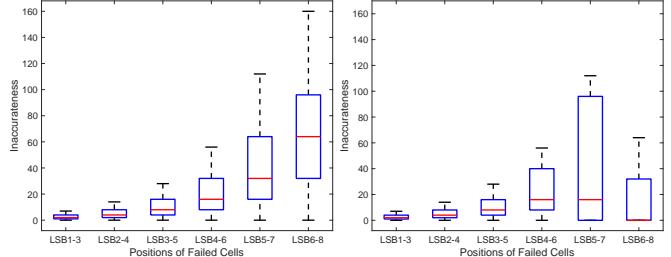
refer every bit using “LSB*”. For instance, LSB1 indicates the least significant bit whereas LSB1-3 indicates the least three significant bits.

6.4 Simulation Results and Analysis

Simulation Setup: To simulate the algorithmic procedures of `SRAM_DP`, we use MATLAB_R2022a to implement the key steps in Section 3.1. Specifically, we first sample 1,000 random numbers, convert them into binary values, shuffle their bit positions, and follow an i.i.d., Bernoulli multivariate vector (mimicking the SRAM cell failure probabilities) to decide if any binary values need to be changed into an constant value (e.g., 1s, which reflects the *fixed output upon cell failure* property). Next, we replace the prior changed values with 1s or 0s randomly. In the end, we reverse the binary values back to their original bit positions, which completes the `SRAM_DP` randomization life-cycle.

Privacy Analysis: To preserve the ϵ_∞ -DP guarantee in Theorem 4.1, we must select appropriate cell failure rates f , the number of failed cells z , and the positions of failed cells. For the sake of simplicity, we assume homogeneous failure rates among memory cells. Then, we plot how the cell failure rate changes w.r.t. ϵ_∞ as shown in Figure 5a. The result indicates that a stronger privacy (i.e., smaller ϵ_∞) is achieved by increasing cell failure rates and confining failures in a small number of memory cells. For example, a single cell failure with probability 0.5, i.e., $z = 1$ with $f = 0.5$, yields $\ln 3$ -DP. Nevertheless, a small *indistinguishable set* for a small z gives rises a higher chance for adversaries to infer the correct value. As shown in Figure 5b, a larger z offers a higher IA under the adversarial knowledge **K1**.

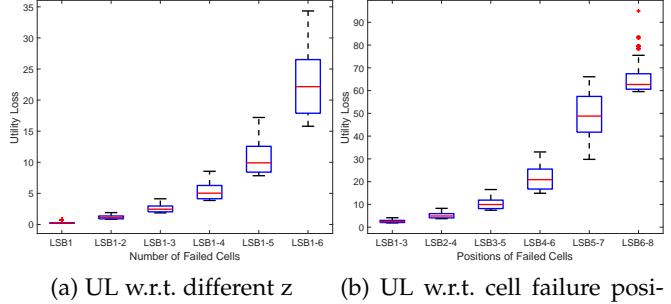
When it comes to different adversarial knowledge, we evaluate how it can affect adversaries’ capability in compromising data privacy. We present simulation results in Figure 6. When compared with an adversary with **K1**, its peer with **K2** outperforms in inference accuracy at the high end of the spectrum (i.e., failed cells close to MSB). This can be reasoned by looking into the characteristics of the input data. As shown in Figure 8, there are nearly no samples beyond $X = 192$ (in binary form 110*****), so the **K2** can confidently eliminate 110**** and 111**** from the *indistinguishable set* leading to higher inference accuracy. Nevertheless, we notice that the **K1** adversary does not have an evident advantage over **K2** when failed cells are clustered at low-to-medium bit positions. This provides us with a great insight when rendering cell failures at the design of `SRAM_DP`.



(a) IA with prior knowledge K1

(b) IA with prior knowledge K2

Fig. 6: Privacy level under different prior knowledge.

(a) UL w.r.t. different z

(b) UL w.r.t. cell failure positions

Fig. 7: Utility analysis.

Utility Analysis: In addition to the privacy analysis, we evaluate utility losses. In this simulation, we set $\epsilon = \ln 3$. As shown in Figure 7a, it is obvious that utility loss increases exponentially as there are more failed cells. This finding matches with our theoretical analysis in Theorem 4.2. Nevertheless, it is not advisable to select a small number of failed cells for the reasons that were discussed in the privacy analysis.

Moreover, Figure 7b shows that the utility loss is moderately low when failed cells are placed at the low end of bit positions, which is intuitive.

Data Re-construction: Here, we demonstrate how well original data can be re-constructed from the sanitized one by using EM-based and CLR-based algorithms. Specifically, in the EM-based algorithm, we set the convergence criteria of $\delta = 10^{-3}$ and assume no prior information about input statistics. Likewise, no information about original distribution is assumed in the CLR-based algorithm. We consider three failed cells with failure patterns **F1**, **F2** and **F3** for $\epsilon = \ln 3$. Figure 8 shows the reconstructed frequency histograms versus the original one. There are two take-away from the results. First, both algorithms in general perform better when the failed positions are near LSB. Second, the EM-based algorithm is preferred at low-end bit failures while the CLR-based one slightly outperforms its peer at high-end bit failures. Another note is that if any prior information were available such as variance, we could significantly improve upon reconstructing the shape of the distribution via smoothing.

6.5 Experiment Results and Analysis

A key take-away from the above simulation results is that the number of, the location of, and the failure rate of failed

