Local AI Agents as Research Assistants for Historical Cryptography

Tomáš Pavuk, Eugen Antal and Pavol Zajac

Slovak University of Technology in Bratislava
Ilkovičova 3, 841 04 Bratislava
Slovakia

Abstract

Research of historical cryptography can significantly benefit from modern AI methods. Currently available commercial and open solutions include multimodal processing of historical manuscripts, reasoning methods that support decryption tasks, and specific AI agents that can improve the workflow of the researcher. One of the problems with available solutions is questions related to the privacy of the processed data. Some of the research materials can be private, with specific use restrictions, and can even contain some information that is still secret (or should stay restricted until the research is finished). To overcome the privacy limitations, we have designed and developed a custom local AI agent system. The system can analyse encrypted historical documents, compare different approaches to solving the cryptograms, and even automate the whole pipeline of the historical research without the need to process data in cloud services. The experiments show that while the capabilities of the local AI agents are weaker than the available cloud-based solutions, they can still be efficient enough for our research purposes. However, careful tuning and preparation of local agents are required for the best results.

Keywords

LLM, local LLM, agents, historical ciphers, cryptography

1. Introduction

Recent progress in AI methods has opened new possibilities for supporting selected tasks in historical cryptography research. Large language models, multimodal models, and agent-based systems may assist researchers in activities such as transcription support, reconstruction of partially damaged text, exploration of possible cipher types, and overall automation of the decryption or decipherment process. These methods are not replacements for expert cryptanalytic work, but they can provide useful assistance in workflows where the material is incomplete, ambiguous, or noisy.

However, the use of AI tools in this domain introduces an important practical security problem. The most capable and convenient AI systems are often available as cloud-based services. This is problematic when the processed material is unpublished, copyrighted, privately owned, or subject to archive-specific usage restrictions. Even if the historical document itself is not secret, intermediate research artifacts may represent sensitive research output that should remain restricted until the research is completed.

The goal of this paper is to present a local, modular, agent-based LLM framework for privacy-sensitive cryptanalytic research workflows. The framework builds on the team project *Lokálne jazykové modely*, developed at the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava [1]. The original project developed the local LLM infrastructure, experiment runner, tool integration, logging, evaluation metrics, and initial cryptanalytic workflows. In this paper, we present this work from the perspective of privacy-preserving AI-assisted cryptanalytic research.

The framework was developed for experiments in historical cryptography, where AI assistance must be combined with exact and verifiable cryptanalytic operations. At the architectural level, however, the same design principles apply more broadly to privacy-sensitive research workflows: data should remain

ITAT'26: Information technologies – Applications and Theory, September 25–29, 2026, Biele Karpaty, Slovakia
xpavukt@stuba.sk (T. Pavuk); eugen.antal@stuba.sk (E. Antal); pavol.zajac@stuba.sk (P. Zajac)
https://www.hcportal.eu (E. Antal)
0000-0002-9685-6051 (E. Antal); 0000-0003-1909-9453 (P. Zajac)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

CEUR-WS.org/Vol-4323/paper39.pdf

local, model outputs should be logged and evaluated, and tasks requiring exactness should be delegated to deterministic tools. In this paper, we use cryptanalytic tasks to validate these design principles.

The contribution of this paper is not a new cipher solver. Instead, we present a reusable framework that allows researchers to test different local LLM-based approaches to cryptanalytic tasks without sending data to external cloud services. The framework supports multiple execution modes, including scripted experiment runs, iterative feedback loops, tool-assisted agents, deterministic brute-force workflows with LLM-based candidate ranking, heuristic search, and orchestrator-worker workflows.

The main contributions of this paper are:

- a local framework for AI-assisted cryptanalytic research workflows without dependence on external cloud-based AI services;
- a modular architecture separating LLM inference, experiment orchestration, deterministic tools, datasets, metrics, and logging;
- support for multiple execution modes, including scripted experiments, tool-assisted agents, and orchestrator-worker workflows;
- a security-oriented discussion of local deployment for sensitive historical cryptography data;
- an experimental validation showing which parts of cryptanalytic workflows can be supported by local LLM agents and which require deterministic tools.

