Loyal: A Privacy-Preserving Decentralized Intelligence Network

Chris Cherniakov
kcherniakov@colgate.edu

May 26, 2025

Abstract

A purely peer-to-peer network of AI agents would allow artificial intelligence services to be accessed directly without going through centralized providers. Trusted Execution Environments provide part of the solution, but the main benefits are lost if users must still surrender their data to access AI. We propose a solution using agent-to-agent payments for inference. The network consists of specialized agents running on cryptographic processors, verified through hardware attestation that proves both the code and isolation of computations. Client agents discover and pay service agents per inference, chaining together capabilities to complete complex tasks. Developers deploy agents to earn from their models, while users' personal agents handle routing and micropayments automatically. As long as the hardware attestation remains unforgeable and spending limits are enforced through smart contracts, agents can transact without exposing user data or requiring trust. The network itself requires minimal structure. Agents advertise capabilities, client agents route requests to the most suitable providers, and payments flow directly between participants, accepting the hardware attestation as proof of privacy preservation.

1 Introduction

The internet we knew is evolving toward a new operating model. Every safe harbor is polluted with AI-generated slop, thousands of computer hours are spent optimizing user engagement, and every single message or query becomes another data in the centralized surveillance machine. Global industry leaders have built their empires on a Faustian bargain: free services for total data surrender. Users have no choice but to feed these systems their most intimate thoughts, financial data, and personal communications. We trust these corporations won't exploit our secrets. This trust is systematically betrayed.

This paper proposes an open-source multi-layered peer-to-peer network for agentic artificial intelligence. *Client* agents discover and pay *service* agents directly for inference, with AMD SEV-SNP and Intel TDX providing hardware attestation of code integrity and data isolation. The network requires no central coordination: agents advertise capabilities through a decentralized registry, and payments flow per inference through smart contracts.

Don't trust the system. Trust the math.

2 Related Work

Our work draws from research in decentralized systems, privacy-preserving machine learning, and efficient neural architectures. We discuss how existing approaches address subsets of our requirements and position our contributions.

Decentralized AI and computation markets. Decentralized approaches to AI computation have emerged to address centralization concerns. Bittensor [Rao, 2021] implements a peer-to-peer network for collaborative training with blockchain-based incentives, while Golem [Golem Team, 2016] creates markets for general computation. These systems focus on batch processing and lack privacy guarantees. iExec [Claverie et al., 2018] combines blockchain with TEEs but targets general computation rather than AI-specific optimizations. Federated learning systems [McMahan et al., 2017; He et al., 2020; Beutel et al., 2020] keep data distributed but require central coordination and address training rather than inference. In contrast, we target real-time inference with cryptographic privacy guarantees and no central coordinator.

Privacy-preserving inference. Cryptographic techniques for private ML inference fall into three categories. Homomorphic encryption approaches [Gilad-Bachrach et al., 2016; Brutzkus et al., 2019] enable computation on encrypted data but increase inference latency by 4-6 orders of magnitude. Secure multi-party computation [Mohassel and Zhang, 2017; Kumar et al., 2020] splits computation between parties but requires multiple rounds of communication. Hybrid approaches [Mishra et al., 2020; Hao et al., 2022] reduce overhead to 2-3× but remain impractical for interactive applications. TEE-based systems [Tramer and Boneh, 2019; Wang et al., 2021; Chandar et al., 2022] achieve near-native performance through hardware isolation. However, existing TEE approaches assume centralized deployment. We extend hardware-based isolation to decentralized networks, combining attestation with peer-to-peer routing.