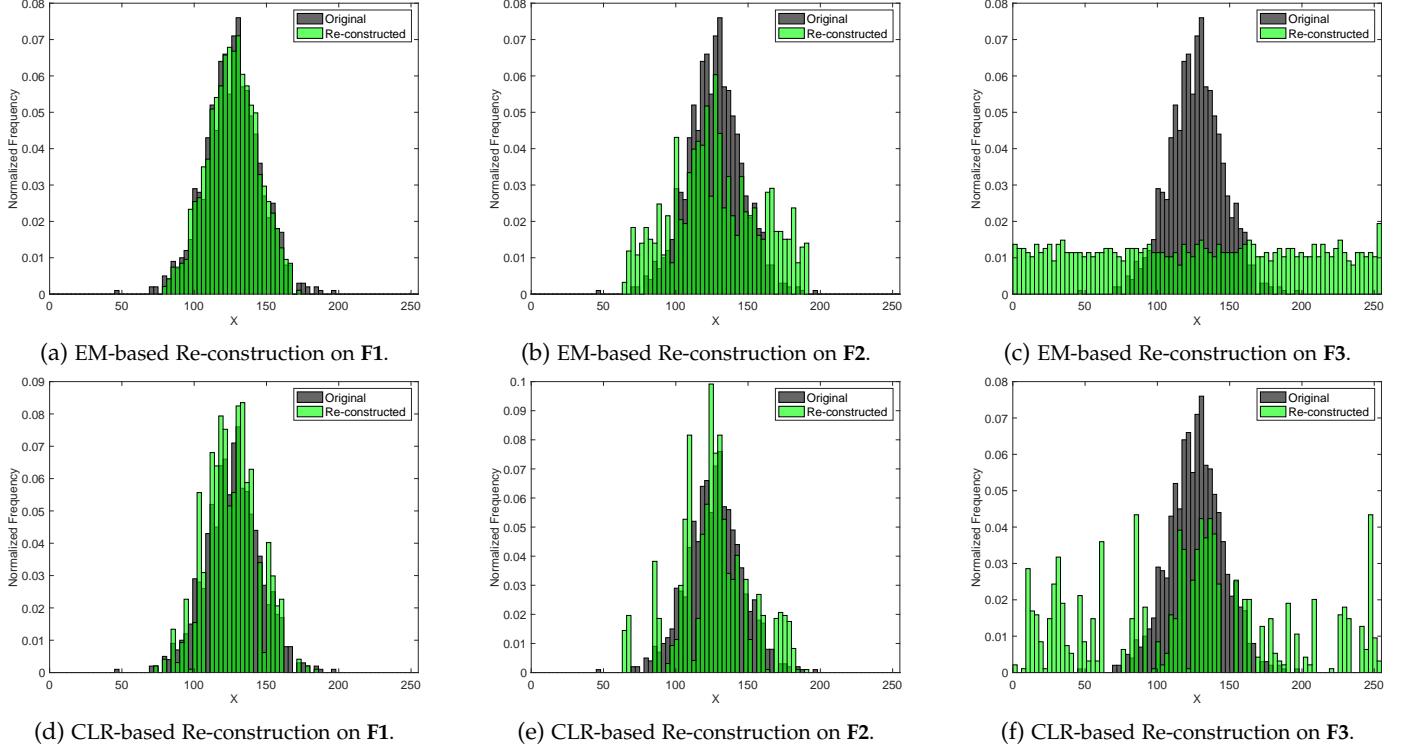


Fig. 8: The normalized frequencies of input data are plotted in grey. Software-generated DP noises are drawn based on three failure patterns F1: $f = [0, 0, 0, 0, 0.82, 0.82, 0.82]$, F2: $f = [0, 0, 0.82, 0.82, 0.82, 0, 0, 0]$ and F3: $f = [0.82, 0.82, 0.82, 0, 0, 0, 0, 0]$ for $\epsilon = \ln 3$. Green plots show the re-constructed distributions under EM-based and CLR-based algorithms.

cells jointly affect IA, UL, and data re-construction accuracy. A fair principle is to place failed cells as close to LSBs as possible and keep the number of failed cells moderate. In this subsection, we instantiate a memory configuration for implementation that balances IA, UL, and data re-construction accuracy. That is, we set four failed cells at the four LSBs (i.e., LSB1-4) with the same failure rate whose value is guided by Theorem 4.1.

The design details of SRAM_{DP} are outlined in the Appendix B. In brief, SRAM_{DP} is implemented based on a 45 nm CMOS technology and the standard supply voltage is 1V. SRAM_{DP} is designed with a layout of 8 memory banks and each bank has 128 words × 10 bits.

Analysis on IA, UL and Data Re-construction: Our experiment is based on the setting of 0.50V supply voltage. We write and then read out the same 1,000 data, and analyze the performance of IA, UL and data re-construction compared with the previous simulation results.

First, as shown in Figure 9a, the re-constructed data is plotted against the original data. Obviously, the EM-based algorithm has a quite low miss-detection rate and its frequency estimation for most data records are accurate. Nonetheless, if we use the Mean Squared Error (MSE) metric to capture how well data is reconstructed, the EM-based algorithm achieves MSE = 69.56 in the experiment while MSE = 5.12 in the simulation. Apparently, the experimental performance is inferior to the simulation. The reason is because of the inconsistency of failure rate f in the sense that EM-based algorithm takes the simulated or theoretical value of f (i.e., 81.57%) as the input parameter to recover the experimentally perturbed data, in which the exact value

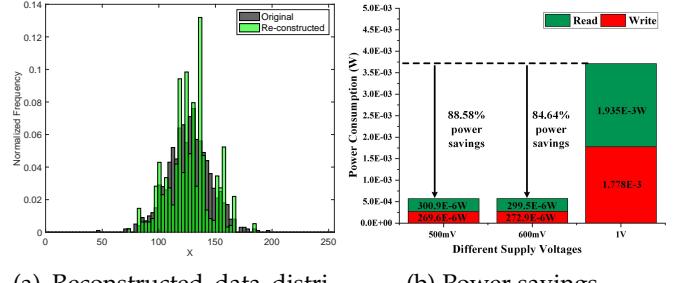


Fig. 9: Estimation accuracy and power saving.

of f may deviate from the simulated one.

Next, we demonstrate and compare the performance of IA and UL. As shown in Figure 10, the simulation and experiment results are comparatively similar, which implies that the perturbed data that is generated in the experiment resembles well to the one generated in the simulation.

