Synthesizing Probabilistic Saturating Counters with Differentially Private Formal Guarantees

Zhiming Chi[1,2,3], Lutan Zhao[3,4], Depeng Liu[1,2], Yong Li[1,2], Pengfei Yang[5], Bow-Yaw Wang[6], Rui Hou[3,4], Cheng-Chao Huang[7], Andrea Turrini[1,2], Lijun Zhang[1,2,3(✉)], and Naijun Zhan[8]

[1] Key Laboratory of System Software (Chinese Academy of Sciences), Beijing, China
[2] Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[5] Institute of AI for Industries. Chinese Academy of Sciences, Nanjing, China
[6] Institute of Information Science, Academia Sinica, Taipei, Taiwan
[7] Nanjing Institute of Software Technology, Chinese Academy of Sciences, Nanjing, China
[8] School of Computer Science & MOE Key Laboratory of High Confidence Software Technologies, Peking University, Beijing, China
zhanglj@ios.ac.cn

Abstract. Branch predictors improve instruction-level parallelism in modern processors and are commonly modeled using saturating counters. However, classical saturating counters are deterministic and thus vulnerable to side-channel attacks: an attacker can manipulate the counter state and infer the branch direction of a victim process. Probabilistic saturating counters (PSCs) have been proposed to mitigate this leakage by randomizing counter updates, but existing evaluations are mainly empirical. In this paper, we give a formal analysis based on differential privacy (DP): we model PSCs and the corresponding Prime+Probe attack strategies as probabilistic Moore machines, derive optimal attack strategies, and quantify the attacker's distinguishing power through DP. Our DP guarantee applies to the PSC primitive under the Prime+Probe observation model; end-to-end security for a full branch predictor under repeated or adaptive attacks is an important direction for future work. We then synthesize parameters for an enhanced PSC that satisfies a target pure DP guarantee. To evaluate utility, we derive the stationary misprediction rate and validate the theoretical predictions on benchmark programs. Compared to deterministic and existing probabilistic saturating counters, the synthesized PSCs provide formal security guarantees while preserving competitive prediction performance.

1 Introduction

Branch prediction is a fundamental technique for enhancing instruction-level parallelism in modern high-performance processors [29, 52, 55]. However, the

shared nature of the predictor's resources, whether in single-core or Simultaneous Multi-Threading processors, has exposed significant security vulnerabilities [24, 32, 38]. The root cause lies in the deterministic update strategy of the Saturating Counter (SC)–the fundamental building block of predictors ranging from GShare to TAGE [44, 51]. This determinism allows attackers to manipulate the counter state and exploit side-channels, such as the Prime+Probe attack [24, 42], to infer fine-grained execution traces and thus extract information about the victim's private data. While Probabilistic Saturating Counters (PSCs) [42] have been proposed to mitigate these attacks by introducing randomized transitions, existing security analyses rely on incomplete simulations, failing to rigorously balance the trade-off between security and performance.

In this paper, we address this gap by applying formal verification to the design and analysis of PSCs, specifically targeting the Prime+Probe attack. In this attack scenario, an adversary first primes the target saturating counter to a known initial state and subsequently probes the counter to detect the cut-off point–the first instance of a correct prediction during the victim's execution. By observing this cut-off point, the adversary can infer the victim's branch direction; repeated attacks allow the attacker to infer the private data used in the branch decision. We model the SC and this adversarial interaction using probabilistic Moore machines [49] and quantify security through Differential Privacy (DP) [19]. DP is a natural fit for this setting because it directly bounds the information leakage from observable outputs and provides a compositional worst-case guarantee [18, 33, 61]. By deriving an optimal attack strategy, we show that current PSC designs fail to provide sufficient DP guarantees under worst-case conditions, rendering the attack always successful.

To overcome these limitations, we propose a novel PSC design framework with additional probabilistic transitions. Given a target privacy budget $\varepsilon > 0$ (smaller ε means stronger privacy), our *parameter synthesis problem* is: find the defense parameter $p \in (0, 1)$ and update probability $m \in (0, 1]$ such that the enhanced PSC satisfies $(\varepsilon, 0)$-DP and the stationary misprediction rate is minimized. We solve this problem analytically (Prop. 1), yielding the optimal $p^* = \frac{1}{1+e^{\varepsilon}}$. Moreover, we assess performance impact by deriving an analytical solution for the misprediction rate based on the PSC's stationary distribution. While we focus on the ubiquitous 2-bit counters, our methodology extends to arbitrary k-bit designs. Our main contributions are: (i) a formal framework for modeling saturating counters and the Prime+Probe attack using probabilistic Moore machines, which reveals inherent privacy vulnerabilities in existing PSC designs through the derivation of the optimal attack strategy; (ii) a robust PSC design suitable for parameter synthesis, enabling the enforcement of rigorous differential privacy guarantees during the design phase; (iii) the stationary misprediction rate as a performance metric and its analytical derivation; and (iv) a formal privacy-utility trade-off analysis, including optimal parameter selection and a comparison with the randomized response mechanism that demonstrates the advantage of our targeted randomization. Empirical results confirm that our enhanced PSCs achieve a superior balance of provable security and high

performance compared to traditional and existing probabilistic counterparts. Compared to Zhao et al. [42], who introduced PSCs with empirical security evaluation, we provide the *first formal DP analysis* with provable guarantees, identify a fundamental vulnerability in the original PSC (the c=1 observation), propose an enhanced PSC achieving pure ε-DP for any target budget, and derive the stationary misprediction rate in closed form.

Related work. Side-channel attacks against branch predictors and shared microarchitectural resources (e.g., SGX, BPUs, and caches) have been extensively studied [6, 9, 10, 17, 24, 28, 30, 31, 34, 36, 39, 43, 50, 63]. Most relevant to our work, Wang et al. [57] utilized symbolic execution to identify attack patterns on predictors. However, their evaluation of probabilistic saturating counters remained largely empirical. In contrast, we provide a theoretical framework that formally explains the effectiveness of attacks and explicitly models how attack outputs lead to branch inference, offering a more rigorous security assessment.

Differential privacy [19] has been widely adopted across various fields, including economics, healthcare, and deep learning [1, 7, 11, 12, 14, 16, 21, 26, 45, 46, 48, 54, 65]. Importantly, DP guarantees imply rigorous bounds on an adversary's ability to make inferences: Wasserman and Zhou [61] established a statistical framework connecting DP with hypothesis testing, and Kairouz et al. [33] studied the composition of DP mechanisms and the resulting attack bounds. From a formal verification perspective, DP can be specified and verified using probabilistic couplings in programming languages [58, 59, 64], automated theorem proving [2–4], and probabilistic model checking [40, 41]. Our work extends these formal techniques to the hardware security domain, leveraging the connection between DP and inference bounds to provide meaningful security guarantees for PSCs.

2 Preliminaries

In this section, we introduce key concepts that will be used later and review side-channel attacks on deterministic saturating counters. Throughout the paper we fix $\Sigma = \Gamma = \{\mathrm{T}, \mathrm{NT}\}$ as alphabets and we just omit them from the definitions.

2.1 Saturating counters and Moore machines

A Saturating Counter (SC) records the branch history and predicts the next branch resolution. The general structure of a k-bit SC is shown in Fig. 1 and has 2^k states: one Strongly and $2^{k-1}-1$ Weakly Not-taken blue-dotted states and, symmetrically, $2^{k-1}-1$ Weakly and one Strongly Taken red-striped states. The predictor operates with two input symbols: T and NT, indicating whether a branch is Taken or Not-Taken during a program execution, respectively. A *misprediction* occurs when the prediction (e.g., T in states ST or WT) conflicts with the actual execution outcome (e.g., NT). The SC requires $n = 2^{k-1}$ consecutive mispredictions to alter for sure its prediction direction due to the hysteresis

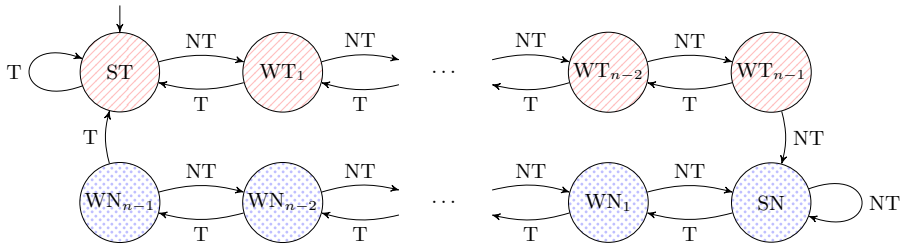


Fig. 1. A k-bit saturating counter, with a total of $2n = 2^k$ states. Red-striped states output T, while blue-dotted states output NT.

provided by the strong/weak (S/W) bit, which allows the SC to adjust its state based on the actual execution direction. An SC can be formally modeled as a Moore machine.

