**SemCom-Synth: Differentially Private Synthetic Semiconductor Operations Text with Hybrid Generation and Open Audits**

Youssef Alothman¹

¹Computer science /University of Portsmouth, Portsmouth, United Kingdom

Youssef.alothman@port.ac.uk

***Abstract***

*We introduce* ***SemCom-Synth****, a privacy-preserving synthetic corpus of semiconductor operations text produced by a dual-track pipeline: a language model trained with* ***DP-SGD*** *and a* ***privacy-aware paraphraser****. A risk-aware hybrid selector enforces* ***k-syntheticity*** *(string/edit and embedding distances), layered* ***PII/domain-jargon redaction****, and canary audits before release. Utility is assessed with* ***Train-on-Synthetic, Test-on-Real (TSTR)*** *across five tasks—root-cause taxonomy, actionability, role/shift, severity, and fault-span extraction—reporting macro-F1, macro-recall, AUROC, calibration (ECE), and asymmetric cost curves. We chart the privacy–utility frontier across ε∈{1,2,4,8} (δ≈10⁻⁵) and show that* ***≤10% few-shot*** *fine-tuning on real text closes most of the remaining gap to full-real baselines. We release dataset shards, prompts, cards (Data/Privacy/Model), and an* ***open audit harness*** *(membership inference, nearest-neighbor, canary) to support reproducible assessment and safe reuse.*

***Keywords***

*Differential Privacy; Synthetic Text; Semiconductor Manufacturing; Root Cause Analysis; Benchmarking; Membership Inference*

**1. Introduction**

Process text from semiconductor fab environments—shift turnovers, tool logbooks, root-cause analyses, and corrective/preventive action (CAPA) memoranda—hold core process understanding. However, embedded in this kind of text is sensitive intellectual property (e.g., recipe settings, tool serial numbers) and possibly identifiable information. Combined, these make open data exchange not possible and reproducibility for natural language processing in manufacturing challenging. Synthetic data is a good compromise that is desired, but naive generation risks verbatim leakage and linkage attacks, and strong privacy weakens downstream utility [4–7, 16].

This paper argues that the privacy-conserving syntactic text of fabs needs to meet three practical desiderata: utility, verifiability, and governance. Utility is measured in terms of models acquired from the release being competitive on fab-applicable tasks such as multi-label root-cause taxonomy, actionability prediction, role/shift inference, severity estimation, and span extraction of fault signatures. Verifiability calls for open, reproducible audits against concrete leakage threats (e.g., canary exposure, nearest-neighbor overlap, and membership inference) rather than loose guarantees [4–7, 15]. Governance encompasses the policies and artifacts—data, privacy, and model cards; acceptable-use terms; do-not-relink clauses—that enable responsible downstream use within industrial applications and cooperative collaboration between companies [14, 16].

We present SemCom-Synth, a synthetic dataset and set of methods to harmonize these conflicts. Our approach integrates formal differential privacy with practical safeguards and sound audits. A differentially-private (DP) text generator, trained using DP-SGD, provides quantifiable () guarantees on per-contribution individuality [1–3, 13, 14]. A privacy-aware paraphrasing pipeline augments linguistic variety and task effectiveness with constrained decoding (e.g., nucleus sampling with repetition limits) [12]. A risk-aware hybrid chooser chooses the safer option per sample by explicit trials—k-nearest-neighbor distance thresholds in string and embedding space (k-syntheticity, using SimCSE-type embeddings) [10], canary exposure scores [4], and tiered PII/domain-jargon removal. Besides generation, we combine evaluation with Train-on-Synthetic, Test-on-Real (TSTR) regimes and calibration metrics [9] and document few-shot transfer where a synth-trained model is adapted with ≤10% real data.

Threat model and scope. We consider a passive attacker with auxiliary data who attempts (i) membership inference—whether a provided record contributed to training—and (ii) record linkage—connecting generated text to an off-network comment. We control inference risk with DP accounting [1–3, 13] and defend against plausible linkage with distance-based filtering, aggressive redacting, and reject-and-regenerate rules. Active poisoning or side-channel attacks on deployment infrastructure are out of scope but align with our framework.

Research questions:

RQ1. How does downstream utility vary as a function of privacy budget over fab-relevant tasks?

RQ2. To what extent do k-syntheticity and layered redaction alleviate memorization and overlap with private text [4–7, 10, 15]?

RQ3. Can few-shot fine-tuning (1–10% real) minimize the residual utility gap without sacrificing privacy protection?

RQ4. How do generation choices (DP vs. paraphrase) and decoding/threshold parameters in the hybrid selector [11, 12] impact outcomes?

Contributions:

(C1) A DP-SGD generator + privacy-aware paraphraser with role, shift, label set, and time bucket controllable conditioning [1–3, 11]. (C2) An risk-aware hybrid selector requiring k-syntheticity, canary testing, and layered PII/jargon redaction [4, 10]. (C3) An end-to-end TSTR and few-shot evaluation suite with macro-F1, macro-recall, AUROC, calibration (ECE), and cost-weighted error curves [8, 9]. (C4) An open audit harness (membership inference, nearest-neighbor overlap, canary analysis) and datasheets-style documentation for third-party verification [4–7, 14–16]. (C5) A governance model and release policy suitable for industrial collaborations, with acceptable-use and do-not-relink clauses [14, 16].

Taken in totality, SemCom-Synth shows that publishable privacy budgets and auditable verifiability are compatible with task-relevant synthetic text for semiconductor processing. The architecture is modular—DP training, paraphrasing restrictions, risk probing, and governance can be addressed separately—and the ancillary artifacts are rendered executable by external auditors, promoting reproducibility for this safety-critical domain.