Power Saving: Figure 9b shows the power consumption of SRAM_{DP} at three different voltages including standard supply voltage 1V and two low voltages 0.60V and 0.50V. For each test case shown in Figure 9b, the average power consumption is measured for writing 8-input input data 10100110₂ to a random word in a 128 word × 10 bit memory bank, which is initialized to 10101001₂, followed immediately by reading 10100110₂ from the same word, such that all read/write memory operations are equally included (i.e., reading "0" and "1," and writing "0" to "0," "0" to "1," "1" to "0," and "1" to "1"). During this process, the two MSBs

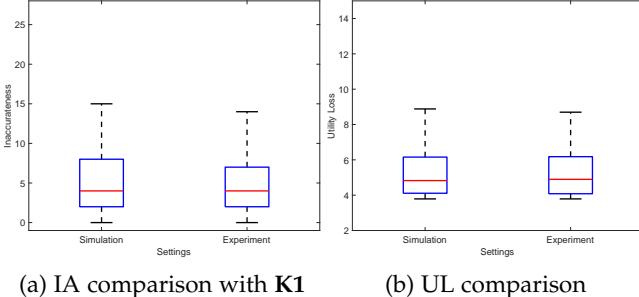


Fig. 10: Performance Comparison on IA and UL.

of the selected 10-bit word store the pattern selection signal $S[1:0]$, which is generated by the 2-bit random generator (Figure 17). Our first test case is that SRAM_DP operates at 1V, which is the standard supply voltage without memory failures, and it shows that the total power consumption is 3.713×10^{-3} W. Our next test case is to let SRAM_DP operate at 0.60V, the cell failure rate is 60.26%, and the corresponding ϵ value is 3.36 (Table 3). With noise injection, SRAM_DP consumes 5.724×10^{-4} W, which is 84.64% lower than base case. Our final test case is that SRAM_DP operates at 0.50V and the cell failure rate is increased to 81.57%. In this case, SRAM_DP achieves the LDP with $\epsilon = 1.49$ and it obtains 88.58% power savings as compared to the base case.

Timing Diagram: The timing diagram of SRAM_DP is shown in Figure 13. Two supply voltages are used in SRAM_DP including Vdd! and Vdd1!. Vdd1! is applied for memory cells and it can be adjusted to enable the target failure rate for noise injection, as shown in Table 3. In this timing diagram, the value of Vdd1! is 0.50V. Vdd! is used as a global supply voltage for all other designs to support the functionality of the memory and its values is 0.60V. Also, two clock signals are adopted in SRAM_DP: Clk is the global clock signal for the entire memory and Clk_1 is used to generate random noise ranNoise[3:0] for four LSBs in the noise injection process. writeEn and readEn are write enable and read enable signals, respectively. dataIn[7:0] is 8-bit input data and dataOut[7:0] is the final output of the memory. The 2-bit control signal S[1:0] are the generated 2-bit random bits for permutation pattern selection.

The working process is detailed as follows. As one example which is highlighted in green in Figure 13, 8-bit input data dataIn[7:0] = 11110101₂ is applied to the memory. The generated random signal S[1:0] is 01₂, so the permutation pattern $\pi_2-[0, 1, 2, 3, 5, 4, 7, 6]$ is selected for data shuffling and the four LSBs DS[3:0] are reordered to 1010₂. Then, 10-bit out[9:0] including 8-bit shuffled data DS[7:0] and 2-bit pattern selection signal S[1:0] are written to memory for storage. During the reading process, the generated random noise ranNoise will be injected according to the memory failure map at 0.50V. For the selected word, three bits out of the four LSBs are failed and the control signal Sfailure[3:0] are generated accordingly. After noise injection, the value of four LSBs outNoise[3:0] is 1001₂. Then, based on the control signal out[9:8], i.e., S[1:0], the data re-shuffler is enabled to reverse the bit shift and generate the final output dataOut[7:0], which is 11110110₂. As another example highlighted in red in Figure 13, the 8-bit input data dataIn[7:0]

is 10101001₂ and the pattern selection signal S[1:0] is 10₂, and therefore the permutation pattern $\pi_3-[0, 1, 2, 3, 6, 7, 4, 5]$ is selected for data shuffling. The generated 10-bit out[9:0] including 8-bit shuffled data 10100110₂ and 2-bit pattern selection signal 10₂ are stored in memory. During the read operation, based on the control signal Sfailure[3:0], ranNoise<0>, ranNoise<1>, out(2), and ranNoise<3> are selected as four LSBs of the data (1100₂). After re-shuffling, the final output data is 10100011₂.

6.6 Comparison with Software-based LDP

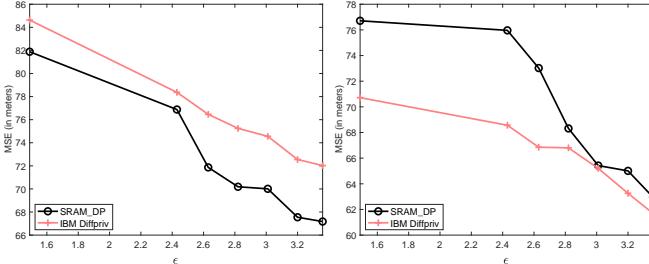
In this section, we compare our SRAM_DP mechanism with the widely accepted software-based RR mechanisms for their performance in utility, estimation error, and system overhead.

Setup: The comparison is drawn by using a real-life dataset — *Foursquare* which is one of the largest location datasets [24]. *Foursquare* contains 227,428 check-ins in New York City, each of which is associated with a user ID, time stamp and GPS coordinate. Specifically, we narrow down the geographical scope to a 2.8Km×2.8Km region within (40.7999N, -73.9700W) and (40.7744N, -73.9445W). A total of 4,759 check-ins are found within this region. We further divide this region into 64×64 areas of interests (AOIs) — each AOI is a 43.75m×43.75m grid. For this selected region, all check-in coordinates (longitude and latitude) are the same to their tenths decimal, so the SRAM_DP and RR mechanisms are only applied to the hundredths, thousandths and ten thousandths, i.e., **.*999-**.*744 for latitude while **.*700-**.*445 for longitude.

To implement RR, we employ the IBM Diffprvlib [25] for which we call the *diffprivlib.mechanisms.Binary* class to add LDP noises. To draw a fair comparison, we consider an utility-optimal RR that the bit randomization is only applied to the same LSBs as SRAM_DP. RR as software-based LDP mechanisms are programmed, compiled and run in the VS Code IDE in a macOS v12.3 computer with a Apple M1 chip of 39-watt standard power and a 32GB memory of 12-watt standard power.

UL and Estimation Accuracy: We adopt the metric MSE in meters to assess the utility losses and estimation errors. EM algorithm is employed for estimating original data. As shown in Figure 11a, SRAM_DP achieves lower utility loss by as much as 6.89%, but the estimation error can be 8.45% higher than the RR mechanism. The reason lies within the calculation and application of cell failure rate vector $\{f_1, \dots, f_8\}$. For simplicity, our SRAM_DP mechanism calculates the average cell failure rate across all 1,000 word-lines and uses it for recovering original data. Yet, each wordline may not exactly follow the failure rate $\{f_1, \dots, f_8\}$, thus leading to high estimation errors. One possible remedy to this problem is to keep track of each wordline's cell failure rate, but it comes with a sacrifice in storage overhead. Recall that our insights from Figure 9a coincide with our observation here, that is the incapability of tracking f_s in fine granularity deteriorates estimation accuracy.

