

**QSyn Finance**

TECHNICAL WHITEPAPER · V1.0

Architecture, Differential Privacy & *Benchmark Methodology*

What a quantitative research and model-risk team needs to evaluate QSyn before a first meeting: how the platform is built, how its privacy guarantees are constructed and tested, and exactly how its performance claims are measured and reproduced.

PRODUCTS

QSyn-Data · QSyn-Risk

AUDIENCE

Quant Research · Model Risk · InfoSec

VERSION

1.0 — September 2026

CONTACT

founders@qsyn.finance

— ABSTRACT

What this document is — and is not.

QSyn Finance is an integrated infrastructure layer for regulated finance. It couples **QSyn-Data**, a differentially-private synthetic-data platform, with **QSyn-Risk**, a GPU-accelerated, quantum-ready risk and optimization engine. The two connect in a closed loop: synthetic data feeds forecasting and risk analytics; the risk engine emits stress scenarios that regenerate conditioned synthetic data. This paper documents the three things a technical evaluator asks for first — the **architecture and closed-loop design**, the **(ϵ , δ)-differential-privacy method and its privacy test battery**, and the **benchmark methodology** behind our performance figures.

READ THIS FIRST — INTELLECTUAL HONESTY

Performance numbers in this paper are stated as **engineering targets** derived from the system design and component-level measurements, not as audited production results across customer workloads. Where a figure is a target, it is labelled as one. Our commitment is operational, not rhetorical: **every pilot ships a live, reproducible benchmark on your workload (or public data) before any number is relied upon.** Section 4 defines exactly how those benchmarks are run and reproduced.

— CONTENTS

01	Architecture & closed-loop design	03
	System context, containers, the closed loop, data contracts, deployment topologies	
02	The (ϵ , δ)-differential-privacy method	06
	Definitions, DP-SGD mechanism, privacy accounting & budget management	
03	Privacy tests & the release gate	08
	Membership inference, distance-to-closest-record, attribute inference, blocking	
04	Benchmark methodology	10
	Baselines, hardware, metric definitions, utility & fidelity, reproducibility protocol	
05	Governance & model-risk alignment	13
	SR 11-7 / OCC 2011-12, auditability, reproducibility	
—	Glossary & references	14

 $\geq 50M$

Monte Carlo paths / sec per GPU node (target)

 $< 30s$

Market VaR latency, 4-GPU node (target)

 ≥ 0.90

Synthetic utility score vs. real (target)

100%

DP budget accounted & enforced

SECTION 01

Architecture & closed-loop design

QSyn is two independently deployable products that share one integration contract. Each is a modular, microservice-compatible system that runs entirely inside the customer's trust boundary — on-prem, in their cloud account, or hybrid. No real data ever leaves customer-controlled infrastructure.

1.1 System context

Three actor classes interact with the platform: **quantitative and data-science users** who upload reference data and consume synthetic datasets and risk results; **risk and compliance officers** who review privacy and model-risk reports and approve releases; and **platform administrators** who manage roles, clusters and policy. All sensitive computation is co-located with the data — the design goal is **zero data exfiltration**.