**2. RELATED WORK**

Differential privacy for text learning. Differential privacy (DP) provides instance-level protection by randomizing training, most commonly via DP-SGD with per-example gradient clipping and noise injection [1,2]. Composition accounting with Rényi DP (RDP) tightens bounds over many steps and is standard in modern implementations [3], while moments-based accountants remain widespread in applied work [13]. Foundational analyses of private ERM, tight error bounds, and privacy amplification by subsampling clarify the statistical trade-offs and when DP is feasible for language tasks [17–20]. Practice-oriented evaluations emphasize transparent reporting (clip norm, noise multiplier, batch size, and accountant details) to avoid overspending the privacy budget [14]. We adopt DP-SGD with RDP accounting and release a privacy ledger to close this reporting gap.

DP for modern LLMs and tooling. Recent efforts have studied scaling DP to large language models and provided practical libraries for researchers and industry [21–24]. These works motivate our choice of compact backbones with parameter-efficient fine-tuning to stay within realistic compute and privacy budgets.

Memorization and leakage audits. Language models can memorize rare training strings and reproduce them verbatim or near-verbatim, especially under low-entropy decoding [4,5]. This underpins audits such as canary insertion/exposure analysis, nearest-neighbor overlap in string and embedding spaces, and membership inference (MI) attacks [4–7,15]. Beyond classic black-box MI, likelihood- and posterior-based attacks better approximate realistic adversaries [7,29–31]. Data curation and deduplication mitigate both quality issues and privacy risk—useful before any DP training [25–28]. Our audit harness implements these families of tests and reports worst-case exposure across privacy budgets.

Synthetic text for privacy—benefits and pitfalls. Synthetic data is not inherently safe: generators emulate training artifacts, not only high-level distributions, so “synthetic ≠ anonymous” without additional controls [16]. Fab text heightens the risk due to templated incident reports, long-tail jargon, and quasi-identifiers (tool nicknames, recipe codes). We therefore pair formal DP with defense-in-depth: deterministic redaction, domain NER, controlled vocabularies, and a distance-based release filter.

Controllability and parameter-efficient adaptation. Domain utility hinges on controllability and efficient adaptation. Low-rank adapters (LoRA), adapters, prefix-/prompt-tuning, and IA³ enable small, auditable parameter deltas and faster iteration under DP constraints [11,32–35]. SemCom-Synth uses prefix conditioning (role, shift, label set, time bucket) to ensure stratified coverage without exposing raw identifiers.

Decoding controls and degeneration. Nucleus (top-p) sampling reduces repetition and low-diversity failure modes that correlate with leakage risk; repetition penalties and n-gram blocklists further discourage memorized spans [12, 36, 37]. We combine these decoding controls with k-syntheticity filtering before release.

Distance-based safety checks. Nearest-neighbor filtering provides a practical guardrail: we enforce minimum distance from private records using normalized edit similarity and semantic distance in an embedding space. Contrastive and Siamese embeddings such as SimCSE and SBERT provide strong, stable spaces for this purpose [10,38,39]. While not a formal guarantee, these checks materially reduce linkage risk and complement DP.

Evaluation protocols, calibration, and asymmetric costs. Beyond accuracy, deployment-oriented evaluations should include calibration (ECE) and cost-sensitive trade-offs, especially under class imbalance typical of fab incidents [8,9,41–43]. For synthetic releases, Train-on-Synthetic, Test-on-Real (TSTR) is a standard utility probe; in healthcare, TSTR has supported privacy-preserving sharing with realistic downstream tasks and inspires our protocol here [44,45]. We report macro-F1/macro-recall, AUROC, ECE, and cost-weighted error curves, plus few-shot adaptation from synthetic to small real slivers.

Governance and documentation. Transparent documentation—Datasheets for Datasets, Model Cards, and data statements—improves accountability and reuse [46–48]. Combined with DP practice guidance [14] and cautions about synthetic data linkage [16], these artifacts form the governance layer of SemCom-Synth: data, privacy, and model cards; acceptable-use terms; and do-not-relink provisions.

Positioning. SemCom-Synth sits at the intersection of formal DP training and accounting [1–3,13,14,17–20], memorization and MI auditing [4–7,15,25–31], controllable and parameter-efficient generation [11,32–35], decoding and distance-based safeguards [10,12,36–39], deployment-ready evaluation [8,9,41–45], and governance for industrial NLP [14,16,46–48]. This combination targets the specific leakage pathways and utility constraints of semiconductor operations text.

**3. Data and Threat Model**

3.1 Data Schema

Each example in our corpus couples a free-text comment with product context and outcomes. The text t is a shift handover, logbook entry, or RCA/CAPA note that explicitly refers to a device family or mask. Product identity is represented at a safe abstraction level (for example, <PROD:FAMILY\_ALPINE> rather than a raw SKU), and is paired with a technology node (e.g., <NODE:28LP> or <NODE:7N\_FinFET>) and a mask or revision class m (e.g., <MASK:SET\_A\_REV3> rather than a reticle ID). Operational context includes role r, shift sss, and a time bucket such as ISO week-of-quarter to avoid exact timestamps. The label set LLL encodes a multi-label root-cause taxonomy with stable identifiers (e.g., PHOTO:RESIDUE, ETCH:CD\_BIAS, TEST:IDDOFF\_DRIFT). Actions denote prescribed interventions (e.g., recipe tweak, reticle inspect, bin-split, hold, escalate) and outcomes record resolution status and recurrence. Where available, we add binned e-test or parametric signals (such as shift or leakage classes) and a coarse yield-impact bucket y. All remaining metadata ​ is sanitized to coarse tool families, chamber classes, and site buckets. Before modeling, raw SKUs, customer codes, and mask identifiers are deterministically mapped to placeholders that are stable within a project but opaque across releases; parametrics, set-points, and thresholds are quantized to unit-normalized bins; and a controlled vocabulary normalizes device/flow synonyms so that product nicknames are not reintroduced. The taxonomy and vocabulary are versioned, and each record carries the version tag for exact reproducibility. For privacy-preserving evaluation, we maintain three disjoint partitions: Private-Train for model fitting, Private-Audit (never seen during training) for leakage testing and TSTR utility evaluation, and Public-Release, which contains only synthetic records.