System Overhead: By using the *psutil* tool and excluding the most time-consuming library loading process, we obtained the 8.71% CPU usage and 0.287% memory usage for the 8-bit Binary randomization process, which



(a) Utility Losses.

(b) Estimation Errors.

Fig. 11: Comparison of SRAM_DP and the utility-optimal RR mechanism from IBM Diffprvlib.

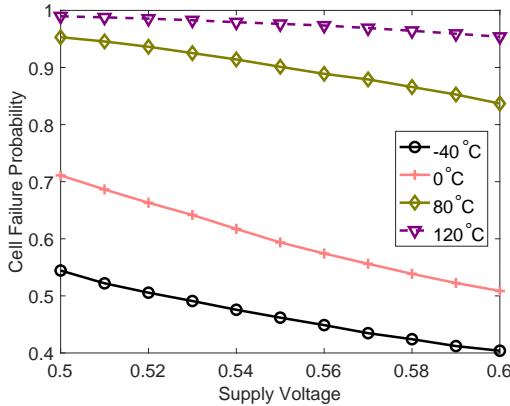


Fig. 12: Cell failure probabilities at various temperatures.

accounts for 7.61×10^{-5} seconds runtime and consumes roughly $39 \times 8.71\% + 12 \times 0.287\% = 3.43$ watt power.

Moreover, we consider a standard memory (45 nm CMOS, 8 memory banks, each bank has 128 words \times 10 bits) without any customization as the baseline for comparison. As shown in Table 2, despite minor increase in chip peripheral overhead due to added circuits for voltage control and bit manipulation, SRAM_DP significantly outperforms IBM Diffprvlib in terms of system responsiveness (measured in latency) and power saving. Note that we neglect the storage overhead for the bit-shuffle pattern as it is data-dependent and can be easily optimized using data piggybacking, peripheral look-up table, and many other methods.

7 DISCUSSION ON SRAM_DP RELIABILITY

Temperature Variations: The reliability of the proposed SRAM_DP in various operating conditions such as high aging and temperature variation is one important design consideration. Figure 12 shows the simulated failure probability of the 6T cells used in our design under temperature fluctuations. The temperature was varied from -40°C to 120°C for the target voltage range (0.5V-0.6V), with 25°C as the reference temperature. As shown, the memory cells become more inclined to failure as the temperature increases (2 times from -40°C to 125°C). This is mainly due to the weakened PMOS transistors in the 6T SRAM cells.

Voltage Precision Control: Supply voltage droop is another factor which will cause variation of memory failure probability, thereby impacting the reliability of SRAM_DP. However, its effect is much smaller than the temperature

variation for today's mainstream voltage regulators and power management Integrated Circuits (PMIC). For example, the voltage droop of a commercial PMIC for IoT devices can be well controlled within $\pm 1.5\%$ [26]. Such a small voltage droop can cause less than $\pm 1\%$ change in the cell failure probability, as shown in Figure 2 and 12. When mapping the cell failure probability variation into the the drift of LDP parameter ϵ , we have the following theorem.

Theorem 7.1. For a SRAM cell whose failure probability changes from f_i to $\alpha_i f_i$ ($0 \leq \alpha_i \leq 1/f_i$) due to the supply voltage droop, the variation of ϵ is bounded by $|\Delta\epsilon| = |\epsilon' - \epsilon| \leq \sum_{i=1}^n |\ln(2\alpha_i - 1)|$.

Proof. According to Theorem 4.1, we can calculate that

$$\begin{aligned} |\Delta\epsilon| &= \left| \sum_{i=1}^n \ln\left(\frac{1 - \frac{1}{2}\alpha_i f_i}{\frac{1}{2}\alpha_i f_i}\right) - \sum_{i=1}^n \ln\left(\frac{1 - \frac{1}{2}f_i}{\frac{1}{2}f_i}\right) \right| \\ &= \left| \sum_{i=1}^n \ln(2 - \alpha_i f_i) - \ln(2 - f_i) - \ln(\alpha_i) \right| \\ &\leq \sum_{i=1}^n |\ln(2 - \alpha_i f_i) - \ln(2 - f_i)| + \sum_{i=1}^n |\ln(\alpha_i)| \end{aligned}$$

For the case $1 \leq \alpha_i \leq 1/f_i$ implying that the supply voltage droop causes the cell failure probability to increase, we can deduce that for every SRAM cell, $\ln(2 - f_i) - \ln(2 - \alpha_i f_i) \leq \ln(2 - \frac{1}{\alpha_i})$ by observing that $\frac{2-f_i}{2-\alpha_i f_i} - (2 - \frac{1}{\alpha_i}) = 2(\alpha_i - 1)(\alpha_i f_i - 1) \leq 0$. Similarly, for the cell probability downscaling case where $0 \leq \alpha_i \leq 1$, we can arrive at the same upper bound. Then, $|\Delta\epsilon|$ can be further written as

$$|\Delta\epsilon| \leq \sum_{i=1}^n |\ln(2 - \frac{1}{\alpha_i})| + \sum_{i=1}^n |\ln(\alpha_i)| = \sum_{i=1}^n |\ln(2\alpha_i - 1)|.$$

Specifically for our developed SRAM_DP that adopts 6T cells and renders cell failures in the 4 LSBs, a $\pm 1\%$ drift in cell probability will result in at most ± 0.08 variations in ϵ for all possible selection of ϵ 's. The privacy customer should thus be cautioned of this worst-case variation when using the SRAM_DP chip.

Reliability Enhancement Solutions: Amid the reliability concerns, different design strategies can be adopted as remedies. First, designing variation-aware cells is one possible solution. The current SRAM_DP adopts 6T cells to achieve the target failure probability for noise injection. To enhance the reliability, the transistor sizing and structure of memory cells can be further custom designed to enable the target mean failure probability while minimizing its variance in the presence of temperature variation or voltage voltage droop [27].