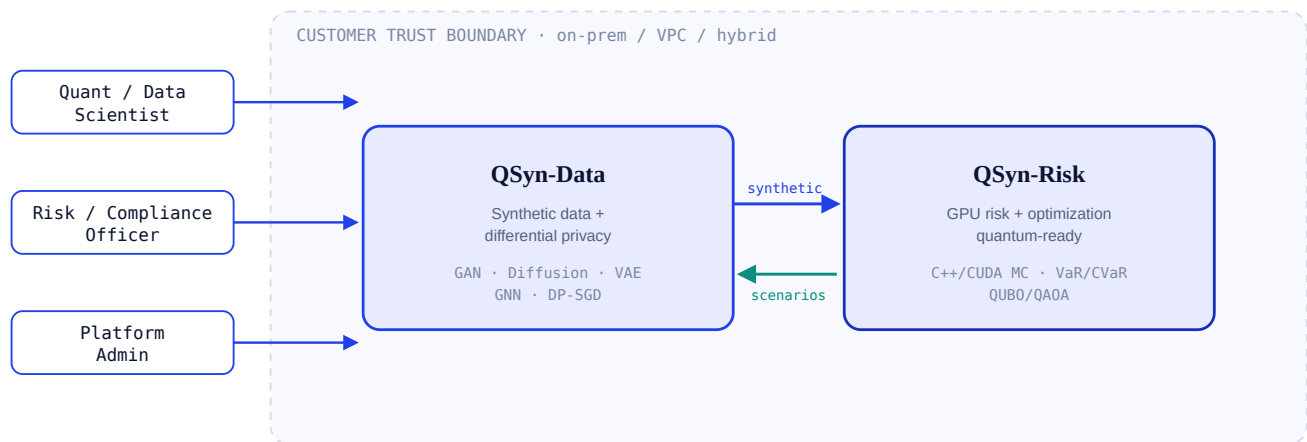


Figure 1. System context. Both products deploy inside the customer boundary; QSyn-Data emits privacy-annotated synthetic datasets, QSyn-Risk returns stress scenarios that condition new synthetic data.

1.2 Containers & technology

Each product decomposes into a small set of services with a clear separation between a stateless API tier, an orchestration tier, and GPU/CPU compute workers. Metadata lives in PostgreSQL; artifacts live in S3-compatible object storage; jobs move over a Redis/Kafka queue.

CONTAINER	QSYN-DATA	QSYN-RISK	STACK
API tier	Dataset, model, policy & report APIs	Risk-job, quantum-optimize & data-integration APIs	FastAPI · REST/WS
Orchestrator	Training/generation job control	Simulation DAG scheduling	Celery/Redis · Ray/Dask
Compute workers	PyTorch generative training & sampling (multi-GPU DDP)	SIMD C++17 + CUDA Monte-Carlo kernels	PyTorch · C++/CUDA
Specialized	DP engine (per-sample clipping, noise, accountant)	Quantum layer (Qiskit/IonQ/D-Wave); scenario engine	Opacus-class · Qiskit
State	Metadata in PostgreSQL · artifacts in S3/GCS/Blob · queue in Redis/Kafka · audit index in OpenSearch/ELK		Postgres · S3
Deploy	Docker images, Kubernetes operators for training/inference/simulation, Terraform IaC, CI/CD		K8s · Terraform

DETERMINISM BY CONSTRUCTION

GPU Monte-Carlo uses a counter-based **Philox** RNG; every job records its seed, code & model versions, dataset IDs, scenario IDs and hardware. Re-running a job with the same seed reproduces bit-identical results — the property model validators and auditors require.

1.3 The closed loop

The architectural thesis is that synthetic data and risk analytics are more valuable joined than separate. The loop has four legs, each an explicit, versioned, auditable hand-off rather than an ad-hoc export:

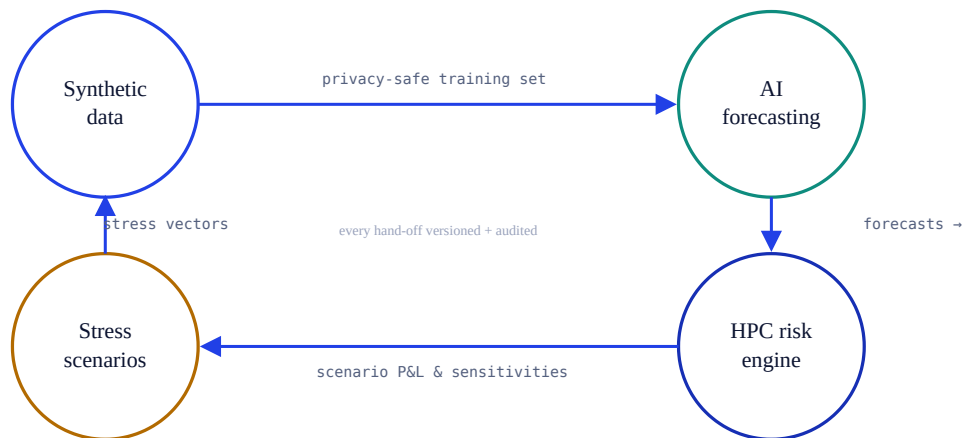


Figure 2. The closed loop. QSyn-Risk submits stress vectors to QSyn-Data (POST `/data/scenario/request`), which returns a stress-conditioned synthetic dataset for the next cycle.

- 1. Synthetic → forecasting/risk.** QSyn-Risk fetches a dataset by ID, validates its privacy and utility metadata, and rejects anything failing the gate (below).
- 2. Risk → scenarios.** The risk engine computes VaR/CVaR/ES, credit and liquidity risk, then derives stress vectors (rate, equity, credit-spread, liquidity shocks).
- 3. Scenarios → stressed synthetic.** Those vectors are posted back to QSyn-Data, which regenerates synthetic data *conditioned* on the stress — extreme-but-plausible states the original history may lack.
- 4. Feedback.** Observed factor sensitivities and correlation breakdowns optionally refine the generative models, so the loop sharpens each cycle.