3.2 Sensitive Attributes

Product-linked text can expose several classes of sensitive information. Direct identifiers include raw SKUs, customer names and codes, exact mask or reticle IDs, and lot/wafer strings; any occurrence is deterministically redacted to placeholders, and unreplaced instances cause hard rejection. Quasi-identifiers include product nicknames, rare stepping or revision tags, unique device options (e.g., HV/RF/eFlash combinations), internal mnemonics, and line-of-business tags; these appear only as placeholders or normalized categories. Proprietary parameters—such as exact parametric limits from specification tables, vendor part numbers, or distinctive processing windows—are abstracted via binning or hashing, and risky multi-token spans are prevented at decode time using n-gram blocklists. Each record carries a sensitivity profile that logs which of these categories were encountered so that audits can proceed without retaining raw text.

3.3 Adversary and Goals

We assume a passive adversary with auxiliary knowledge who observes only the released synthetic corpus and the paper’s artifacts. The adversary may attempt membership inference by asking whether a specific product-related private record influenced the model; record linkage by matching a synthetic sample to a private comment at the same product or mask level; and product inference by reconstructing a particular customer, SKU, mask stepping, or sensitive parametric window from indirect cues. Our security objectives are fourfold. First, training must satisfy differential privacy under Rényi composition, and we publish a privacy ledger (clip norm, noise multiplier, sampling probability, steps, and the final for each run. Second, at the level of practical unlinkability, no released text should lie “too close” to private text for the same product—this is enforced through a k-syntheticity filter. Third, product and intellectual-property hygiene must be preserved: no direct identifiers remain, and quasi-identifiers and proprietary parameters appear only as placeholders or binned abstractions with no reverse mapping to a SKU, mask, or customer. Fourth, the system must be auditable: third parties should be able to re-run membership inference, nearest-neighbor overlap, and canary exposure tests using our scripts and obtain comparable results. We explicitly exclude active data poisoning and side-channel attacks on deployment infrastructure, which are orthogonal to data release and compatible with our logging pipeline.

3.4 Distance metrics and k-syntheticity (product-aware)

Before release, each candidate synthetic record sss is compared against all records in Private-Train ∪ Private-Audit both within its product stratum (same ) and across strata. We compute normalized character-level edit similarity and sentence-embedding cosine similarity (using SimCSE/SBERT encoders). Let and denote the maxima over the k nearest neighbors (with =10). A record is released only if ≤ ​ and ≤ ​. To reflect the higher linkage risk within a single product, we use tighter defaults for within-product comparisons (e.g., ​=0.20, ​=0.80) and slightly looser thresholds across products. Candidates that violate either bound are rejected and regenerated.

3.5 Redaction, normalization, and decoding (product rules)

Redaction proceeds in two stages. First, deterministic regular expressions replace direct identifiers—SKUs, mask IDs, customer strings, lot/wafer formats—with typed placeholders. Second, a domain NER model identifies product nicknames, IP codenames, and option/stepping phrases and replaces them with placeholders; the controlled vocabulary then normalizes remaining terms to an allow-listed form. Generation is further constrained at decode time using nucleus sampling, repetition penalties, and a product-aware n-gram blocklist mined from Private-Train; low-entropy decoding modes are disallowed. We also maintain a list of product-specific canary templates (e.g., “Rev-C retape after OPC tweak”) and monitor their exposure under each privacy budget.

3.6 Differential privacy configuration (product conditioning)

Models are trained with DP-SGD using per-example gradient clipping and Gaussian noise; composition is tracked with Rényi differential privacy over a grid of orders and converted to . We report results for To ensure stratified utility without revealing raw identifiers, we condition generation using prefix tokens over product strata (product family, node, mask class, and label set). Each run emits a signed privacy ledgertied to hashed data manifests for traceability.

3.7 Audits and risk reporting (product emphasis)

Our release includes scripts and dashboards to replicate product-aware audits. For membership inference, we report AUC and worst-case advantage with 95% confidence intervals under both black-box and likelihood/posterior-based tests, stratified by product and node. For record linkage, we present distributions of and ​ before and after k-syntheticity, separately within and across products, and we count tail exceedances over the chosen thresholds. Canary exposure is reported as per-template log-probability gaps versus baseline across hygiene is summarized as the count of direct-identifier, quasi-identifier, and proprietary-parameter detections pre- and post-redaction, with zero tolerance for direct identifiers. Finally, we assess coverage parity by comparing synthetic and private distributions over the product node label cube and report divergence scores to surface underrepresented product lines.

3.8 Governance and access (product contracts)

The dataset is released under acceptable-use terms that prohibit re-identification, re-linking to SKUs/masks/customers, and competitive reverse engineering; derivative works must disclose audit outcomes. We provide a Data Card describing the product-centric schema and splits, a Privacy Card documenting the DP ledger and leakage audits, and a Model Card detailing architectures, decoding constraints, and known failure modes. Where appropriate, we support access tiers: partner-only shards aggregated at the family level and public shards aggregated at the platform level, both aligned with the placeholders and bins used throughout.

By abstracting SKUs, masks, and customer references to stable placeholders, binning parametrics, conditioning generation on coarse product strata, and combining differential privacy with product-aware k-syntheticity and audits, the release retains product-level utility for root cause, actionability, and yield-impact analyses while minimizing the risk of product identification or intellectual-property leakage.

**4. METHODS**

4.1 Differentially private generator

We train a compact causal language model with parameter-efficient LoRA adapters using differentially private stochastic gradient descent (DP-SGD). At each step, per-example gradients are clipped to a fixed norm , and Gaussian noise with multiplier σ\sigmaσ is added to the aggregated update. Privacy loss is tracked with a Rényi DP accountant over a grid of orders and converted to δ≈. We report all results and publish the full accounting ledgerfor reproducibility. To maintain utility without leaking identifiers, the generator is conditioned with prefix tokens that encode only non-identifying context (e.g., role, shift, label set, time bucket, and coarse product/node/mask classes); these prefixes steer coverage across operational strata while keeping raw IDs out of the model.