Efficient transformer architectures. The quadratic complexity of transformers motivates linear alternatives. Kernel-based approaches [Choromanski et al., 2021; Peng et al., 2021] approximate attention through feature maps, while low-rank methods [Wang et al., 2020; Xiong et al., 2021] project to smaller dimensions. State-space models [Gu et al., 2022; Peng et al., 2023; Sun et al., 2023] reformulate attention as recurrence, achieving O(n) complexity. We look toward new approaches that offer a promising balance of efficiency and performance; for example, linear-attention RWKV models enable constant-memory serving across multiple users — a critical requirement for decentralized deployment that prior work doesn't address. The final architecture will be chosen after head-to-head evaluations of long-context robustness, multi-tenant serving stability, quantization tolerance, and tool-use performance.

Agent composition and communication. Multi-agent frameworks demonstrate task decomposition across specialized models [Richards, 2023; Nakajima, 2023; Wu et al., 2023]. DSPy [Khattab et al., 2023] provides declarative abstractions for LLM pipelines, while Octopus [Chen et al., 2024] introduces functional tokens for efficient tool calling. These systems assume centralized orchestration and single-user contexts. We extend functional tokens to enable decentralized agent discovery and routing without central coordination.

Our system uniquely combines hardware-attested isolation, decentralized architecture without coordinators, cryptocurrency micropayments with sub-second settlement, and linear attention optimized for multi-tenant serving. While individual components exist, their integration into a practical system for privacy-preserving AI inference represents our primary contribution.

3 System Architecture

3.1 Overview

Loyal enables privacy-preserving AI inference through a peer-to-peer network of specialized agents. Users interact with personal *client* agents that discover and pay *service* agents for specific capabilities. All agents execute within hardware-attested Trusted Execution Environments, ensuring cryptographic privacy without trusted intermediaries.

The network consists of three primary components: (1) agents executing in TEEs on distributed nodes, (2) a decentralized capability registry for service discovery, and (3) a blockchain layer for micropayments and spending controls. Agents communicate through encrypted channels established via TEE remote attestation. No central coordinator exists—the network self-organizes through market dynamics and gossip protocols.

3.2 Workflow

Consider a user asking their client agent to "analyze my tax documents and suggest deductions." The interaction proceeds as follows:

1. **Query Processing**: The client agent parses the query and generates embedding vector q representing the required capabilities.

2. **Service Discovery**: The agent queries the decentralized registry, computing cosine similarity between q and advertised capability vectors. It identifies TaxAnalyzer agent A (document parsing) and DeductionOptimizer agent B (tax law expertise).

3. **Attestation Verification**: Before sending data, the client verifies hardware attestations. This proves A runs expected code within a genuine TEE.

4. **Secure Communication**: Agents establish encrypted channels using ephemeral keys from TEE key agreement. User documents encrypt with session keys that never leave hardware boundaries.

5. **Chained Execution**: A processes documents and invokes B through functional tokens, single vocabulary items representing entire function calls. B returns deduction recommendations without ever accessing raw documents.

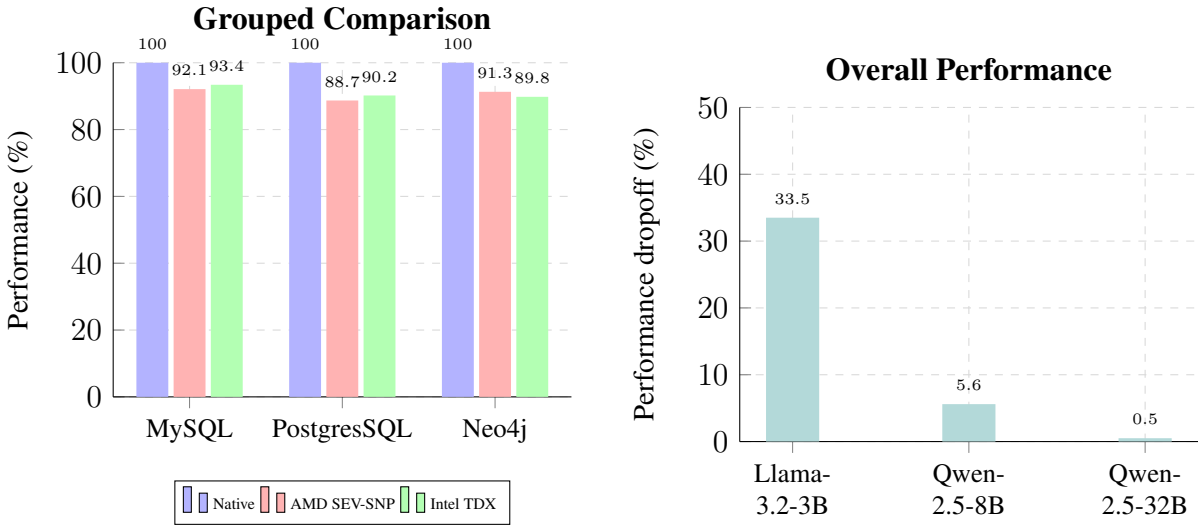
6. **Micropayment Settlement**: Payments flow automatically: User$\rightarrow$A and A$\rightarrow$B. Smart contracts enforce spending limits, preventing unauthorized drainage even if agents are compromised.