1.4 The integration contract

The hand-off is governed by an explicit data contract. QSyn-Risk never accepts real data directly; it consumes only QSyn-Data artifacts carrying privacy provenance:

```
// synthetic dataset contract consumed by QSyn-Risk
{
  "synthetic_dataset_id": "UUID",
  "schema": { ... },
  "dp_policy_id": "pol_standard_v3",
  "dp_applied": true,
  "utility_score": 0.94,
  "privacy_risk_level": "LOW",    // LOW | MEDIUM | HIGH
  "source_dataset_ids": [ ... ],
  "trained_model_id": "mdl_ctgan_...",
  "storage_path": "s3://.../synthetic/v3.parquet"
}
```

ENFORCED INGESTION RULES

If `privacy_risk_level == "HIGH"` → QSyn-Risk **refuses ingestion by default**. If `dp_applied == false` → the run is flagged `dp_protected: false` and requires explicit **Risk-Manager acknowledgement**. Utility below the configured floor is likewise rejected. Every decision is written to the immutable audit index.

1.5 Deployment & security posture

TLS 1.3 everywhere; optional mutual-TLS between services; SSO/OIDC; role-based access (risk_analyst · risk_manager · quant_developer · admin). Data at rest is AES-256-GCM encrypted with 90-day key rotation. Deployments scale horizontally: generation scales ~linearly with available GPUs; simulation workers scale out under Ray/Dask. Observability is first-class — Prometheus/Grafana metrics, OpenSearch/ELK logs, PagerDuty alerting — which is also what DORA operational-resilience expectations require.

SECTION 02

The (ϵ, δ) -differential-privacy method

Differential privacy is applied per dataset, as a configurable policy, during model training — not as a cosmetic post-process. It gives a mathematically stated, composable bound on how much any single real record can influence the synthetic output.

2.1 Definition

A randomized mechanism M satisfies **(ϵ, δ) -differential privacy** if, for every pair of datasets D, D' differing in one record, and every measurable set of outputs S :

$$Pr[M(D) \in S] \leq e^{\epsilon} \cdot Pr[M(D') \in S] + \delta$$

Interpreted operationally: ϵ (the privacy loss / budget) bounds how much the presence or absence of one individual can change the distribution of outputs — smaller ϵ is stronger privacy; δ is the small probability that the ϵ -bound fails, conventionally held below the inverse dataset size ($\delta < 1/N$). Because the guarantee is a property of the *mechanism*, it holds against adversaries with arbitrary side information — the property linkage-attack resistance ultimately rests on.

2.2 The mechanism — DP-SGD

Generative models (CTGAN, TVAE, diffusion, transformer and GNN generators) are trained with **differentially-private stochastic gradient descent**. Two operations convert ordinary training into a DP mechanism at each step: per-sample gradient **clipping** bounds any one record's contribution, and calibrated **Gaussian noise** masks it.

$$\tilde{g}_t = (1/B) \left(\sum_{i \in B} \text{clip}(g_i, C) + \mathcal{N}(0, \sigma^2 C^2 I) \right)$$

where $\text{clip}(g, C) = g \cdot \min(1, C / \|g\|_2)$ — the per-sample ℓ_2 clipping norm C bounds sensitivity; σ is the noise multiplier; B is the lot size.

The implementation uses a well-reviewed DP library (Opacus-class) rather than bespoke cryptographic code, with per-layer clipping and a privacy accountant. This is a deliberate risk control: DP guarantees are silently voided by subtle bugs in noise calibration or RNG, so we favour audited primitives and add DP-specific regression tests (Section 3.4).

2.3 Privacy accounting & composition

Training takes many noisy steps, and each spends privacy. A **privacy accountant** (Rényi-DP / moments accountant) composes the per-step loss into a single tight (ϵ, δ) for the released model — far tighter than naïve summation. The accountant makes the budget an explicit, first-class quantity that can be governed:

- **Per-dataset and per-user budgets.** Cumulative ϵ is tracked across generations from the same source. Repeated synthesis *composes* — five datasets from one source do not each get a fresh budget.
- **Hard limits.** Once cumulative ϵ exceeds the policy ceiling, further DP-protected generation is disallowed pending re-approval or re-ingest.
- **Visibility.** Spent and remaining budget appear in the UI and in every exported DP report.