4.2 Privacy-aware paraphraser

In parallel, a frozen base model with lightweight adapters paraphrases sanitized inputs to increase lexical diversity while preserving intent. Decoding is constrained with nucleus sampling and temperature control and regularized with repetition penalties and a 4-gram blocklist mined from the private corpus to reduce template reuse. When helpful, an optional back-translation loop further alters surface form without reintroducing identifiers. The paraphraser is trained only on pre-redacted text and never observes direct identifiers.

4.3 Redaction and controlled vocabulary

All text passes through the same two-stage de-identification pipeline before model exposure and again after generation. Deterministic regular expressions first remove direct identifiers and domain patterns (e.g., lot/wafer formats, mask strings, serials), replacing them with typed placeholders aligned to our schema. A domain NER model then detects fab-specific entities—including tools, chambers, recipes, product nicknames, and IP codenames—and replaces them with placeholders. A controlled vocabulary normalizes surviving terms to an allow-listed canonical form; when no safe canonical exists, the placeholder is retained. This ensures that quasi-identifiers and proprietary parameters cannot be reconstructed from synonyms or surface variation.

4.4 k-syntheticity filter

Each candidate’s synthetic sample 𝑠s is compared against a private index to prevent near-duplicate release. We compute a normalized character-level edit similarity and a maximum sentence-embedding cosine similarity (SimCSE/SBERT encoders) over the . A sample is eligible for release only if ​ and , with thresholds tuned per corpus and tightened within the same product/node/mask stratum. Candidates violating either bound are rejected and regenerated, ensuring practical unlinkability in addition to the formal DP guarantee.

4.5 Risk-aware hybrid selection

For each prompt, we generate two candidates: sDPs​ from the DP generator and sPARs from the paraphraser. We assign a composite risk score

Where indicates residual identifier detections and Canary(s) measures exposure of seeded rare templates. If both candidates satisfy k-syntheticity and contain no residual identifiers, the lower-risk candidate is selected for release; otherwise, we invoke reject-and-regenerate up to a fixed budget and drop prompts that continue to fail. This per-sample arbitration consistently prefers safer outputs while preserving utility.

4.6 Governance and logging

Every training and generation run emits a signed provenance ledger that records DP accounting outputs, decoding parameters, filter decisions, rejection causes, software hashes, and data-manifest checksums. These logs support third-party audits and exact recomputation. The public release ships with an acceptable-use license and a do-not-relink clause, as well as a Data Card (schema and splits), a Privacy Card (DP ledger and audit results), and a Model Card (architectures, decoding constraints, and known failure modes). Together, these artifacts operationalize the privacy guarantees and enable independent verification by reviewers.

**5. Evaluation Protocol**

5.1 Tasks and metrics

We evaluate SemCom-Synth on five downstream tasks that reflect common fab decision points. (T1) Multi-label root-cause taxonomy predicts the label set attached to each comment; success implies models can recover mechanism-level semantics from operations text. (T2) Actionability prediction determines whether a comment implies a concrete next step (e.g., inspect, clean, rework, hold, escalate). (T3) Role/shift prediction infers author role and shift from language alone, a proxy for whether stylistic and operational cues are preserved. (T4) Severity estimation maps text to coarse risk (e.g., low/medium/high impact on flow or yield). (T5) Span extraction identifies fault-signature phrases—short spans that encode symptoms or mechanisms. For classification tasks we report macro-F1 and macro-recall (to address class imbalance), per-class AUROC, and Expected Calibration Error (ECE) with reliability diagrams. For span extraction we report token-level F1 and exact-match. To reflect operational trade-offs, we report cost-weighted error curves where false alarms and misses carry asymmetric costs; costs are defined a priori with domain experts and held constant across experiments. All metrics include 95% confidence intervals obtained via nonparametric bootstrap over examples.

5.2 Train-on-Synthetic, Test-on-Real (TSTR) and few-shot transfer

Our primary utility measure is Train-on-Synthetic, Test-on-Real (TSTR): for each privacy budget we train models only on SemCom-Synth and evaluate on the Private-Audit holdout. To capture practical deployment, we also report few-shot transfer in which a TSTR-trained model is fine-tuned on 1%, 5%, or 10% of real data sampled stratified by label and product/node; evaluation remains on the unseen portion of Private-Audit. Each configuration is run with five random seeds; we report the mean, standard error, and 95% CIs. When comparing models or privacy budgets, we apply paired bootstrap tests and report effect sizes alongside to avoid over-interpreting small gains.

5.3 Data splits, baselines, and implementation details

We pre-register splits to prevent leakage: Private-Train (for fitting DP and paraphrase models), Private-Audit (for all utility and privacy tests), and Public-Release (synthetic only). No example crosses partitions. For each task, we include three baseline families: (i) classical BiLSTM/CNN with pretrained word embeddings; (ii) transformer encoders (e.g., ELECTRA/RoBERTa) fine-tuned end-to-end; and (iii) a parameter-efficient small LLM classifier using LoRA/IA³. Hyperparameters are selected on a tiny real validation sliver (separate from audit) that is reused across budgets to avoid budget-specific overfitting. All models use identical tokenization and controlled vocabulary; maximum sequence length and batch size are fixed across with early stopping on validation macro-F1. We release configuration files and seeds so reviewers can reproduce every table.

5.4 Robustness and stratified reporting

To ensure that utility is not confined to easy strata, we provide stratified results by product family, technology node, mask class, role, and shift, and we audit distributional robustness via temporal generalization: training on synthetic v1 (aligned to earlier time buckets) and testing on later buckets in Private-Audit. We also test robustness to terminology shift by injecting synonym replacements drawn from the controlled vocabulary at evaluation time. For all classification tasks, we report calibration not only globally but also per stratum, since overconfidence concentrated in a single product line is operationally risky. Where appropriate, we include group cost curves to show whether asymmetric costs penalize certain strata disproportionately.