Definition 1. *A* Moore machine *is as a tuple $M = (S, \bar{s}, T, O)$, where S is a finite set of states, $\bar{s} \in S$ is the initial state, $T\colon S \times \Sigma \to S$ is the transition function that maps the current state and the input to the next state, and $O\colon S \to \Gamma$ is the output function that maps a state to an output symbol.*

The Moore machine corresponding to the k-bit saturating counter shown in Fig. 1 has the depicted nodes as states, the initial state is ST, as indicated by the sourceless incoming arrow, the transition function is given by the edges between nodes, and the output function assigns T to the red-striped states and NT to the blue-dotted states. A 2-bit saturating counter has only four states: ST, WT, WN, and SN, i.e., the states on the left and right of the counter shown in Fig. 1.

In addition to ordinary Moore machines, we will also use *probabilistic Moore machines* to model branch predictors when probabilities are introduced.

Definition 2. *A* probabilistic Moore machine *is a tuple $\mathcal{D} = (S, \bar{\iota}, \mathbf{P}, O)$, where S and O are as in Def. 1, $\bar{\iota}\colon S \to [0,1]$ is the initial distribution such that $\sum_{s \in S} \bar{\iota}(s) = 1$, and $\mathbf{P}\colon S \times \Sigma \times S \to [0,1]$ is the probability transition function such that $\sum_{t \in S} \mathbf{P}(s, \sigma, t) = 1$ for each state $s \in S$ and $\sigma \in \Sigma$.*

Let $P_{s,\sigma}$ be the probability of receiving the input symbol $\sigma \in \Sigma$ when being in state $s \in S$. These probabilities are specified externally by the workload or by the phase of the attack, and are not determined by Def. 2. Then we can represent the transition probability function $\mathbf{P}$ as the matrix $\mathrm{M} = (\mathbf{P}(s,t))_{S \times S}$ where each entry $\mathrm{M}(s,t)$ is defined as $\mathrm{M}(s,t) = \sum_{\sigma \in \Sigma} \mathbf{P}(s, \sigma, t) \cdot P_{s,\sigma}$. A probability distribution μ on S is stationary if $\mu \mathrm{M} = \mu$.

2.2 Prime+Probe attacks on branch predictors

Branch predictor side-channel attacks are based on the ability to prime and probe them. The predictor encodes the execution history of branch instructions, potentially including sensitive branches; an attacker cannot observe the internal

Algorithm 1 Algorithm for finding the cut off point

Input: Number of branch executions n; the victim thread's direction $v \in \{\mathrm{T}, \mathrm{NT}\}$
Output: The number of probes before reaching the cut-off point

```
1: function FindCutOffPoint(n, v)
2:     execute the attacker branch with direction T for n times
3:     execute the victim branch with direction v
4:     c ← 0
5:     while c ≤ n do
6:         execute the attacker branch with direction NT
7:         if prediction is correct then return c
8:         c ← c + 1
9:     return c
```

state of the predictor, but can observe its output. By providing appropriate inputs, the attacker can guide the predictor to a known state and then infer the internal state after the victim has taken its branch direction. In [24, 42], this is achieved by identifying the *cut-off point* of the saturating counter: the first instance of a correct prediction following multiple mispredictions. By locating this cut-off point, the attacker can deduce whether the victim thread's execution direction was T or NT. The cut-off point can be found by means of Alg. 1 that we can divide into three phases:

- Phase 1 (line 2): the attacker first primes the target saturating counter to a predefined state, by repeatedly issuing T to take the branch at least n times; if $n \geq 2^{k-1}$, then state ST is for sure reached.
- Phase 2 (line 3): the victim thread executes its branch with $v = \mathrm{T}$ or $v = \mathrm{NT}$; the counter either remains in ST or moves to WT_1, respectively.
- Phase 3 (lines 4–8): the attacker probes the predictor by counting how many NT branch executions are needed to reach SN, the first state where the prediction agrees with the input NT.

The core of distinguishing the victim's behavior lies in observing differences in prediction outcomes during the probing phase. A critical observation is the *cut-off point* in Phase 3, which corresponds to the SN state of the predictor. Before this point, only mispredictions are observed; afterwards, all predictions are correct. By counting the number of mispredictions c before reaching the cut-off point, the attacker can infer the direction of the victim's branch, by comparing c with the known value 2^{k-1}: if $c = 2^{k-1}$, then the victim's direction was $v = \mathrm{T}$; if $c < 2^{k-1}$, then the victim's direction was $v = \mathrm{NT}$. Through repeated execution of this algorithm, the attacker can continuously decode the victim's sensitive branch executions, posing significant security risks.

2.3 Differential privacy

To measure how much a branch predictor is resistant to attacks we use differential privacy [18, 20, 47], a framework designed for the protection of individual

data during publication and analysis. Given a set of *data entries* $\mathcal{X}$, a *dataset* of size n is an element of $\mathcal{X}^n$. Two datasets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}^n$ are *neighbors*, denoted by $\Delta(\mathcal{D}, \mathcal{D}') \leq 1$, if they differ in at most one data entry. To ensure individual privacy, random noise is introduced to the dataset so that true values cannot be disclosed or inferred. A *data publishing mechanism* $\mathcal{M}$ is a randomized algorithm that takes a dataset $\mathcal{D}$ as input; let range($\mathcal{M}$) denote the set of all possible outputs of $\mathcal{M}$ and assume we are given a probability space over range($\mathcal{M}$). Differential privacy is satisfied by $\mathcal{M}$ if its output distributions over all neighboring datasets are close enough.

Definition 3. *Given $\varepsilon, \delta \in \mathbb{R}_{\geq 0}$, a mechanism $\mathcal{M}$ is called* (ε, δ)-differentially private *if for all measurable sets $O \subseteq$ range($\mathcal{M}$) and all datasets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}^n$ such that $\Delta(\mathcal{D}, \mathcal{D}') \leq 1$, it holds that* $\Pr(\mathcal{M}(\mathcal{D}) \in O) \leq e^{\varepsilon} \cdot \Pr(\mathcal{M}(\mathcal{D}') \in O) + \delta$.

The parameters ε and δ control the probability differences between the outputs in any two neighboring input datasets regarding an output set O. Evidently, smaller values of these parameters result in smaller discrepancies in the probability distributions, thus offering enhanced privacy protections. The special case $\delta = 0$ is known as *pure* ε-differential privacy, a strictly stronger guarantee than (ε, δ)-DP with $\delta > 0$ [18]. In subsequent sections, we adapt this definition to safeguard sensitive information of the victim thread within the context of branch predictors. Concretely, the "dataset" is the victim's single branch direction ($v \in \{\mathrm{T}, \mathrm{NT}\}$); two "neighboring datasets" differ only in this one bit; the "mechanism" is the Prime+Probe attack output (the cut-off point c); and DP guarantees that the distribution of c is nearly the same regardless of v. This is a *local* security guarantee for the PSC primitive under the Prime+Probe observation model. We note that alternative quantitative security notions, such as quantitative information flow [53], could also be applied; DP is particularly well-suited here because (1) it provides a compositional worst-case guarantee for arbitrary post-processing, (2) it directly bounds the adversary's inference advantage as $\frac{e^{\varepsilon}}{1+e^{\varepsilon}}$, and (3) the resulting constraints on parameters m and p admit efficient synthesis.

3 Security Analysis of Existing PSCs

In this section, we first review the existing PSCs presented in [42]. We then provide two contributions related to these PSCs. First, we employ probabilistic Moore machines to formally model both the PSC mechanism and its associated attack surface, proposing a rigorous security specification grounded in differential privacy. Second, by deriving the optimal attack strategy and quantifying the probability of a successful attack, we conduct a comprehensive theoretical analysis that reveals a critical security vulnerability in the existing PSC design.