2.4 Policy presets & guardrails

ϵ is not left to untrained intuition. Datasets are generated under one of three **labelled policy presets**, with a documented risk level and recommended use; anything outside the range requires a Compliance/Admin override with a written reason.

PRESET	TYPICAL ϵ (PER RELEASE)	POSTURE	RECOMMENDED USE
Strict low risk	0.1 – 1	Maximal privacy; more noise, lower fidelity	External sharing, cross-border, vendor/OEM
Standard balanced	1 – 4	Balanced privacy / utility	Cross-team internal analytics & model dev
Experimental use with care	4 – 8	Higher fidelity; weaker guarantee	Internal R&D under access control
Blocked by policy: $\epsilon \gg 8$ without documented override — the UI raises a hard warning and requires explicit sign-off, because a very large ϵ yields the <i>appearance</i> of DP with negligible protection.			

WHAT DP DOES AND DOES NOT PROMISE

DP **reduces** the influence of, and the inference risk around, any single record; it does **not** render data non-sensitive or grant unrestricted sharing rights. Every QSyn-Data report states this explicitly, and DP is treated as one layer among several — combined with the empirical privacy tests in Section 3 and governance controls on release.

SECTION 03

Privacy tests & the release gate

Formal DP bounds are necessary but not sufficient. Every synthetic dataset is also subjected to an **empirical privacy battery** that attacks it the way an adversary would, and the results drive an automatic release decision. A dataset can be blocked even when DP was applied.

3.1 The test battery

TEST	WHAT IT MEASURES	PASS CRITERION (TARGET)
Membership inference	Can an attacker decide whether a specific real record was in the training set, from the synthetic output? Reported as attack AUC.	AUC $\approx$ 0.5 (no better than chance)
Distance-to-closest-record (DCR)	Distribution of each synthetic record's distance to its nearest <i>real</i> neighbour — detects copies & near-duplicates.	DCR distribution $\approx$ real-vs-real holdout; no zero-distance matches
Nearest-neighbour distance ratio	Ratio of distance-to-real vs distance-to-synthetic; flags records that sit implausibly close to individuals.	No systematic collapse toward real records
Attribute inference	Can sensitive fields be predicted from quasi-identifiers in the synthetic data beyond a baseline?	Lift over baseline within tolerance
Exact / near-duplicate scan	Direct search for real records reproduced verbatim or within a similarity threshold.	Zero exact matches; near-matches below threshold

3.2 The release gate

Test outputs are compressed into a **privacy risk report** and a single `privacy_risk_level` $\in$ {LOW, MEDIUM, HIGH}. The gate is mechanical:

- **LOW** / **MEDIUM** — released with the report attached to the dataset's lineage.
- **HIGH** — the dataset is marked `blocked = true`; **export is prevented** until a documented manual review overrides it. Downstream, QSyn-Risk independently refuses HIGH datasets, so the control is enforced twice.

For rare or outlier segments — the records most exposed to re-identification — the platform can **suppress or additionally perturb** the tail rather than release it.

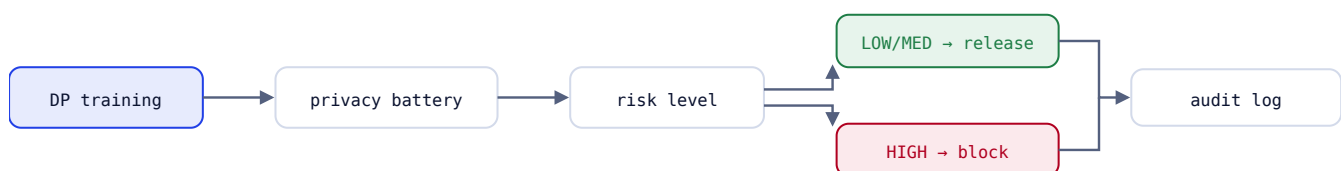


Figure 3. The release gate. Empirical tests, not just the DP parameters, determine whether a dataset can leave the platform.

3.3 Utility is scored alongside privacy

Privacy without utility is trivial (pure noise is perfectly private and useless). Each dataset therefore also carries a utility report so the trade-off is explicit and governable — see Section 4.3 for the metric definitions and the ≥ 0.90 target.