Also, SRAM_DP can utilize the well-studied variation-adaptive solutions [28], which integrate with on-chip sensors and adaptive control circuits to measure specific parameters (e.g., temperature, voltage droop, or aging effect) and then to adjust the supply voltage accordingly. Existing variation-adaptive schemes (such as [29]) have been developed mainly to compensate for the impact of the parameter changes on performance or power consumption. However, they can be easily adapted to our SRAM_DP design, which requires compensation to achieve a stable failure probability (or ϵ).

TABLE 2: Performance comparison between SRAM_DP and utility-optimal RR for $\epsilon = 1.49$ (bold ones are better)

Figure of Merits	Baseline	SRAM_DP	IBM Diffprivlib
Utility Loss (MSE)	0	78.8861 (-6.79%)	84.6308
Estimation Error (MSE)	0	76.7106	70.717 (-7.81%)
Chip Peripheral Overhead (# of transistors)	66,000	67,623 (+2.459%)	+0%
System Latency (ns)	0.84	1.02 (+21.4%)	7.61×10^4 (+9.06 $\times 10^6$ %)
Power Consumption (mW)	3.713	0.5724 (-88.58%)	3.43×10^3 (+922.78%)

8 RELATED WORK

Broadly speaking, most recent works on DP focus on developing domain-specific DP mechanisms for various applications [30], [31], [32]. However, little to no attention has been paid to the realization of DP mechanisms due to the presumption that DP noises can be easily, correctly, and even securely generated and injected in software. This is unfortunately challenged by many recent studies [7], [8], [9] and the emerging lightweight IoT devices that have no rich software modules. Ilvento et al. [33] is among the very few works to investigate the realization of DP mechanisms, and they switch from base-e to base-2 arithmetic — offering higher precision in existing arithmetic libraries — when implementing the popular exponential DP mechanism in software.

In recent years, we observe a growing trend of realizing DP, or more broadly noise-injection, mechanisms by utilizing hardware characteristics. In general, the hardware-based approach is superior than the software-based one because of its true randomness of noise and scalability in implementation (i.e., independence from complex software modules), although the former approach falls short in real-time adaptability after hardware is fabricated. Nonetheless, there are only limited number of works along this theme line. Yang et al. [34] proposed to introduce DP Gaussian noises to deep learning model training by scaling down supply voltage and creating SRAM bit errors. Fu et al. [13] leveraged the inherent Gaussian noises due to the imperfectionness of memristor operations and developed a differentially private deep learning model. Although these works are innovative in their ideas, they are neither implemented in hardware nor rigorous when it comes to conforming to DP notions. For instance, Gaussian mechanism only ensures (ϵ, δ) -DP with $\epsilon < 1$, but some works neglected this constraint.

9 CONCLUSION

In this paper, we designed, implemented, and evaluated a new memory architecture to achieve local differential privacy in hardware rather than in software. By downscaling supply voltages, LDP noise can be introduced to data, especially those in least significant bits, when it is stored in memory. Compared with existing software-based LDP mechanisms, this paper’s analytical, simulated and experimental results all supported the feasibility and superiority of our design in terms of privacy, utility, latency, and system power consumption with moderate compromise in estimation errors and chip overhead.

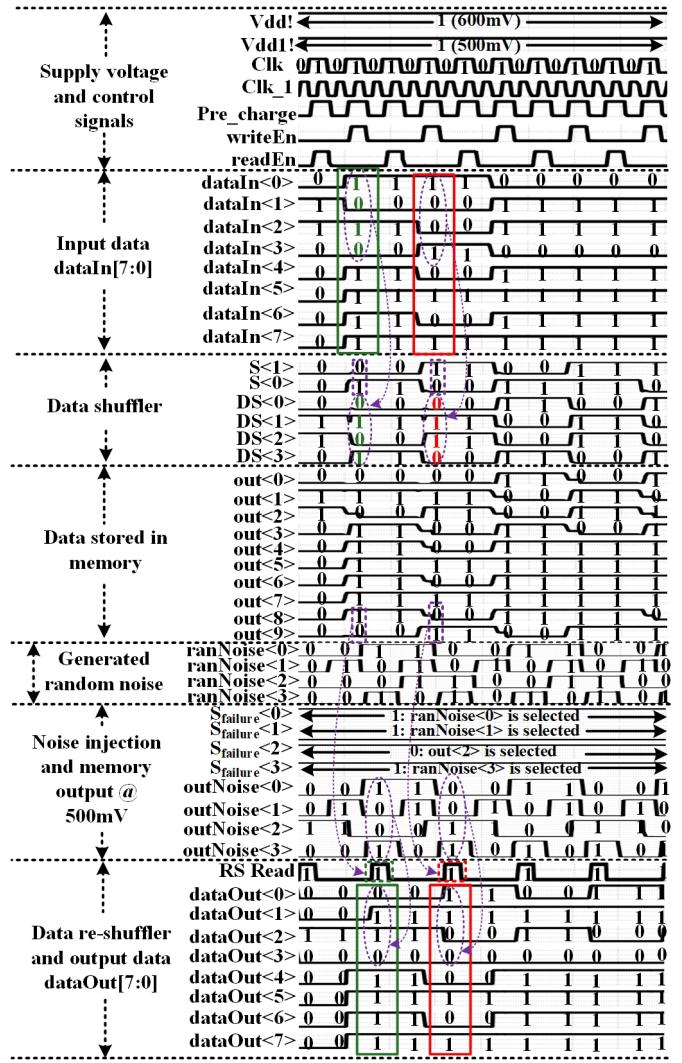


Fig. 13: Timing diagram of the implemented SRAM_DP.