3.1 Differential privacy analysis of PSCs

The attack described in Alg. 1 exploits the deterministic nature of classical saturating counters, where the next state is uniquely determined by the current

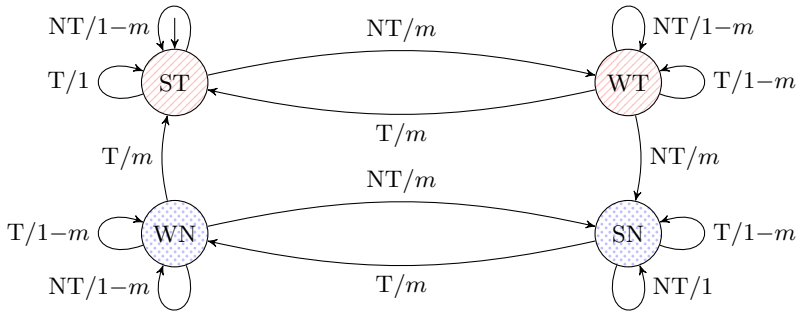


Fig. 2. A 2-bit PSC, shown with initial state ST after the attacker's priming phase

state and input. To defend against such attacks, Zhao et al. [42] proposed to use probabilistic saturating counters. This approach introduces a probabilistic threshold into the counter mechanism: upon a state transition, a random number is generated and compared to the threshold; if the number is below the threshold, the transition proceeds. Otherwise, the counter remains in its current state. Taking the 2-bit PSC illustrated in Fig. 2 as an example, given the threshold $m \in [0, 1]$, for each state s and input letter $b \in \{\mathrm{T}, \mathrm{NT}\}$, with probability m the PSC transitions from s to the intended target state of b; with the remaining probability $1 - m$ the PSC ignores the input b and remains in s.

By adding the threshold m, the attacker cannot precisely predict the state of the counter due to the probabilistic nature of the steps leading to the cut-off point. The experimental results given in [42] show the success rate of the attack under various values of m, yet a theoretical analysis is essential to understand the optimal attack strategies and the misprediction rates, crucial to performance evaluation. We apply differential privacy to establish theoretical guarantees for defense mechanisms in PSCs. Additionally, we analyze the optimal attack strategy for Alg. 1, identifying scenarios where the attacker can successfully infer the victim's execution. To mitigate this, we propose a new PSC model and calculate its misprediction rate; we begin by defining differential privacy for PSCs.

In the context of branch prediction side-channel attacks, the sensitive data corresponds to the direction of a single victim's branch. Therefore, we refine the standard notion of neighboring datasets to reflect this specific threat model: two datasets (i.e., execution histories) are considered neighbors if they differ only for the outcome of the single victim's branch. As data publishing mechanism, we consider the attacker based on FindCutOffPoint but we reverse the meaning of its outcome: the mechanism provides privacy if the attacker fails in computing the right cut-off point with large enough probability. This results in the following definition of (ε, δ)-differential privacy for PSCs.

Definition 4. *Given $n \in \mathbb{N}$, let out be the random output returned by Alg. 1 on input n and victim direction v. A PSC satisfies (ε, δ)-differential privacy if, for*

every output c of Alg. 1, the probability of observing c satisfies

$$\Pr[out = c|v = \mathrm{T}] \leq \mathrm{e}^{\varepsilon} \Pr[out = c|v = \mathrm{NT}] + \delta,$$
$$\Pr[out = c|v = \mathrm{NT}] \leq \mathrm{e}^{\varepsilon} \Pr[out = c|v = \mathrm{T}] + \delta.$$

In concrete scenario attacks, we assume the target branch has been executed sufficiently many times with direction T that the probability of the counter being in a state other than ST is in practice negligible (cf. the experiments in Sect. 5).

This definition provides a natural security metric: the smaller the achievable ε, the harder it is for the attacker to distinguish between the two branch directions. When $\varepsilon = 0$ and $\delta = 0$, the output distributions are identical and the attacker gains no information—corresponding to perfect privacy at the cost of a 50% misprediction rate. When ε is large, the output distributions diverge significantly, allowing the attacker to infer the branch direction with high probability. Our goal in the remainder of this section is to determine the achievable ε for the existing PSC design and, in Sect. 4, to synthesize an enhanced PSC that satisfies a given ε-DP requirement.

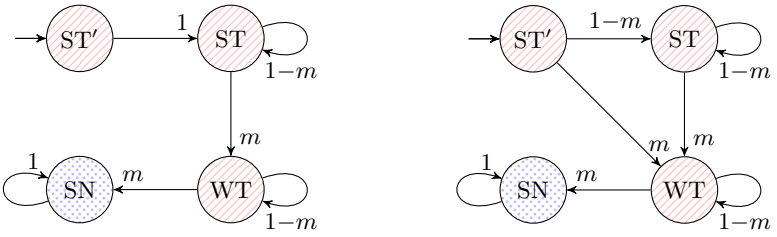
3.2 Modeling attacks to PSCs with probabilistic Moore machines

The concept of differential privacy on PSCs guides us in analyzing the probability of obtaining each output under different executions of the victim's branch. Specifically, given an observation c, our goal is to determine the attacker's strategy to infer whether the branch was taken or not. In [42], experiments fix a parameter m and simulate the victim's behavior multiple times. The attacker records the frequency of different outputs when the branch is taken versus not-taken, then guesses the victim's action based on the most frequent outcome. However, the limitation of this method is that it requires a large number of tests and does not provide general conclusions for arbitrary values of m.

To address this limitation, we model the attack algorithm using parametric Moore machines, as they allow us to derive the optimal attack strategy. This approach allows for more generalized insight regardless of the specific value of m. During the three phases of Alg. 1, the directions of the branches executed are different, as well as their probabilities in the different states s:

- In Phase 1, the branch is always taken. Thus, $P_{s,\mathrm{T}} = 1$ and $P_{s,\mathrm{NT}} = 0$.
- In Phase 2, the branch is executed once depending on the victim's thread actual direction v, so $P_{s,v} = 1$ and $P_{s,\neg v} = 0$ for the opposite direction $\neg v$.
- In Phase 3, the branch is always not-taken. Thus, $P_{s,\mathrm{T}} = 0$ and $P_{s,\mathrm{NT}} = 1$.

When $m = 0$, only self-loops exist, so c is independent of v and perfect privacy is achieved—at the cost of a 50% misprediction rate. When $m = 1$, the PSC reduces to the deterministic SC, and the attacker can distinguish v from the number of Phase 3 steps (3 for taken vs. 2 for not-taken). For $0 < m < 1$, the probabilistic transitions make the probe count c uncertain; the attacker can only infer v with some probability. For instance, with $m = 0.5$, observing $c = 1$ is impossible when the victim takes the branch but possible when it



(a) The victim takes the branch (b) The victim does not take the branch

Fig. 3. The probabilistic Moore machines for the attack algorithm

does not take the branch, already indicating a structural asymmetry in the two output distributions. Our goal is to compare $\Pr[out = c \mid v = \mathrm{T}]$ and $\Pr[out = c \mid v = \mathrm{NT}]$ for each c to derive the optimal strategy. To achieve our goal, we model Alg. 1 for both taken and not-taken scenarios of the victim's branch execution; the corresponding probabilistic models are shown in Fig. 3 and they encode both the attacker and the victim behaviors at the beginning of the victim's Phase 2 execution: the transition from ST' in each machine represents the victim's branch execution, reaching the appropriate state of the PSC ST or WT from where we simulate Phase 3 of Alg. 1. Formally, let M_{T} and M_{NT} represent the transition matrices of the probabilistic Moore machine when the branch is taken or not-taken, respectively. Let $\mathbf{1}_{\mathrm{ST}}, \mathbf{1}_{\mathrm{SN}} \in \mathbb{R}^4_{\geq 0}$ be two column vectors, with value 0 everywhere except for value 1 in the entry corresponding to state ST in $\mathbf{1}_{\mathrm{ST}}$ and to state SN in $\mathbf{1}_{\mathrm{SN}}$, respectively. Then $\mathbf{1}_{\mathrm{ST}}^{\intercal} M_{\mathrm{T}}^{c+1} \mathbf{1}_{\mathrm{SN}}$ gives the probability of being in the absorbing state SN after $c+1$ steps when the victim takes the branch, which coincides with observing the cut-off point c, that is, $\Pr[out = c | v = \mathrm{T}] = \mathbf{1}_{\mathrm{ST}}^{\intercal} M_{\mathrm{T}}^{c+1} \mathbf{1}_{\mathrm{SN}}$. Similarly, $\Pr[out = c | v = \mathrm{NT}] = \mathbf{1}_{\mathrm{ST}}^{\intercal} M_{\mathrm{NT}}^{c+1} \mathbf{1}_{\mathrm{SN}}$. These probabilities are the conditional probabilities required to ensure differential privacy in PSCs as defined in Def. 4.