4 Cryptographic Isolation

AMD SEV-SNP and Intel TDX implement hardware-based confidential computing through VM-level isolation. These architectures employ on-die hardware random number generators (HRNGs) to derive encryption keys for full memory encryption. The keys reside exclusively within the processor's security perimeter, protected from extraction via software interfaces. Memory controllers encrypt data at the cache line granularity (64 bytes) using AES-128 with a tweak function incorporating the physical address, preventing block relocation attacks. The hypervisor and all system software observe only ciphertext in DRAM.

Remote attestation enables cryptographic verification of execution environments. Each processor embeds a device-specific attestation key during manufacturing, certified by the vendor's PKI infrastructure. The attestation protocol generates reports containing: (i) SHA-384 digest of the initial VM image, (ii) security configuration registers encoding enabled features, (iii) platform state measurements, and (iv) caller-provided nonce for freshness. The processor signs these reports using ECDSA P-384, enabling remote parties to verify $\text{Verify}_{pk}(\text{Report}, \sigma) = 1$ where pk chains to the vendor's root certificate.

GPU confidential computing extends the trusted execution boundary to accelerators. The attestation protocol follows PCIe Component Measurement and Authentication (CMA) with Security Protocol and Data Model (SPDM) 1.2. Upon successful mutual authentication, the CPU and GPU establish a secure session using ECDHE key agreement, protecting all PCIe transactions with AES-256-GCM. This cryptographic tunnel ensures model parameters θ and activations h_i remain encrypted except within authenticated execution units.



(a) Database performance in sealed environments

(b) TEE-enabled model performance

Figure 1: Performance overhead of confidential computing across workloads. SQL operations incur 8-12% overhead due to memory encryption. LLM inference overhead varies inversely with model size: 30% for models <3B parameters (batch sizes 4-16), 5% for 8B models, and <1% for models >10B parameters. Larger models amortize encryption costs across more computation.

Reproducible builds complete the trust chain through deterministic compilation. Given source code S and build environment E, the compiler produces binary B such that $H(B) = h$ deterministically. The attestation measurement $m = H(B_{\text{loaded}})$ enables verification $m \stackrel{?}{=} h$, proving execution of published code. This establishes end-to-end verifiability: silicon authenticates the measurement, the measurement identifies the binary, the binary corresponds to auditable source code.

5 Action Tokens and Agent Communication

The network distinguishes between client agents that interface with users and service agents that provide specialized computation. Client agents maintain user context and orchestrate re-

quests, while service agents execute specific tasks with domain expertise. Communication between agents requires a protocol more efficient than natural language parsing.

Traditional tool-calling approaches suffer from fundamental inefficiencies. Language models attempting to invoke functions must parse documentation, construct JSON payloads, and hope their output conforms to expected schemas. This performance wastes computational resources: GPT-4 requires approximately 1,000 tokens to invoke a simple calculator function.