The rest of the paper is organized as follows. Section 2 summarizes background and related work. Section 3 explains the security and privacy motivation for local processing. Section 4 presents the framework architecture. Section 5 discusses the security properties and local deployment model. Section 6 presents an evaluation of the framework on representative cryptanalytic workflows. Section 7 discusses the implications of the evaluation and limitations of our current work.

2. Background and Related Work

Historical cryptography studies the evolution of encryption and cryptanalytic methods before modern mathematical methods were developed. Current computing power and technologies can enable modern methods that can be applied to the study of historical ciphers, including methods that can process and decipher previously secret materials [3]. The ultimate goal is a fully automated decryption pipeline [2] that includes all related tasks, such as transcription, analysis, and decipherment.

Large language models, which build on the Transformer architecture [8], have been shown to perform strongly across a wide range of language-oriented tasks [9]. These capabilities are relevant for historical cryptography because the research workflow often requires interpretation of incomplete or noisy textual material, generation of possible readings, and evaluation of candidate plaintexts.

At the same time, cryptanalytic tasks expose the limitations of LLMs. Classical ciphers often require exact symbolic manipulation, strict character inventory checking, enumeration of permutations, and reproducible validation. A language model may produce text that looks plausible but violates the exact constraints of the cipher. This is especially problematic in cryptanalysis, where a near-readable phrase is not necessarily a valid solution.

Recent cryptographic benchmarks such as CipherBank evaluate the reasoning capabilities of LLMs on cryptographic challenges [7]. Previous work in historical cryptology has also investigated AI-assisted processing of historical encrypted material, including encrypted postcards and historical anagrams [4, 5, 6]. These works show that AI tools can support historical cryptography, but also that direct AI-based solving is not sufficient for all tasks.

The framework proposed in this paper also relates to the broader direction of agent-based LLM systems. In such systems, the language model is not used only as a text generator, but as a component that can interact with tools, external state, and orchestration logic [10, 11]. This pattern is useful for cryptanalytic workflows, where some steps require flexible language interpretation while other steps require deterministic and verifiable computation.

Our work differs from a pure model evaluation. We focus on building a local framework that combines LLMs with deterministic tools, experiment orchestration, logging, and evaluation. The goal is not only to test whether a model can solve a cipher, but to provide a controlled environment in which multiple local AI-assisted cryptanalytic workflows can be implemented and compared.

3. Security and Privacy Motivation

The security motivation of this work comes from the way AI tools are commonly used today. In many cases, a researcher uploads a document, transcription, or partial analysis to a commercial or external AI service. This workflow is convenient, but it also means that the data leaves the researcher's controlled environment.

In historical cryptography, this can be undesirable. The processed data may include historical manuscripts, scans from archives, private collections, unpublished transcriptions, candidate decryptions, suspected keys, or research notes. Some of these materials may be restricted by archive rules or copyright. Others may not be legally secret, but may still represent unpublished research that should not be disclosed before the work is completed.

We identify several practical risks:

- **Data leakage.** Uploading material to an external service can expose unpublished or restricted research data.
- **Copyright and licensing restrictions.** Historical scans, edited datasets, or transcriptions may have usage restrictions even if the original document is old.
- **Loss of research control.** Candidate keys, partial decryptions, and hypotheses may represent original research output.
- **Institutional constraints.** Some institutions or archives may not permit processing their material through third-party cloud services.
- **Audit and reproducibility.** External AI services may not provide sufficient control over model version, logs, prompts, and intermediate outputs.

For these reasons, the central requirement is local processing. The framework should allow researchers to use AI assistance while keeping cryptographic material and intermediate research artifacts inside a local machine, university server, or private infrastructure.

