Universität Siegen

Wirtschaftswissenschaften
Wirtschaftsinformatik
Wirtschaftsrecht

Master Thesis

for the degree of Master of Science in Business Information Systems

Integrating an LLM-Based Agent into CrypTool 2 for Cryptography and Cryptanalysis Education

Mentor	Prof. Dr. Nils Kopal
Supervisor	Prof. Bernhard Esslinger
Second Supervisor	Prof. Dr. Roland Wismüller

Submitted by	Marc Philipp Kray
Student Number	1566750
Submission Date	May 18, 2026
Updated	July 23, 2026

Abstract

This thesis examines whether integrating an LLM-based agent directly into CrypTool 2 (CT2) can improve the usability of the software. In the following, this LLM-based agent is referred to simply as "the agent". CT2 is a software environment for cryptography and cryptanalysis, and the agent is intended to lower the entry barrier for learning these topics. CT2 provides over 200 components, around 280 templates, and many configuration options, which can make it difficult for beginners to use effectively. Existing support mechanisms such as documentation, templates, and tutorials are static and not directly connected to the user's current workspace or task.

To address this problem, this thesis presents CrypLLM, a context-aware conversational agent integrated into CT2. CrypLLM is implemented in C# and uses Microsoft Semantic Kernel to connect CT2 with cloud-based and local LLMs. The agent uses tools provided by plugins to access information about the current workspace, including components, connectors, settings, validation errors, logs, and open tabs. The prototype also supports selected workspace modifications such as inserting components, creating connections, adjusting parameters, and starting workspace execution.

The prototype was evaluated qualitatively with three participants who had no prior experience with CT2. The evaluation included think-aloud protocols, semi-structured interviews, and practical tasks involving workspace exploration, conceptual understanding, debugging, and cryptanalysis. The results show that the agent was especially useful for first-time users. It reduced uncertainty, helped participants understand cryptographic concepts and CT2 workspaces, and supported debugging and problem solving. Participants particularly valued the direct access to the current workspace context, which enabled more specific guidance than an external chatbot.

At the same time, the evaluation revealed several limitations. The agent sometimes referred to internal connector names or interface terms that did not match the language of the user's question. Longer conversations occasionally caused context-related problems during the evaluation, which motivated the later integration of context-window management mechanisms.

Overall, the thesis demonstrates that context-aware AI integration can provide meaningful support in CT2 and improve both usability and learning in cryptography education.

Kurzzusammenfassung

Diese Arbeit untersucht, ob die direkte Integration eines LLM-basierten Agenten in CT2 die Nutzung der Software erleichtern kann. Im Folgenden wird dieser LLM-basierte Agent vereinfachend als „Agent“ bezeichnet. CT2 ist eine Softwareumgebung für Kryptografie und Kryptoanalyse, und der Agent soll die Einstiegshürde beim Erlernen dieser Themen senken. CT2 bietet über 200 Komponenten, rund 280 Vorlagen und zahlreiche Konfigurationsmöglichkeiten, was die Nutzung insbesondere für Einsteiger erschweren kann. Bestehende Unterstützungsmechanismen wie Dokumentation, Vorlagen und Tutorials sind statisch und nicht direkt mit dem aktuellen Workspace oder der konkreten Aufgabe des Nutzers verbunden.

Als Lösungsansatz stellt diese Arbeit CrypLLM vor, einen kontextbewussten, konversationellen Agenten, der in CT2 integriert ist. CrypLLM wurde in C# implementiert und verwendet Microsoft Semantic Kernel, um CT2 mit cloudbasierten und lokalen LLMs zu verbinden. Der Agent nutzt von Plugins bereitgestellte Tools, um auf Informationen über den aktuellen Workspace zuzugreifen, darunter Komponenten, Konnektoren, Einstellungen, Validierungsfehler, Logs und geöffnete Tabs. Zusätzlich unterstützt der Prototyp ausgewählte Änderungen an Workspaces, etwa das Einfügen von Komponenten, das Erstellen von Verbindungen, das Anpassen von Parametern und das Starten der Workspace-Ausführung.

Der Prototyp wurde qualitativ mit drei Teilnehmern evaluiert, die keine Vorerfahrung mit CT2 hatten. Die Evaluation umfasste Think-Aloud-Protokolle, semi-strukturierte Interviews und praktische Aufgaben zur Exploration von Workspaces, zum konzeptionellen Verständnis, zur Fehlersuche und zur Kryptoanalyse. Die Ergebnisse zeigen, dass der Agent insbesondere für Erstnutzer hilfreich war. Er reduzierte Unsicherheit, unterstützte die Teilnehmer beim Verständnis kryptografischer Konzepte und CT2-Workspaces sowie bei der Fehlersuche und Problemlösung. Besonders geschätzt wurde der direkte Zugriff auf den aktuellen Workspace-Kontext, der spezifischere Hilfestellungen ermöglichte als ein externer Chatbot.

Gleichzeitig zeigte die Evaluation mehrere Einschränkungen. Der Agent verwendete teilweise interne Konnektornamen oder Begriffe der Benutzeroberfläche, die nicht zur Sprache der Nutzerfrage passten. Längere Gespräche führten während der Evaluation gelegentlich zu Kontextproblemen, was die spätere Integration von Mechanismen zum Kontextfenster-Management motivierte.

Insgesamt zeigt die Arbeit, dass eine kontextbewusste KI-Integration in CT2 einen sinnvollen Mehrwert bieten und sowohl die Benutzbarkeit als auch den Lernerfolg in der Kryptografieausbildung verbessern kann.

Contents

Abstract 2

Kurzzusammenfassung 3

Contents 4

1. Introduction 7
- 1.1. Motivation . . . 7
- 1.2. Problem Statement . . . 8
- 1.3. Research Gap . . . 9
- 1.4. Research Question . . . 10
- 1.5. Objectives and Contributions . . . 11
- 1.6. Thesis Structure . . . 12
- 1.7. Working Environment . . . 13

2. Foundation 14
- 2.1. Cryptography and Cryptanalysis Fundamentals . . . 14
- 2.2. CrypTool Project . . . 15
- 2.3. CrypTool 2 . . . 15
  - 2.3.1. User Interface and Interaction Model . . . 16
  - 2.3.2. Workspace Model . . . 17
  - 2.3.3. Workspace Manager . . . 19
  - 2.3.4. Plugin-Based Architecture . . . 20
  - 2.3.5. Educational Features and Current Interaction Limitations . . . 21
- 2.4. Large Language Models . . . 21
- 2.5. LLM-Based Agents . . . 22
  - 2.5.1. Tool-Calling . . . 23
  - 2.5.2. Model Context Protocol . . . 24

3. Related Work 25
- 3.1. AI-Enhanced Learning and Intelligent Tutoring Systems . . . 25
- 3.2. LLM-Based Agents for Educational Applications . . . 26
- 3.3. Cryptology Education and Learning Tools . . . 27
- 3.4. CrypTool 2: Platform and Recent Developments . . . 28

4. System Design and Architecture 31
- 4.1. Design Objectives and System Requirements . . . 32
  - 4.1.1. Functional Requirements . . . 32
  - 4.1.2. Non-Functional Requirements . . . 33
- 4.2. System Overview . . . 33
- 4.3. Integration into CrypTool 2 . . . 35
  - 4.3.1. User Interface Integration . . . 35

4.3.2. System Boundary . . . 36
4.4. Architecture of the LLM-Based Agent . . . 36
4.4.1. Agent Overview . . . 36
4.4.2. Conversation Management . . . 37
4.4.3. Agent Execution Workflow . . . 38
4.4.4. Context Management and Token Budgeting . . . 38
4.4.5. Tool-Based Interaction . . . 39
4.4.6. Error Handling and Failure Modes . . . 39
4.5. Context Acquisition and Representation . . . 40
4.5.1. Workspace-Centric Context Extraction . . . 40
4.5.2. Structured Graph Representation . . . 41
4.5.3. Snapshot-Based Workspace Context Retrieval . . . 41
4.6. Use of the Semantic Kernel Framework . . . 41
4.7. Security and Privacy Considerations . . . 42
4.7.1. Security . . . 42
4.7.2. Privacy . . . 43
4.7.3. Safety . . . 44
4.8. Architectural Decisions and Trade-offs . . . 44

5. Implementation **46**
5.1. Development Environment and Technologies . . . 46
5.2. Integration into CrypTool 2 . . . 47
5.2.1. User Interface Integration . . . 47
5.2.2. Port and Adapter Integration . . . 47
5.2.3. Runtime Initialization . . . 48
5.3. Implementation of the LLM-Based Agent . . . 48
5.3.1. Agent Overview . . . 48
5.3.2. Conversation Management . . . 50
5.3.3. LLM Integration . . . 50
5.3.4. Agent Execution Workflow . . . 51
5.3.5. Context Window Management . . . 53
5.3.6. Tool-Based Interaction . . . 54
5.4. Context Acquisition Mechanisms . . . 56
5.5. User Interface . . . 58
5.6. Configuration, Security, and Consistency . . . 59

6. Evaluation **62**
6.1. Evaluation Methodology . . . 62
6.2. Participant Profile . . . 64
6.3. Task Design and Procedure . . . 64
6.4. Evaluation with Respect to the Research Questions . . . 66
6.4.1. Usability . . . 66
6.4.2. Understanding of Components and Workspaces . . . 67
6.4.3. Learning Support . . . 69

6.4.4. Problem Solving and Error Correction . . . 70
6.4.5. Response Quality . . . 71
6.4.6. Summary . . . 72

7. Discussion and Limitations **73**
7.1. Interpretation of the Results . . . 73
7.2. Limitations of the AI Chat . . . 74
7.3. Limitations of the Evaluation . . . 76
7.4. Threats to Validity . . . 77

8. Conclusion and Future Work **78**
8.1. Summary and Main Findings . . . 78
8.2. Implications for CT2 and Cryptography Education . . . 78
8.3. Future Work . . . 79
8.4. Final Conclusion . . . 81

Bibliography **82**

List of Abbreviations **89**

List of Tables **90**

List of Listings **91**

List of Figures **92**

Appendix **93**

A. Default System Prompt **93**

B. Think Aloud Protocols and Interview Transcripts **95**
B.1. Participant 1 . . . 96
B.2. Participant 2 . . . 99
B.3. Participant 3 . . . 102

C. Chat Histories **105**
C.1. Participant 1 . . . 106
C.2. Participant 2 . . . 109
C.3. Participant 3 . . . 112

Use of Artificial Intelligence **115**

Eidesstattliche Erklärung **116**

Content of the USB stick **116**

1. Introduction

This chapter introduces the background and motivation of the thesis. It outlines the importance of cryptography and cryptanalysis education and discusses current support mechanisms in CT2. Based on their limitations, the chapter identifies the research gap and presents the research question, objectives, and overall structure of the thesis.

1.1. Motivation

Cyberattacks are becoming more frequent, more sophisticated, and more expensive. According to the German Federal Office for Information Security BSI, the IT security situation in Germany remains tense, with ransomware, phishing, stolen credentials, and attacks on critical infrastructure representing major threats. The number of publicly known vulnerabilities is also increasing. In 2025, the BSI observed an average of 119 new vulnerabilities in IT systems per day. [6, p. 3]

In parallel, the financial impact of cyber incidents continues to grow. IBM reported that the average cost of a data breach in Germany had risen to approximately 4.9 million euros in 2024 [25], while the global average reached 4.88 million US dollars [4].

These developments demonstrate the increasing importance of education in cryptology. Secure communication, authentication, confidentiality, and digital trust all depend on cryptographic methods. At the same time, understanding cryptanalysis is necessary to identify weaknesses, evaluate security mechanisms, and understand how attacks are performed.

Practical learning environments are particularly important in this context because cryptographic concepts are often difficult to understand through theory alone. Learners benefit from experimenting with encryption methods, cryptanalytic techniques, attack scenarios, and protocol workflows in realistic environments.

CT2, first published in 2011, is one of the most comprehensive educational environments for cryptology. In its current version (CT2.1 Nightly Build 9843.1) [14], it provides 211 components and 281 templates for exploring classical and modern cryptographic methods. Users can create visual workflows, called workspaces, by placing functional building blocks, connecting them, and adjusting their settings. This enables them to execute cryptographic processes and visually inspect how data flows through the system. [15, 13]

However, the size and complexity of CT2 can make the software difficult to approach, especially for first-time users. The large number of available components, settings, and

possible workspace configurations can quickly become overwhelming. Users may struggle to identify which components are relevant for a given task, how they should be connected, or which component controls need to be adjusted.

To address these challenges, additional support can be beneficial not only for beginners but also for experienced users. Although they are already familiar with the general structure of the software, they may still need more information about individual components, possible workflows, parameter settings, or the interpretation of existing workspaces. Accessing this information through documentation, tutorials, or manual exploration can be time-consuming.

An integrated LLM-based agent could address these issues by providing context-aware support directly within the application. It can help new users understand the structure of the software, explain how workspaces can be constructed, and provide suggestions for selecting and connecting suitable components. At the same time, experienced users can retrieve relevant information more quickly and ask targeted questions about the current workspace, its structure, and its settings. Beyond answering questions and giving recommendations, the agent can also support users by directly interacting with the workspace, for example, by inserting components, creating connections, or adjusting selected parameters. In this work, the term system refers to the overall integration of the agent into CT2.

Recent software systems increasingly integrate LLM-based agents or agent-like conversational functionality directly into their user interfaces. Examples include development environments such as GitHub Copilot [20], as well as enterprise platforms like Microsoft Copilot [42], Salesforce Agentforce [63], and SAP Joule [64]. In addition, open-source projects such as Continue.dev [10] and Tabby [70] demonstrate similar approaches to embedding Artificial Intelligence (AI)-based support into development environments. These systems reflect a broader trend toward supporting users in complex tasks through context-aware, interactive assistance. However, such approaches are typically applied in general-purpose or domain-specific systems such as software development or enterprise applications and are not yet integrated into complex cryptographic learning environments like CT2.

1.2. Problem Statement

Although CT2 provides a large collection of templates for cryptographic methods, users still need prior knowledge in order to select the appropriate template for a given task. Users who are unfamiliar with specific encryption methods, cryptanalytic techniques, or cryptographic terminology may therefore struggle to identify a suitable starting point.

Users who attempt to construct their own workspaces may encounter additional difficulties. Problems such as incorrect connections, unsuitable parameter settings, hidden buttons, disabled editing options, or unexpected execution results are not always easy to understand. These challenges become even more pronounced when users want to modify an existing workspace, for example, by adding new components, changing connections, or adjusting parameters without fully understanding the consequences of these changes. While CT2 provides logs and technical error messages, these are often aimed at experienced users and may not clearly explain the actual cause of a problem. External resources, such as YouTube tutorials, are available and explain many concepts and workflows in detail. However, finding the relevant video, navigating to the appropriate section, and determining whether it addresses the specific problem can still require considerable time and effort.

In addition, CT2 provides detailed documentation for its components. These descriptions often include the purpose of the component, its functionality, connectors, component controls, templates, references, and examples of use. However, reading through the available documentation can still require considerable time, especially when users are looking for a very specific piece of information. To keep the documentation concise and manageable, highly specific questions or uncommon use cases may not always be covered in detail. [15]

As a result, particularly self-directed users may require substantial effort and prior knowledge before they are able to use CT2 effectively. This limits the usability of the platform, particularly in educational settings where users may be unfamiliar with both cryptographic concepts and the internal structure of the software.

1.3. Research Gap

Several approaches already exist for supporting the teaching and learning of cryptology. These include software tools [19], curriculum frameworks [33], competitions [21], Intelligent Tutoring Systems (ITSs) [22, 82], and AI-based tutoring systems [22, 82]. In the field of cryptology education, systems such as CryptoScratch [56] and CryptoEL [61] demonstrate how interactive learning environments can improve accessibility and learner engagement. More general educational systems such as Khanmigo [30] and Duolingo Max [17] further demonstrate the potential of conversational AI support, adaptive feedback, and personalized assistance in educational contexts.

However, such conversational AI capabilities are rarely integrated into existing learning systems for cryptology. Most tools focus on predefined exercises, templates, visualizations, or guided workflows without allowing users to ask individual questions or receive adaptive explanations. CryptoEL is one of the few exceptions, as it combines cryptographic learning content with conversational assistance. [61] However, it is primarily

designed for middle and high school students and therefore focuses on introductory concepts rather than the more advanced requirements of university-level education. In addition, CryptoEL is not yet publicly available, although the authors already indicated in 2024 that a public release was planned [60]. This limits the promises about accessibility and practical relevance given in the paper.

Overall, existing systems either provide conversational support without access to a realistic cryptographic workspace or offer advanced cryptographic functionality without interactive and context-aware assistance. As a result, there is currently no solution that combines a conversational LLM-based educational agent with direct awareness of the current CT2 workspace and its components, settings, and structure.