We adopt the functional token framework demonstrated in Octopus v2 [Chen et al., 2024]. Functions are embedded directly into the model's vocabulary as discrete tokens. Instead of generating text descriptions of function calls, models emit action tokens—single vocabulary items that represent entire function invocations. Formally, given function $f : \mathcal{X} \to \mathcal{Y}$, we introduce token τ_f and train a LoRA adapter such that:

$$P(\tau_f|\text{context}) = \text{softmax}(W_{\text{base}} + \Delta W_{\text{LoRA}}) \cdot h_{\text{context}} \tag{1}$$

where h_{context} represents the hidden state encoding conversational context. The adapter learns to emit function tokens based on semantic patterns rather than syntactic parsing.

Service agents advertise capabilities through embedding vectors. Each agent publishes $C = \{\tau_1, \tau_2, ..., \tau_n\}$ where τ_i represents an action token embedding. Client agents discover services by computing cosine similarity between query embeddings and advertised capabilities. Given user intent q, routing becomes:

$$\text{route}(q) = \arg \max_{s \in \text{Services}} \sum_{c \in C_s} \text{cosine}(q, c) \tag{2}$$

This mathematical routing eliminates natural language intermediation. The client agent simultaneously selects the optimal service and reformulates the query for that service's expected format. Empirical results demonstrate 10x improvement in routing speed compared to text-based approaches, with accuracy reaching 98.5% on standardized benchmarks.

The architecture enables true specialization. A tax preparation agent learns tokens for `calculate_deduction()` and `file_return()`. A medical diagnosis agent learns `analyze_symptoms()` and `suggest_treatment()`. Each service requires only an 8MB LoRA adapter rather than a full model fine-tune. The base model remains unchanged across all agents, with only the lightweight adapter differing. This modularity allows the network to scale to thousands of specialized services without multiplicative storage costs.

6 The Agent Graph

The agent network can be represented as a graph in which nodes correspond to agents and edges represent capability dependencies. We distinguish two node types: master nodes that interface with users and worker nodes that provide specialized computation. This architecture mirrors the findings of Xu et al. [2024] on distributed language model systems, where specialized models outperform monolithic ones on heterogeneous tasks.

Master nodes maintain user context and orchestrate request routing. Each master node stores user state in a knowledge graph — e.g., using Neo4j, known for native graph operations and enabling O(1) neighbor lookups. Relationships encode user preferences, conversation history, and learned behavior patterns. Raw messages persist in PostgreSQL for 72 hours before distillation into graph representations. This dual-storage approach optimizes for both immediate retrieval and long-term pattern recognition.

Worker nodes advertise capabilities through a gossip protocol. Each node broadcasts a capability vector $C = \{c_1, c_2, ..., c_n\}$ where c_i represents the embedding of action token τ_i. Nodes maintain a routing table of discovered workers, updated through periodic heartbeats. The absence of central coordination ensures no single point of failure or censorship.

Request routing employs embedding-based similarity matching. Given user intent embedding q, the master node computes:

$$\text{route}(q) = \underset{w \in Workers}{\arg\max} \sum_{c \in C_w} \text{cosine}(q, c) \tag{3}$$

This mathematical routing eliminates natural language parsing. Workers may chain requests to other workers, forming execution graphs bounded only by spending limits. The network naturally evolves specialized clusters: financial agents discover tax preparers and investment analyzers; creative agents find image generators and music composers. Market dynamics drive this organization—frequently co-accessed capabilities migrate closer in the graph, reducing latency and payment overhead.

7 Cryptographic Payments

7.1 Payment Requirements

Agent-to-agent micropayments impose stringent requirements: (i) sub-second finality for interactive workloads, (ii) a sufficiently small fee ($\lesssim 10^{-3}$ USD) to enable penny-scale inference pricing, (iii) programmatic spending controls preventing unauthorized drainage, and (iv) censorship resistance ensuring global accessibility. Traditional payment systems fail these requirements categorically. We require a cryptocurrency optimized for high-frequency micropayments with programmable restrictions.