This places the paper naturally within an applied security workshop. We do not propose a new cryptographic primitive. Instead, we address a practical security problem in the use of AI for cryptanalytic research: how to benefit from LLM-based assistance without transferring sensitive data to cloud-based systems.

4. Framework Architecture

The main result of the work is a modular framework for local LLM-assisted cryptanalytic workflows. The implementation builds on the team project *Lokálne jazykové modely* [1], but the framework is more focused here as a reusable architecture for privacy-preserving AI-assisted research. The original project provided the local model abstraction, experiment runner, tool integration, logging infrastructure, and initial cryptanalytic workflows.

The novelty of the framework is not in any single component, but in the integration of local LLM inference, deterministic cryptanalytic tools, configurable experiment execution, and privacy-aware logging into a reusable workflow for sensitive cryptanalytic research.

The framework is designed around the separation of three concerns. First, LLM inference is separated from experiment logic, so that different local models can be tested without rewriting the workflow. Second, exact cryptanalytic operations are implemented as deterministic tools rather than delegated to

the language model. Third, all inputs, intermediate outputs, tool calls, and metrics can be logged and evaluated locally, while allowing sensitive logging to be reduced or disabled.

Figure 1 shows the high-level architecture. The whole workflow is intended to run inside a local or private deployment boundary. The experiment runner loads the workflow configuration, initializes the selected model, dataset, tools, metrics, and logging configuration. The workflow layer then executes one of the supported execution modes.

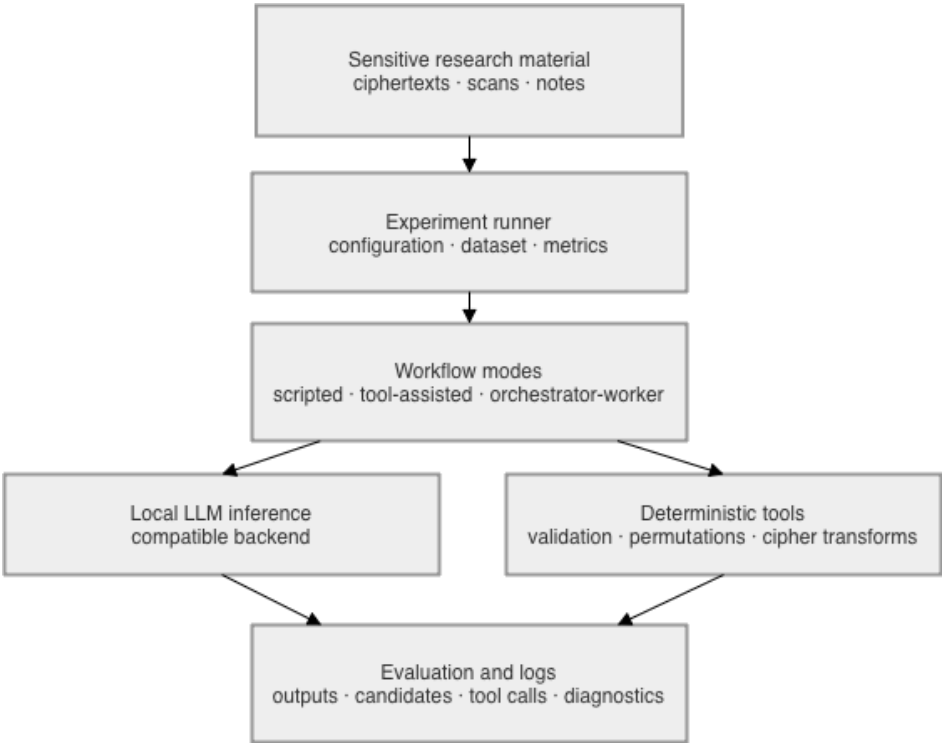


Figure 1: Compact architecture of the local LLM framework. Sensitive research material remains inside a local or private deployment boundary. The framework separates configuration-driven execution, workflow modes, local LLM inference, deterministic tools, and logging/evaluation.