1.4. Research Question

The central research question is therefore:

> Can an LLM-based agent improve the usability of CT2 and reduce its entry barrier in order to better support users in exploring, understanding, and applying cryptography and cryptanalysis?

This question addresses both beginners and more advanced users. The aim is not only to support users in understanding cryptographic concepts but also to improve their ability to work effectively with CT2 itself.

To answer the main research question, the following sub-questions are considered:

- Can an LLM-based agent help users understand the purpose, functionality, and application of components in CT2?
- Can the agent support users in constructing, modifying, and interpreting workspaces in CT2, including performing selected workspace modifications such as inserting components, creating connections, or adjusting parameters?
- Can the agent help users identify suitable components, parameter settings, and workflows for specific cryptographic or cryptanalytic tasks?
- Can the integration of the agent reduce the time and effort required to solve problems compared to existing support mechanisms such as documentation, templates, tutorials, or videos?

- Does the integration of the agent provide advantages over general-purpose external chatbots that do not have access to the current CT2 workspace?

1.5. Objectives and Contributions

The agent is intended to support users directly within the application and thereby reduce the entry barrier and improve the usability of CT2 for cryptography and cryptanalysis education.

A central goal of the proposed system is to help users understand the current workspace and the role of individual components within it. This includes explaining the purpose, functionality, connectors, component controls, and typical use cases of components, as well as supporting users in understanding how workspaces are structured and how they can be modified.

To achieve this, the proposed system enables context-aware interaction with the current workspace. Through a set of callable tools, the agent can access information about the workspace, including its structure, component settings, and related documentation. In addition, the agent can perform selected modifications using existing CT2 mechanisms, such as inserting components, creating connections, or updating parameters. By combining structured workspace information with the general knowledge of modern Large Language Models (LLMs), the agent generates responses that are tailored to the current user context.

The main contributions of this thesis are as follows:

- Design and implementation of a system integrating a context-aware LLM-based agent into CT2, developed in C#.
- Support for multiple LLMs, including both locally hosted and Application Programming Interface (API)-based models, enabled through the integration of Microsoft Semantic Kernel.
- Development of a tool-enhanced agent that can access and manipulate CT2 workspaces, including components, connectors, and parameters.
- Integration of the agent into the existing CT2 user interface in a way that enables seamless interaction within the application.
- Implementation of context-window management mechanisms, including token-budget-based reduction of conversation history and tool outputs.

The primary focus of this thesis lies on the design and implementation of the proposed system. An evaluation with users was conducted as an additional step to gain initial insights into its usability and practical usefulness. As this evaluation was not originally part of the core scope of the thesis, it is intended as an exploratory study rather than a comprehensive evaluation.

1.6. Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2 provides the theoretical foundations required for this work. It introduces CT2, explains its architecture and educational features, and discusses the fundamentals of LLMs, LLM-based agents, tool-calling, and the MCP.

Chapter 3 reviews related work in the areas of AI-enhanced learning, ITSs, LLM-based educational agents, cryptology education, and existing cryptographic learning tools. It also discusses existing work related to CT2 and identifies the research gap addressed in this thesis.

Chapter 4 describes the design and architecture of the proposed system. It defines the system requirements, explains how the agent is integrated into CT2, and presents the internal architecture of the LLM-based agent, including its conversation management, tool-based interaction with the workspace, and controlled modification capabilities.

Chapter 5 presents the implementation of the system. It explains the chat interface, the integration of external frameworks, the design of the available tools, context-window management, and the interaction between the agent and the CT2 environment, including tools for modifying workspaces.

Chapter 6 describes the evaluation design and presents the evaluation results. It explains the study setup, the participant group, the evaluated tasks, and the collected feedback regarding usability, the reduction of the entry barrier, and limitations of the agent.

Chapter 7 discusses the main findings of the evaluation, reflects on limitations of the current prototype, and considers potential risks and challenges related to the use of LLM-based assistance in educational software.

Finally, Chapter 8 concludes the thesis by summarizing the main results and outlining possible directions for future work.

1.7. Working Environment

This thesis was developed in a Windows-based working environment, since CT2 is a desktop application for Windows. The implementation was carried out using the CT2 source code. The implementation setup is described in Section 5.1.

The written thesis was prepared in LaTeX [73] using LuaLaTeX and TeX Live 2025 [72]. The document was edited in Overleaf [54] hosted within Sciebo [66]. Bibliographic references were managed using Citavi 7 [38].

AI tools were used during the preparation of this thesis as documented in Table 6. The tools and setup used specifically for the evaluation are described separately in Section 6.1.

2. Foundation

This chapter introduces the theoretical and technical foundations relevant to this thesis. It first presents basic concepts of cryptology and introduces the CrypTool (CT) project and CT2 as the environment in which the proposed system is integrated. Finally, it discusses the workspace model, plugin-based architecture, educational characteristics of CT2, and the fundamentals of LLMs, LLM-based agents, and tool calling.

2.1. Cryptography and Cryptanalysis Fundamentals

Cryptology is the broader field concerned with secure communication and information protection. It is commonly divided into cryptography and cryptanalysis. While cryptography focuses on protecting information, cryptanalysis examines how such methods can be attacked, evaluated, or broken. [40, 55] [19, p. 22]

The primary objectives of cryptographic systems are confidentiality, integrity, authenticity, and non-repudiation. Confidentiality ensures that information can only be read by authorized parties, integrity protects data from unauthorized changes, authenticity verifies the identity of communication partners, and non-repudiation prevents a party from denying a previously performed action. [40]

In symmetric cryptography, the same secret key is used for both encryption and decryption. Examples include the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES). Asymmetric cryptography uses two different keys for encryption and decryption: a public key and a private key. Common examples include Rivest-Shamir-Adleman (RSA) and Elliptic-Curve Cryptography (ECC). In addition to encryption methods, cryptography also includes hash functions, digital signatures, and protocols for secure communication. [55, 69]

Cryptanalysis aims to identify weaknesses in algorithms, implementations, or configurations in order to recover protected information or keys. Techniques include brute-force attacks, frequency analysis, side-channel attacks, or attacks based on weaknesses in protocol design. In practice, cryptanalysis is important not only for attackers but also for evaluating the security of cryptographic systems and improving their design.

CT2 supports both cryptographic and cryptanalytic methods. Users can experiment with encryption and decryption methods, analyze ciphertexts, investigate weaknesses in algorithms, and study the behavior of cryptographic protocols. As a result, CT2 can be used not only to understand how cryptographic methods work but also to explore how and why they can fail.

2.2. CrypTool Project

CT is an educational initiative designed to support the teaching and learning of cryptology. It provides interactive tools that allow users to explore cryptographic algorithms and cryptanalytic techniques. The project is hosted by the Research Institute CODE and supported by several academic institutions [12]. The CT project comprises many software projects, most notably CT1 and CT2, as well as CT-Online, JCrypTool, and CrypTool Transcriber & Solver. In addition, the initiative includes educational and community projects such as the pupil and teacher events "Schülerkrypto", the "Lernkurs Klasse 11" CT-Learn, the cryptographic puzzle platform MysteryTwister, and the CT Portal, which serves as the main entry website.

The development of the CT software began in 1998 as an awareness project for employees of Deutsche Bank, a major German multinational investment bank. Due to its internal success, the IT board approved in 2000 that the software be released as freeware. In 2003, CT was released as an open-source project. From that point onward, both the CT program and its source code were publicly available. CT has since been continuously developed and maintained by a broad community. [35] All CT programs can be downloaded free of charge via the official project website [11]. In addition to providing software tools, the project also offers educational resources that support self-learning and the teaching of cryptology.

2.3. CrypTool 2

CrypTool 2 is one of the main software applications of the CT project. It is an open-source Windows program designed for the exploration and visualization of cryptographic algorithms and cryptanalysis methods. [15] The software is implemented in C# using the Windows Presentation Foundation (WPF) framework and serves as an interactive e-learning environment for studying modern cryptology.

CT2 provides a large collection of components and templates that allow users to experiment with various cryptographic techniques, including encryption, decryption, and cryptanalytic attacks. Each component is accompanied by documentation and background information to support the learning process. [13]

A central feature of CT2 is its graphical user interface for visual programming. Cryptographic processes are modeled within a workspace as interconnected components that exchange data during execution. These workspaces can be visually inspected, modified, and executed, allowing users to observe how data is processed step by step. [13]

2.3.1. User Interface and Interaction Model

When CT2 is launched, the user is presented with the Startcenter, which serves as the central entry point to the application. As shown in Figure 1, the Startcenter provides direct access to the main functions of the system, including creating a new workspace or launching the Wizard, a step-by-step assistant (left navigation bar), opening previously created user workspaces (bottom), and loading application-provided templates (center).

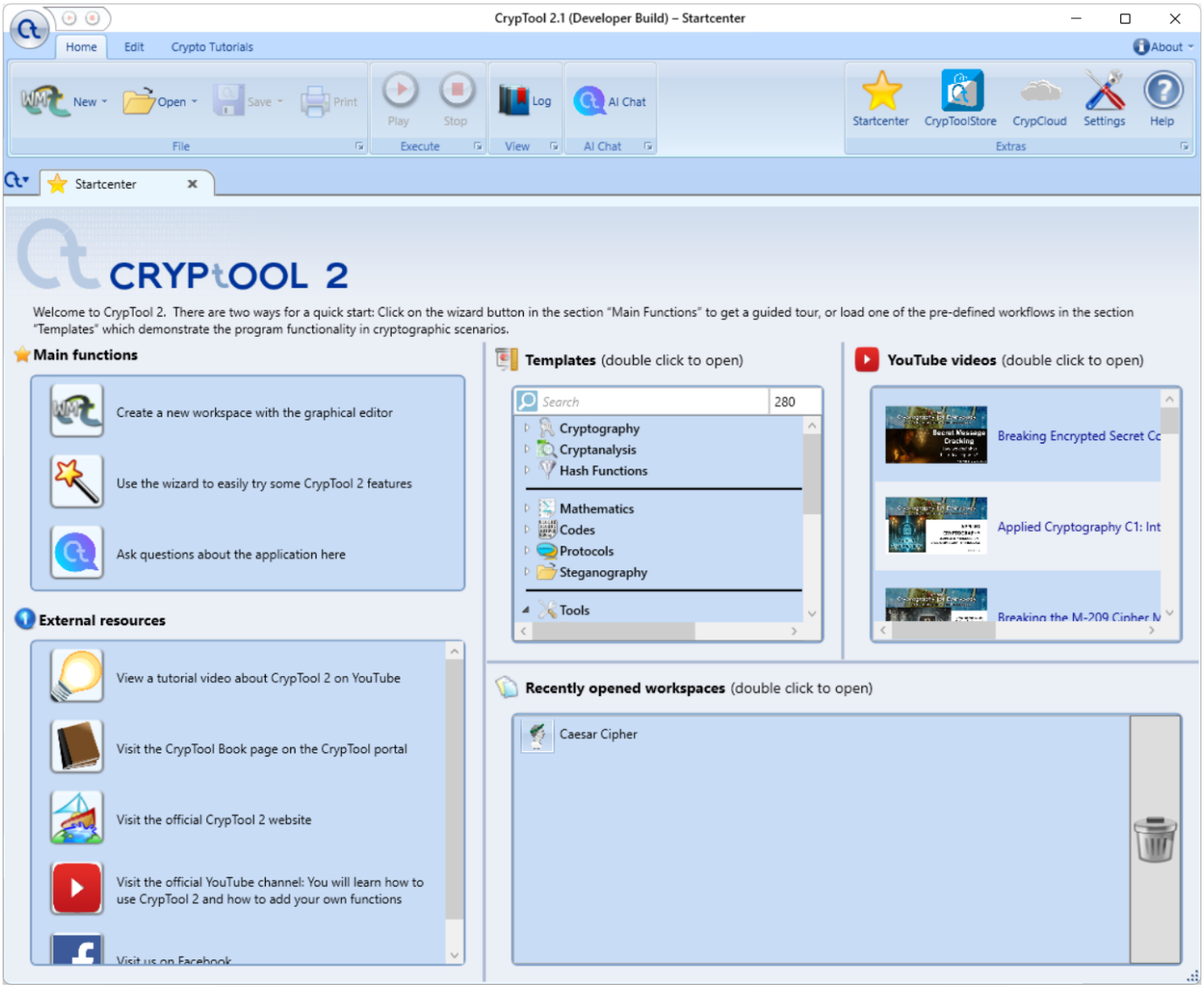

Fig. 1: CrypTool 2 Startcenter providing access to workflow templates, main functions, and learning resources.

The templates are organized into categories covering different areas of cryptology, such as cryptography, cryptanalysis, and hash functions. These templates contain predefined

configurations of components that demonstrate cryptographic algorithms and analysis techniques, allowing users to quickly explore the functionality of the system.

In addition to templates, the Startcenter provides links to external resources such as tutorials, documentation, and instructional videos that support the learning process. CT2 also includes integrated help mechanisms such as component documentation, online help pages, and the Wizard. The Wizard is a step-by-step assistant that helps users identify suitable templates through a decision-tree-like process, thereby reducing the complexity of navigating the large number of available components and templates. [36] [19, p. 762 ff.] After selecting or creating a workspace, the user is taken to the graphical workflow editor, where cryptographic and cryptanalytic tasks can be constructed and modified.

2.3.2. Workspace Model

In CT2, cryptographic and cryptanalytic tasks are constructed within a workspace that represents a directed graph of interacting elements. The workspace acts as a container that manages all components involved in a task, as well as their relationships. Figure 2 shows an example of such a workspace in the graphical user interface.

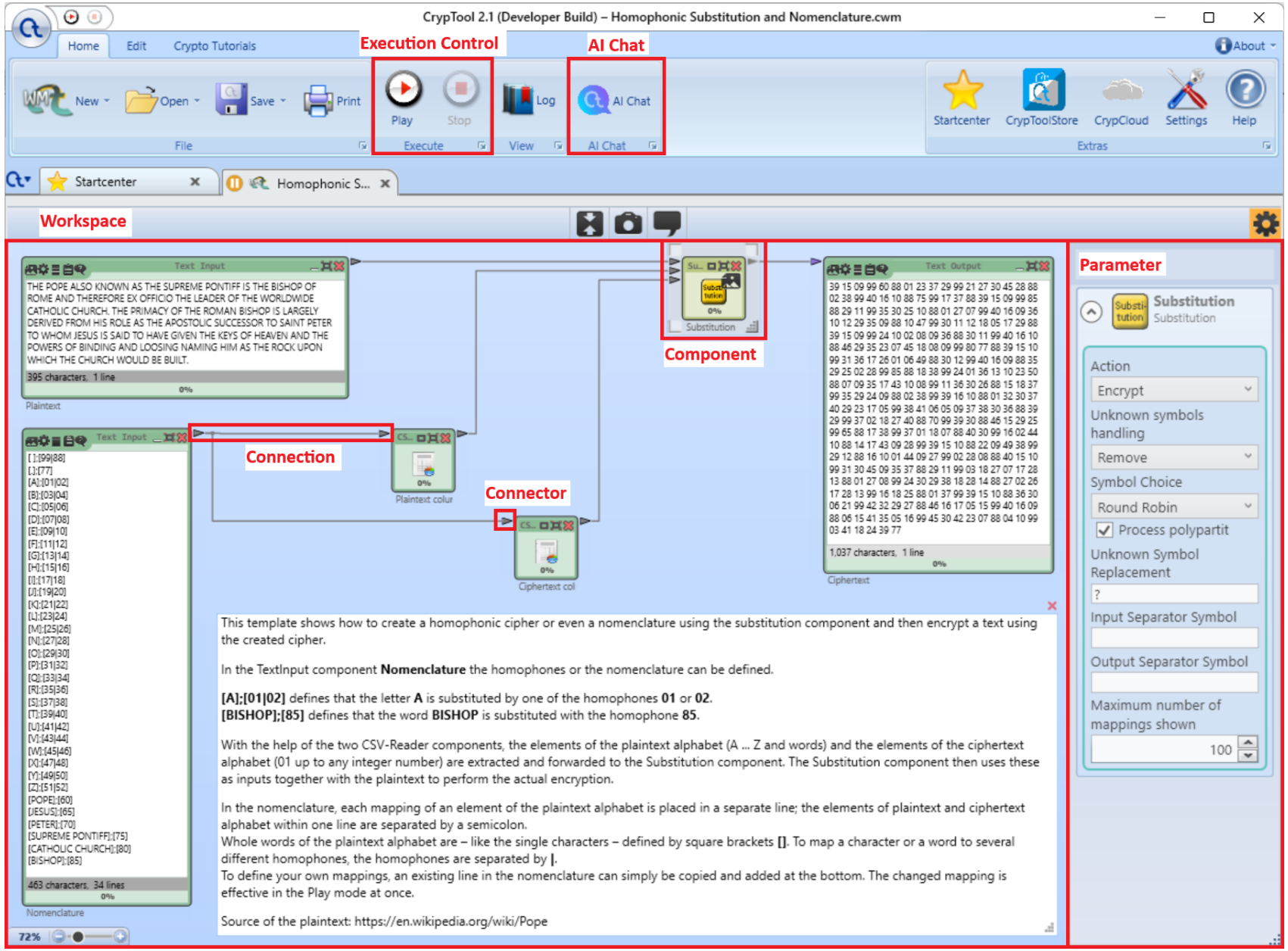

Fig. 2: Screenshot of the CrypTool 2 workspace showing the template "Homophonic Substitution Cipher and Nomenclature – Encryption" with highlighted interface elements.