7.2 Transaction Finality Analysis

Bitcoin achieves probabilistic finality through proof-of-work, requiring approximately 6 blocks (60 minutes) for 99.9% double-spend resistance. Ethereum's proof-of-stake provides finality after 2 epochs (12.8 minutes). For comparison, LLM inference completes in 100-500ms. These finality periods exceed inference time by factors of 10^4-10^5, making synchronous payment-for-inference impossible.

Modern architectures enable atomic payment-inference transactions, where payment confirmation completes within the model's inference window. In particular, under PoH + Tower BFT, Solana offers sub-second optimistic confirmation (500–600 ms); the forthcoming Alpenglow consensus is designed for 100–150 ms finality (occasionally 100 ms), enabling near-real-time settlement for model calls.

7.3 Spending Control Mechanism

We implement spending limits through programmable constraints enforced on-chain. Let V be a vault address controlled by multisignature, A be an authorized agent address, L be a spending limit, and T be a time window. The smart contract enforces:

$$\forall t : \sum_{i \in \text{Tx}(A,V,[t-T,t])} \text{amount}_i \leq L \tag{4}$$

where $\text{Tx}(A, V, [t_1, t_2])$ denotes transactions from A drawing from V within time interval $[t_1, t_2]$. This provides cryptographic rate limiting: agents cannot exceed authorized amounts even under complete compromise. Users retain sovereignty through the multisignature mechanism while enabling autonomous operation within defined bounds.

8 Privacy Guarantees

Conventional privacy architectures rely on policy-based controls enforced through access control lists, audit logs, and legal agreements. These mechanisms fail systematically: access controls succumb to privilege escalation, audit logs provide only post-hoc detection, and legal agreements offer no technical prevention. We require privacy guarantees rooted in cryptographic impossibility rather than institutional compliance.

The system achieves privacy through compositional cryptographic proofs. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{384}$ denote the hardware attestation function, C the code measurement, K_{TEE} the ephemeral keys within the trusted execution environment, and D the user data. The system maintains the invariant:

$$\mathcal{H}(C) = h_{\text{published}} \land D' = \text{AES-GCM}_{K_{\text{TEE}}}(D) \land K_{\text{TEE}} \cap \text{Accessible}_{\text{SW}} = \emptyset \tag{5}$$

This conjunction establishes three properties: (1) executing code matches published source through reproducible builds, (2) data exists only in encrypted form outside the TEE, (3) encryption keys remain inaccessible to all software layers including hypervisor and OS kernel. Breaking this guarantee requires either forging processor signatures (computationally infeasible without the root key) or physical extraction of keys from silicon (requiring nation-state resources).

Data lifecycle enforces ephemeral processing. Upon request receipt at time t_0, the system generates keys $K_{\text{session}} \leftarrow \text{HRNG}()$ within the TEE. All user data encrypts immediately: $D_{\text{enc}} = E_{K_{\text{session}}}(D_{\text{user}})$. Processing occurs in memory regions marked with architectural flags preventing DMA access and cache-line attacks. At request completion t_1, the system executes secure erasure: $\text{memset}(K_{\text{session}}, \text{random}, |K_{\text{session}}|)$ before memory deallocation. The hypervisor observes only the tuple $\langle D_{\text{enc}}, t_1 - t_0, \{\text{access}_i\}\rangle$ containing ciphertext, timing, and memory access patterns.

Information leakage through side channels remains theoretically possible but practically limited. Timing attacks require distinguishing inference patterns across millions of possible queries. Memory access patterns reveal at most the model architecture, not user data. Power analysis attacks require physical access to the hardware. The system implements standard mitigations: constant-time cryptographic operations where feasible, memory access obfuscation through padding, and rate limiting to prevent statistical analysis. Perfect security remains impossible, but the attack surface shrinks to academic curiosities rather than practical exploits.