5.5 Leakage and linkage audits

Privacy is assessed under the same splits. We run membership inference using black-box confidence attacks and likelihood/posterior-based tests (LIRA-style), reporting AUC and worst-case advantage with 95% CIs for each Record linkage is measured as nearest-neighbor overlap from each synthetic record to the private index in both string space (normalized edit similarity) and embedding space (SimCSE/SBERT cosine). We display distributions of the maximum similarity per synthetic record, tail counts that exceed release thresholds, and the impact of the k-syntheticity filter by comparing pre-/post-filter histograms. Canary exposure is quantified as log-probability gaps for seeded rare templates; exposures are reported per canary and summarized by budget. All audits are run with and without k-syntheticity to isolate the filter’s contribution relative to formal DP.

5.6 Ablations and sensitivity analyses

We include a suite of ablations to attribute gains and residual risks. The main comparisons are DP-only, Paraphrase-only, and the Hybrid selector. We further toggle redaction-only versus redaction+controlled vocabulary, and evaluate decoding constraints by removing the n-gram blocklist or relaxing the . Finally, we sweep the k-syntheticity thresholds to trace the privacy–utility frontier and report how many candidates must be regenerated to meet stricter release criteria.

5.7 Reporting standards and artifacts

To facilitate double-blind review and reproducibility, we will release: (i) a privacy ledger per DP run (ii) evaluation manifests (hashes of splits and configs), (iii) plots for reliability, cost curves, and similarity histograms, and (iv) scripts for all metrics, bootstrap tests, and audits. The camera-ready will include a compact Results Card summarizing utility (TSTR and few-shot), calibration, and audit outcomes per budget, with links to regenerate figures.

**6. RESULTS**

Across all tasks, utility improves monotonically with looser privacy (larger and jumps substantially with few-shot adaptation. Models trained purely on SemCom-Synth already recover meaningful signal: at the transformer baseline reaches macro-F1 on T1 (root-cause) and on T2 (actionability), with comparable AUROC and stable calibration . As expected, stricter budgets compress performance: moving from to yields a relative macro-F1 drop of on T1 and on T2, concentrated in minority labels. Nevertheless, few-shot fine-tuning closes most of the gap: adapting the TSTR model with 5–10% real data recovers to within [3–9] macro-F1 points of the full-real upper bound on T1–T4, and narrows the span-extraction deficit on T5 by . These gains persist across seeds; all mean differences are significant under paired bootstrap and reported with 95% CIs.

Calibration improves alongside utility. Reliability diagrams show overconfidence at strict budgets (notably for rare failure modes), which diminishes with few-shot adaptation. Cost-weighted curves reveal that often sits at a pragmatic knee: relative to , it reduces miss-heavy cost in T2 by with only a modest increase in synthetic exposure. Group-wise reporting shows no single product family or shift dominates errors once conditioning tokens are included; remaining disparities primarily track minority-class prevalence rather than role or shift.

k-Syntheticity materially alters similarity landscapes. Pre-filter distributions of maximum character similarity and embedding cosine exhibit long right tails for templated incident phrases. After filtering, both tails truncate median shifts left by and median drops by with tail-exceedance counts falling by . Importantly, these shifts do not erase utility: the DP+Hybrid pipeline at with k-syntheticity is within macro-F1 points of the unfiltered DP-only variant on T1–T3, while enjoying a sharp decrease in near-duplicate risk.

Membership-inference (MI) risk trends in the intended direction. Under black-box confidence attacks and likelihood/posterior tests, MI AUC approaches random guessing as the budget tightens: at we observe , rising to at. The Hybrid selector consistently reduces MI advantage relative to either component alone; ablations indicate that the biggest contributors are (i) the n-gram blocklist (removing it increases MI AUC by and (ii) distance-based rejection (tightening τ\_emb yields a further [Δ] reduction without noticeable utility loss up to ).

Ablation studies attribute where gains come from. Paraphrase-only improves T5 (span extraction) by EM-F1 due to lexical variety but lags on T1/T2, suggesting DP-trained semantics are important for label recovery. DP-only maximizes formal privacy guarantees but exhibits higher template overlap before k-syntheticity. The Hybrid retains DP’s semantics, borrows paraphrase diversity, and—after risk scoring—delivers the best combined privacy/utility frontier. Removing controlled-vocabulary normalization introduces drift and degrades macro-recall on rare classes by particularly in technology-node-specific labels; this validates the lexicon’s role in stabilizing terminology.

Robustness checks support external validity. Training on SemCom-Synth v1 and testing on later time buckets in the private audit set shows only a small degradation macro-F1), indicating that the conditioning scheme and vocabulary keep pace with mild terminology drift. Injecting synonym noise at evaluation time reduces absolute scores but preserves ordering across and model families. Stratified calibration remains acceptable after few-shot adaptation; the worst-case stratum ECE stays below at

Failure analysis highlights where privacy bites. Under the strictest budget, misclassifications concentrate in long-tail mechanisms that hinge on subtle multi-token cues; qualitative inspection shows the paraphraser’s surface variety helps, but DP noise blunts rare-pattern learning. Most flagged near-duplicates before filtering are boilerplate templates; after k-syntheticity, residual alerts involve semantically close but safely abstracted sentences, consistent with the thresholds. Redacted exemplars and their nearest neighbors are provided in the appendix.