Components placed in the workspace correspond to functional units that process data, such as cryptographic algorithms, analysis tools, or auxiliary operations. Depending on their purpose, components may provide input connectors, output connectors, or both. These connectors define how data can enter or leave a component. Connections between connectors establish directed edges along which data is transferred between components. The conceptual structure of these elements is illustrated in Figure 3.

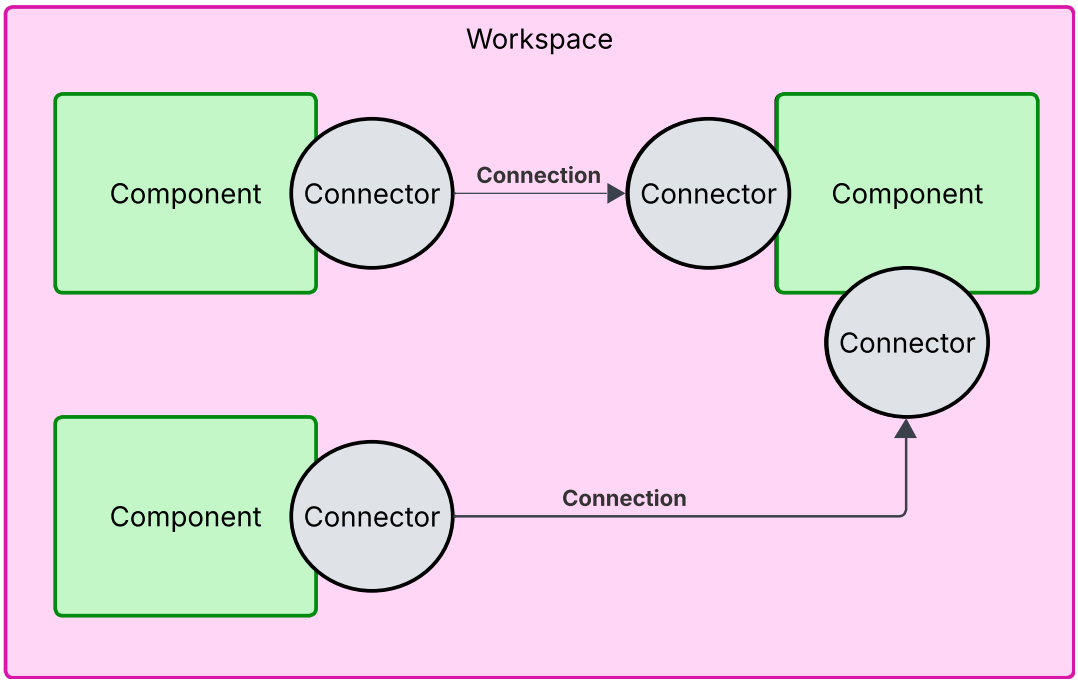

Fig. 3: Conceptual structure of a CrypTool 2 workspace showing components, their connectors, and their connections.

Together, components, connectors, and connections form a directed graph that describes how data is processed during execution. [34] Data produced by one component becomes available to connected components, enabling sequential or parallel processing steps. This model allows complex cryptographic and cryptanalytic processes to be constructed by combining simple functional units while maintaining a clear visual representation of the data flow.

2.3.3. Workspace Manager

The workspace manager is the central component of CT2 responsible for displaying, editing, and executing workspaces. It provides the graphical programming environment in which the conceptual workspace model described above is visualized and manipulated.

Within this environment, components are represented as visual elements that can be placed, connected, and configured by the user. Interactions such as drag-and-drop operations, the creation of connections, and parameter adjustments are handled by the workspace manager [35].

In addition, the workspace manager controls the execution of the workspace by instantiating and coordinating the components defined in the model. During runtime, it provides feedback about warnings, errors, and execution progress.

Furthermore, the workspace manager offers several features for working with complex workspaces. Components can be displayed in different views, including presentations, settings, data views, and logs. Users can zoom within the workspace, select multiple components, and add text and image elements to document their workspaces.

2.3.4. Plugin-Based Architecture

CT2 follows a modular architecture in which most functionality is implemented through plugins. Cryptographic algorithms, analysis methods, visualization tools, and auxiliary functions are encapsulated in independent plugin modules, allowing the system to be extended without modifying the core platform.

Plugins do not operate directly within the workspace. Instead, they provide one or more components that can be instantiated and used within a workspace. In this context, plugins represent the underlying implementation units that define and supply functionality, while components correspond to the visual and functional elements that are instantiated and manipulated by users within the graphical user interface.

This means that plugins are primarily relevant at the implementation level, whereas users interact exclusively with components when constructing and modifying workspaces. This separation enables a single plugin to supply multiple related components and supports the integration of complex functionality. At runtime, plugins are loaded once by the system, whereas components can be instantiated multiple times within a workspace and configured independently.

The relationship between plugins and components has evolved over the course of the development of CT2. In earlier versions of the system, the terms “plugin” and “component” were often used synonymously, since each plugin typically corresponded directly to a single element in the workspace. [34] In the current architecture, plugins serve as underlying modules that can provide multiple components, increasing flexibility and extensibility.

2.3.5. Educational Features and Current Interaction Limitations

CT2 is designed as an e-learning platform for cryptology and is used in educational, training, and research contexts. Its visual and component-based approach allows users to explore cryptographic algorithms and cryptanalytic techniques interactively rather than only through theoretical descriptions.

By modifying parameters, executing workspaces, and observing intermediate results, users can experiment with different configurations and directly compare their effects. The visual structure of workspaces also supports self-learning and classroom teaching, since intermediate results and algorithmic behavior can be inspected directly. In addition, CT2 provides templates, component documentation, tutorials, videos, and online help resources that further support the learning process.

Despite these educational features, the current CT2 primarily relies on traditional interaction mechanisms such as graphical editing, static documentation, templates, tutorials, and log messages. While these resources provide valuable support, they require users to actively search for relevant information and interpret it on their own, which can become difficult for beginners in more complex workspaces.

2.4. Large Language Models

LLMs have emerged as powerful components for processing natural language and enabling new forms of human-computer interaction. An LLM is a mathematical model designed to understand and generate human language. [57] These models are trained on large and diverse text corpora, allowing them to handle a wide range of topics and tasks. [67, 47] LLMs are predominantly based on the Transformer architecture. [59, 75] They learn statistical patterns of language from large text corpora so they can generate coherent and contextually appropriate responses. [5] In interactive applications, pretrained models are often further refined to better follow user instructions and conversational behavior. [53]

LLMs can understand natural language instructions and generate explanations, summaries, recommendations, or answers to questions. They are able to adapt their responses to the current context of a conversation and can provide information across a wide range of domains. [84] These capabilities make LLMs suitable as conversational interfaces for software systems, where users interact with the system through dialogue-like natural language exchanges. In the context of this work, they can help users understand components, interpret workspaces, and retrieve relevant information about CT2 through natural language interaction.

Despite their capabilities, LLMs have several limitations. Because they rely on pretrained knowledge, they may provide outdated or incomplete information. [32] In addition, they can generate hallucinations, meaning that they may produce incorrect or fabricated information while presenting it confidently. [23, 27] Furthermore, the quality of the generated response can depend strongly on the phrasing of the input. [83] In addition, LLMs are limited by the size of their context window, meaning that only a restricted amount of information can be processed at once. These limitations are particularly important in educational settings, where users may rely on the correctness of the provided information. For this reason, LLMs are often combined with external tools and application-specific data, for example, to retrieve current information or perform calculations.

2.5. LLM-Based Agents

LLM-based agents are software entities that embed LLMs within a broader system architecture to perform tasks with a degree of autonomy. Unlike standalone LLM applications that primarily generate responses to user prompts, such agents can pursue objectives, maintain contextual state, and interact with external systems. In general, they interact with their environment through tool use. [8, 76]

The scope and definition of LLM-based agents vary in the literature, particularly with regard to their degree of autonomy. Common capabilities include memory, tool use, and goal-directed behavior, which allow agents to perform tasks beyond simple text generation.

For this thesis, the agent is designed not only to provide explanations and recommendations based on user input and contextual information but also to support workspace editing through controlled tool calls. In addition to answering questions and providing guidance, it can actively modify the current workspace through controlled tool calls.

According to [8], LLM-based single-agent systems typically exhibit the following core capabilities:

1. **Planning**: The agent can determine which steps or actions are required to achieve a given objective and can decompose more complex tasks into smaller subtasks.

2. **Memory**: The agent can use previous interactions, contextual information, and stored knowledge as part of its reasoning process. Memory may include short-term context, such as the current conversation history, as well as long-term knowledge stored in external systems.

3. **Rethinking**: The agent can evaluate intermediate results and reconsider previous decisions based on newly available information or tool outputs.

4. **Environments**: The agent can interact with different types of environments and use information from them as part of its reasoning process. These environments may include computer, code, real-world, simulation, or gaming environments.

5. **Action**: The agent can perform actions and use tools to interact with external systems. These actions may include retrieving information, calling APIs, executing code, performing calculations, or controlling software systems.

Not all LLM-based agents implement these capabilities to the same extent. Depending on the application domain, some capabilities may play a more important role than others. In the system developed in this thesis, memory, environment interaction, and action through tool use are particularly relevant, as the agent uses conversational and workspace context and can perform selected operations within CT2.

2.5.1. Tool-Calling

LLM-based agents can invoke tools to access information or perform operations that are not directly available through the language model itself. Tools can provide functions such as information retrieval, numerical computation, code execution, or interaction with software systems. [76]

In modern agent architectures, tools are typically exposed through structured interfaces that define their functionality, input parameters, and outputs. During operation, the agent determines whether a tool is required, generates an appropriate tool call, and incorporates the returned result into its subsequent reasoning. This often results in an iterative process in which reasoning steps alternate with tool usage until a satisfactory solution is reached. [81, 65, 76]

Tool-calling separates natural language reasoning from action execution. While the language model decides which action should be taken, the external system performs the actual operation and returns a verifiable result. This improves reliability, reduces hallucinations, and enables the agent to interact with external environments while maintaining a conversational interface. [81, 65, 58, 76]

In application-specific systems, tool-calling is particularly important because tools can expose selected internal program functions to the agent and enable controlled interaction with the software environment. In this thesis, this principle is applied to CT2: the user interacts with the agent through the chat interface, while the tool functions are provided by the application itself. The agent may invoke these tools to retrieve workspace

information or perform selected operations, but the actual execution remains within the controlled CT2 application layer rather than giving the agent unrestricted access to internal data structures.

2.5.2. Model Context Protocol

The Model Context Protocol (MCP) is an emerging open standard for connecting LLMs and LLM-based agents with external tools, systems, and data sources. [48] It defines a client–server architecture in which an LLM-based application acts as an MCP client and communicates with one or more external MCP servers. [52] These servers expose their available tools, resources, and prompts through a standardized interface. The client can first discover which capabilities are available and can then invoke them in a structured way by providing the required parameters and processing the returned results.

Although MCP is not used in the implementation presented in this thesis, it is relevant as a possible future extension. The prototype instead uses a custom tool interface integrated directly into CT2, because the required tools operate on internal application structures such as the active workspace, components, connectors, and settings. The decision to use this custom interface rather than an MCP-based integration is discussed in Section 4.8. In principle, CT2 could later expose selected functions through an MCP server or connect to external MCP-compatible systems. [68]

3. Related Work

This chapter reviews related work relevant to the integration of an LLM-based agent into CT2. It first discusses AI-enhanced learning systems, ITSs, and LLM-based educational agents. Subsequently, existing approaches for cryptology education and related learning tools are examined. Finally, the chapter considers recent developments around CT2 and identifies the research gap addressed in this thesis.

3.1. AI-Enhanced Learning and Intelligent Tutoring Systems

AI is increasingly transforming many aspects of modern society, including education. **AI-enhanced learning** is a broad term describing the use of AI-based technologies to support teaching and learning processes through automation, personalization, and data-driven insights. Traditional educational settings often struggle to address differences in prior knowledge, learning speed, and individual support needs, particularly in large classrooms or online environments. Consequently, there is a growing demand for technological solutions that can provide scalable, individualized learning support. Examples include adaptive learning platforms that adjust task difficulty, automated feedback tools that provide immediate responses, learning analytics systems that identify knowledge gaps, recommendation systems that suggest relevant materials, and conversational agents capable of interactive assistance. [22, 82]

A particularly important and long-established subgroup of AI-enhanced learning systems is ITSs. ITSs have been studied for several decades and existed long before the recent rise of generative AI and large language models. Traditional ITSs are typically based on rule-based systems, predefined knowledge representations, and explicit learner models. A typical ITS architecture comprises a domain model representing expert knowledge, a student model estimating the learner's current understanding, a tutoring (or pedagogical) model that selects instructional strategies, and a user interface that enables interaction. Based on these components, ITSs can diagnose misconceptions, adapt task difficulty, provide step-by-step guidance, and deliver individualized feedback. [79]

Advantages of AI-based tutors include the ability to provide individualized support at scale, deliver immediate feedback, remain available at all times, and maintain consistent instructional behavior across users. Interaction data can further be used to refine models and improve future recommendations, making such systems particularly suitable for self-directed learning and large-scale educational settings. [29]

Despite these benefits, AI-enhanced learning systems also face significant challenges. The quality of generated explanations may vary, and incorrect or misleading responses can negatively affect learning outcomes. Biases in training data, limited pedagogical

understanding, lack of transparency, and privacy concerns related to learner data further complicate the deployment of such systems. [74, 29]

Recent advances in generative AI, particularly LLMs, have introduced a new generation of digital tutors capable of engaging in natural dialogue, generating explanations on demand, and adapting responses to individual learners in real time. LLM-based tutors differ from earlier rule-based ITSs in that they can handle open-ended questions and provide flexible assistance across a wide range of topics. These developments highlight the potential of agent-based systems to deliver context-aware, interactive support within complex learning environments such as CT2. [29, 3, 77]

3.2. LLM-Based Agents for Educational Applications

Traditional ITSs and other AI-enhanced learning systems often provide only limited contextual understanding, restricted interactivity, and little flexibility in handling open-ended questions. LLM-based agents aim to overcome these limitations by integrating capabilities such as memory, tool use, and planning. In particular, they can process substantially richer contextual information and incorporate it into their responses, enabling more adaptive and individualized learning interactions. In this sense, LLM-based educational agents can be regarded as a newer generation of ITS-like systems that extend traditional tutoring approaches through natural language interaction, flexible reasoning, and access to external tools.

According to Chu et al., educational LLM agents can be broadly divided into pedagogical agents and domain-specific educational agents. Pedagogical agents support general teaching and learning activities, for example, through automated feedback or adaptive tutoring. Domain-specific agents, in contrast, are tailored to specific subject areas such as mathematics, language learning, or medical education. By specializing in a particular domain, these agents can provide more accurate explanations, targeted practice, and context-aware guidance. [9]

Despite their enhanced capabilities compared to traditional AI-supported systems, LLM-based educational agents still face significant challenges. First, privacy concerns arise because these systems often process sensitive personal data, including performance metrics and behavioral information. Second, hallucinations—plausible but incorrect outputs generated by LLMs—remain a critical reliability issue, although the integration of external tools and retrieval mechanisms can mitigate their frequency. Third, there is concern that excessive reliance on AI support may inhibit the development of independent problem-solving skills among learners. Finally, the large-scale integration of such systems into educational institutions remains limited due to insufficient infrastructure, lack of standardized deployment frameworks, and unequal access to AI technologies across institutions. [9]

Examples of such systems include *Khanmigo* [30], developed by Khan Academy, an AI-powered personal tutor and teaching assistant for teachers, learners, and their families. It can generate lesson plans, prepare quizzes, and provide guided problem-solving support, demonstrating agent-like capabilities such as contextual awareness, task automation, and personalized assistance. Another example is *Duolingo Max* [17], which offers interactive role-play scenarios and video-based conversations that allow learners to practice realistic dialogues and receive immediate feedback. [18] While primarily conversational, these features exhibit domain-specific agent characteristics through adaptive interaction and task-oriented language practice and can therefore be considered LLM-based educational agents specialized in language learning.