4.1. Workflow Modes

The framework supports three main workflow modes. In a scripted workflow, the experiment is controlled by deterministic code and configuration, and the LLM is called only at selected points. This is useful for reproducible workflows, for example when code generates candidate keys and the model is used only to rank candidate plaintexts.

In a tool-assisted agent workflow, the model can call predefined tools during the solving process. The model may ask for possible words, validate a phrase, inspect intermediate state, or submit an answer. The framework logs all tool calls and their outputs. This mode is useful for interactive or semi-autonomous workflows where the model can guide parts of the process, but the available operations remain controlled.

In an orchestrator-worker workflow, one model acts as the orchestrator or judge, while worker models perform repeated subtasks. This is useful when many candidate decryptions or partial results must be evaluated in batches. In our experiments, this mode was used to combine a larger model for orchestration or final evaluation with a smaller model for repeated sub-tasks.

4.2. Local LLM Abstraction

The local LLM abstraction hides the specific inference backend from the experiment logic. The same experiment can therefore be configured to use different local models without changing the implemen-

tation of the workflow. In our implementation, the models were accessed through an Ollama local inference interface.

This abstraction is important for experimentation. It allows the researcher to compare different local models, model sizes, quantization settings, and inference configurations while preserving the same datasets, tools, metrics, and logging structure.

4.3. Deterministic Tool Layer

The deterministic tool layer is an essential part of the framework. It provides operations that should not be delegated to the model, such as dictionary search, permutation generation, cipher decryption, and exact text metrics computation.

This separation is important because LLMs are useful for language interpretation and plausibility judgments, but unreliable for exact symbolic manipulation. A model may produce an answer that looks natural while still violating the constraints of the cipher. The framework therefore uses LLMs for tasks such as candidate ranking, reconstruction of near-readable outputs, and orchestration, while deterministic tools handle operations that require exactness.

4.4. Logging and Evaluation

The logging and evaluation layer records model outputs, selected candidates, tool calls, metric values, and runtime diagnostics. This supports reproducibility, comparison of workflows, and debugging. It also allows the same task to be evaluated under multiple workflow configurations.

Because logs may contain sensitive research material, logging is treated as part of the framework design. Detailed logs are useful during development and evaluation, but they may need to be reduced or disabled when working with restricted data. The framework therefore assumes configurable logging rather than unconditional storage of all prompts and outputs.

4.5. Framework Extensibility

The framework is designed to be extensible. New experiments can be added by implementing a new experiment class and a configuration file. New tools, metrics, datasets, and processors can be registered as independent components and then used from the configuration.

This is important for historical cryptography because different materials and cipher types often require different processing strategies. A substitution cipher, a transposition cipher, a symbolic alphabet, and a partially damaged manuscript may require different tools, but they can still be evaluated inside the same local framework.

5. Security Properties and Local Deployment

The security contribution of the framework is practical rather than cryptographic. The framework does not introduce a new cryptographic primitive or a new access-control mechanism. Instead, it reduces the exposure of sensitive research material by allowing the AI-assisted workflow to run locally or inside a controlled private environment.

The main security property is data locality: all input material, intermediate research artifacts, prompts, and logs remain under the control of the researcher or institution (see Section 3).

The intended deployment model is a local workstation, university server, or private infrastructure. In our evaluation setup, the framework ran on a dedicated server and was accessed through the existing login and remote-access mechanisms of the environment, such as SSH or a private network. This keeps the model endpoint, input data, intermediate candidates, and logs inside the controlled infrastructure instead of sending them to external cloud services.

This deployment model relies on the existing security boundary of the environment, including operating-system user accounts, server access control, network isolation, and configurable logging. It is suitable for the intended research setting with known users and controlled access to the server.

Detailed logs are useful for reproducibility, debugging, and comparison of workflows. However, logs may contain sensitive prompts, candidate plaintexts, partial keys, or research hypotheses. For this reason, logging is configurable and can be reduced or disabled when restricted material is processed.