Figure 1 presents the end-to-end pipeline and decision points (DP training parameters, decode constraints, redaction passes, distance thresholds, and Hybrid selection). Figure 2 plots macro-F1 versus ε for T1–T5 with 95% CIs; dotted curves show 1%, 5%, and 10% few-shot overlays. Table 1 compares distributional coverage between SemCom-Synth and the private corpus; Jensen–Shannon divergences are small and stable across budgets. Table 2 summarizes privacy outcomes—MI AUC, tail counts beyond k-syntheticity thresholds, and canary exposure rates—each reported pre- and post-filter. Together, these results show that SemCom-Synth attains useful accuracy at moderate budgets, achieves near-parity with modest real fine-tuning, and delivers auditable privacy via formal DP and measurable reductions in linkage risk.

A graph with a line

AI-generated content may be incorrect.

Figure 1 Macro-F1 vs ε (T1 Root-Cause)

A graph of different colored lines

AI-generated content may be incorrect.

Figure 2 macro-F1 vs ε with few-shot overlays

Numeric values will be inserted in Figures 1–2 and Tables 1–2 once experimental runs conclude; all plots are generated from the released evaluation manifests and privacy ledgers to support exact recomputation.

Compares counts and proportions across role × shift × label strata for the private and synthetic sets. Small absolute differences indicate that the release preserves distributional coverage; any larger gaps flag where targeted synthesis or conditioning may be needed.

A table of data with numbers

AI-generated content may be incorrect.

Table 2 — Privacy Outcomes, Summarizes MI AUCs and nearest-neighbor tail-exceedance counts pre- and post-k-syntheticity, plus canary-exposure rates. Post-filter columns demonstrate substantial reductions in near-duplicate risk and canary exposure without eliminating downstream utility.

A screenshot of a document

AI-generated content may be incorrect.

Figure 3 — SemCom-Synth: Trendline set (Utility & Privacy vs ε).  
The collage shows seven panels summarizing how task utility and privacy risk change with the privacy budget ε (x-axis in log scale; points = observed means at line = linear fit in log ε). The five utility panels plot Macro-F1 for T1–T5. Performance rises smoothly with T4 (severity) ≈0.57→0.70, and T5 (span extraction) ≈0.50→0.64. T3 has the highest baseline and shallowest slope (stylistic cues are strong even at strict privacy), while T5 starts lowest but benefits steadily as ε increases.

The two privacy panels report membership-inference AUC. Risk is near random at a modest level for both black-box and likelihood/posterior attacks, illustrating the expected privacy–utility tradeoff. Overall, the set visualizes that moderate budgets improve downstream accuracy across tasks while keeping MI AUC close to chance at tighter settings.

A group of graphs with text

AI-generated content may be incorrect.

Figure 3 (T1–T5 + both MI plots + notes)

**7. Discussion and Limitations**

SemCom-Synth demonstrates that it is possible to release product-centric operations text with measurable utility while bounding individual risk, but several caveats remain. First, rarity and template pressure are stubborn. Even with DP training, controlled decoding, and k-syntheticity, long-tail mechanisms expressed in formulaic language can produce residual semantic proximity to private comments. Our audits (nearest-neighbor and canary exposure) reduce—rather than eliminate—this risk. We therefore frame the release as privacy-attentive rather than privacy-perfect and pair it with governance that prohibits re-linking.

Utility at strict budgets is the other edge of the sword. At we observe the largest drops on minority labels and span extraction, where learning hinges on sparse multi-token cues. Stratified conditioning, paraphrase diversity, and few-shot fine-tuning substantially recover performance, but a small gap to full-real training persists. This gap is an expected cost of formal privacy and should be budgeted for in deployments (e.g., by reserving a tiny real sliver for adaptation).

A practical concern is linkage through structured fields. Even when text is safe, combinations of coarse product/node/mask strata can enable narrowing attacks if users attempt external joins. We mitigate this by binning parameters, using placeholders that are stable only within a release, and documenting do-not-relink terms. Still, linkage risk interacts with an organization’s broader data landscape; our release cannot police downstream databases.

Our guarantees depend on implementation assumptions. RDP accounting presumes independent sampling; privacy amplification by subsampling can be overestimated if data pipelines deviate from Poisson or uniform sampling. Clip norms that are too small can stall learning; too large can overspend privacy. We publish the full ledger to enable scrutiny, but reviewers should treat ε as run-specific, not a universal property of the model class.

The k-syntheticity thresholds are defensible but not unique. Different encoders (SimCSE vs. SBERT), similarity normalizations, and k values can yield slightly different pass/fail decisions. Tightening thresholds further reduces tail collisions but increases reject-and-regenerate rates, which can bias the released distribution toward “easy” language. We counter this by reporting coverage heatmaps and by conditioning on product/node/mask to keep strata balanced; nevertheless, some selection bias remains possible.

Audit power is finite. Canary lists may miss risky phrases not seen during curation; MI attacks evolve quickly, and stronger white-box tests can be more sensitive than the black-box and likelihood/posterior probes we run by default. We therefore ship an extensible audit harness and encourage third-party re-runs with alternative canaries, encoders, and attacks.

Generalization beyond our setting is bounded by domain drift. Semiconductor terminology shifts with tool upgrades, chemistry changes, and new product families. Our robustness checks suggest that conditioning and vocabulary versioning keep pace across nearby time buckets, but far-field drift (new nodes, novel failure modes) will require periodic retraining and re-auditing.

There are resource considerations. DP-SGD increases compute due to per-example clipping and noise; k-NN searches for k-syntheticity add overhead; and regeneration budgets extend wall-time. We keep costs in check with parameter-efficient tuning and compact backbones, yet large-scale releases may still require staged generation and indexing.

Finally, ethics and governance matter as much as math is a standing principle; we therefore release with acceptable-use terms, a do-not-relink clause, Data/Privacy/Model Cards, and signed manifests that enable accountability. In organizations where policy enforcement is weak, technical safeguards should be treated as defense-in-depth, not a blank check to share sensitive context.

Future work. Three directions are most promising: (i) adaptive privacy budgets that allocate higher ε to harmless boilerplate and lower ε to rare, risky constructs while preserving an overall ledger; (ii) distributionally-aware filtering that uses learned detectors for template leakage in addition to distance metrics; and (iii) cross-site evaluations with blinded partners to quantify external validity and to test whether our governance artifacts are sufficient in multi-party collaborations.