3.3 Optimal attack strategy

The attacker's strategy is based on comparing the probabilities $\Pr[out = c | v = \mathrm{T}]$ and $\Pr[out = c | v = \mathrm{NT}]$ and selecting the branch direction with the highest probability. Specifically, we evaluate whether $c > 1/m$ satisfies the inequality

$$\mathbf{1}_{\mathrm{ST}}^{\intercal} M_{\mathrm{T}}^{c+1} \mathbf{1}_{\mathrm{SN}} > \mathbf{1}_{\mathrm{ST}}^{\intercal} M_{\mathrm{NT}}^{c+1} \mathbf{1}_{\mathrm{SN}}. \tag{1}$$

This defines the attacker's optimal strategy: if $c = \textsc{FindCutOffPoint}(n, v)$ satisfies $c > 1/m$, then it infers that $v = \mathrm{T}$; otherwise, it concludes that $v = \mathrm{NT}$. In fact, the expected number of steps from WT to SN is $1/m$, so having $c > 1/m$ hints that the PSC was in state ST after Phase 2 of Alg. 1 instead of being in WT, thus it is more likely that the victim took the branch (i.e., $v = \mathrm{T}$). Regardless of the value of m, if $c = 1$ is observed, the attack succeeds with probability 1, indicating that no (ε, δ)-differential privacy can be achieved when $\delta < 1$. This

limitation arises from the PSC structure: if the victim takes the branch, at least two steps are needed to reach SN from ST, unlike the case where the branch is not taken, as one step suffices to reach SN. We design an enhanced PSC with improved security to address this issue in Sect. 4.

We now quantify the success probability of this strategy. Bayes' theorem gives the posterior probability of the attacker's guess being correct:

$$\Pr[v{=}b|\,out{=}c] = \frac{\Pr[out{=}c|v{=}b]\,\Pr[v{=}b]}{\sum_{b'\in\{\mathrm{T},\mathrm{NT}\}}\Pr[out{=}c|v{=}b']\,\Pr[v{=}b']},$$

where b is the attacker's guess (T if $c > 1/m$, NT otherwise). When the attacker has no prior knowledge, we use the uniform prior $\Pr[v{=}\mathrm{T}] = \Pr[v{=}\mathrm{NT}] = 0.5$, and the prior factor cancels from the numerator and denominator. The likelihoods come from the matrix powers M_{T}^{c+1} and M_{NT}^{c+1} (Sect. 3.2); when one likelihood dominates the other, the posterior approaches 1 and the attack succeeds with near certainty.

The connection to differential privacy is immediate. If the PSC provides $(\varepsilon, 0)$-differential privacy, Def. 4 gives $\Pr[out{=}c|v{=}\mathrm{T}] \le \mathrm{e}^{\varepsilon} \cdot \Pr[out{=}c|v{=}\mathrm{NT}]$, yielding the well-known upper bound on the success probability [33, 61]:

$$\Pr[v{=}b|\,out{=}c] \le \frac{\mathrm{e}^{\varepsilon}}{1+\mathrm{e}^{\varepsilon}} = \frac{1}{1+\mathrm{e}^{-\varepsilon}}.$$

This bound applies specifically to pure ε-differential privacy ($\delta = 0$); for general (ε, δ)-DP with $\delta > 0$, the bound requires additional correction terms [33]. As $\varepsilon \to 0$ this bound approaches 0.5 (random guessing, meaning the defense is effective); a large ε allows it to approach 1 (the attacker can determine the branch direction with near certainty). This provides a clear quantitative relationship: the privacy budget ε directly governs the worst-case adversarial advantage.

In summary, the attacker's strategy operates as follows: for each observed probe count c, the attacker computes the posterior probability $\Pr[v{=}b \mid out{=}c]$ using Bayes' theorem and selects the direction b with the higher posterior. The key insight is that the PSC's transition structure creates an asymmetry in the distribution of c under different victim directions: when the victim takes the branch, the PSC remains at ST and requires more NT steps to reach SN, leading to larger c values; when the victim does not take, the PSC moves to WT (or stays at ST with reduced probability in the enhanced model), leading to smaller c values. This structural asymmetry is precisely what the attacker exploits: the ratio between $\Pr[out{=}c \mid v{=}\mathrm{T}]$ and $\Pr[out{=}c \mid v{=}\mathrm{NT}]$ grows as c deviates from the expected transition count $1/m$, enabling the attacker to infer the branch direction with increasing confidence. The DP guarantee ensures that this ratio, which means the attacker's advantage, is bounded by ε.

4 Enhanced PSC Model and Parameter Synthesis

As seen in the previous section, it is still possible for an attacker to accurately infer the branch execution of the victim because there exist deterministic transitions in the PSC. In particular, the observation $c = 1$ always reveals that

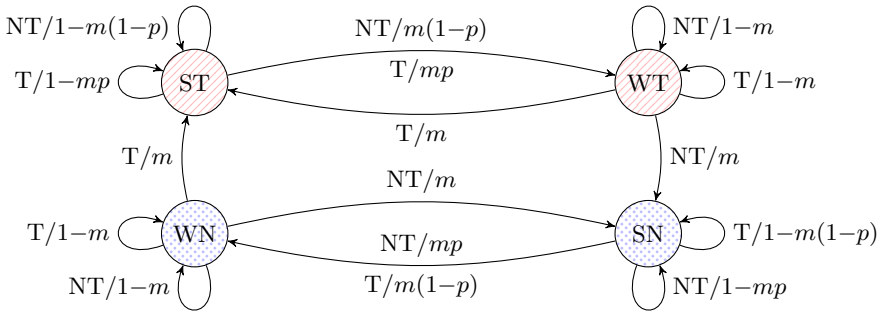


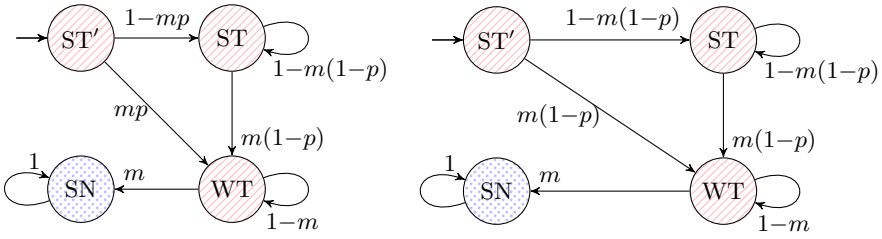
Fig. 4. The probabilistic Moore machine for the enhanced PSC

$v = \mathrm{NT}$, since reaching SN in one step is impossible when the victim takes the branch. This constitutes a fundamental vulnerability that cannot be mitigated by adjusting the existing parameter m alone. In this section, we design a new model based on the original PSC and establish its differential privacy properties by parameter synthesis.

4.1 The enhanced PSC model

We enhance the PSC model by introducing additional probabilistic choices. The original probability update mechanism is retained, where the transitions are updated with probability m. When a transition is triggered, the actual state reached by the transition is probabilistically decided by means of a new parameter $p \in [0, 1]$, termed the *defense parameter*: for state ST and input T, instead of remaining in ST for sure, with probability p the PSC moves to WT and for input NT, instead of going to WT for sure, with probability p the PSC remains in ST; for state SN the defense is applied symmetrically. The resulting enhanced PSC is shown in Fig. 4. Notably, forcing all transitions to be probabilistic from both ST and SN is essential to achieve security: as we will see below, they prevent an attack based on Alg. 1 similar to the one presented in Sect. 3.3. Note that our enhanced PSC generalizes both the classical saturating counters (by setting $m = 1$ and $p = 0$) and the probabilistic saturating counters (when $p = 0$).