Yao et al. propose instructional agents, a multi-agent framework that supports educators in generating teaching materials such as curricula, lecture notes, slides, and assessment tasks. The framework combines multiple specialized agents and supports different levels of human supervision, ranging from autonomous execution to a copilot mode. Although this approach focuses on teacher support rather than student learning, it illustrates the broader potential of LLM-based agents in educational settings. [80]

However, there is still limited research on LLM-based educational agents for highly specialized domains such as cryptography and cryptanalysis, particularly in environments that combine conversational support with interactive visual programming.

3.3. Cryptology Education and Learning Tools

Cryptology education is important for helping learners understand the principles of secure communication, data protection, and cybersecurity. Because many cryptographic concepts are abstract and mathematically complex, practical examples and interactive learning environments are often required to make them easier to understand. Several studies emphasize that cryptographic principles should be taught using realistic scenarios and hands-on activities to improve understanding and motivation. [7]

Various approaches have been proposed to support cryptology education, including dedicated software tools, curriculum frameworks, competitions, and AI-based tutoring systems. Among programming-oriented learning environments, CT2 represents a comprehensive platform for experimenting with classical and modern cryptographic algorithms. In contrast, *CryptoScratch* targets primarily K–12 students, meaning students in the school system from kindergarten to 12th grade, and extends the graphical programming language Scratch with cryptographic primitives such as AES and RSA. [56] By leveraging a block-based interface familiar from school settings, the tool lowers the barrier to entry while still enabling meaningful learning outcomes. [62] In addition, curriculum proposals have been formulated to systematically integrate cryptology into school ed-

ucation. For example, one framework explicitly incorporates hands-on activities using CT2 to support experiential learning in cybersecurity contexts. [33]

Beyond programming tools and formal curricula, gamification has also been explored as a means to increase engagement and motivation. Several studies investigate game-based approaches to introduce fundamental cryptographic concepts in an understandable and engaging manner. [24] A related and widely used approach in cybersecurity education is Capture-the-Flag (CTF) platforms, in which learners solve hands-on challenges in areas such as cryptography, web security, reverse engineering, or forensics. Such platforms combine competition with practical problem solving and have been shown to improve engagement, technical skills, and motivation. [28] Well-known examples include picoCTF and pwn.college.

More recently, web-based interactive systems have emerged that combine visualization, simulation, and conversational support. *CryptoEL* is one such system designed for K–12 learners, integrating interactive scenarios with AI-assisted dialogue to teach core concepts such as hashing, symmetric cryptography, and asymmetric cryptography. [61] Evaluation results indicate that students can achieve a solid understanding of these principles through guided exploration. However, the tool is not yet publicly available, limiting its accessibility and reproducibility.

In addition to educational software and curricula, competitions play an important role in promoting interest in cryptology. The *NSUCRYPTO* Olympiad (Non-Stop University CRYPTO) has been organized annually since 2014 as an international contest presenting unusual and often unsolved problems in cryptography. By challenging participants to solve authentic research-level tasks, the competition aims to inspire talented students and increase awareness of the field. [21]

Summary. Overall, existing approaches demonstrate that effective cryptology education benefits from interactive tools, practical exercises, and motivating formats such as games or competitions. Nevertheless, many solutions either target beginners with simplified environments or remain external to professional-grade software systems. The integration of intelligent assistance directly into a comprehensive cryptographic platform, such as CT2, therefore represents a promising direction to support learners across different expertise levels. This motivates the work presented in this thesis, which investigates how an LLM-based agent can enhance cryptology education within an authentic, feature-rich environment.

3.4. CrypTool 2: Platform and Recent Developments

As discussed in the foundations chapter, CT2 is a comprehensive platform for experimenting with cryptographic and cryptanalytic methods. Compared to many educational

tools, it offers a significantly broader range of functionality, including attack simulations and protocol analyses within a single environment. [15, 35]

However, this flexibility also introduces usability challenges. For novice users, it may be difficult to identify suitable components, understand the meaning of connectors, or determine why a workspace does not execute as expected. In addition, complex templates with many visible components can quickly become overwhelming, especially when users are confronted with unfamiliar terminology or densely connected structures. Newly inserted components often appear as small windows in which important fields and settings are not immediately visible. Users may not realize that these windows can be expanded, resized, or moved. Components may also appear outside the currently visible workspace area, causing users to lose track of the overall structure. Furthermore, execution controls such as the play button, progress indicators, and error messages are not always immediately obvious, making it difficult to understand whether a workspace is still running, has failed, or requires additional actions. As a result, the graphical user interface can appear complex and may increase cognitive load during the initial learning process. [39]

At the same time, several recent student projects demonstrate the educational potential of extending CT2 with more specialized and interactive functionality. Bender developed an interactive tutorial for differential cryptanalysis that guides users through the individual steps of the attack process using toy ciphers, visualizations, and dedicated components. [2] Addad implemented numerous image and text steganography methods together with graphical presentations that allow users to experiment with different configurations and better understand the underlying techniques. [1] Weimann added a range of historical hand ciphers and codes to CT2, including dedicated visualizations and a simulated annealing attack against the Josse cipher. [78] These projects illustrate that CT2 continues to evolve through new educational extensions and interactive visualizations. They also show that the platform is increasingly used not only to demonstrate cryptographic algorithms but also to support guided learning and experimentation.

Recent research directions suggest that embedding conversational assistants directly into complex software environments could help reduce such usability barriers while preserving functionality. In the context of CT2, an agent could provide contextual guidance, explain components and connectors, propose suitable workspace structures, assist in diagnosing configuration problems, and support users in modifying workspaces through natural language instructions. [9, 3]

By leveraging modern LLMs, such a system could additionally provide conceptual explanations, recommend suitable cryptographic approaches, answer questions based on the current workspace state, and support selected editing actions such as inserting components or creating connections. This would be particularly valuable because existing educational tools often either simplify functionality or provide conversational support detached from the operational software environment.

At the time of writing, no prior work appears to exist that integrates an LLM-based conversational agent directly into a professional cryptographic software environment for educational purposes to provide context-aware support.

4. System Design and Architecture

This chapter presents the design and architecture of CrypLLM, the proposed LLM-based agent for CT2. It first defines the design objectives and system requirements, then explains the integration into the existing CT2 architecture and the internal structure of the agent. Finally, it discusses security, privacy, and the main architectural decisions and trade-offs.

Figure 4 provides a simplified conceptual overview of the CrypLLM integration at application level. It shows the LLM-based agent as the central runtime component of CrypLLM inside CT2. The agent interacts with the active workspace through embedded inspection and modification tools; these tools are part of the agent subsystem and do not represent a separate tool server or external process. The agent communicates with an external or locally hosted LLM backend through an API.

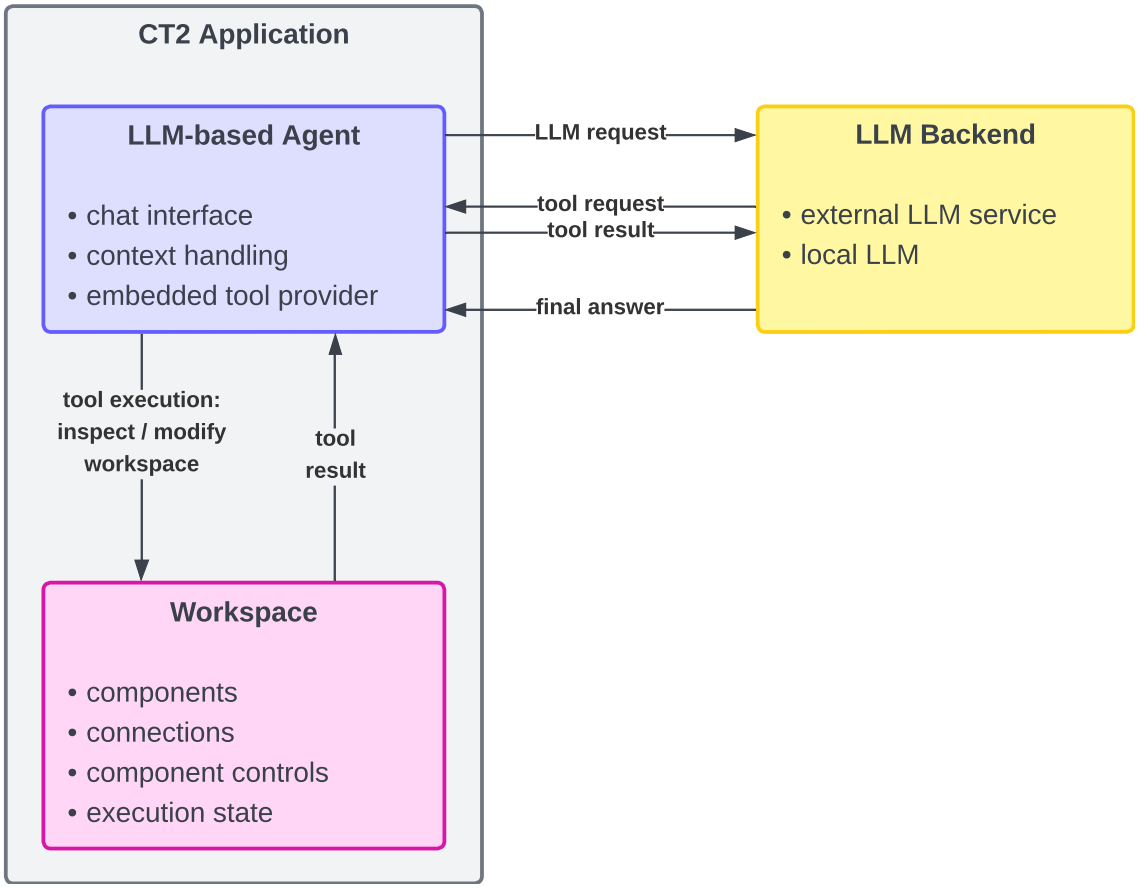

Fig. 4: Simplified architecture of the CrypLLM integration into CT2.

4.1. Design Objectives and System Requirements

This section defines the primary objectives and requirements for integrating an LLM-based agent into CT2. The central goal of the system is to provide context-aware conversational support that helps users understand and work with cryptographic and cryptanalytic tasks within the software environment. In particular, the agent is intended to support users in interpreting the structure and behavior of workspaces, understanding the purpose of individual components and connections, and navigating the functionality of CT2. The agent is therefore designed to analyze the current application state and provide explanations, guidance, and recommendations based on the active workspace and the user's questions.

4.1.1. Functional Requirements

The system shall not only provide explanations but also support the active modification of the current workspace. Users shall be able to instruct the agent to insert, delete, or move components, create or remove connections, and update component parameters. In addition, the agent shall be able to start and stop workspace execution and wait for execution results before continuing its reasoning.

Workspace modifications shall be executed directly through the existing mechanisms of the workspace manager, as described in Chapter 2. The agent shall therefore use the same internal operations that are available for manual user interactions.

The system shall support multi-step modifications in which several actions are combined into a larger workflow. For example, the agent may insert multiple components, create connections, update parameters, and execute the resulting workspace as part of a single request. Such chained operations are common in tool-using LLM-based agents. [76]

The system should identify the currently active tab and extract only the information associated with that tab. If the active tab contains a workspace, only the information required to generate a context-specific response should be transmitted.

The system should support both external and locally hosted LLMs. Compatibility with local models is required so that users can benefit from the extension without incurring additional API costs.

The amount of transmitted context should generally be minimized. Reducing the number of tokens lowers API costs and improves compatibility with local models that provide smaller context windows.

The agent shall also support the identification and correction of workspace errors. This includes detecting missing connections, invalid parameter settings, or incomplete configurations and modifying the workspace accordingly. Since such corrections may require several reasoning steps, the agent may perform them over multiple interaction cycles. For example, the agent may first identify a missing connection, then recognize that additional parameter adjustments are required before the workspace can be executed successfully.

All tools that modify the workspace shall be configurable and can be disabled if required. This allows the system to operate either as a purely advisory agent or as an interactive editing agent.

4.1.2. Non-Functional Requirements

Responses generated by the LLM should be delivered with low latency to preserve interactive usability. Explanations should remain understandable for users with different levels of prior knowledge and should emphasize clarity and educational value.

The chat interface should support multi-turn conversations so that users can ask consecutive questions while preserving contextual coherence within the limits of the model context window. The integration must not reduce the extensibility of CT2. In particular, the existing plugin-based architecture should remain unchanged so that additional components can be integrated without modifications to the core system.

The system should explicitly manage the limited context window of the selected LLM. Long conversation histories, large tool outputs, and tool definitions should be reduced or restricted when necessary to avoid request failures. This is especially important for locally hosted models, which often provide smaller context windows than current cloud-based models.

When communicating with an external LLM service, the system should transmit only the information required to generate a response. Sensitive workspace data should be excluded whenever possible to reduce the risk of exposing locally processed information to external services.

4.2. System Overview

At the top of the architecture, CrypWin acts as the presentation layer and main application shell. It contains the main application window, the ribbon interface, and the docking system used to display different editors and views.

Figure 5 provides an overview of the layered architecture of CT2 and the position of the CrypLLM subsystem within the application.

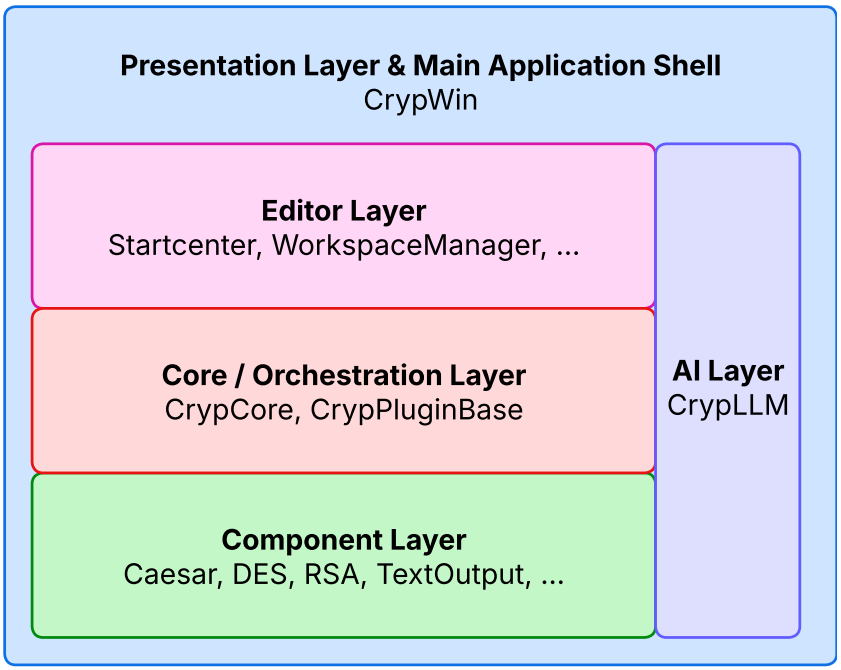

Fig. 5: Layered architecture of CrypTool 2 with the separate CrypLLM AI layer.

Below the presentation layer, the editor layer contains the main working environments of CT2. This layer includes the Startcenter, the WorkspaceManager, and additional editors. The Startcenter serves as the entry point of the application and provides access to templates, tutorials, and recently opened files. In the context of this work, the WorkspaceManager is the most important editor. It provides the visual canvas for workspace graphs and controls their execution. In addition, it is responsible for displaying components, managing connections, and executing workflows. [15]

The core/orchestration layer provides the shared infrastructure used by the editor layer and the cryptographic plugins. This layer includes CrypCore and CrypPluginBase, which define common interfaces, execution mechanisms, and shared services. The component layer contains the actual cryptographic plugins, such as Caesar, DES, RSA, or TextOutput. These plugins are instantiated and executed by the WorkspaceManager and remain independent from the LLM-based agent. [15]

CrypLLM forms a separate AI layer within the application. It is implemented as an independent project within the CT2 solution and contains the agent logic, tool plugins, conversation management, and communication with external LLM services. The chat interface is integrated into CrypWin, while the remaining functionality remains part of the CrypLLM subsystem.

Overall, the architecture preserves the existing separation between the presentation layer, the editor layer, the core/orchestration layer, and the component layer. The CrypLLM AI layer extends this structure with conversational interaction and controlled workspace access without requiring changes to the existing plugin architecture.

4.3. Integration into CrypTool 2

As outlined in the previous section, CrypLLM creates a separate AI layer within the overall CT2 architecture. Unlike CT2 plugins that provide components for use inside workspaces, CrypLLM is not instantiated as a workspace component and is not executed as part of a workspace graph. Instead, it is implemented as a dedicated subsystem that is integrated directly into the application codebase and operates at a higher architectural level.