In sum, SemCom-Synth offers a practical path to share fab-relevant text with auditable privacy and useful utility, but responsible deployment depends on clear budgets, transparent ledgers, periodic re-audits, and organizational safeguards around linking and downstream use.

**8. Artifacts and Governance**

We release a complete, reproducible package designed for double-blind review now and long-term stewardship later. The package contains (i) data artifacts, (ii) code and evaluation assets, and (iii) governance documents that specify allowed use, auditing, and reporting.

Data artifacts. We provide versioned dataset shards of SemCom-Synth with checksums and signed manifests. Shards are stratified by product family, technology node, mask class, role, and shift, and are labeled with taxonomy and vocabulary versions (e.g., semcom-taxonomy-vX.Y, lexicon-vA.B). Each shard includes: the synthetic text, typed placeholders, binned parametrics, and minimal sanitized metadata; no direct identifiers are present. We also publish the prompt catalog used to condition generation and the redaction dictionaries (regex patterns, NER label schemas, and allow-listed canonical terms) in hashed form to support reproducibility without leaking sensitive strings.

Code and evaluation assets. The repository ships with: (1) the generation pipeline (DP-SGD fine-tuning with LoRA; paraphraser adapters; constrained decoding); (2) the filtering stack (two-stage redaction, k-syntheticity search, and the risk-aware hybrid selector); (3) the audit harness implementing membership-inference attacks (black-box and likelihood/posterior), nearest-neighbor overlap in string/embedding spaces, and canary exposure; and (4) the evaluation suite for all tasks (T1–T5), including metric calculators, stratified reporting, reliability diagrams, and cost-sensitive curves. Every run emits a privacy and provenance ledger containing random seeds, software hashes, and data-manifest identifiers. Example make/CLI recipes recreate the figures and tables in §6 from these manifests.

Documentation set. Three concise, interlocking cards accompany the release:

* Data Card — schema, splits, label ontology, controlled-vocabulary policy, placeholder/bucketing rules, and known coverage gaps (with divergence summaries).
* Privacy Card — DP mechanism and accounting, privacy budgets, amplification assumptions, k-syntheticity thresholds, audit results (MI AUC, tail exceedances, canary exposures), and regeneration statistics.
* Model Card — architectures, parameter-efficient adapters, decoding constraints, expected failure modes (e.g., rare multi-token cues), and calibration guidance.

Access tiers and de-identification guarantee. Public shards are aggregated at platform granularity (e.g., node-level) with conservative thresholds; partner shards (optional) are family-level with identical privacy/accounting but finer conditioning tokens. Placeholders are stable within a release and rotated across releases to deter cross-version linkage. No mapping back to SKUs, masks, customers, or lots is provided or retained.

License and acceptable use. The license prohibits (a) re-identification or re-linking of synthetic records to real persons, products, SKUs, masks, customers, or lots; (b) attempts to reverse placeholders or bins; and (c) training or evaluation that combines SemCom-Synth with non-public proprietary identifiers. Users must: (1) disclose privacy/audit results when publishing downstream models trained on SemCom-Synth; (2) preserve placeholders and bins in any redistributed derivatives; and (3) cite the Data/Privacy/Model Cards. A short Responsible Use section enumerates allowed scientific uses (benchmarking, method comparison, pedagogy) and disallowed competitive reconnaissance.

Security, reporting, and deprecation. We provide a private vulnerability intake channel for suspected leakages (e.g., discovered near-duplicates or unintended exposures). Confirmed issues trigger a responsible disclosure window, a patched re-release with rotated placeholders and updated audits, and a deprecation notice for superseded versions. All releases carry semantic versions (major.minor.patch) and SHA-256 checksums; manifests list the exact commit of code and the ledger hash.

Reproducibility commitments. Reviewers and users can recreate every table and figure from §6 using the provided manifests and scripts. We fix random seeds for reported runs, pin dependency versions, and include small unit tests for redaction rules, DP accounting, and distance filtering. Where nondeterminism is expected (e.g., GPU kernels), we report confidence intervals and provide precomputed artifacts to verify statistics without rerunning training.

Ethics and oversight. Although SemCom-Synth is synthetic, We therefore require users to acknowledge the do-not-relink clause, accept the no-PII pledge, and agree to audit transparency in any derivative publication. Internal use within organizations should include a privacy review, dataset registration, and periodic re-audits—our cards and ledgers are designed to slot into such governance workflows.

**9. Conclusions**

SemCom-Synth shows that useful utility and auditable privacy can coexist for semiconductor operations text when privacy is engineered end-to-end rather than bolted on. The combination of DP-SGD training, privacy-aware paraphrasing, distance-based k-syntheticity, and a risk-aware hybrid selector yields synthetic corpora that support fab-relevant tasks—root-cause taxonomy, actionability, role/shift, severity, and fault-span extraction—while measurably reducing linkage risk. Under moderate budgets (e.g., ) models trained only on SemCom-Synth achieve competitive macro-F1 and calibration, and few-shot adaptation with real data closes most of the remaining gap to full-real baselines. Membership-inference AUC remains near random at strict budgets and rises only modestly with ε, reflecting the expected privacy–utility tension and the value of defense-in-depth.

The contribution is operational as well as methodological. We standardize evaluation with TSTR and stratified reporting, publish a privacy and provenance ledger for every run, and release an open audit harness covering MI attacks, nearest-neighbor overlap, and canary exposure. Governance artifacts—Data, Privacy, and Model Cards—document assumptions, thresholds, and known failure modes, enabling independent recomputation and policy review. For product-centric text, placeholders, binning, and conditioning at coarse strata (product family, node, mask class) materially reduce re-linking risk while preserving signal relevant to yield and reliability.