The network architecture amplifies privacy guarantees through structural properties. Traffic analysis becomes exponentially harder as node count increases—distinguishing one user among n requires $O(n^2)$ correlation attempts. Multi-hop routing creates unintentional mix networks, where each hop sees only immediate neighbors. Payment flows reveal aggregate usage statistics but cannot link specific queries to users, as the same wallet funds multiple unrelated requests. Unlike centralized systems where scale enables surveillance, our architecture ensures that growth strengthens anonymity. The system achieves privacy not through policy but through physics: the computational cost of surveillance exceeds the value of any feasible attack.

9 Conclusion

We have proposed a system for artificial intelligence without surveillance. The architecture rests on cryptographic primitives rather than corporate promises: processors that prove their state, payments that enforce limits, and routing that requires no coordinator. Each component exists today. AMD manufactures the chips. Solana processes the payments. The code compiles deterministically. Only integration remains.

Hardware attestation completes the cryptographic toolkit. We can now prove what executes where, with what data, under whose control. Combined with efficient architectures (linear attention), practical payments (100ms finality), and mathematical routing (embedding similarity), we achieve what seemed impossible: artificial intelligence that scales without surveillance.

The implications extend beyond privacy. When users control their agents, when developers deploy without permission, when payments flow without intermediaries, we create markets instead of monopolies. Quality emerges through competition. Innovation requires no approval. The network grows through voluntary association rather than platform lock-in.

The code is law. The math is truth. Privacy is non-negotiable.

References

[1] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., and Lane, N. D. (2020). *Flower: A friendly federated learning research framework*. arXiv preprint arXiv:2007.14390.

[2] Brutzkus, A., Gilad-Bachrach, R., and Elisha, O. (2019). *Low latency privacy preserving inference*. International Conference on Machine Learning (ICML).

[3] Chandar, B., Hada, R., and Campbell, C. (2022). *Privacy-preserving machine learning with trusted execution environments*. arXiv preprint arXiv:2205.12704.

[4] Chen, W., Wang, S., Zhang, C., Li, Z., Sun, X., and Zhang, M. (2024). *Octopus v2: On-device language model for function calling*. arXiv preprint arXiv:2404.01549.

[5] Chen, W. and Li, Z. (2024). *Octopus v4: Graph of language models*. arXiv preprint arXiv:2404.19296.

[6] Choe, W., Ji, Y., and Lin, F. X. (2025). *RWKV-Lite: Deeply Compressed RWKV for Resource-Constrained Devices*. arXiv preprint arXiv:2412.10856.

[7] Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. (2021). *Rethinking attention with performers*. International Conference on Learning Representations (ICLR).

[8] Claverie, T., Boz, T., and Bottoni, M. (2018). *iExec: Blockchain-based decentralized cloud computing*. iExec Whitepaper.

[9] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). *CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy*. International Conference on Machine Learning (ICML).

[10] Goldstein, A., Liu, H., Ranganathan, A., and Stoica, I. (2025). *RADLADS: Rapid Architecture Distillation for Linear Attention via Decomposed Supervision*. Proceedings of the International Conference on Learning Representations (ICLR).

[11] Golem Team (2016). *The Golem Project: Crowdsourced supercomputer*. Golem Whitepaper.

[12] Gu, A., Goel, K., and Ré, C. (2022). *Efficiently modeling long sequences with structured state spaces*. International Conference on Learning Representations (ICLR).

[13] Hao, M., Li, H., Chen, H., Xing, P., Xu, G., and Zhang, T. (2022). *Iron: Private inference on transformers*. Advances in Neural Information Processing Systems (NeurIPS).

[14] He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., et al. (2020). *FedML: A research library and benchmark for federated machine learning*. arXiv preprint arXiv:2007.13518.

[15] Khattab, O., Santhanam, K., Li, X. L., Hall, D., Liang, P., Potts, C., and Zaharia, M. (2022). *Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP*. arXiv preprint arXiv:2212.14024.