CrypLLM is implemented as a separate project within the CT2 solution. It contains the conversation management, tool execution, context processing, and communication with external LLM services. Only the chat interface is embedded into CrypWin as a docking panel, while the remaining logic remains part of the CrypLLM subsystem.

The subsystem primarily interacts with the WorkspaceManager through controlled interfaces. It can access information about the current workspace, inspect available plugins, and perform selected modifications such as inserting components, changing parameters, or creating connections. These operations use the same internal mechanisms as manual user interactions. Communication with CrypWin and the active editor is handled through dedicated adapter classes, which translate generic CrypLLM requests into calls to the corresponding CT2 functionality.

4.3.1. User Interface Integration

The chat interface is integrated into the main application window of CT2 as a dedicated docking panel. The visual chat control is implemented in the CrypLLM project as a separate WPF user control, while the actual docking window is defined and managed within CrypWin. Its visibility can be controlled through ribbon buttons and settings, and it is not shown by default at startup unless configured otherwise. Since CT2 already uses docking panels extensively, this integration preserves interface consistency.

The docking panel is only the visible part of the subsystem. Conversation management, tool execution, and communication with external LLM services remain part of CrypLLM and continue to run independently of whether the panel is currently visible.

Interaction with the agent is possible at any time and does not interfere with the execution of workflows or other application functions. The chat interface operates independently of the current editing state but remains context-aware. It can access information from all open tabs, while responses are primarily grounded in the currently active editor and workspace.

Conversations remain available across application sessions and can be reopened after restarting the program. The agent follows a reactive interaction model in which it only responds to explicit user input and does not initiate actions autonomously.

4.3.2. System Boundary

From a system perspective, the agent is part of the CT2 client application. All components responsible for user interaction, context acquisition, and coordination with the host environment operate within the same process as the main application.

External to this boundary are the language model backends themselves, which may be hosted remotely or executed locally. Communication with these services occurs through standard APIs. CrypLLM therefore depends on external LLM services while remaining an integrated subsystem of CT2.

Overall, the integration approach enables the agent to function as a context-aware extension of the application rather than as an independent tool. It leverages existing infrastructure for workspace management, user interaction, and execution control while augmenting the system with conversational capabilities based on external LLM services.

4.4. Architecture of the LLM-Based Agent

While the previous section described the integration of CrypLLM into CT2, this section focuses on the internal structure of the subsystem itself. CrypLLM implements an LLM-based agent that runs within the application process and interacts with the host environment through controlled interfaces.

4.4.1. Agent Overview

CrypLLM is implemented as a dedicated subsystem within the CT2 application and forms the core of the integrated LLM-based agent. It is based on the Microsoft Semantic

Kernel framework, which provides the abstractions required for connecting to external LLMs, managing conversations, and invoking tools.

CrypLLM supports both remotely hosted LLM services and locally hosted models. Within the application settings, users can select the model, specify an organization identifier, and provide an API key. If supported by the selected model, users can also configure the reasoning level as low, medium, or high. The current implementation has been developed and tested with APIs compatible with OpenAI.

Model selection, service endpoints, and inference parameters are loaded dynamically when an agent instance is created. Sensitive information such as API keys are stored separately and are not exposed to the LLM during operation.

The behavior of the agent is controlled through a configurable main system prompt. This prompt defines the role, capabilities, and interaction principles of CrypLLM within the CT2 environment. It is embedded into the application, can be modified through the settings, and can be reset to a default configuration if required. In addition to the static system prompt, CrypLLM injects dynamic metadata into each request, including information such as the current date and time. The default prompt configuration used in the implementation is documented in Listing 3. The main system prompt was developed using the Valisory Prompt Studio to achieve consistent behavior across a wide range of user requests. This framework supports the structured creation, testing, and comparison of different prompt variants. [31]

The subsystem consists of several internal components responsible for different tasks. Conversation management handles message histories and thread persistence. Tool plugins provide controlled access to the host application and the active workspace. Additional components manage prompt construction, communication with the selected LLM service, and response processing.

4.4.2. Conversation Management

Conversations are organized into independent conversation threads. Each thread contains the message history, the associated workspace tab, timestamps, and model settings. This allows CrypLLM to preserve context even when various workspaces are open simultaneously.

Conversation threads are stored persistently in a local history file so that they remain available across application restarts. The serialized thread data is written to *AIThreads.history.json* in the local application data directory. Only visible user and assistant messages are restored in the user interface. Internal system messages, tool invocations, and additional Semantic Kernel metadata are intentionally excluded from

serialization since they are defined by the application itself and may change due to updates or user-specific configuration settings. Tool invocations can instead be executed again when needed, allowing the agent to retrieve the current workspace state with up-to-date component and connector identifiers.

When a user submits a message, CrypLLM either continues the active thread or creates a new one. The request is combined with the current thread history and the global system instructions described in the previous section.

Responses are generated internally before being transferred to the user interface. Once the complete response has been generated, it is propagated through an event-driven mechanism and displayed in the chat interface as a single message.

4.4.3. Agent Execution Workflow

At the architectural level, a user request is processed in three phases: request preparation, model interaction, and response delivery. During request preparation, the user message is associated with the active conversation thread and the current application context. During model interaction, the request is sent to the selected LLM together with the available tool definitions. If additional information or actions are required, the model may request tool calls, which are executed through controlled interfaces of CT2. The resulting response is then returned to the chat interface. The detailed implementation of this workflow is described in Section 5.3.4.

4.4.4. Context Management and Token Budgeting

A central architectural challenge of CrypLLM is the limited context window of the selected LLM. Each request may contain system instructions, tool definitions, the user message, previous conversation history, and tool results. Without explicit context management, longer conversations, large tool schemas, or verbose tool outputs could exceed the available context window and cause request failures.

CrypLLM therefore follows a budget-oriented context management strategy. Dynamic parts of the request, such as conversation history and tool outputs, can be reduced when necessary, while sufficient space is reserved for the model's response. This is particularly important for locally hosted models, which often provide smaller context windows than current cloud-based models.

In addition, the number of enabled tools directly affects context usage because their definitions must be included in the model request. Some tools or tool combinations may therefore only be practical with models that provide sufficiently large context windows.

4.4.5. Tool-Based Interaction

CrypLLM interacts with the CT2 environment through callable tools. These tools provide controlled access to selected parts of the application and can be used to retrieve information or perform modifications.

The available tools are grouped into several functional categories, including workspace inspection, workspace modification, access to component information, template retrieval, and user interface support.

Depending on the user input, the agent may invoke one or more tools before generating the final response. The detailed implementation of these tools and their technical integration are described in Section 5.

4.4.6. Error Handling and Failure Modes

CrypLLM must handle failures both in the communication with external LLM services and in the interaction with the host application. Since the subsystem depends on network communication, external providers, and probabilistic model generation, it uses a multi-layered exception handling strategy. The goal is graceful degradation: failures within CrypLLM must not terminate CT2, block the user interface, or corrupt the active workspace.

Typical failure cases include missing internet connectivity, unavailable local inference servers, invalid API keys, timeouts, rate limits, failed tool invocations, and incomplete workspace information.

Connection failures to external or local LLM services are caught and displayed directly in the chat interface. This includes issues such as authentication failures, invalid or expired API keys, timeouts, rate limits, or unavailable endpoints.

Tool execution introduces additional failure modes because the LLM determines autonomously which tools should be called and which parameters should be passed. Invalid requests may include references to non-existing components or unsupported parameter combinations. To prevent such failures from affecting the host application, all tools validate their input before performing any operation.

If a tool invocation still fails, the exception is intercepted by the execution pipeline and converted into a controlled error response. Depending on the situation, the error is either shown directly to the user or returned to the LLM as additional context. This allows the model to recognize invalid actions and attempt a corrected tool call in the next reasoning step.

All non-recoverable errors are transformed into readable chat messages instead of causing hard crashes or unhandled exceptions. These messages are visually separated from normal responses so that users can distinguish between generated content and technical failures.

4.5. Context Acquisition and Representation

To operate within CT2, the LLM-based agent requires access to the current application state, especially the structure and execution status of the active workspace. Context acquisition is therefore implemented as a controlled process that retrieves selected information from the running system without tightly coupling CrypLLM to internal components. This information is exposed to the LLM through dedicated tools that can be invoked when additional workspace context is required.

4.5.1. Workspace-Centric Context Extraction

The primary source of contextual information is the workspace, which represents a workspace graph composed of components and connections.

Instead of exposing the original internal model directly, CrypLLM constructs an intermediate representation tailored for machine processing. This representation contains components, connectors, connections, annotations, images, and execution state information.

Each component is represented together with its input and output connectors, descriptive metadata, execution progress, and operational state. Connections are represented as directed edges between connectors, preserving the graph structure of the workspace.

4.5.2. Structured Graph Representation

The extracted context is organized as a strongly typed object structure and serialized into JSON. Conceptually, the workspace is represented as a graph in which components form nodes and connections form edges.

Each component object contains identifiers, localized captions, descriptions, execution state, and collections of input and output connectors. Connector objects additionally store connector direction, data type, mandatory status, and visibility information.

By combining structural and runtime information, CrypLLM can reason about both the configuration and the behavior of the workspace.

4.5.3. Snapshot-Based Workspace Context Retrieval

To preserve consistency across multiple open workspaces, CrypLLM stores a reference to the active tab whenever the user submits a message. This ensures that the agent can still identify which workspace the request refers to, even if the user switches to another tab while the response is being generated.

However, the workspace state itself is not captured immediately when the message is submitted. Instead, the current workspace information is retrieved only when the corresponding tools are executed. As a result, modifications made by the user after submitting the message but before the tool invocation are still included in the retrieved context.

This snapshot-based approach avoids inconsistencies during processing and allows the user to continue working while a response is generated.

4.6. Use of the Semantic Kernel Framework

CrypLLM is implemented using the Microsoft Semantic Kernel framework [44], which acts as an orchestration layer between CT2 and external or locally hosted LLMs. The framework provides abstractions for model communication, conversation handling, and tool invocation. As an open-source project released under the MIT License, Semantic Kernel can be integrated into the Apache 2.0 licensed CT2 codebase without licensing conflicts.

A central reason for using Semantic Kernel is its support for tool-based agents. Functions implemented in the host application can be exposed to the model as callable tools, while

the framework handles structured tool invocation and returns tool results to the model. This allows CrypLLM to ground responses in the current application state rather than relying only on pretrained model knowledge.

Semantic Kernel also abstracts the underlying model provider. CrypLLM can therefore communicate with cloud-based LLM services as well as locally hosted models that expose compatible interfaces. Model-specific settings such as the selected provider, endpoint, model, API credentials, and reasoning level are incorporated dynamically when the agent is created.

Compared to a custom implementation of LLM communication and tool invocation, Semantic Kernel reduces development effort and improves maintainability. Changes in provider APIs can often be handled through framework updates instead of extensive modifications to the CrypLLM codebase.

4.7. Security and Privacy Considerations

The integration of an LLM-based agent into CT2 introduces security, privacy, and safety risks related to external services, local application data, and the probabilistic behavior of language models. The system therefore follows a conservative design that emphasizes user control, transparency, and restricted default behavior.

4.7.1. Security

CrypLLM runs within the existing application process and accesses application data only through interfaces. Read access includes information such as component configurations, connector structures, textual inputs and outputs, memo contents, execution states, log messages, and tab names. Write access is more restricted and limited to predefined actions such as inserting or deleting components, modifying parameters, creating connections, and starting or stopping workspace execution.

Since CrypLLM can interact with existing CT2 components, the security implications depend not only on the agent itself but also on the capabilities of the available components. Components that can access the file system, external resources, or the network are particularly relevant. For example, file output components can write generated content to local files. In the worst case, a manipulated prompt could cause the agent to overwrite existing files, save harmful script content, or create files in sensitive locations such as startup directories. Future CT2 components could further extend these risks depending on their capabilities, especially if they introduce additional access to the operating system, external services, or executable content.

To reduce these risks, modification tools are disabled by default and can be enabled or disabled individually in the settings. This limits the risk of unintended or malicious changes caused by incorrect model behavior. Future versions should additionally provide confirmation dialogs for high-risk operations such as writing files, changing network settings, or modifying executable content.

Communication with external LLM providers is performed using Hypertext Transfer Protocol Secure (HTTPS). When locally hosted models are used, communication may also occur over Hypertext Transfer Protocol (HTTP), typically in local environments. API keys are stored in encrypted form using the Windows Data Protection API (DPAPI), ensuring that only the current operating system user can decrypt them. Chat histories are likewise stored locally as encrypted JSON files in the application data directory.

Prompt injection attacks are possible because the model may receive untrusted content from user messages, workspace texts, memo components, annotations, connector names, or component names. A manipulated workspace could therefore contain hidden instructions that influence the model and trigger unintended tool calls or workspace modifications.

The current implementation does not provide dedicated prompt injection protection. Instead, the system relies on restricted tool permissions, conservative defaults, optional write access, and user supervision to reduce the impact of manipulated prompts. In addition, all tool invocations are recorded in the application log to support transparency, debugging, and later inspection of performed actions.

4.7.2. Privacy

Privacy risks primarily arise when online LLM services are used. In this case, user messages and selected workspace information may be transmitted to external providers. Depending on the active tools, this may include workspace structures, textual component content, memo fields, connector information, execution logs, file paths, and cryptographic material such as plaintexts, ciphertexts, passwords, keys, or initialization vectors.

Log messages may also contain sensitive local system information, including file locations, configuration details, or technical error information. The system does not automatically filter or redact sensitive information before transmission. Responsibility for submitted data therefore remains with the user.

To reduce unnecessary disclosure, users can enable or disable individual context tools and modification tools in the settings. Chat histories remain on the local machine and

are not synchronized with external services. They can be deleted manually or through the application settings.

Since the retention policies of external providers vary, the system cannot guarantee how transmitted data is stored or processed after it leaves the local application. Before the first use of CrypLLM, users are informed that responses are generated by an AI system, may contain errors, and may require the transmission of workspace data to external services.

4.7.3. Safety

Beyond technical security and privacy, the use of generative models introduces risks related to correctness and reliability. CrypLLM may produce inaccurate, incomplete, or misleading information, including incorrect explanations of cryptographic concepts or unsuitable recommendations for workspace modifications.

Modification tools create additional operational risks. CrypLLM may misinterpret a request and delete components, remove connections, overwrite parameters, or otherwise damage the current workspace. Incorrect parameter choices may also weaken cryptographic security, for example, by selecting outdated algorithms, weak keys, insecure initialization vectors, or unsuitable iteration counts.

The agent may also configure components with excessive workload parameters, which can significantly increase execution time or freeze the application. This could happen, for example, if a component is configured with extremely high iteration counts or very large search spaces.

CrypLLM is designed as an educational aid rather than an authoritative source. Even when modification tools are enabled, the system only reacts to explicit user requests and does not perform autonomous actions without user interaction. Nevertheless, users must still review generated explanations and workspace changes carefully, since incorrect recommendations or unintended modifications cannot be fully prevented.

The use of external AI services and probabilistic models, therefore, introduces residual risks that cannot be eliminated completely.

4.8. Architectural Decisions and Trade-offs

CrypLLM balances functionality, performance, usability, and security within the constraints of integrating an LLM-based agent into an existing desktop application.

One important decision concerns the handling of workspace consistency. When the user submits a message, CrypLLM stores the active tab so that the request remains associated with the correct workspace even if the user switches tabs during response generation. However, the workspace state itself is not frozen. Instead, tools retrieve the current state when they are executed. This avoids blocking the user while the agent is generating a response and allows recent user changes or agent-generated modifications to be included in later tool calls.

An alternative approach would be to lock the workspace during response generation. This would make the referenced workspace state more predictable but would also restrict the user's ability to continue working interactively. CrypLLM therefore prioritizes interactivity over strict state isolation. The trade-off is that the exact workspace state used for a response may not always be transparent to the user. If the workspace changes while a response is being generated, the agent may refer to a state that differs from the one visible when the message was submitted, which can lead to minor inconsistencies.

Another central decision is the direct integration of CrypLLM into CT2. This enables the agent to access the current workspace, inspect application state, and perform selected modifications through existing CT2 mechanisms. The trade-off is a stronger coupling to the internal architecture of CT2. A more loosely coupled approach, for example through an MCP-based interface, could improve interoperability with other agent systems but would have introduced additional architectural complexity and would not have provided the same direct access to internal workspace functionality. For this reason, CrypLLM uses a custom tool interface integrated directly into CT2.