3.4 Assurance of the DP implementation itself

The most dangerous privacy failure is a silent one — a bug that disables noise while the report still claims DP. Controls specific to that risk:

- **Audited primitives** (Opacus-class) over bespoke DP code; code review required for any DP change.
- **DP regression tests** that assert noise scales correctly with ϵ and δ , that clipping is applied as designed, and that a "zero-noise" configuration raises an alarm rather than shipping.
- **Independent review** of the DP implementation by a privacy specialist, and re-validation on any change to the mechanism.

SECTION 04

Benchmark methodology

This is the section a quant reads most carefully, so it is the most precise. A speedup number is meaningless without its baseline, its hardware, and its definition. Here is exactly how we measure, and how you reproduce it.

4.1 Principles

1. **Same problem, same answer.** A speedup only counts if QSyn computes the *same* risk quantity to the *same* statistical accuracy as the baseline — verified by comparing point estimates and confidence intervals, not just runtime.
2. **Named baseline.** Every figure names what it is measured against (below). "Faster" against an unstated baseline is marketing; we don't publish it.
3. **Disclosed hardware.** Every figure names the CPU/GPU, node shape, and precision.
4. **Reproducible by you.** Fixed seeds, pinned versions, and a published notebook (Section 4.4).

4.2 Performance benchmarks & how each is defined

METRIC	TARGET	DEFINITION & CONDITIONS
MC path throughput	$\geq 50\text{M paths/s (GPU)}$ $\geq 8\text{M paths/s (CPU node)}$	Paths generated & evaluated per second for a standard multi-factor GBM/Heston set-up, single 4-GPU node vs. CPU cluster node; steady-state, warm caches; Philox RNG.
Market-VaR latency	$< 30\text{ s}$	Wall-clock for a full portfolio Monte-Carlo VaR/ES job at production path counts on a 4-GPU node, submission→result.
API responsiveness	$p99 < 200\text{ ms}$	Control-plane API latency (job submit/status/fetch), excluding compute time.
Quantum-opt quality	$\leq 5\%$ from optimum	QUBO/QAOA portfolio solution objective vs. the classical optimum on the same instance; a classical baseline is <i>always</i> computed alongside.

Projected speedups vs. a legacy CPU baseline

The figures below are **engineering targets** relative to a single-node, general-purpose CPU implementation (NumPy/pandas-class analytics), the stack most quant teams recognise as today's overnight-batch reality. They are what we design to and instrument for — not a claim about your specific workload, which the pilot measures directly.

WORKLOAD	TARGET SPEEDUP	PRINCIPAL SOURCE OF THE GAIN
Monte-Carlo path generation	20–50×	GPU vectorization, fused kernels, counter-based RNG
CVaR-constrained optimization	10–20×	GPU linear algebra + warm-started solvers
Scenario generation	5–10×	Batched multi-factor simulation
Synthetic model training (GPU)	3–5×	Multi-GPU data-parallel (DDP) training

VARIANCE REDUCTION IS DISCLOSED, NOT HIDDEN

Part of a "faster VaR" can come from needing fewer paths, not just faster paths. Where we use antithetic variates, control variates or quasi-Monte-Carlo (Sobol) sequences, the benchmark **states it and reports the**

effective accuracy, so a like-for-like comparison is always available. We never present a lower-accuracy result as a speedup.

4.3 Utility & fidelity metrics (QSyn-Data)

For synthetic data, "performance" means statistical fidelity at fixed privacy. Every dataset is scored on a battery whose headline is an **Overall Utility Score** (target ≥ 0.90 vs. real), decomposed into:

DIMENSION	METRICS
Marginal fidelity	Per-column Kolmogorov-Smirnov distance; Wasserstein distance; KL divergence
Dependency fidelity	Correlation-matrix difference (Frobenius norm); for time series, autocorrelation & partial-autocorrelation agreement, volatility clustering
Structure fidelity	PCA/UMAP embedding trustworthiness; for graphs, degree distribution, clustering coefficient, connected components, community structure
Task fidelity (TSTR)	<i>Train-on-Synthetic, Test-on-Real</i> — train a downstream model on synthetic, evaluate on real, and report the accuracy/AUC/PD/EL delta vs. training on real. The metric that actually matters for model development.
Tail fidelity	Preservation of rare/extreme events (fat tails) — the states that dominate risk and are hardest to synthesize

4.4 Reproducibility protocol

Every published or pilot benchmark ships with the means to re-run it:

- **Fixed seeds** (Philox) and **pinned software versions**, recorded in the run metadata.
- **Disclosed hardware** — exact GPU/CPU, node shape, precision (FP32/FP64/TF32 where relevant).
- **Open data** — the public benchmark runs on public or synthetic data so you can reproduce it without touching your own; a companion **benchmark notebook** is provided.
- **Cold vs. warm** stated explicitly; steady-state throughput and end-to-end latency reported separately.
- **Accuracy alongside speed** — every runtime is paired with the accuracy achieved, so speed is never bought silently with precision.