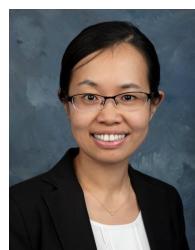
REFERENCES

- [1] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, “Large-scale parallel collaborative filtering for the netflix prize,” in *International conference on algorithmic applications in management*. Springer, 2008, pp. 337–348.
- [2] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.
- [3] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber, “Privacy: Theory meets practice on the map,” in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. IEEE Computer Society, 2008, pp. 277–286.
- [4] F. McSherry and I. Mironov, “Differentially private recommender systems: Building privacy into the netflix prize contenders,” in

- Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 627–636.
- [5] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, “Heavy hitter estimation over set-valued data with local differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 192–203.
- [6] Ú. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. ACM, 2014, pp. 1054–1067.
- [7] I. Mironov, “On significance of the least significant bits for differential privacy,” in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 650–661.
- [8] M. Andryesco, D. Kohlbrenner, K. Mowery, R. Jhala, S. Lerner, and H. Shacham, “On subnormal floating point and abnormal timing,” in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 623–639.
- [9] A. Haeberlen, B. C. Pierce, and A. Narayan, “Differential privacy under fire,” in *USENIX Security Symposium*, 2011.
- [10] J. Liu and N. Gong, “Privacy by memory design: Visions and open problems,” *IEEE Micro*, vol. 44, no. 1, pp. 49–58, 2024.
- [11] S. Y. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, “Compute-in-memory chips for deep learning: Recent trends and prospects,” *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.
- [12] S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
- [13] J. Fu, Z. Liao, J. Liu, S. C. Smith, and J. Wang, “Memristor-based variation-enabled differentially private learning systems for edge computing in iot,” *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9672–9682, 2020.
- [14] Y. Xu, H. Das, Y. Gong, and N. Gong, “On mathematical models of optimal video memory design,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 1, pp. 256–266, 2019.
- [15] N. Gong, S. Jiang, J. Wang, B. Aravamudhan, K. Sekar, and R. Sridhar, “Hybrid-cell register files design for improving nbit reliability,” *Microelectronics Reliability*, vol. 52, no. 9–10, pp. 1865–1869, 2012.
- [16] J. Croon, S. Decoutere, W. Sansen, and H. Maes, “Physical modeling and prediction of the matching properties of mosfets,” in *Proceedings of the 30th European Solid-State Circuits Conference (IEEE Cat. No. 04EX850)*. IEEE, 2004, pp. 193–196.
- [17] H. Das, A. A. Haidous, S. C. Smith, and N. Gong, “Flexible low-cost power-efficient video memory with ecc-adaptation,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 10, pp. 1693–1706, 2021.
- [18] M. Gottscho, A. BanaiyanMofrad, N. D. Dutt, A. Nicolau, and P. Gupta, “Dpcsc: Dynamic power/capacity scaling for sram caches in the nanoscale era,” *ACM Transactions on Architecture and Code Optimization*, vol. 12, no. 3, pp. 1–27, 2015.
- [19] J. Edstrom, D. Chen, Y. Gong, J. Wang, and N. Gong, “Data-pattern enabled self-recovery low-power storage system for big video data,” *IEEE Transactions on Big Data*, vol. 5, no. 1, pp. 95–105, 2017.
- [20] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, “Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86,” in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 679–697.
- [21] D. R. Dipta and B. Gulmezoglu, “Df-sca: dynamic frequency side channel attacks are practical,” in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 841–853.
- [22] T. Murakami and Y. Kawamoto, “Utility-optimized local differential privacy mechanisms for distribution estimation,” in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1877–1894.
- [23] T. Wang, J. Blocki, N. Li, and S. Jha, “Locally differentially private protocols for frequency estimation,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 729–745.
- [24] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, “Modeling user activity preference by leveraging user spatial temporal characteristics in lsns,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2015.
- [25] N. Holohan, S. Braghin, P. Mac Aonghusa, and K. Levacher, “Diffprivlib: the ibm differential privacy library,” *arXiv preprint arXiv:1907.02444*, 2019.
- [26] Digi-Key, “Use advanced ldos to meet iot wireless sensor power supply design challenges,” 2019. [Online]. Available: <https://www.digikey.com/en/articles/use-advanced-ldos-iot-wireless-sensor-power-supply-design>
- [27] H. Kwon, D. Kim, Y. H. Kim, and S. Kang, “Variation-aware sram cell optimization using deep neural network-based sensitivity analysis,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 4, pp. 1567–1577, 2021.
- [28] K. A. Bowman, “Adaptive and resilient circuits: A tutorial on improving processor performance, energy efficiency, and yield via dynamic variation,” *IEEE Solid-State Circuits Magazine*, vol. 10, no. 3, pp. 16–25, 2018.
- [29] J. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vangal, S. Narendra, Y. Hoskote, H. Wilson, C. Lam *et al.*, “Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging,” in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. IEEE, 2007, pp. 292–604.
- [30] J. Liu, C. Zhang, B. Lorenzo, and Y. Fang, “Dpavatar: A real-time location protection framework for incumbent users in cognitive radio networks,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 552–565, 2019.
- [31] X. Pei, X. Deng, S. Tian, J. Liu, and K. Xue, “Privacy-enhanced graph neural network for decentralized local graphs,” *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1614–1629, 2023.
- [32] N. Gai, K. Xue, B. Zhu, J. Yang, J. Liu, and D. He, “An efficient data aggregation scheme with local differential privacy in smart grid,” *Digital Communications and Networks*, vol. 8, no. 3, pp. 333–342, 2022.
- [33] C. Ilvento, “Implementing the exponential mechanism with base-2 differential privacy,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 717–742.
- [34] L. Yang and B. Murmann, “Approximate sram for energy-efficient, privacy-preserving convolutional neural networks,” in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 689–694.
- [35] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth, “Differential privacy: An economic method for choosing epsilon,” in *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 398–410.
- [36] L. T. Clark, S. B. Medapuram, and D. K. Kadiyala, “Sram circuits for true random number generation using intrinsic bit instability,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 2027–2037, 2018.



Jianqing Liu (M’18) received the Ph.D. degree from University of Florida in 2018. He is currently an assistant professor with the Department of Computer Science at NC State University. His research interest is wireless networking, security, and privacy. He received the U.S. National Science Foundation Career Award in 2021. He is also the recipient of several best paper awards including the 2018 Best Journal Paper Award from IEEE TCGCC.



Na Gong (M’13) received the Ph.D. degree in computer science and engineering from the State University of New York, Buffalo, in 2013. Currently, Dr. Gong is a professor with the Department of Electrical and Computer Engineering at University of South Alabama. Her research interests include power-efficient computing circuits and systems, memory optimization, AI hardware, and neuromorphic computing. She is the recipient of the best paper nomination from ISVLSI’19, best paper award from EIT’16, best paper nominations from ISQED’16 and ISLPED’16.



Hritom Das (M'20) received the Ph.D. degree in Electrical and Computer Engineering from North Dakota State University in 2020. He was a visiting Assistant Professor with the Department of Electrical and Computer Engineering at the University of South Alabama. Currently, he is a Post-Doctoral Research Associate with the Department of Electrical Engineering and Computer Science at The University of Tennessee. His research interests include neuromorphic computing and data privacy for edge devices.