We now show how to synthesize parameters m and p such that the enhanced PSC satisfies a given (ε, δ)-differential privacy requirement. Similar to Sect. 3.2, we consider the probabilistic Moore machine corresponding to the attack model, depicted in Fig. 5. To ensure (ε, δ)-differential privacy, we must satisfy the requirements of Def. 4. This involves solving the two differential privacy bounds to determine the feasible range for m and p. Given that the condition in Eq. (1) underlying the attacker's strategy involves polynomials in m, p, and the exponent c, finding explicit analytic solutions is challenging. However, we can fix one parameter and derive a suitable range for the other: suppose that we are verifying the enhanced PSC for (ε, δ)-differential privacy with $\varepsilon = 10^{-2}$ and $\delta = 10^{-5}$. We start by fixing the probability m at 0.5, which has shown good



(a) The victim takes the branch (b) The victim does not take the branch

Fig. 5. The enhanced probabilistic Moore machines for the attack algorithm

performance in [42]. Using SageMath [66] for symbolic algebra computation, we obtain $p \in [0.4899, 0.5017]$ fulfilling the differential privacy constraints.

We also establish a formal relationship between the defense parameter p and the achievable privacy level, as presented in Thm. 1:

Theorem 1. *Given an enhanced PSC $\mathcal{D}$ with defense parameter $p \in (0,1)$, if $p \geq \frac{1}{2}$, then $\mathcal{D}$ satisfies $(\ln \frac{p}{1-p}, 0)$-differential privacy; otherwise, it satisfies $(\ln \frac{1-p}{p}, 0)$-differential privacy.*

The detailed proof of Thm. 1 is provided in Appendix C of the arxiv version [13]. A notable feature of this result is that the achievable differential privacy level depends solely on p and is *independent of* m. Intuitively, m acts as a uniform "speed" parameter that scales all non-self-loop transition probabilities equally in both the taken and not-taken attack models (cf. Fig. 5). The differential privacy guarantee is determined by the *ratio* of the output probabilities $\Pr[out = c \mid v = \mathrm{T}]$ and $\Pr[out = c \mid v = \mathrm{NT}]$; since m appears as a common multiplicative factor in both, it cancels out in this ratio via the binomial theorem (see the proof in Appendix C of the arxiv version [13]). In contrast, p introduces an *asymmetry* exclusively at the strong states (ST/SN), directly affecting how differently the two models behave and thus governing the privacy level.

Moreover, we remark that Thm. 1 establishes *pure* ε-differential privacy (i.e., $\delta = 0$) for all values of $p \in (0,1)$. We adopted the more general (ε, δ)-DP framework (Def. 4) as the starting point of our analysis because it is the standard formulation in the DP literature [18] and, importantly, because we first needed to show that existing PSCs *fail* to satisfy even this weaker notion (Sect. 3.3). The fact that our enhanced PSC achieves the stronger $\delta = 0$ guarantee is a positive outcome of the design.

The interplay between p and the resulting privacy–utility trade-off is analyzed in detail in Sect. 4.3.

4.2 Estimating the misprediction rate

The branch execution of a program can be modeled as a sequence σ in $\{\mathrm{T}, \mathrm{NT}\}^*$. Exact misprediction rates over all finite traces are generally infeasible to compute: traces grow exponentially, and real executions may be history-dependent.

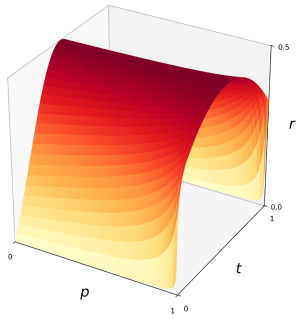


Fig. 6. The stationary misprediction rate as a function of the defense parameter p and branch taken probability t

Worst-case sequences can force deterministic SCs to mispredict every step, but such traces are rare in practice; we therefore use a statistical estimate.

For a given branch, let the probability of executing T be $t \in (0, 1)$ and of NT be $s = 1 - t$; let $q = 1 - p$. Here, t abstracts the empirical taken probability of one static branch, not a full generative model of program traces. For $m > 0$ and non-degenerate branch behavior, the induced finite Markov chain is irreducible and aperiodic, and hence has a unique stationary distribution. Let $\mu = [a, b, c, d]$ denote this stationary distribution over the states ST, WT, WN, and SN, respectively. The stationary distribution is obtained by solving $\mu M = \mu$ for the transition matrix M of Fig. 4; the misprediction rate is then $r = (a+b)s+(c+d)t$. Solving these equations yields

$$r = \frac{ts(qt + ps)(1 + qs + pt) + ts(qs + pt)(1 + qt + ps)}{t(qt + ps)(1 + qs + pt) + s(qs + pt)(1 + qt + ps)}.$$

Appendix A of the arxiv version [13] gives the full calculation.

Since m uniformly scales all non-self-loop transitions (cf. Fig. 5), it affects convergence speed but not equilibrium: smaller m only makes the PSC dwell longer in each state, while transition directions are governed by t and p. Thus the stationary distribution and misprediction rate are determined entirely by p and t. For $p = 0$, the rate reduces to that of the deterministic SC. Fig. 6 shows r for $p \in [0, 1]$ and $t \in [0.001, 0.999]$: for fixed $t \in (0, 1)$, r increases monotonically with p, reaching 0.5 at $p = 1$; Sect. 4.3 discusses the privacy–utility implication.

4.3 Privacy-utility trade-off

The DP guarantee of the enhanced PSC (Thm. 1) and its stationary misprediction rate (Sect. 4.2) induce a privacy-utility curve. Following the DP literature [18, 25, 27], we define utility as prediction accuracy $u = 1 - r$.

The privacy-utility curve. By Thm. 1, $\varepsilon = |\ln \frac{p}{1-p}|$, decreasing on $p \in (0, 0.5]$ and increasing on $p \in [0.5, 1)$. For fixed $t \in (0, 1)$, r increases monotonically

with p (Sect. 4.2). Thus improving privacy requires moving p toward 0.5, which increases r. At $p = 0.5$ we obtain perfect privacy ($\varepsilon = 0$) but $r = 0.5$; as p tends to 0 (or 1), prediction approaches the deterministic SC while ε tends to ∞.

Optimal defense parameter selection. For any target privacy budget $\varepsilon > 0$, there exist two values $p_1 = \frac{1}{1+e^{\varepsilon}} < \frac{1}{2}$ and $p_2 = \frac{e^{\varepsilon}}{1+e^{\varepsilon}} > \frac{1}{2}$ that achieve the same $(\varepsilon, 0)$-differential privacy. Since r is monotonically increasing in p, choosing $p_1 < 0.5$ always yields a strictly lower misprediction rate than $p_2 > 0.5$ for the same privacy guarantee.

Proposition 1. *For any target privacy budget $\varepsilon > 0$ and branch probability $t \in (0, 1)$, the defense parameter $p^* = \frac{1}{1+e^{\varepsilon}}$ minimizes the misprediction rate among all values of p that satisfy $(\varepsilon, 0)$-differential privacy.*

Proof. By Thm. 1, the set of p values achieving exactly $(\varepsilon, 0)$-DP is $\{p_1, p_2\}$ where $p_1 = \frac{1}{1+e^{\varepsilon}}$ and $p_2 = \frac{e^{\varepsilon}}{1+e^{\varepsilon}} = 1 - p_1$. Since r is monotonically increasing in p and $p_1 < p_2$, we have $r(p_1, t) < r(p_2, t)$ for all $t \in (0, 1)$. Moreover, any p' satisfying $(\varepsilon', 0)$-DP with $\varepsilon' \leq \varepsilon$ must have $p' \in [p_1, p_2]$, which gives $r(p', t) \geq r(p_1, t)$. □

Consequently, the defense parameter should always be chosen from the interval $(0, 0.5]$. For example, to achieve $(\ln 9, 0)$-DP, one should set $p^* = 0.1$ (not $p = 0.9$), which significantly reduces the misprediction rate while providing the same privacy guarantee.

Comparison with the randomized response mechanism. An alternative is the *randomized response* (RR) mechanism [18, 60], which in our setting randomizes the branch input at *every* counter state: the true direction b is retained with probability $p_{\mathsf{RR}} = \frac{e^{\varepsilon}}{1+e^{\varepsilon}}$ and flipped otherwise. By DP post-processing [18], composing RR with any deterministic saturating counter yields an $(\varepsilon, 0)$-DP mechanism for the attack output.