The framework uses deterministic tools to reduce the risk of unverifiable model output. Candidate plaintexts can be checked against character inventories, candidate keys can be generated or verified by code, and outputs can be compared using explicit metrics. This reduces the risk that a plausible hallucinated answer is treated as a valid cryptanalytic result.

The current implementation should be understood as a local research framework rather than a public multi-user service. If the framework were deployed as a public or broadly multi-user service, additional hardening would be required, including stronger application-level authentication, sandboxing, audit logs, rate limiting, secret management, and operational monitoring.

6. Framework Evaluation

We evaluated the implemented framework on representative cryptanalytic workflows. The goal of the evaluation is not to introduce a new state-of-the-art cipher solver, but to verify that the framework supports the intended modes of local AI-assisted cryptanalytic work: scripted execution, tool-assisted agents, orchestrator-worker workflows, deterministic validation, logging, and metric-based comparison.

The validation tasks were selected because they combine language interpretation with exact symbolic constraints. This makes them useful for testing whether the framework correctly separates LLM-based assistance from deterministic computation. Table 1 summarizes the tasks and the framework capability exercised by each one.

Table 1
Validation tasks used to exercise the framework.

Task	Validated framework capability
Best-of-N reconstruction	Repeated local inference, candidate logging, and deterministic verification.
Iterative correction	Feedback loop, attempt tracking, and deterministic error reports.
Tool-assisted agent	Tool calling, agent state, and validation of submitted answers.
Columnar transposition brute force	Scripted deterministic search with LLM-based candidate ranking.
LLM-generated permutations	Structured model output parsing, deduplication, and coverage checking.
Simulated annealing	Hybrid heuristic search with LLM evaluator, cache, and diagnostics.
Autonomous solving	Boundary test showing the need for deterministic support.

6.1. Validation Setup

The validation experiments use two local model roles. The following setup was used to test whether the framework can run larger local models and worker-style workflows in a controlled research environment without relying on external cloud AI services.

The larger model, `gpt-oss:120b`, is used when the workflow requires stronger reasoning, ranking, synthesis, or final reconstruction. The model has 116.8B parameters, and with MXFP4 quantisation it requires 65 GB of disk space (the model supports a context window of up to 131 072 tokens). Whole model takes 66 GiB VRAM of GPU (35.5 s load time from disk, if preloaded in VRAM it takes 100–200

ms to start). The smaller model, `qwen3:8b`, is used for repeated worker tasks such as candidate scoring, permutation generation, and batch comparison. The model has 8.2B parameters, and with Q4_K_M quantisation it takes 5.2 GB of disk space (the model supports a context window of 32 768 tokens). Whole model takes 6 GB VRAM of GPU (2.8–4 s of disk load time). Both models are accessed through the same local inference abstraction, which allows the experiment runner to switch models through configuration. The architecture relies on pure Python and CLI (with no web UI). Each component is independent, replaceable (registered under a unique name), configurable through a specific YAML file. The tools support plain text, JSON, and visual (PNG/JPEG/base64 WebP) inputs. Both outputs and logs are in JSON format.

The evaluation was performed on a dedicated server with 2 Intel Xeon 6737P CPUs, and 503 GiB RAM, running OS: Debian GNU/Linux 13, kernel 6.12. The server was equipped with an NVIDIA H200 NVL GPU [12] with 140.4 GiB VRAM. We used a local Ollama 0.17.6 instance, accessed through its REST API via an SSH tunnel.

The performance depended on the context. With small context windows (hundreds to thousands of tokens) gpt-oss processed 149 t/s (with prompt processing 1 400–2 750 t/s), and qwen processed 178–216 t/s (with prompt processing 7 000–8 700 t/s). Larger contexts degraded performance significantly. However, context cache improved performance of repeated prompts up to 34 000 t/s.

Note that our architecture has been run in a secure environment to prevent potential security issues. The models were isolated from the internet via loopback-only binding and SSH tunnels, and they were in a passive mode (under unprivileged user account). All tools were predefined, and no agents had access to shell or external code execution. All arguments were treated as data only, thus the models could not execute any additional code, or contact external services.