THE PILOT IS THE REAL BENCHMARK

The numbers here characterise the engine's design envelope. The number that should decide your evaluation is the one produced **on your workload, on your hardware, in a 4-6 week pilot** — a reproducible result your own quants can re-run and your model-risk team can sign off. That is the deliverable, not this document.

SECTION 05

Governance & model-risk alignment

QSyn is built to clear the model-risk and compliance review, not just the technical demo. The platform treats itself as a model under management and produces the evidence a validation function expects.

5.1 SR 11-7 / OCC 2011-12 alignment

QSyn-Data is registered as both a **model** (the generative engine) and a **data-transformation tool** used in the development and validation of other models. The platform supports the three pillars of the framework:

PILLAR	HOW QSYN SUPPORTS IT
Conceptual soundness	Model cards per generative model; documented architecture, assumptions, known failure modes (mode collapse, tail under-representation), and appropriate/inappropriate uses.
Ongoing monitoring	Tracked utility scores, re-identification metrics and DP budget usage across datasets; alerts on drift and on blocked releases.
Outcomes analysis & independent validation	TSTR downstream deltas; deterministic replay via seed control; independent validation hooks; full change logs and versioning of models, DP policies, scenarios and quantum circuits.

5.2 Auditability & reproducibility

Every risk run and every synthetic-data release writes an immutable audit record — user, dataset IDs, DP policy, scenario, model & code version, seed, hardware, runtime — to an OpenSearch/ELK index, with a cryptographic hash of the results. Any run can be replayed to bit-identical output. This is the substrate that makes both regulatory defensibility and independent validation possible.

5.3 Known limitations, stated plainly

- Synthetic data is **not a substitute for real data** as sole evidence of population behaviour; it is labelled with intended use and limitations, and high-risk uses are gated.
- DP **reduces but does not eliminate** privacy risk; the empirical battery and governance controls exist precisely because the formal bound is necessary but not sufficient.
- Quantum optimization is **experimental**; a classical baseline is always computed, and quantum is positioned as future option value rather than a present-day performance claim.
- Regulatory approval to replace a legacy VaR engine is a customer-side process; QSyn is designed to *run alongside* incumbents during validation rather than demand rip-and-replace.

— GLOSSARY

(ϵ , δ)-DP — differential privacy; ϵ bounds per-record privacy loss, δ the failure probability.

DP-SGD — differentially-private SGD; per-sample gradient clipping + Gaussian noise.

DCR — distance-to-closest-record; synthetic-to-real nearest-neighbour distance.

TSTR — train-on-synthetic, test-on-real; downstream utility measure.

VaR / CVaR / ES — value-at-risk / conditional VaR / expected shortfall.

PD / LGD / EAD — probability of default / loss given default / exposure at default.

QUBO / QAOA — quadratic unconstrained binary optimization / quantum approximate optimization algorithm.

CTGAN / TVAE — conditional-tabular GAN / tabular variational autoencoder.

Philox — counter-based RNG enabling deterministic, parallel simulation.

SR 11-7 — US supervisory guidance on model risk management.

— SELECTED REFERENCES

Dwork & Roth, *The Algorithmic Foundations of Differential Privacy* (2014). · Abadi et al., *Deep Learning with Differential Privacy* (DP-SGD), CCS 2016. · Mironov, *Rényi Differential Privacy*, 2017. · Xu et al., *Modeling Tabular Data using Conditional GAN* (CTGAN), NeurIPS 2019. · Shokri et al., *Membership Inference Attacks against Machine Learning Models*, IEEE S&P 2017. · Board of Governors of the Federal Reserve / OCC, *Supervisory Guidance on Model Risk Management* (SR 11-7 / OCC 2011-12).

NEXT STEP

The companion **benchmark notebook** reproduces the benchmark methodology of Section 4 on synthetic data. To run it against your own workload, book a design-partner pilot at **qsyn.finance** or write to **founders@qsyn.finance**.