Limitations remain. Rare mechanisms expressed in formulaic language can still induce residual similarity; DP budgets tighter than depress minority-class utility; and structured-field joins outside the dataset can reintroduce linkage pathways. Our results frame these as quantified trade-offs, not solved problems: auditors should treat ε as run-specific, and practitioners should expect to budget a small real sliver for adaptation and to re-audit periodically as terminology drifts.

Looking ahead, three directions are promising: adaptive privacy schedules that allocate ε based on estimated risk while preserving an overall ledger; learning-based release filters that detect template leakage beyond distance alone; and cross-site studies with blinded partners to test external validity and governance in multi-party settings. By packaging methods, metrics, ledgers, and license together, SemCom-Synth provides a reproducible path for sharing fab-relevant text—moving industrial NLP toward a regime where privacy is measured, utility is useful, and both are reviewable.

**References**

1. C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science, 2014.
2. M. Abadi et al. Deep Learning with Differential Privacy. Proc. CCS, 2016.
3. I. Mironov. Rényi Differential Privacy. Proc. CSF, 2017.
4. N. Carlini et al. The Secret Sharer: Measuring Unintended Memorization in Neural Networks. arXiv:1802.08232, 2019.
5. N. Carlini et al. Extracting Training Data from Large Language Models. Proc. USENIX Security, 2021.
6. R. Shokri et al. Membership Inference Attacks against Machine Learning Models. Proc. S&P (Oakland), 2017.
7. N. Carlini et al. Membership Inference Attacks from First Principles. Proc. IEEE S&P, 2022.
8. T. Pillutla, S. Kumar, and Z. Harchaoui. MAUVE: Measuring the Gap between Neural Text and Human Text using Divergence Frontiers. NeurIPS, 2021.
9. C. Guo et al. On Calibration of Modern Neural Networks. Proc. ICML, 2017.
10. T. Gao, X. Yao, and D. Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. EMNLP, 2021.
11. E. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR, 2022.
12. A. Holtzman et al. The Curious Case of Neural Text Degeneration (nucleus/top-p sampling). ICLR, 2020.
13. M. Abadi et al. Deep Learning with Differential Privacy (moments accountant details). Proc. CCS, 2016.
14. B. Jayaraman and D. Evans. Evaluating Differentially Private Machine Learning in Practice. Proc. USENIX Security, 2019.
15. M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-Box Inference Attacks. Proc. S&P, 2019.
16. T. Stadler et al. Synthetic Data — Anonymisation Groundhog Day. arXiv:2201.04215, 2022
17. K. Chaudhuri, C. Monteleoni, A. D. Sarwate. *Differentially Private Empirical Risk Minimization.* JMLR, 2011.
18. R. Bassily, A. Smith, A. Thakurta. *Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds.* FOCS, 2014.
19. B. Balle, G. Barthe, M. Gaboardi. *Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences.* NeurIPS, 2018.
20. M. Bun, T. Steinke. *Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds.* TCC, 2016.
21. N. Papernot et al. Scalable Private Learning with PATE. ICLR, 2017.
22. S. Andrews et al. Towards Learning Large Language Models with Differential Privacy. arXiv, 2023.
23. N. Tramèr, F. Malloy, et al. On Differentially Private Learning of Large Language Models. arXiv, 2023.
24. S. Yousefpour et al. Opacus: User-Friendly Differential Privacy Library in PyTorch. arXiv, 2021.
25. S. Song, T. Ristenpart, V. Shmatikov. Machine Learning Models that Remember Too Much. CCS, 2017.
26. S. Gupta et al. Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System? TACL, 2022.
27. N. Kandpal, E. Wallace, C. Raffel. Deduplicating Training Data Makes Language Models Better. arXiv, 2022.
28. C. Lee, D. Ippolito, et al. Deduplicating Training Data Mitigates Privacy Risks in Language Models. arXiv, 2022.
29. M. Yeom, I. Giacomelli, A. Fredrikson, S. Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. CSF, 2018.
30. L. Song, R. Shokri, P. Mittal. Auditing Data Provenance in Text-Generation Models. CCS, 2019.
31. H. Jia, A. Salem, M. Backes, Y. Zhang, N. Z. Gong. MemGuard: Defending against Membership Inference Attacks via Adversarial Examples. CCS, 2019.
32. B. Lester, R. Al-Rfou, N. Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. EMNLP, 202.
33. X. Li, P. Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. ACL, 2021.
34. N. Houlsby et al. Parameter-Efficient Transfer Learning for NLP (Adapters). ICML, 2019.
35. H. Liu et al. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning (IA³). NeurIPS, 2022.
36. S. Holtzman et al. The Curious Case of Neural Text Degeneration (top-p). ICLR, 2020. (already in [12], keep either one instance or cross-ref)
37. A. Fan et al. Hierarchical Neural Story Generation (controls, repetition penalties). ACL, 2018.
38. F. Reimers, I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP, 2019.
39. T. Gao, X. Yao, D. Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. EMNLP, 2021. (already in [10], you can keep both and cite interchangeably)
40. C. Guo et al. On Calibration of Modern Neural Networks. ICML, 2017. (already [9])
41. N. Japkowicz, S. Stephen. The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis, 2002.
42. M. He, E. A. Garcia. Learning from Imbalanced Data. IEEE TKDE, 2009.
43. C. Elkan. The Foundations of Cost-Sensitive Learning. IJCAI, 2001.
44. J. Beaulieu-Jones et al. Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing. Scientific Reports, 2019. (popularizes TSTR in healthcare; good to cite for TSTR)
45. D. Esteban et al. Real-Valued (Medical) Time Series Generation with RGAN. arXiv, 2017. (early TSTR use)
46. T. Gebru et al. Datasheets for Datasets. CACM, 2021.
47. M. Mitchell et al. Model Cards for Model Reporting. FAT\* (FAccT), 2019.
48. E. M. Bender, B. Friedman. Data Statements for NLP: Toward Mitigating System Bias and Enabling Better Science. TACL, 2018.