Appendix A

Recall that ΔA is defined as $O - X$, which measures how far apart the SRAM_DP perturbed output O is from the original input X . The closed form PMF of ΔA is unfortunately intractable because calculating $P(\Delta A)$ requires enumerating all combinations of $\{\Delta a_{n-1}, \dots, \Delta a_0\}$ in an equality constraint (i.e., ΔA equals to a specific value) which is similar to the bounded knapsack problem that is NP-complete. This poses a great challenge to analyze the l_1 utility loss of SRAM_DP in Theorem 4.2. In light of it, this section will not seek to derive the closed form PMF of ΔA , but rather to prove that the PMF of ΔA has a zero mean and is symmetric w.r.t. $\Delta A = 0$. This key property will facilitate us to arrive at the conclusion in Theorem 4.2.

First, for notation simplicity, we introduce a new variable $S_n(a)$ to denote the probability of $\Delta A = a$ for any n -bit data ($n \geq 1$), i.e., $S_n(a) = P(\Delta A = a)$. Note that $S_n(a)$ has two key properties. (i) a is defined as $a \in \mathbb{Z}$, but $S_n(a) = 0$ for $a \notin \{-2^{n-1} + 1, \dots, 2^n - 1\}$; (ii) $S_n(a)$ has a following boundary condition concurring with Eq.(2)

$$S_1(a) = \begin{cases} \frac{1}{4}f_1 & a = -1 \\ 1 - \frac{1}{2}f_1 & a = 0 \\ \frac{1}{4}f_1 & a = 1 \end{cases}$$

Next, we develop a recursive approach to prove Lemma 4.3. First of all, $S_n(a)$ has the following the recursion relation

$$\begin{aligned} S_n(a) = & \frac{1}{4}f_n S_{n-1}(a - 2^{n-1}) + \left(1 - \frac{1}{2}f_n\right) S_{n-1}(a) \\ & + \frac{1}{4}f_n S_{n-1}(a + 2^{n-1}). \end{aligned} \quad (6)$$

The intuition behind Eq.(6) is that a n -bit data can have $\Delta A = a$ when all bits except the n^{th} bit are (i) equal to $(a - 2^{n-1})$ in decimal while the n^{th} bit fails (i.e., $\Delta a_n = 1$); or (ii) equal to a in decimal while the n^{th} bit is intact (i.e., $\Delta a_n = 0$); or (3) equal to $(a + 2^{n-1})$ in decimal while the n^{th} bit fails (i.e., $\Delta a_n = -1$).

Then, we follow the principle of *proof by induction* to prove Lemma 4.2. The standard proof procedures are as follows. (i) From the the boundary condition, we know that Lemma 4.2 stands true for the base case $S_1(a)$. (ii) Assume Lemma 4.2 also holds for $S_k(a)$ where $2 \leq k < n$. This implies that $S_k(a) = S_k(-a)$, $S_k(a - 2^k) = S_k(2^k - a)$, and $S_k(-a - 2^k) = S_k(a + 2^k)$. Then, according to Eq.(6), we can derive

$$\begin{aligned} S_{k+1}(-a) = & \frac{1}{4}f_{k+1}S_k(-a - 2^k) + \left(1 - \frac{1}{2}f_{k+1}\right) S_k(-a) \\ & + \frac{1}{4}f_{k+1}S_k(-a + 2^k) \\ = & S_{k+1}(a). \end{aligned}$$

TABLE 3: Cell Failure Rates and Corresponding ϵ 's

Supply Voltage (V)	Failure Rate (%)	ϵ
0.50	81.57	1.49
0.55	70.57	2.43
0.56	68.31	2.63
0.57	66.15	2.82
0.58	64.09	3.01
0.59	62.03	3.20
0.60	60.26	3.36

Therefore, by the principle of *proof by induction*, we can assert that $S_n(a)$ is symmetric w.r.t. to $a = 0$, $\forall n \geq 1$ (naturally, it has a zero mean). In other words, the PMF of ΔA also obeys the same property, which concludes the correctness of Lemma 4.2.

Appendix B

B.1. Voltage Control

SRAM_DP adopts four 6T cells to store LSB1-4 in order to inject noise. Its failure rate under different supply voltages is shown in Table 3. The corresponding ϵ at any specific supply voltage can be easily calculated according to Theorem 4.1. In differential privacy, ϵ is a rather abstract quantity and recent research articles can select ϵ from as little as 0.01 to as much as 7 [35]. Our proposed memory can also achieve a wide-range adaptation of ϵ under broad tuning of supply voltages. Yet, the chip stability can be hardly controlled at extreme low-voltage operations (i.e., when cell failure rate exceeds 90%). Therefore, we only showcase the results in Table 3.

Based on the failure rates listed in Table 3, the following four permutation patterns from MSB to LSB are considered for **Bit Shift** in Step 1: $\pi_1-[0, 1, 2, 3, 4, 5, 6, 7]$; $\pi_2-[0, 1, 2, 3, 5, 4, 7, 6]$; $\pi_3-[0, 1, 2, 3, 6, 7, 4, 5]$; and $\pi_4-[0, 1, 2, 3, 7, 6, 5, 4]$. Accordingly, during the data storage process, two bits are sufficient for permutation pattern selection and they are stored in memory as shown in Figure 4.

B.2. Custom Cells Design

As discussed earlier, to implement the proposed SRAM_DP, we need to render cell failures to the four LSBs (i.e., LSB1-4), while maintaining four MSBs (i.e., LSB5-8) intact. To enable it, we propose a hybrid-cell design scheme, as shown in Figure 14. Reliable 8T cell are custom designed to enable zero failure rate and they are used to store two pattern selection bits and four MSBs of the input data (i.e., LSB5-8)) to minimize the utility loss. Area-efficient 6T cells are custom designed to store the four LSBs (i.e., LSB1-4). To achieve zero area overhead of 6T-8T integration, two separate word lines (RWL and WWL) are used for 6T cells [15].

B.3. Bit Shuffler and Re-Shuffler Design

A MUX-based bit Shuffler is designed to enable **Bit Shift** in Step 1, as shown in Figure 15. The 8-bit input data called dataIn will be shuffled according to a 2-bit pattern selection signal $S[1:0]$. Specifically, four 4-to-1 MUXs are adopted and each MUX is used to shift one LSB. As $S[1:0]$ is "00", "01", "10", or "11", π_1 to π_4 will be selected accordingly.