Context-window management represents another relevant trade-off. CrypLLM reduces conversation history and tool outputs when necessary to keep requests within the limits of the selected LLM. This improves robustness and increases compatibility with locally hosted models that provide smaller context windows. However, reducing context may remove information that could still be relevant for later reasoning. In addition, enabling many tools increases the size of the tool definitions sent to the model, so some tools may only be practical with models that provide sufficiently large context windows.

Finally, CrypLLM follows a reactive and permission-based interaction model. The agent only acts in response to explicit user input, and tools that modify the workspace can be enabled or disabled by the user. This reduces the risk of unwanted modifications and keeps the user in control. The trade-off is that the agent cannot proactively detect errors, suggest improvements, or intervene when it notices potential problems unless the user explicitly asks for assistance.

5. Implementation

This chapter describes how the architectural concepts introduced in Chapter 4 were implemented in CrypLLM. The structure of this chapter intentionally follows the same overall organization as the previous chapter so that the conceptual design can be compared directly with the corresponding implementation details.

The chapter first introduces the development environment and the technologies used during implementation. It then explains how CrypLLM is integrated into CT2, how the agent is implemented internally, and how context information is extracted from the application state. Finally, the chapter describes the user interface, configuration mechanisms, and selected implementation details related to security, persistence, and consistency.

5.1. Development Environment and Technologies

CrypLLM was implemented in C# as part of the existing CT2 codebase. The user interface was developed with WPF [46] and Extensible Application Markup Language (XAML) in order to remain compatible with the graphical framework already used by CT2. Structured data exchanged between the host application, the agent, and external LLM services is represented in JSON format.

The implementation targets Microsoft .NET Framework 4.7.2 and was developed for a 64-bit Windows environment. Development was carried out with Microsoft Visual Studio [45], initially version 2022 and later version 2026 after its release. [43, 44]

Microsoft Semantic Kernel version 1.67 [44] was used during development. The implementation supports OpenAI-compatible APIs and was tested with GPT-5.2, GPT-5.2 mini, GPT-5.4, GPT-5.4 mini, and GPT-OSS 20B through LM Studio [37]. [51]

Additional libraries include *Newtonsoft.Json* [26] for serialization and deserialization of structured data exchanged with the model and stored locally, as well as Markdig and MdXaml for Markdown rendering inside the chat interface. The implementation was developed and tested using CT2 source code from the main branch as of June 23, 2025. [15]

5.2. Integration into CrypTool 2

The CrypLLM project is referenced by CrypWin and initialized during startup in MainWindow. Integration points include the docking layout and the registration of the CrypWinAdapter instance in CrypWinPort.

5.2.1. User Interface Integration

The chat interface is implemented as a WPF control named *AIChatContent*. The control is embedded directly into the XAML definition of the main application window and integrated into the existing docking layout of CrypWin.

Since the control is created during application startup, the chat subsystem is available immediately without requiring additional initialization steps when the user opens the panel. The panel itself can be shown or hidden through the ribbon interface.

5.2.2. Port and Adapter Integration

Communication between CrypWin and CrypLLM is implemented through a port-and-adapter architecture in order to decouple the LLM-based agent from the internal structures of CT2. The central abstraction is the `ICrypWinAdapter` interface in the namespace `CrypLLM.Ports`. This interface defines the operations that CrypLLM may use to access information from CrypWin, such as retrieving the active workspace, reading log messages, obtaining the currently selected tab, or accessing open editors.

The concrete implementation of this interface is provided by the `CrypWinAdapter` class located in `CrypWin.Adapter`. During initialization, the adapter receives a reference to the active `MainWindow` instance and uses this reference to delegate requests to the existing CrypWin functionality. For example, methods such as `GetOpenTabs()`, `GetRecentLogMessages()`, `GetActiveEditor()`, or workspace-related accessors retrieve their data directly from the internal state of the main application.

To make the adapter accessible throughout CrypLLM, the project uses a static class named `CrypWinPort`. This class stores the active `ICrypWinAdapter` instance during startup and exposes it globally to Semantic Kernel plugins, services, and the LLM-based agent. As a result, these components can access CrypWin functionality without requiring direct references to `MainWindow` or other CrypWin-specific classes.

5.2.3. Runtime Initialization

CrypLLM is initialized automatically during startup together with the main application window. Core services such as the thread manager, configuration manager, and agent manager are created lazily when they are first required.

Communication with external model endpoints is executed asynchronously to avoid blocking the user interface. Exceptions such as missing API keys, unavailable endpoints, or invalid model responses are intercepted and converted into readable error messages shown in the chat interface.

5.3. Implementation of the LLM-Based Agent

The CrypLLM subsystem is organized into several components responsible for user interaction, conversation management, tool execution, persistence, and communication with the host application. Figure 6 illustrates the most important classes and their relationships as a Unified Modeling Language (UML) component diagram.

5.3.1. Agent Overview

The central component of the subsystem is the *AIThreadManager*. It acts as the central orchestration component for conversation handling, agent lifecycle management, persistence, and communication with the selected LLM service.

Each conversation is represented by an *AIThread* object. A thread stores the message history, the corresponding Semantic Kernel execution thread, the associated workspace tab, timestamps, thread identifiers, and model-specific settings.

CrypLLM uses a single shared *ChatCompletionAgent* instance for all conversation threads. The subsystem reuses this agent across all requests and combines it with the state of the currently active thread. A new agent instance is only created when relevant settings change, such as the selected provider, model, API key, reasoning level, or system prompt.

The *AIThreadManager* also serves as the central integration point between conversation handling, tool execution, and model communication. It coordinates the interaction between the Semantic Kernel environment, the registered plugins, and the active conversation thread.

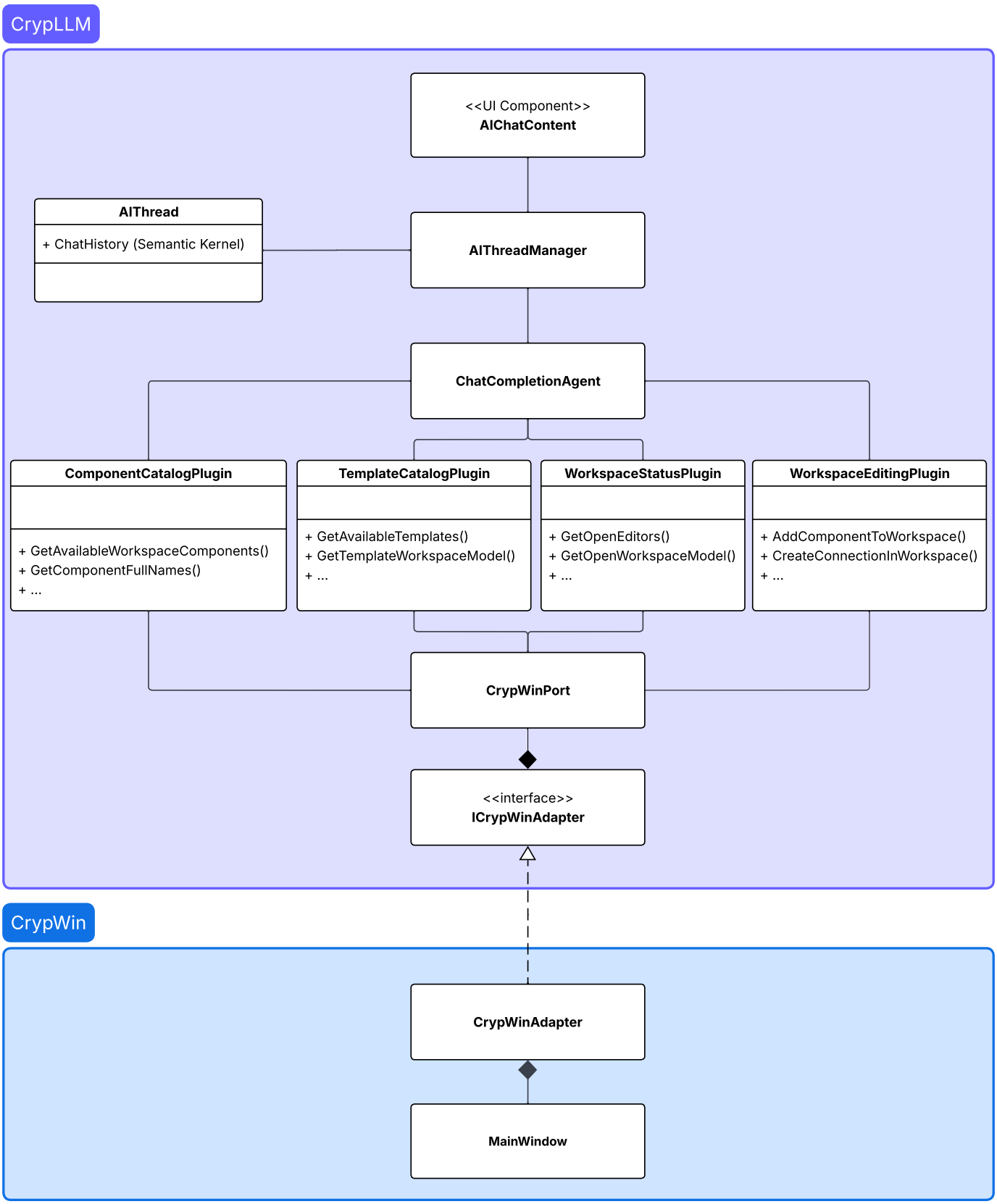

Fig. 6: UML component diagram of the CrypLLM subsystem illustrating the main classes, tool plugins, and communication with the CrypTool 2 host application. The *UiTextCatalogPlugin* was omitted for readability.

As shown in Figure 6, the *AIThreadManager* is connected to the chat interface, the Semantic Kernel agent, and the available plugin classes. Centralizing these responsibilities in a single component ensures that conversation state, agent configuration, and tool availability remain consistent across all requests.

5.3.2. Conversation Management

Conversation handling is implemented through the classes *AIThreadManager* and *AIThread*. As described in Section 4.4.2, conversations are organized into independent threads that preserve the message history and the associated workspace context.

Each conversation is represented by an *AIThread* object. The class acts as a lightweight wrapper around the Semantic Kernel *ChatHistoryAgentThread*. In addition to the message history, an *AIThread* stores metadata such as a unique identifier, a user-visible title, and the identifier of the last associated workspace tab.

New threads are created automatically if no previous history exists or when the user explicitly selects the *Plus* action in the interface. Newly created threads initially receive a default title. After the first user message has been submitted, the thread is automatically renamed based on the first non-empty line of the prompt, truncated to a fixed maximum length. This avoids the need for manual thread naming and improves the usability of the chat interface.

Before each request, the conversation history is prepared for model invocation using the context-window management mechanism described in Section 5.3.5. The complete conversation remains visible in the user interface and is preserved for persistence, while only a request-specific reduced version is sent to the model when necessary.

5.3.3. LLM Integration

The integration of language models is managed centrally by the *AIThreadManager*. As described in Section 4.6, CrypLLM uses Microsoft Semantic Kernel as an abstraction layer between the application and the selected LLM provider.

The implementation supports both cloud-based and locally hosted models. Relevant settings include the provider, model identifier, API key, local endpoint, and reasoning level.

Cloud-based models are accessed through the OpenAI API using the configured API key and optional organization identifier. Locally hosted models are integrated through the

same OpenAI-compatible interface by replacing the default endpoint with a local URL, for example, an endpoint provided by LM Studio.

Internally, both variants use the Semantic Kernel method *AddOpenAIChatCompletion(…)*. For local models, the configured endpoint is redirected to the local inference server instead of the official OpenAI endpoint. This allows local OpenAI-compatible services to be integrated without changing the remaining communication logic.

The provider abstraction also simplifies the integration of additional model providers. However, this mainly applies to providers that are already supported by Semantic Kernel and provide compatible chat completion services with tool-calling support. At the time of writing, Semantic Kernel provides connectors for providers such as OpenAI, Azure OpenAI, Google Gemini, Anthropic, Mistral, Ollama, Hugging Face, Amazon Bedrock, Open Neural Network Exchange (ONNX) and others. Locally hosted AI services from these providers are also supported. [41]

Since CrypLLM is based on the *ChatCompletionAgent*, additional providers must support chat completion and function calling so that Semantic Kernel can expose the registered plugins as callable tools. Supporting another provider therefore mainly requires registering a compatible Semantic Kernel connector and adding the corresponding provider settings to the user interface. Existing plugins, conversation handling, and tool logic remain unchanged.

In addition to function-calling support, the usable tool set also depends on the context window of the selected model. The definitions of all enabled tools are included in the model request, which means that many tools can consume a substantial part of the available context before conversation history or workspace information is added. This is especially relevant for local models with smaller context windows. For this reason, CrypLLM allows tools to be enabled or disabled depending on the selected model and usage scenario.

5.3.4. Agent Execution Workflow

When a user submits a message, CrypLLM starts a request-specific execution workflow. Figure 7 illustrates this process from the initial user input in the chat interface to the final response returned by the agent.

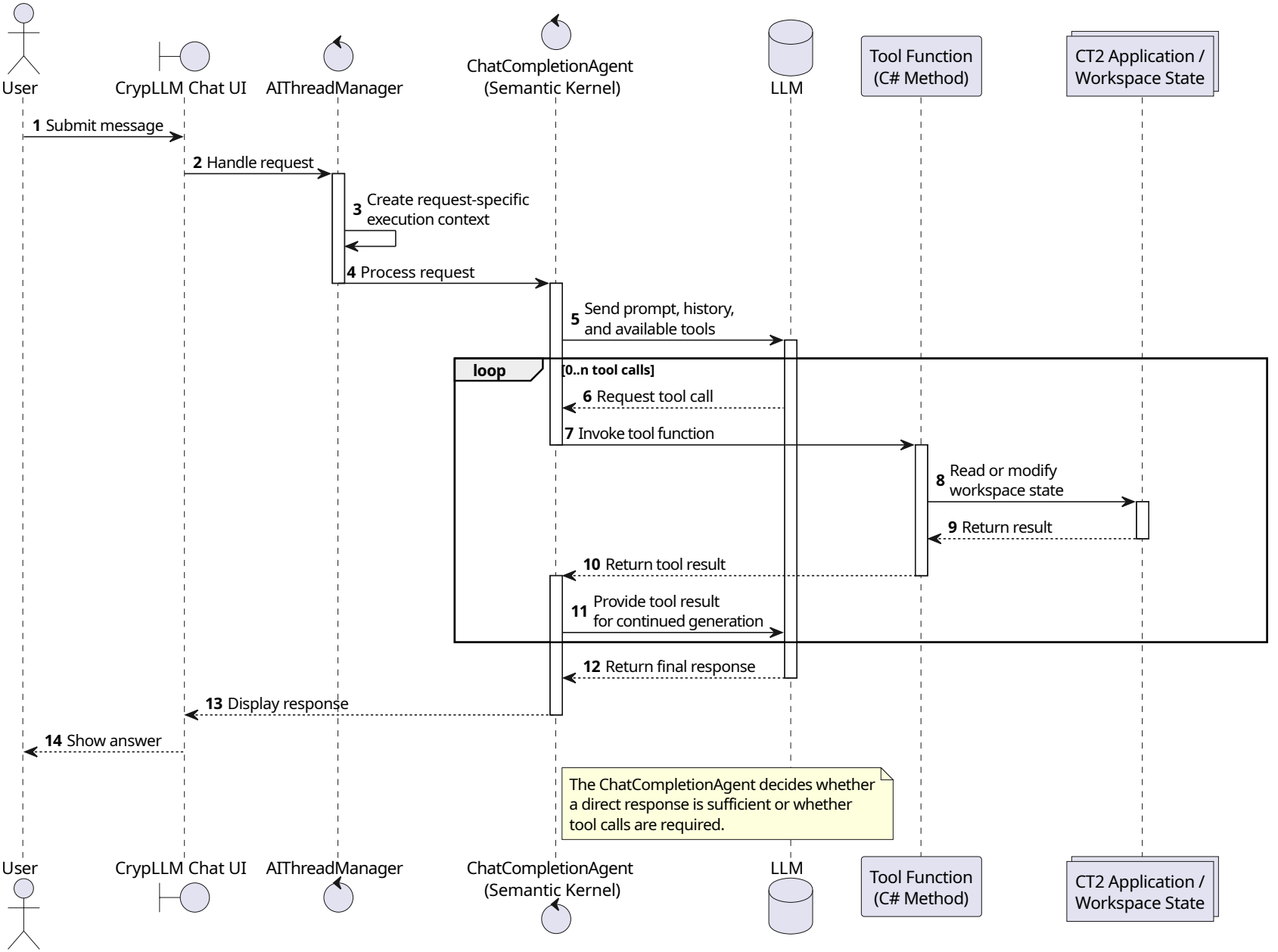

Fig. 7: Execution workflow of a user request in CrypLLM, including optional tool calls through Semantic Kernel.