6.2. Candidate Generation and Feedback

The first two validation tasks test repeated generation and iterative correction. In the repeated reconstruction workflow, the model generated multiple candidates for a scrambled input and the framework selected the candidate that best matched the expected inventory. Simple word-level permutations were handled reliably, while character-level reconstruction produced plausible near-misses such as `artificial inteligence` for `artificial intelligence`.

The iterative correction workflow tested feedback handling. In one example, the model corrected `artificial intelgence` to `artificial intelligence` after receiving feedback about a missing character. In another example, the model converged to the plausible but invalid phrase `natural network` instead of `neural network`. This second case is not a valid character-inventory-preserving reconstruction; it illustrates that iterative feedback can also lead the model toward a linguistically plausible answer that violates the original constraints. These tasks validate candidate logging, feedback loops, and deterministic verification.

6.3. Tool-Assisted Agent Workflow

The tool-assisted agent workflow tested controlled tool use. For the input `nignealrnmachie`, the agent used dictionary and validation tools to construct and submit `machine learning`. Deterministic metrics, computed by the framework (not LLM) were used for evaluation. On character level, `char_match` is a ratio of correct letter positions and the string size (normalized matching character count). Similarly, on word level, `word_match` is a fraction of correct words in the candidate phrase compared to the original.

Note that on harder inputs, the agent sometimes submitted a near-miss too early. This validates the agent mode and shows that tool access must be combined with enforced validation.

6.4. Scripted and Structured Workflows

The scripted brute-force workflow was tested on a columnar transposition cipher with key length five. The framework generated all $5! = 120$ key permutations using deterministic code, decrypted the

ciphertext under each candidate key, and used the LLM only as a language judge. The correct candidate was selected with `char_match=1.0` and `word_match=1.0`.

The structured output workflow asked the worker model to enumerate all 120 permutations of $[0, 1, 2, 3, 4]$. The framework parsed, deduplicated, and validated the output. In the tested run, all 120 permutations were generated on the first attempt. This validates structured output handling, but deterministic enumeration remains preferable when complete coverage must be guaranteed.

6.5. Hybrid Search and Boundary Test

The hybrid heuristic search workflow used simulated annealing for a columnar transposition task with key length ten, where exhaustive search would require $10! = 3\,628\,800$ permutations. Deterministic code generated neighboring keys, while a worker LLM compared candidate plaintexts. A representative run recorded 438 LLM comparison calls, 91 cache hits, and 3 JSON parsing failures. The best candidate improved from `char_match=0.921` before final reconstruction to 0.974 after orchestrator reconstruction.

Finally, a boundary test removed most deterministic support and asked the LLM to solve a short columnar transposition example directly. The best output achieved only `char_match=0.583` and `word_match=0.0`. This negative result supports the framework design: exact positional manipulation should remain in deterministic tools, while the LLM should be used for ranking, interpretation, and orchestration.

7. Discussion

The evaluation supports the main architectural decision of the framework: local LLMs should be used as components of controlled workflows, not as autonomous cryptanalytic solvers.

First, deterministic validation is necessary. Several generated outputs were linguistically plausible but cryptanalytically invalid. In cryptanalytic workflows, a near-readable phrase is not sufficient if it violates character counts, ordering constraints, or cipher rules.

Second, hybrid workflows are the most reliable execution mode. The most robust workflows in our evaluation were hybrid workflows, where deterministic code generated or verified candidates and the LLM ranked them by language plausibility. This pattern is more reproducible than direct prompting and safer than fully autonomous solving.

Third, tool-assisted agents are useful, but their freedom must be limited by the framework. The model may call validation tools and still decide too early that an answer is acceptable. Therefore, validation should be enforced by the workflow rather than left only to the prompt.

Fourth, local deployment changes the trade-off. Local models may be weaker than cloud-based systems, but deterministic tools, orchestration, and validation can compensate for part of this difference.