The key difference is *where* randomization is applied: RR perturbs every branch input, whereas our enhanced PSC randomizes only at the strong states (ST/SN), preserving weak-state updates and yielding better accuracy for the same privacy guarantee. Under RR, the SC perceives an effective taken probability $t_{\mathsf{eff}} = t \cdot p_{\mathsf{RR}} + (1 - t)(1 - p_{\mathsf{RR}})$; the resulting misprediction rate is:

$$r_{\mathsf{RR}} = \frac{(1-t) \cdot t_{\mathsf{eff}}^2 (2 - t_{\mathsf{eff}}) + t \cdot (1 - t_{\mathsf{eff}})^2 (1 + t_{\mathsf{eff}})}{1 - t_{\mathsf{eff}}(1 - t_{\mathsf{eff}})}. \tag{2}$$

Table 1 compares the two approaches for representative privacy budgets and branch probabilities. In all cases, the enhanced PSC achieves a lower misprediction rate than RR; the advantage is most pronounced for highly biased branches and tighter privacy budgets (up to 78% reduction at $\varepsilon = \ln 3$, $t = 0.9$).

Practical parameter guidelines. Given a target privacy budget ε, one should set $p^* = \frac{1}{1+e^{\varepsilon}}$ (Prop. 1). Since neither ε nor r depends on m (cf. Thm. 1 and Sect. 4.2), the update probability m can be tuned independently for convergence

Table 1. Misprediction rates: enhanced PSC (with $p^* = \frac{1}{1+e^{\varepsilon}}$) vs. randomized response (RR) on a deterministic 2-bit SC, for selected ε and branch probability t. r_{SC} is the rate of the baseline deterministic SC ($p = 0, m = 1$), and Reduction is $(r_{\mathsf{RR}} - r_{\mathsf{PSC}})/(r_{\mathsf{RR}} - r_{\mathrm{SC}})$.

ε	t	r_{SC}	r_{PSC}	r_{RR}	Reduction
$\ln 9 \approx 2.20$	0.9	0.117	0.129	0.155	68%
	0.8	0.251	0.265	0.287	63%
	0.7	0.377	0.386	0.399	61%
$\ln 3 \approx 1.10$	0.9	0.117	0.147	0.255	78%
	0.8	0.251	0.285	0.357	68%
	0.7	0.377	0.399	0.435	62%

speed; $m = 0.5$ [42] provides a balanced default. For example, $\varepsilon = \ln 9 \approx 2.20$ with $p = 0.1$ and $m = 0.5$ yields at most 1.2 percentage points of additional misprediction (at $t = 0.9$) while bounding the adversary's success probability by $\frac{e^{\varepsilon}}{1+e^{\varepsilon}} = 0.9$.

4.4 General k-bit PSCs

Our methodology generalizes to arbitrary k-bit designs; all formal definitions and proofs are in Appendix C of the arxiv version [13].

We first consider a standard k-bit PSC, obtained by extending the probabilistic transition function to a k-bit linear chain with $2^{k-1} - 1$ WT and WN states. Our analysis reveals that this naive extension remains vulnerable to the Prime+Probe attack. Specifically, by generalizing the transition matrix analysis of Sect. 3.2 to the k-bit case (see Appendix C of the arxiv version [13] for the full proof), we can show that the comparison $\Pr[\mathsf{out}_{\mathrm{T}} = c] \bowtie \Pr[\mathsf{out}_{\mathrm{NT}} = c]$ reduces to $c \bowtie (2^{k-1} - 1)/m$ for any comparison operator $\bowtie$. Hence, an attacker can distinguish branch outcomes with high confidence once the probe count c significantly exceeds $(2^{k-1} - 1)/m$, using the same threshold-based strategy as in the 2-bit case. This confirms that merely adding probabilistic updates without targeted defense transitions is insufficient for higher-order counters.

To address this, we propose the enhanced k-bit PSC, injecting the defense parameter p exclusively into transitions from ST and SN (Fig. 7); intermediate states retain their original update m. This generalized design, shown in Fig. 7, preserves the rigorous security guarantees of the 2-bit version:

Theorem 2. *Given an enhanced k-bit PSC and $p \in (0, 1)$, if $p \geq \frac{1}{2}$, then it is $(\ln \frac{p}{1-p}, 0)$-differentially private; otherwise, it is $(\ln \frac{1-p}{p}, 0)$-differentially private.*

For the general enhanced k-bit PSC, we can prove that its misprediction rate is independent of the update probability m.

Theorem 3. *Given an enhanced k-bit PSC, its misprediction rate is independent of m.*

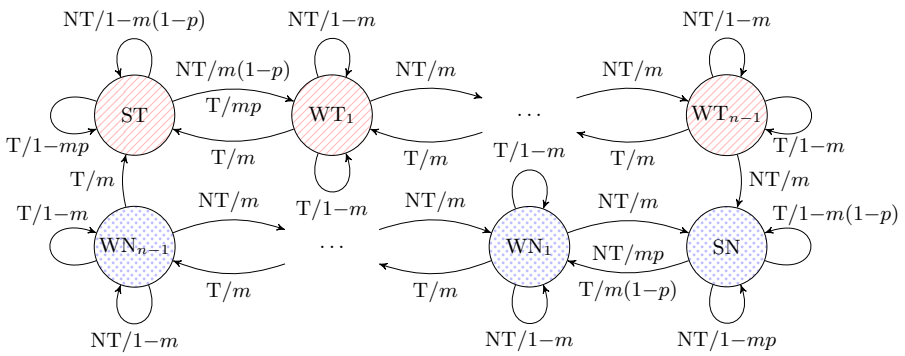


Fig. 7. The probabilistic Moore machine for the enhanced k-bit PSC; $n = 2^{k-1}$

The proof (Appendix C of the arxiv version [13]) relies on the decomposition $M = mB + (1-m)I$, where B is independent of m; solving $\mu M = \mu$ then reduces to $\mu(B - I) = 0$, which is independent of m.

In summary, extending the PSC model to the k-bit case not only enhances its flexibility but also maintains its security and performance characteristics. The fact that both the DP guarantee and the misprediction rate are independent of the counter width k (in the sense that the same ε-p relationship holds) further underscores the generality of the proposed design.

5 Experimental Evaluation

In this section, we evaluate the performance of PSCs under various parameters using real-world programs. For comparison, we also analyze the experimental results of deterministic saturating counters and original PSCs. We utilize the clock-level Gem5 simulator [5] to simulate an out-of-order execution processor based on the latest Intel Sunny Cove core model [62]. The PSCs model is implemented within the classic Tournament branch predictor architecture [35], with probabilistic mechanisms realized through Register Transfer Level code. The formal model maps to the implementation as follows: each Moore machine state corresponds to a saturating counter value; m and p are realized by linear-feedback shift registers. Experiments validate *utility* (prediction accuracy) under realistic workloads; privacy is established by the formal theorems of Sect. 4, not by empirical observation. To assess performance, we employ the SPEC CPU 2017 benchmark suite [8]. After warming up with 100 million instructions, the simulator executes further 100 million instructions in clock-precise mode. We measure *mispredictions per kilo-instruction (MPKI)* to gauge branch prediction accuracy and *instructions per cycle (IPC)* to evaluate overall performance.

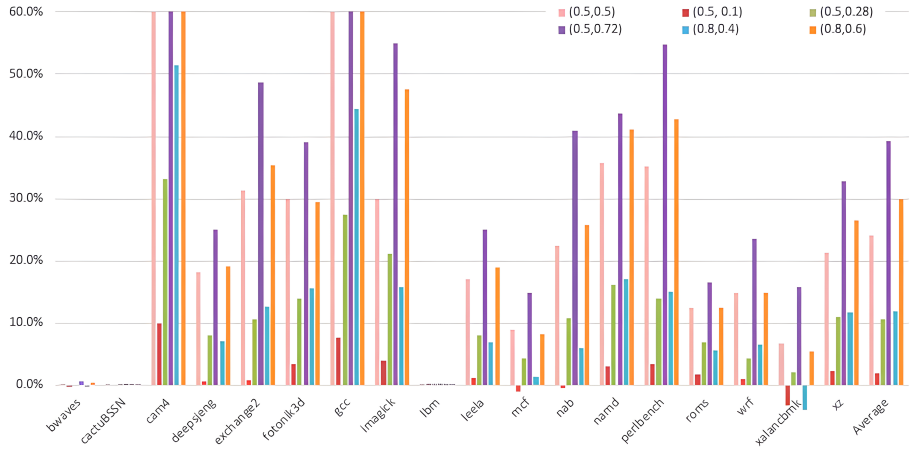


Fig. 8. The normalised performance overhead for PSCs under different settings of parameters (m, p), where the baseline is the deterministic saturating counter

5.1 Performance evaluation

Figure 8 presents the performance evaluation results. We measure Enhanced PSC overhead relative to the deterministic baseline across six configurations grouped by DP guarantee: (i) $(m, p) = (0.5, 0.5)$, yielding perfect $(0, 0)$-DP; (ii) $(0.5, 0.1)$, satisfying $(\ln 9, 0)$-DP; and (iii) $\{(0.5, 0.28), (0.5, 0.72), (0.8, 0.4), (0.8, 0.6)\}$, all providing $(0.94, 0)$-DP. These budgets align with real-world deployments [15, 23, 56]. The $(m, p) = (0.5, 0.1)$ setting gives a good balance, with 1.8% average normalized overhead under $(\ln 9, 0)$-DP over all benchmarks except the outlier `cam4`. The perfect-privacy endpoint $(0.5, 0.5)$ incurs 24.1% average overhead because it drives the stationary misprediction rate toward 50%; it marks the privacy-utility frontier rather than a recommended operating point. Among the $(0.94, 0)$-DP configurations, overhead increases with p, confirming that smaller p with larger m yields better performance. Some stochastic configurations even outperform the deterministic baseline (e.g., a 4.1% gain on `xalancbmk` for $(0.8, 0.4)$), consistent with prior observations that randomness can prevent predictors from getting stuck in suboptimal states [42].