[16] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M., and Potts, C. (2023). *DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines*. arXiv preprint arXiv:2310.03714.

[17] Kumar, N., Rathee, M., Chandran, N., Gupta, D., Rastogi, A., and Sharma, R. (2020). *CrypTFlow: Secure tensorflow inference*. IEEE Symposium on Security and Privacy (S&P).

[18] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). *Communication-efficient learning of deep networks from decentralized data*. Artificial Intelligence and Statistics (AISTATS).

[19] Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., and Popa, R. A. (2020). *Delphi: A cryptographic inference service for neural networks*. USENIX Security Symposium.

[20] Mohassel, P. and Zhang, Y. (2017). *SecureML: A system for scalable privacy-preserving machine learning*. IEEE Symposium on Security and Privacy (S&P).

[21] Nakajima, Y. (2023). *BabyAGI: Task-driven autonomous agent*. GitHub repository.

[22] Narayan, A., Biderman, D., Eyuboglu, S., May, A., Linderman, S., Zou, J., and Re, C. (2025). *Minions: Cost-efficient Collaboration Between On-device and Cloud Language Models*. arXiv preprint arXiv:2502.15964.

[23] Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. (2021). *Random feature attention*. International Conference on Learning Representations (ICLR).

[24] Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., Kiran, G., et al. (2023). *RWKV: Reinventing RNNs for the Transformer Era*. arXiv preprint arXiv:2305.13048.

[25] Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., Kiran, G., et al. (2023). *RWKV: Reinventing RNNs for the Transformer Era*. arXiv preprint arXiv:2305.13048.

[26] Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., Kiran, G., et al. (2025). *RWKV: Reinventing RNNs for the Transformer Era*. Advances in Neural Information Processing Systems (NeurIPS).

[27] Rao, J. (2021). *Bittensor: A peer-to-peer intelligence market*. arXiv preprint arXiv:2111.00255.

[28] Richards, T. (2023). *AutoGPT: An autonomous GPT-4 experiment*. GitHub repository.

[29] Siyan, L., Raghuram, V. C., Khattab, O., Hirschberg, J., and Yu, Z. (2025). *PAPILLON: Privacy Preservation from Internet-based and Local Language Model Ensembles*. arXiv preprint arXiv:2410.17127.

[30] Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. (2023). *Retentive network: A successor to transformer for large language models*. arXiv preprint arXiv:2307.08621.

[31] Tramer, F. and Boneh, D. (2019). *Slalom: Fast, verifiable and private execution of neural networks in trusted hardware*. International Conference on Learning Representations (ICLR).

[32] Tran, K.-T., Dao, D., Nguyen, M.-D., Pham, Q.-V., O'Sullivan, B., and Nguyen, H. D. (2025). *Multi-Agent Collaboration Mechanisms: A Survey of LLMs*. arXiv preprint arXiv:2501.06322.

[33] Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). *Linformer: Self-attention with linear complexity*. arXiv preprint arXiv:2006.04768.

[34] Wang, W., Wang, G., Bhatnagar, A., Singh, Y., Song, Y., and Zhu, J. (2021). *Secure neural network inference with encrypted data using trusted execution environments*. IEEE Transactions on Dependable and Secure Computing.

[35] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., and Wang, C. (2023). *AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework*. arXiv preprint arXiv:2308.08155.

[36] Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. (2021). *Nyströmformer: A Nyström-based algorithm for approximating self-attention*. AAAI Conference on Artificial Intelligence.

[37] Xu, Y., Zhang, L., Wei, C., and Zhou, D. (2024). *Distributed Language Model Systems: A Comparative Study of Specialized vs. Monolithic Architectures*. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[38] Yueyu, L., Zhiyuan, L., Yue, P., and Xiao, L. (2025). *ARWKV: Pretrain is not what we need, an RNN-Attention-Based Language Model Born from Transformer*. arXiv preprint arXiv:2501.15570.