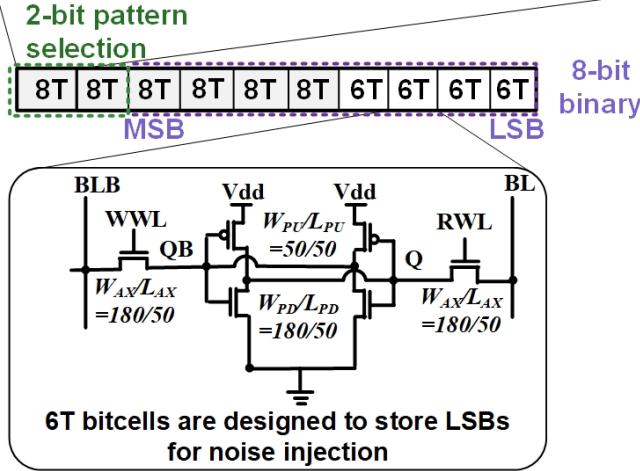
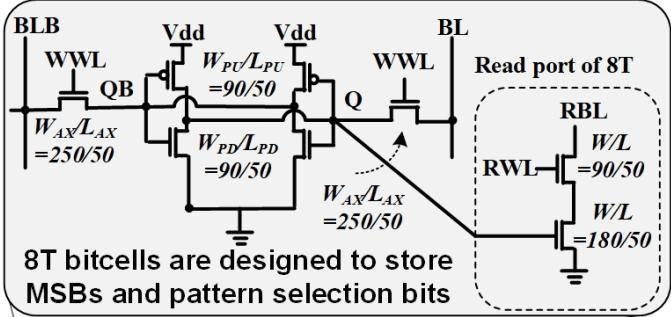


Fig. 14: Cell design for the proposed memory.

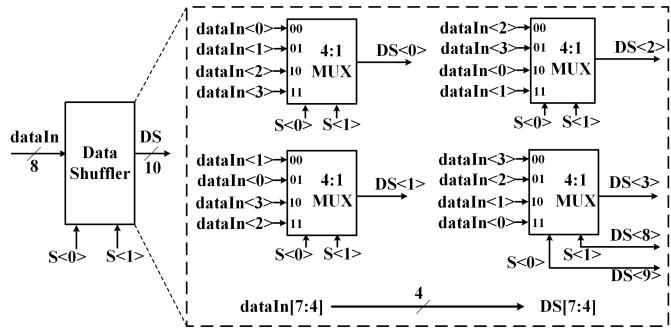


Fig. 15: Bit shuffler.

For example, if S is "11", $\text{dataIn}(3)$ is selected as $\text{DS}[0]$, $\text{dataIn}(2)$ is selected as $\text{DS}[1]$, $\text{dataIn}(1)$ is selected as $\text{DS}[2]$, and $\text{dataIn}(0)$ is selected as $\text{DS}[3]$. The four MSBs of the output DS will remain the same as dataIn . As a result, $\pi_4-[0, 1, 2, 3, 7, 6, 5, 4]$ is selected for shuffling. During this process, a pattern selection signal S is generated by a 2-bit random number generator, which will be stored in the same memory word (row) together with the shuffled data for the **Reverse Bit Shift** in Step 4. Accordingly, the output of the bit shuffler DS has 10 bits, including 8-bit shuffled data and 2-bit pattern selection signal S , as shown in Figure 15.

To avoid performance penalty, the Bit Shuffler is operated in parallel with the word-line decoder of the memory. Due to its small size, the access delay of Bit Shuffler is quite marginal and can be overlapped with the word-line decoding. Therefore, the Bit Shuffler will not create new critical paths in the memory. At the same time, the Bit Shuffler also introduces a small amount of power overhead.

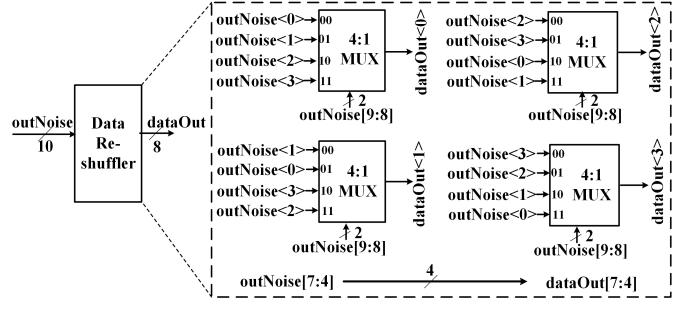


Fig. 16: Bit reshuffler.

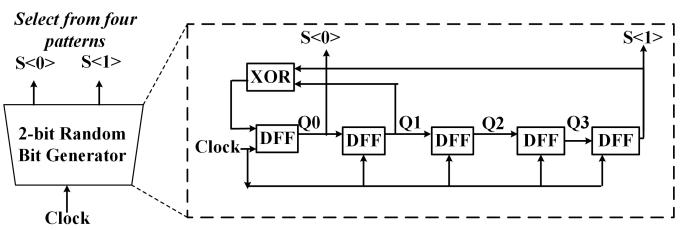


Fig. 17: 2-bit random bit generator for pattern selection.

However, the power savings achieved by the proposed low-voltage memory can offset this overhead.

A MUX-based Bit Re-Shuffler is also designed to implement the **Reverse Bit Shift** in Step 4, as shown in Figure 16. The loaded two pattern selection bits (outNoise(9) and outNoise(8), i.e., $S(1)$ and $S(0)$) are used to revert the memory output data outNoise[7:0] according to the original bit orders. It is worthwhile to emphasize that The proposed MUX-based Shuffler and Re-Shuffler schemes is only dependent on the permutation vector set, and it can easily adapt to different permutation patterns, with minimal implementation cost.

B.4. Random Bit Generator

This module is needed in the **Step 1: Bit Shift** to randomly select one permutation pattern, and in the **Step 3: Noise Injection** to add random noise to the failed positions of the memory. In this paper, we adopt a light-weight Random Bit Generator design using Linear Feedback Shift Registers (LFSR), as shown in Figure 17. To improve the randomness, a true random generator, such as [36], can be used, which, however, will come with a significant implementation cost.