5.2 Comparing theoretical and experimental misprediction rates

To validate the theoretical rates from Sect. 4.2, we follow [22] and apply PSCs to MergeSort (Appendix B of the arxiv version [13]) on uniform and pre-sorted datasets of 10^5 integers. We compare P_{th} with the observed P_{exp} across four sensitive branches; Table 2 shows close agreement, validating both the stationary analysis and the assumption that this workload reaches stationarity. Consistent with Sect. 4.2, PSCs with $p > 0$ have higher misprediction rates than deterministic SCs when t deviates from 0.5, so p should be minimized subject to the target

Table 2. The experimental and theoretical misprediction rate on the MergeSort algorithm. PSC(m,p) denotes the PSC with parameters m and p.

Data	Branch	t	SC		PSC(0.5, 0.5)		PSC(0.8, 0.4)	
			P_{th}	P_{exp}	P_{th}	P_{exp}	P_{th}	P_{exp}
Uniform	1st	0.939	0.068	0.061	0.114	0.094	0.104	0.089
	2nd	0.495	0.500	0.510	0.500	0.504	0.500	0.506
	3rd	0.355	0.433	0.406	0.458	0.471	0.453	0.469
	4th	0.437	0.487	0.544	0.492	0.514	0.491	0.525
Sorted	1st	0.891	0.128	0.109	0.194	0.154	0.179	0.149
	2nd	1	0	0	0	0	0	0
	3rd	0	0	0	0	0	0	0
	4th	0.895	0.123	0.105	0.188	0.158	0.173	0.154

DP budget. For shorter executions, the stationary rate should be interpreted as an asymptotic long-run reference.

6 Conclusion

We presented a formal DP analysis of PSCs under Prime+Probe attacks by modeling counters and attacks as probabilistic Moore machines. The analysis yields optimal attacks against existing PSCs and motivates an enhanced PSC whose parameter p enforces pure ε-DP while m controls convergence speed. We derived stationary misprediction rates, established the privacy-utility trade-off, and showed that targeted randomization improves utility over randomized response; experiments corroborate the stationary analysis. Our formal guarantees hold for the PSC primitive under the Prime+Probe attack model. Future work includes richer predictors such as TAGE, automated synthesis via PRISM [37], and DP composition under repeated attacks.

Acknowledgements. Work supported in part by National Key R&D Program of China No.2025YFE0220300, the Beijing Natural Science Foundation Project No. IS26039, the CAS Project for Young Scientists in Basic Research under grant No. YSBR-040, NSFC under grant No. 61836005, the CAS Pioneer Hundred Talents Program, and the ISCAS New Cultivation Project ISCAS-PYFX-202201. This project is part of the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant no. 101008233.

References

1. Abadi, M., Chu, A., Goodfellow, I.J., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM

SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 308–318. ACM (2016)
2. Barthe, G., Fong, N., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.: Advanced probabilistic couplings for differential privacy. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 55–67. ACM (2016)
3. Barthe, G., Gaboardi, M., Arias, E.J.G., Hsu, J., Roth, A., Strub, P.: Higher-order approximate relational refinement types for mechanism design and differential privacy. In: Rajamani, S.K., Walker, D. (eds.) Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015. pp. 55–68. ACM (2015)
4. Barthe, G., Köpf, B., Olmedo, F., Béguelin, S.Z.: Probabilistic relational reasoning for differential privacy. ACM Trans. Program. Lang. Syst. **35**(3), 9:1–9:49 (2013)
5. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: The gem5 simulator. ACM SIGARCH computer architecture news **39**(2), 1–7 (2011)
6. Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., Sadeghi, A.: Software grand exposure: SGX cache attacks are practical. In: Enck, W., Mulliner, C. (eds.) 11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017. USENIX Association (2017)
7. Bu, Z., Dong, J., Long, Q., Su, W.J.: Deep learning with gaussian differential privacy. CoRR **abs/1911.11607** (2019)
8. Bucek, J., Lange, K.D., v. Kistowski, J.: Spec cpu2017: Next-generation compute benchmark. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. pp. 41–42 (2018)
9. Bulck, J.V., Weichbrodt, N., Kapitza, R., Piessens, F., Strackx, R.: Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017. pp. 1041–1056. USENIX Association (2017)
10. Chen, G., Chen, S., Xiao, Y., Zhang, Y., Lin, Z., Lai, T.: Sgxpectre: Stealing intel secrets from SGX enclaves via speculative execution. IEEE Secur. Priv. **18**(3), 28–37 (2020)
11. Cheng, A., Wang, J., Zhang, X.S., Chen, Q., Wang, P., Cheng, J.: DPNAS: neural architecture search for deep learning with differential privacy. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022. pp. 6358–6366. AAAI Press (2022)
12. Cheng, H., Yu, P., Hu, H., Zawad, S., Yan, F., Li, S., Li, H.H., Chen, Y.: Towards decentralized deep learning with differential privacy. In: Silva, D.D., Wang, Q., Zhang, L. (eds.) Cloud Computing - CLOUD 2019 - 12th International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25-30, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11513, pp. 130–145. Springer (2019)
13. Chi, Z., Zhao, L., Liu, D., Li, Y., Yang, P., Wang, B.Y., Hou, R., Huang, C.C., Turrini, A., Zhang, L., Zhan, N.: Synthesizing probabilistic saturating counters with differentially private formal guarantees (2026), `https://arxiv.org/abs/2608.10521`

14. Choudhury, O., Gkoulalas-Divanis, A., Salonidis, T., Sylla, I., Park, Y., Hsu, G., Das, A.: Differential privacy-enabled federated learning for sensitive health data. CoRR **abs/1910.02578** (2019)
15. Cormode, G., Kulkarni, T., Srivastava, D.: Marginal release under local differential privacy. In: Proceedings of the 2018 International Conference on Management of Data. p. 131–146. SIGMOD '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183713.3196906, https://doi.org/10.1145/3183713.3196906
16. Dankar, F.K., Emam, K.E.: The application of differential privacy to health data. In: Srivastava, D., Ari, I. (eds.) Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012. pp. 158–166. ACM (2012)
17. Disselkoen, C., Kohlbrenner, D., Porter, L., Tullsen, D.M.: Prime+abort: A timer-free high-precision L3 cache attack using intel TSX. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017. pp. 51–67. USENIX Association (2017)
18. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science **9**(3–4), 211–407 (2014)
19. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II. Lecture Notes in Computer Science, vol. 4052, pp. 1–12. Springer (2006). https://doi.org/10.1007/11787006_1, https://doi.org/10.1007/11787006_1
20. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP. LNCS, vol. 4052, pp. 1–12. Springer (2006)
21. Dyda, A., Purcell, M., Curtis, S., Field, E., Pillai, P., Ricardo, K., Weng, H., Moore, J.C., Hewett, M., Williams, G., Lau, C.L.: Differential privacy for public health data: An innovative tool to optimize information sharing while protecting data confidentiality. Patterns **2**(12), 100366 (2021)
22. Elkhouly, R., El-Mahdy, A., Elmasry, A.: 2-bit branch predictor modeling using markov model. Procedia Computer Science **62**, 650–653 (2015)
23. Erlingsson, U., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 1054–1067. CCS '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2660267.2660348, https://doi.org/10.1145/2660267.2660348
24. Evtyushkin, D., Riley, R., Abu-Ghazaleh, N.C., ECE, Ponomarev, D.: BranchScope: A new side-channel attack on directional branch predictor. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18). pp. 693–707. ACM (2018)
25. Geng, Q., Viswanath, P.: The optimal noise-adding mechanism in differential privacy. IEEE Transactions on Information Theory **62**(2), 925–951 (2016). https://doi.org/10.1109/TIT.2015.2504967
26. Ghazi, B., Golowich, N., Kumar, R., Manurangsi, P., Zhang, C.: Deep learning with label differential privacy. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual. pp. 27131–27145 (2021)
27. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: Proceedings of the Forty-First Annual ACM Symposium on