These implications are important for the design of AI systems in applied security contexts. When AI is used on sensitive data, the architecture should not rely only on model capability. It should also provide locality, verifiability, reproducibility, and controlled tool use.

This paper presents a framework and its experimental validation. It is not intended as a complete benchmark of all local LLMs or all historical cipher types. The selected validation tasks focus on reconstruction and transposition problems because they exercise the main architectural components: local inference, deterministic tools, agent control, structured output parsing, logging, and metric-based evaluation.

The evaluation emphasizes text-based cryptanalytic workflows because they make it possible to measure exact symbolic constraints, such as character inventories, permutations, and candidate plaintext validation. This does not mean that the framework is limited to text inputs. The implementation includes a modality layer, and the team project also tested an image-based workflow in which a historical cipher image was processed through a vision-capable local model. This demonstrates that image inputs can be integrated into the same configuration, execution, logging, and evaluation structure. A broader evaluation of multimodal historical material, including handwriting, symbolic alphabets, and damaged scans, is outside the main scope of this paper.

The results depend on the selected local models, inference backend, prompts, and quantization settings. The framework is designed to make such comparisons possible, but a broader model comparison is outside the scope of this paper.

Acknowledgments

This research was supported by the project "Artificial intelligence for encrypted handwritten document processing", "09I05-03-V02-00031" of call "Support of research projects aimed at digitization of the economy in TRL levels 1-3", Call. No. 09I05-03-V02, managed by the Research Agency and funded by the Recovery and Resilience Facility of the Slovak Republic.

Declaration on Generative AI

During the preparation of this work, the first author used Claude (Anthropic) and ChatGPT (OpenAI) in order to: paraphrase and reword. After using these tools, the author reviewed and edited the content as needed, and all authors take full responsibility for the publication's content.

References

[1] T. Pavuk, M. Bartík, D. Babka, M. Brand, P. Macko, Lokálne jazykové modely, Team project, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2026.

[2] E. Antal, From ink to algorithms: AI-powered processing and cryptanalysis of historical ciphers, in: ITAT 2025: 25th Conference on Information Technologies – Applications and Theory, Telgárt, Slovakia, 2025.

[3] O. Grošek, P. Zajac, Automated cryptanalysis of classical ciphers, in: J. R. Rabuñal Dopico, J. Dorado, A. Pazos (Eds.), Encyclopedia of Artificial Intelligence, Information Science Reference, Hershey, PA, 2008.

[4] M. Bednár, G. Jurga, M. Mokáň, F. Precner, S. M. Sirota, Spracovanie šifrovaných pohľadníc, Team project, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2024.

[5] T. Selep, Riešenie anagramov inteligentnými metódami, Bachelor's thesis, Slovak University of Technology in Bratislava, Bratislava, Slovakia, 2025.

[6] E. Antal, P. Zajac, Can artificial intelligence solve the mysterious anagram from the Church of the Poor Clares in Bratislava?, in: Proceedings of the 7th International Conference on Historical Cryptology (HistoCrypt 2024), Oslo, Norway, 2024.

[7] Y. Li, Q. Pei, M. Sun, H. Lin, C. Ming, X. Gao, J. Wu, C. He, L. Wu, CipherBank: Exploring the boundary of LLM reasoning capabilities through cryptography challenges, in: Findings of the Association for Computational Linguistics: ACL 2025, Association for Computational Linguistics, Vienna, Austria, 2025.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[9] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao, Large language models: A survey, 2024. URL: https://arxiv.org/abs/2402.06196. arXiv:2402.06196.

[10] Anthropic, Building effective agents, 2024. URL: https://www.anthropic.com/research/building-effective-agents.

[11] LangChain, LangGraph: Stateful, multi-actor applications with LLMs, 2024. URL: https://langchain-ai.github.io/langgraph/.

[12] NVIDIA, NVIDIA H200 Tensor Core GPU, 2024. URL: https://www.nvidia.com/en-us/data-center/h200/.