The workflow begins when the user submits a message through the CrypLLM chat interface. Before the request is forwarded to the *ChatCompletionAgent*, the conversation history is reduced according to the available token budget, as described in Section 5.3.5.

The prepared request is then processed by the *ChatCompletionAgent*, which is implemented using the Microsoft Semantic Kernel framework. The agent uses the configured *IChatCompletionService* to send the system instructions, the user message, the conversation history, and the registered tool definitions to the LLM.

Depending on the prompt and the current interaction context, the LLM may either generate a direct response or request one or more tool calls. If a tool call is requested, Semantic Kernel handles the invocation automatically through its tool-calling mechanism. The corresponding C# method is executed and can retrieve information from or perform selected operations on the CT2 application and workspace state.

The result of the executed tool is returned to the agent and incorporated into the continuing generation process. This cycle can repeat multiple times within a single request, allowing the agent to combine reasoning and tool-use steps, such as retrieving workspace information, validating it, and then generating an explanation or performing a modification. Once no further tool calls are required, the final response is displayed in the CrypLLM chat interface.

5.3.5. Context Window Management

CrypLLM implements explicit context-window management to prevent requests from exceeding the token limit of the selected LLM. This is necessary because a single request may contain the system prompt, the user message, the conversation history, registered tool schemas, and tool results returned during automatic tool invocation. This mechanism is particularly important for locally hosted models, since they may provide smaller context windows than current cloud-based models while still requiring the same system prompt, tool schemas, and workspace-related context.

Before a request is sent to the model, the *AIThreadManager* uses the *PromptBudgetPlanner* to create a request-specific budget profile. This profile reserves space for fixed context elements such as the system prompt and tool definitions, as well as a response reserve and a safety margin. The remaining budget is used for dynamic content such as the user message, retained conversation history, and possible tool results.

The conversation history is prepared by the *ChatHistoryReductionEngine*. If the accumulated history exceeds the available budget, older parts of the conversation are reduced before model invocation. Depending on the situation, this may include shortening older tool outputs, summarizing previous interactions, or removing older messages. The com-

plete conversation remains available in the user interface and for persistence, while only the reduced version is sent to the model.

Tool outputs are handled separately during automatic tool invocation. Since tools may return large amounts of text, for example, when describing a complex workspace or listing component information, tool execution is performed within a *ToolInvocationBudgetRuntime*. This runtime tracks the available budget during a request and applies text reduction to tool results when necessary before they are appended to the conversation context. This prevents a single verbose tool result from exceeding the model context window and interrupting the generation process.

5.3.6. Tool-Based Interaction

Domain-specific functionality is implemented through Semantic Kernel plugins. These plugins are registered during agent creation in the *CreateNewAgent(…)* method of the *AIThreadManager* using *kernel.Plugins.AddFromType<…>()*. Table 1 summarizes the registered plugins and their available tools.

The *WorkspaceStatusPlugin* provides read access to the current application state, including workspace structures, execution progress, component controls, and recent log messages. The *WorkspaceEditingPlugin* contains tools for modifying the active workspace, such as inserting components, creating connections, or starting execution. In addition, the *ComponentCatalogPlugin* provides information about available components and *UiTextCatalogPlugin* provides information about specific CT2 user interface topics.

Each plugin exposes selected functionality through C# methods annotated with the Semantic Kernel attributes *[KernelFunction]* and *[Description]*. The first attribute defines the identifier under which the function is made available to the model, while the second provides a natural-language description of the tool's purpose and usage conditions. These descriptions are important because the LLM selects tools primarily based on their names, descriptions, and parameter signatures.

Tool results are typically returned as structured JSON using *Newtonsoft.Json*. This allows the model to process complex information such as workspace structures, component settings, validation results, or component catalogs more reliably. [26]

Dedicated service classes encapsulate recurring operations such as retrieving the current workspace model, resolving component information, or accessing log messages. If these operations require access to GUI-bound objects, they are executed on the main User Interface (UI) thread through dispatcher-based helper methods.

Tab. 1: Overview of the main Microsoft Semantic Kernel plugins and their tools

Plugin	**Tool**	**Purpose**
Workspace-StatusPlugin	`tabs_list`	Returns all currently open tabs in CT2.
	`ws_model`	Returns the components, connections, and execution state of a workspace.
	`ws_validate`	Validates the current workspace and reports missing parameters or connections.
	`ws_io`	Returns the current input and output values of workspace components.
	`ws_settings`	Returns the visible component controls of the active workspace.
	`ws_bounds`	Returns the coordinates and sizes of workspace elements.
	`app_log_recent`	Returns recent log messages from the CT2 user interface.
Workspace-EditingPlugin	`ws_add_component`	Inserts a component at a specific position in the workspace.
	`ws_add_connection`	Creates a connection between two connectors.
	`ws_set_component_text`	Changes component controls or text values.
	`ws_move_component`	Moves a component to a new position.
	`ws_remove_component`	Removes a component from the workspace.
	`ws_remove_connection`	Removes an existing connection.
	`ws_run`	Starts workspace execution.
	`ws_stop`	Stops workspace execution.
	`wait_seconds`	Delays the next tool call for a specified number of seconds.
Component-CatalogPlugin	`comp_list`	Lists all available components grouped by category.
	`comp_fullnames`	Returns the technical type names of registered components.
	`comp_docs`	Returns the generated online help for a component.
	`comp_defaults`	Returns the default component controls for a component type.
UiText-CatalogPlugin	`text_ui`	Returns explanatory texts about selected CT2 user interface topics.

Because many tools access WPF-bound objects, thread safety is an important implementation concern. Most plugin functions therefore delegate their execution to helper methods such as *EditorStatusPluginHelper.InvokeOnUi(…)*. This ensures that workspace queries and UI interactions are executed through the dispatcher of the host application.

The implementation also configures invocation filters. The *ToolPermissionFilter* restricts access to selected tools based on the permissions configured by the user in the settings. The *AddReturnTypeSchemaFilter* combines tool return values with their associated JSON schema before returning them to the LLM. This improves the interpretation of complex or nested data structures.

A simplified example of a tool implementation is shown in Listing 1. The example illustrates the use of the Semantic Kernel attributes and the delegation of the actual execution to the UI thread.

Code Listing 1: Simplified implementation of a Semantic Kernel tool function.

```
[KernelFunction("get_all_component_fullnames")]
[Description("Returns a JSON array of full type names for all workspace
    components.")]
public string GetComponentFullNames()
{
    return EditorStatusPluginHelper.InvokeOnUi(GetComponentFullNamesInternal);
}
```

5.4. Context Acquisition Mechanisms

This section describes how CrypLLM extracts workspace information and transforms it into a representation that can be processed by the LLM.

Workspace information is extracted through dedicated service classes. *WorkspaceInspector* provides methods for retrieving specific subsets of information, such as validation results, visible component settings, graphical properties, and current input and output values. For more comprehensive representations, *AISchemaGenerator* converts the complete workspace into a structured abstraction.

The native workspace model is not serialized directly because it contains cyclic references and implementation-specific details. Instead, the implementation constructs simplified data transfer objects such as *WorkspaceModelAbstraction*, *ComponentAbstraction*, and *ConnectionAbstraction*.

These abstractions replace internal object references with stable identifiers. Components are represented by identifiers, captions, type names, positions, dimensions, and selected runtime information. Connections are represented through source and target identifiers instead of direct object references.

The resulting structures are serialized using *JsonConvert.SerializeObject(…)* from *Newtonsoft.Json* and returned to the LLM as structured JSON. This allows the model to process workspace graphs, validation results, and component settings without requiring access to the original internal object model.

Code Listing 2: Simplified excerpt of the serialized workspace representation returned to the LLM.

```
{
  "Components": [
    {
      "Id": "38555124",
      "Type": "Substitution",
      "Name": "Substitution",
      "Inputs": [
        "InputString",
        "SourceAlphabet",
        "DestinationAlphabet"
      ],
      "Outputs": "OutputString"
    },
    {
      "Id": "26864056",
      "Type": "TextInput",
      "Name": "Plaintext"
    },
    {
      "Id": "40449915",
      "Type": "TextOutput",
      "Name": "Ciphertext"
    }
  ],
  "Connections": [
    {
      "From": {"ComponentId": "26864056", "Connector": "TextOutput"},
      "To": {"ComponentId": "38555124", "Connector": "InputString"}
    },
    {
      "From": {"ComponentId": "38555124", "Connector": "OutputString"},
      "To": {"ComponentId": "40449915", "Connector": "TextOutput"}
    }
  ]
}
```

5.5. User Interface

The chat interface is implemented as a dedicated WPF control named *AIChatContent*. As shown in Figure 8, the interface is divided into three main areas: a header section, a message area, and an input area. The header contains a dropdown menu for selecting conversation threads, as well as buttons for creating, deleting, and exporting chats and opening the AI settings. The central area displays the message history inside a scrollable container. The lower section contains the user input field, a model selection dropdown, and a dynamic send or stop button.

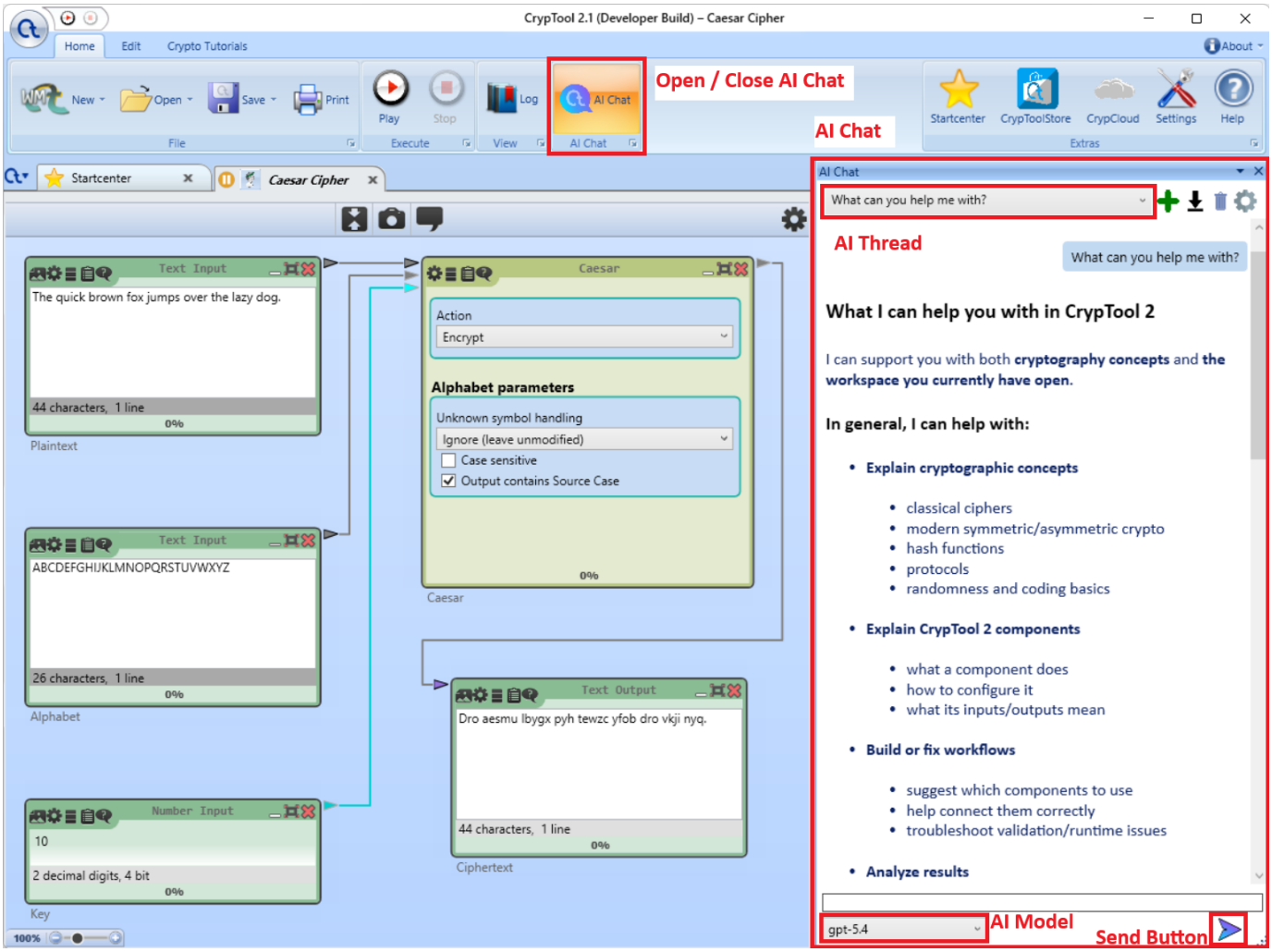

Fig. 8: Embedded AI Chat interface in CrypTool 2.

Messages are displayed with different visual styles depending on their role. User messages are right-aligned and rendered as plain text, whereas agent messages are left-aligned and rendered using a Markdown viewer based on the *Markdig.Wpf* library. This enables support for formatted text, lists, tables, and code blocks. Internal system messages, tool

calls, and tool results are excluded from the visible chat history and are not shown to the user.

Response generation is performed asynchronously so that the user interface remains responsive during longer requests. As shown in Figure 9, the chat control switches into a temporary thinking state while the *AIThreadManager* processes the request in the background. During this phase, a loading indicator is displayed below the most recent message, and the send button changes into a stop button that allows cancellation of the active request through a cancellation token.

The response is not rendered incrementally. Instead, the *AIThreadManager* waits until the complete request, including possible tool calls, has finished. Afterward, the chat control reloads the updated conversation history and displays the final assistant response at once. Messages are stored in *ObservableCollection* objects and synchronized through the WPF dispatcher because updates may originate from asynchronous background tasks.

Fig. 9: AI Chat during response generation.

Long conversations are handled through automatic scrolling, which keeps the latest messages visible. The input field initially behaves like a single-line text box and expands automatically to multiple lines when the entered text exceeds the available width. Automatic wrapping and a maximum height prevent the control from occupying too much screen space. Pressing Enter submits the message, while Shift+Enter inserts a line break.

Errors during request processing are displayed directly in the chat history as agent messages. As shown in Figure 10, failures such as invalid credentials, unavailable endpoints, or invalid model identifiers can therefore be communicated without interrupting the application flow.

5.6. Configuration, Security, and Consistency

As shown in Figure 11, the CrypLLM subsystem provides a dedicated settings interface that is integrated into the existing CT2 settings dialog. The configuration UI is imple-

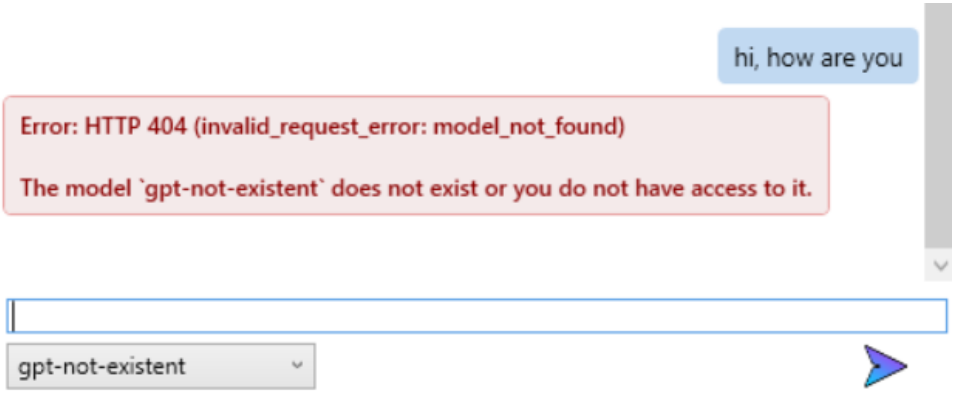

Fig. 10: Error message in the AI Chat.

mented as a separate WPF control and registered as an additional settings tab. Users can configure the model provider, the active model, authentication data, local endpoints, reasoning level, log level, system prompts, and the visibility behavior of the chat window. In addition, users can define fine-grained permissions for Semantic Kernel plugins, grouped by functional categories such as workspace inspection or workspace editing.

Certain texts used by the assistant can also be customized. In addition to the global system prompt, users may adapt selected descriptions of CT2 interface areas that are exposed through the *UiTextCatalogPlugin*, such as descriptions of the Startcenter or the WorkspaceManager. This makes it possible to tailor the assistant more closely to different user groups, languages, or preferred terminology.

For security reasons, tools with write access are disabled by default and can be enabled individually. As a result, users can decide how much control they want to give to the agent.