Theory of Computing. p. 351–360. STOC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536464, https://doi.org/10.1145/1536414.1536464
28. Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: Cache attacks on intel SGX. In: Giuffrida, C., Stavrou, A. (eds.) Proceedings of the 10th European Workshop on Systems Security, EUROSEC 2017, Belgrade, Serbia, April 23, 2017. pp. 2:1–2:6. ACM (2017)
29. Grayson, B., Rupley, J., Zuraski, G.Z., Quinnell, E., Jiménez, D.A., Nakra, T., Kitchin, P., Hensley, R., Brekelbaum, E., Sinha, V., Ghiya, A.: Evolution of the samsung exynos cpu microarchitecture. In: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). pp. 40–51 (2020)
30. Gruss, D., Maurice, C., Wagner, K., Mangard, S.: Flush+flush: A fast and stealthy cache attack. In: Caballero, J., Zurutuza, U., Rodríguez, R.J. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment - 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9721, pp. 279–299. Springer (2016)
31. Hähnel, M., Cui, W., Peinado, M.: High-resolution side channels for untrusted operating systems. In: Silva, D.D., Ford, B. (eds.) 2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017. pp. 299–312. USENIX Association (2017)
32. Huo, T., Meng, X., Wang, W., Hao, C., Zhao, P., Zhai, J., Li, M.: Bluethunder: A 2-level directional predictor based side-channel attack against sgx. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**, 321–347 (2020)
33. Kairouz, P., Oh, S., Viswanath, P.: The composition theorem for differential privacy. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 1376–1385. PMLR, Lille, France (07–09 Jul 2015), https://proceedings.mlr.press/v37/kairouz15.html
34. Kayaalp, M., Abu-Ghazaleh, N.B., Ponomarev, D.V., Jaleel, A.: A high-resolution side-channel attack on last-level cache. In: Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016. pp. 72:1–72:6. ACM (2016)
35. Kessler, R.E.: The alpha 21264 microprocessor. international symposium on microarchitecture **19**(2), 24–36 (1999)
36. Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y.: Spectre attacks: exploiting speculative execution. Commun. ACM **63**(7), 93–101 (2020)
37. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV. LNCS, vol. 6806, pp. 585–591. Springer (2011)
38. Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside {SGX} enclaves with branch shadowing. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 557–574 (2017)
39. Lee, S., Shih, M., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017. pp. 557–574. USENIX Association (2017)
40. Liu, D., Wang, B., Fu, C., Zhang, L.: Model checking differentially private properties. Theor. Comput. Sci. **943**, 153–170 (2023)

41. Liu, D., Wang, B., Zhang, L.: Verifying pufferfish privacy in hidden markov models. In: Finkbeiner, B., Wies, T. (eds.) Verification, Model Checking, and Abstract Interpretation - 23rd International Conference, VMCAI 2022, Philadelphia, PA, USA, January 16-18, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13182, pp. 174–196. Springer (2022)
42. Lu-Tan Zhao, Rui Hou, K.W., Su, Y.L., Li, P.N., Meng, D.: A novel probabilistic saturating counter design. Journal of computer science and technology **36**(5), 1022 (2021)
43. Maisuradze, G., Rossow, C.: ret2spec: Speculative execution using return stack buffers. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 2109–2122. ACM (2018)
44. McFarling, S.: Combining branch predictors. Tech. rep., Technical Report TN-36, Digital Western Research Laboratory (1993)
45. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings. pp. 94–103. IEEE Computer Society (2007)
46. Nasr, M., Shokri, R., Houmansadr, A.: Improving deep learning with differential privacy using gradient encoding and denoising. CoRR **abs/2007.11524** (2020)
47. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing. p. 75–84. Association for Computing Machinery, New York, NY, USA (2007)
48. Nissim, K., Smorodinsky, R., Tennenholtz, M.: Approximately optimal mechanism design via differential privacy. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 203–213. ACM (2012)
49. Rabin, M.O.: Probabilistic automata. Information and control **6**(3), 230–245 (1963)
50. Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard, S.: Malware guard extension: Using SGX to conceal cache attacks. In: Polychronakis, M., Meier, M. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment - 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10327, pp. 3–24. Springer (2017)
51. Seznec, A.: TAGE-SC-L Branch Predictors Again. In: 5th JILP Workshop on Computer Architecture Competitions (JWAC-5): Championship Branch Prediction (CBP-5). Seoul, South Korea (2016)
52. Sinharoy, B., Van Norstrand, J.A., Eickemeyer, R.J., Le, H.Q., Leenstra, J., Nguyen, D.Q., Konigsburg, B., Ward, K., Brown, M.D., Moreira, J.E., Levitan, D., Tung, S., Hrusecky, D., Bishop, J.W., Gschwind, M., Boersma, M., Kroener, M., Kaltenbach, M., Karkhanis, T., Fernsler, K.M.: Ibm power8 processor core microarchitecture. IBM Journal of Research and Development **59**(1), 2:1–2:21 (2015)
53. Smith, G.: On the foundations of quantitative information flow. In: de Alfaro, L. (ed.) Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009. pp. 288–302. Lecture Notes in Computer Science, Springer (2009). https://doi.org/10.1007/978-3-642-00596-1_21
54. Stevens, T., Ngong, I.C., Darais, D., Hirsch, C., Slater, D., Near, J.P.: Backpropagation clipping for deep learning with differential privacy. CoRR **abs/2202.05089** (2022)

55. Suggs, D., Subramony, M., Bouvier, D.: The amd "zen 2" processor. IEEE Micro **40**(2), 45–52 (2020)
56. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in apple's implementation of differential privacy on macos 10.12 (2017), https://arxiv.org/abs/1709.02753
57. Wang, Q., Tang, M., Xu, K., Wang, H.: Modeling, derivation, and automated analysis of branch predictor security vulnerabilities. In: IEEE International Symposium on High-Performance Computer Architecture, HPCA 2024, Edinburgh, United Kingdom, March 2-6, 2024. pp. 409–423. IEEE (2024). https://doi.org/10.1109/HPCA57654.2024.00038, https://doi.org/10.1109/HPCA57654.2024.00038
58. Wang, Y., Ding, Z., Kifer, D., Zhang, D.: Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 919–938. ACM (2020)
59. Wang, Y., Ding, Z., Wang, G., Kifer, D., Zhang, D.: Proving differential privacy with shadow execution. In: McKinley, K.S., Fisher, K. (eds.) Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019. pp. 655–669. ACM (2019)
60. Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association **60**(309), 63–69 (1965), http://www.jstor.org/stable/2283137
61. Wasserman, L., Zhou, S.: A statistical framework for differential privacy. Journal of the American Statistical Association **105**(489), 375–389 (2010). https://doi.org/10.1198/jasa.2009.tm08651
62. Wikichip: Sunny cove (2019), https://en.wikichip.org/wiki/intel/microarchitectures/sunny_cove
63. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 640–656. IEEE Computer Society (2015)
64. Zhang, D., Kifer, D.: Lightdp: towards automating differential privacy proofs. In: Castagna, G., Gordon, A.D. (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017. pp. 888–901. ACM (2017)
65. Zhang, L., Zhu, T., Xiong, P., Zhou, W., Yu, P.S.: More than privacy: Adopting differential privacy in game-theoretic mechanism design. ACM Comput. Surv. **54**(7), 136:1–136:37 (2022)
66. Zimmermann, P., Casamayou, A., Cohen, N., Connan, G., Dumont, T., Fousse, L., Maltey, F., Meulien, M., Mezzarobba, M., Pernet, C., et al.: Computational mathematics with SageMath. SIAM (2018)