General configuration values are stored through the standard CT2 settings mechanism and persisted in the local user configuration directory. System prompts are stored separately in an *AgentInstructions.json* file, while chat histories are written to an *AIThreads.history.json* file in the local CrypLLM data directory. Sensitive information is protected using the Windows Data Protection API (DPAPI) through the *SecretProtector* helper class. API keys and persisted chat histories are encrypted with *DataProtectionScope.CurrentUser*, ensuring that only the current Windows user can decrypt them.

Most configuration changes take effect without requiring an application restart. If settings such as the provider, model, reasoning level, API key, or prompt configuration are modified, the current Semantic Kernel agent is recreated automatically before the next request. This ensures consistent behavior across configuration changes while avoiding unnecessary agent initialization.

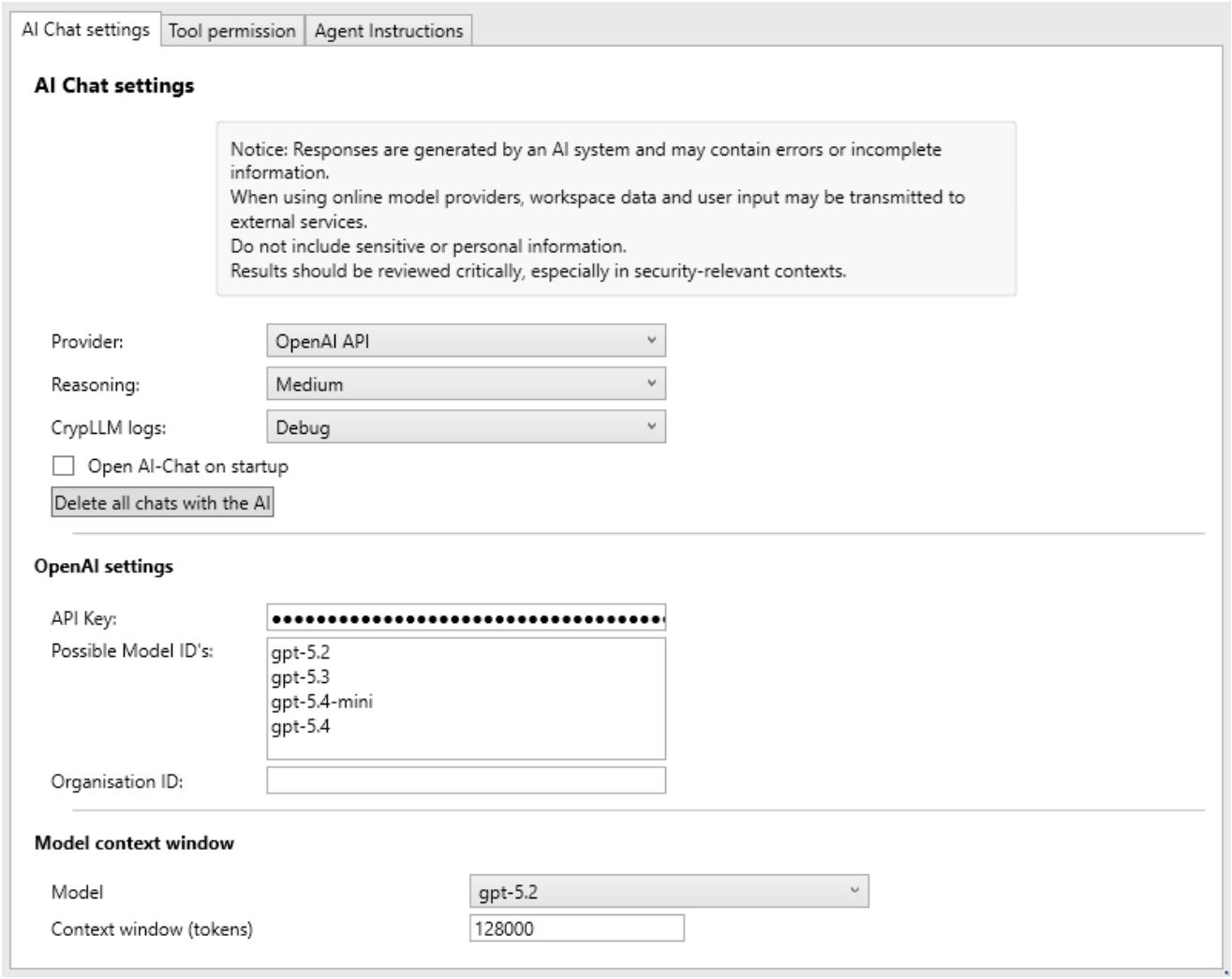

Fig. 11: First page of the AI Chat settings.

6. Evaluation

This chapter evaluates the CrypLLM prototype with regard to the research questions defined in Section 1.4. It first describes the evaluation methodology, participant profiles, and task design. It then discusses the findings with respect to usability, understanding of components and workspaces, learning support, problem solving, and response quality before summarizing the main results.

The primary focus of this thesis lies on the design, integration, and implementation of CrypLLM within CT2. The evaluation was originally planned as a self-evaluation to verify the functionality of the prototype and to reflect on its practical use. In order to obtain broader feedback, the evaluation was voluntarily extended to include three additional participants. Due to the small number of participants, the results are not intended to provide statistically generalizable evidence. Instead, the evaluation should be understood as an exploratory qualitative study that provides initial insights into usability, usefulness, limitations, and possible directions for future improvement.

6.1. Evaluation Methodology

The evaluation followed an exploratory qualitative design based on task observation, think-aloud protocols, and semi-structured interviews. It was conducted with three participants whose backgrounds and self-assessed prior knowledge are described in Section 6.2.

All evaluation sessions were conducted remotely using TeamViewer [71] for screen sharing and Discord [16] for audio communication. Screen and audio recordings were created with OBS [49]. The evaluation was carried out with CT2.1 (Stable Build 9778.2) [14] extended by the unpublished CrypLLM prototype. All participants used the same configuration consisting of the OpenAI GPT-5.2 model [50] with a medium reasoning level.

At the time of the evaluation, the agent could not yet modify workspaces directly, although this functionality is part of the final prototype described in Chapters 4 and 5. It could only provide instructions and suggestions for manual changes. The evaluation therefore focused on guidance, explanation, validation, and problem-solving rather than on direct workspace modification by the agent. Some technical and interface limitations identified during the evaluation, such as chat scrolling behavior and context-window handling, were addressed in later prototype iterations. The findings reported in this chapter, however, refer to the state of the system during the evaluation.

Participants were encouraged to verbalize their thoughts, expectations, uncertainties, and reasoning processes during the experiment. No introduction to the chat interface or the functionality of the agent was provided beyond a short explanation of the study goal and the general purpose of CT2. The participants therefore had to discover how the system worked on their own.

Each session consisted of a short introduction, a task phase, and a final interview. The introduction lasted less than five minutes and emphasized that the evaluation focused on the agent rather than on the participant's performance. The task phase consisted of four tasks, which are described in Section 6.3. The tasks were completed in the same order for all participants. The task phase lasted between 48 and 68 minutes, while the complete sessions lasted between 60 and 84 minutes.

After the task phase, all participants completed a semi-structured interview about their overall experience, the usefulness of the agent, and possible improvements. Participation was voluntary; all participants consented to the recording of audio and screen content, and the collected data was anonymized for later analysis.

The analysis considered interview statements, behavioral observations from the recordings, think-aloud transcriptions, chat histories, the number of chat messages, and the time required to complete the tasks. Interview transcripts, chat histories, and behavioral observations were reviewed and grouped according to the research questions.

The qualitative analysis was structured along a set of analytical dimensions derived from the research questions and the observed interaction patterns. These dimensions were not used as quantitative metrics in a statistical sense, but as categories for organizing the observations, interview statements, and chat histories.

Analytical dimension	**Interpretation in this study**
Task completion	Whether participants were able to complete the assigned tasks with the support of the agent.
Agent usage	How often and in which situations participants used the agent.
Problem-solving support	Whether the agent helped participants identify errors, understand causes, or continue when they were stuck.
Perceived usefulness	How participants described the usefulness of the agent during the final interview.
Cognitive load and uncertainty	Whether participants appeared uncertain, confused, or required additional guidance.
Response quality	Whether the agent's answers were understandable, sufficiently specific, and appropriate for the current task.

Tab. 2: Analytical dimensions used to structure the qualitative evaluation.

6.2. Participant Profile

The evaluation involved three participants with different educational and professional backgrounds, as shown in Table 3. Two participants had a technical background in informatics-related disciplines, whereas one participant had no formal background in computer science. This selection was intended to cover different types of potential users, ranging from technically experienced users to beginners with little prior exposure to cryptography or specialized software.

Person	Age	Gender	Education	Occupation
1	25	Male	B.Sc. Business Informatics (final year)	Student
2	23	Female	B.Sc. Media Informatics	Full stack developer
3	23	Male	M.Sc. Accounting, Auditing, and Taxation	Tax and Audit Assistant

Tab. 3: Participant profiles.

None of the participants had prior experience with CT2. Their self-assessed prior knowledge is summarized in Table 4. Participants 1 and 2 reported some prior knowledge of cryptography and stronger familiarity with software tools, while Participant 3 had no prior knowledge of cryptography. All participants reported familiarity with AI chatbots.

Person	Cryptography	CrypTool 2	AI Chatbots	Software Tools
1	2.5	0	3.5	2.5
2	2	0	4	4
3	0	0	4	3

Tab. 4: Self-assessed experience ratings of the participants on a scale from 0 to 5.

6.3. Task Design and Procedure

The evaluation was based on a sequence of practical tasks designed to cover different forms of interaction with the integrated agent. The tasks were intended to reflect realistic usage scenarios within CT2, ranging from initial exploration of the application to conceptual understanding, workspace repair, and practical cryptanalysis. All participants completed the tasks in the same order and without interruptions between tasks.

Before each task, the instructions were read aloud once by the researcher. No written task sheet was provided. Participants were free to use the AI Chat whenever they considered it useful, but its use was not mandatory. If participants became completely stuck, limited assistance was provided to allow the session to continue. Time limits were treated as approximate guidelines rather than strict constraints.

The first task focused on familiarization with CT2. Participants were asked to explore the application and describe how a typical workflow for learning a cryptographic method might look. This task was intended to evaluate the usability of the interface, the discoverability of features, and the participants' first impressions of the integrated agent.

The second task focused on conceptual understanding. Participants were asked to understand and explain the homophonic substitution cipher. The goal was to examine whether the agent could support users in learning a cryptographic concept and whether the generated explanations were understandable for users with different levels of prior knowledge.

The third task focused on problem solving. Participants were given a prepared DES workspace containing several configuration errors and were asked to repair it so that it produced a meaningful result. The inserted errors included an incorrect connection between components, an input key with one bit too many, and a DES component configured for decryption instead of encryption, as illustrated in Figure 12. This scenario was selected to evaluate whether the agent could help users identify and fix issues in existing workspaces without requiring deep prior knowledge of the DES algorithm.

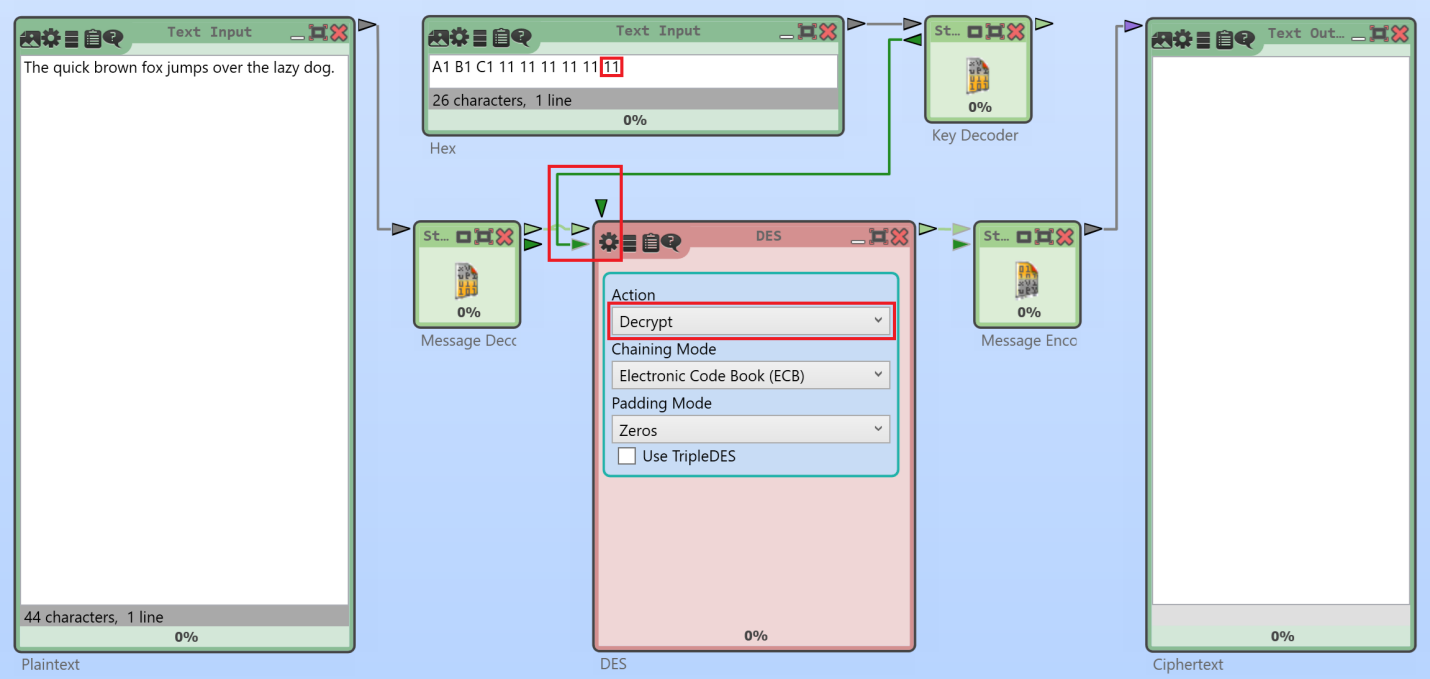

Fig. 12: A prepared DES workspace with intentionally inserted configuration errors used during the evaluation.

The final task represented a more practical end-user scenario. Participants were given a Vigenère-encrypted text without the corresponding key and were only informed about the encryption method and the original language of the text, which was English. They were asked to recover the plaintext without necessarily understanding the underlying

algorithm. This task was intended to evaluate whether the agent could support users who are interested primarily in obtaining a result rather than in learning the theoretical background.

Prepared workspaces and encrypted texts were provided where necessary, especially for the DES repair task and the Vigenère decryption task.

6.4. Evaluation with Respect to the Research Questions

Participants differed in how frequently they used the agent, as shown in Table 5. Participants 1 and 2 interacted with the agent more frequently than Participant 3 and used it continuously for navigation, troubleshooting, and validation. Participant 3 asked fewer questions overall, but relied strongly on the agent whenever conceptual understanding or problem solving was required.

Participant	Total Questions	Total Task Duration
1	25	59 min 26 s
2	26	58 min 15 s
3	11	44 min 57 s

Tab. 5: Overall chat usage and task duration per participant.

6.4.1. Usability

At the beginning of the sessions, participants mainly used the agent to understand the general purpose of CT2, available templates, and possible workflows. During this familiarization phase, the agent primarily acted as an orientation aid rather than as a problem-solving tool.

A central strength of the agent was its ability to support navigation in CT2. Participants used it to find RSA-related components, templates, categories, logs, the wizard, and the next step in a workflow. Several participants described the agent as especially useful for beginners because it provided step-by-step guidance and explained what to search for or which menu to open next. Even participants who initially described CT2 as confusing or old-fashioned reported that they rarely felt unsure because the agent guided them through the tasks. *(see Appendix B.1, Appendix B.2, Appendix C.1, and Appendix C.2)*

Despite this support, participants encountered several usability problems in the surrounding CT2 interface. They struggled to find the WM+ button at the end of the Wizard, which transfers the configuration selected during the Wizard process into a new

workspace. Participants also had difficulties finding the Start button and the correct way to return from specific menus or views. Difficulties also occurred when participants tried to edit running workspaces, because they did not immediately recognize that execution must first be stopped before changes can be made. During the evaluation, the agent could not yet interpret this state correctly and was therefore unable to explain why modifications were temporarily blocked. *(see Appendix B.1, Appendix B.2, Appendix B.3, and Appendix C.3)*

Participants also criticized the chat interface itself. Long messages were difficult to navigate because scrolling only worked through the scrollbar instead of directly with the mouse wheel. Participants further criticized the minimize, maximize, and scrolling behavior of the chat window, as well as the right-aligned text layout. One participant repeatedly requested shorter and simpler explanations, which suggests that some responses were too detailed for beginners. The scrolling behavior of the chat interface was improved after the evaluation. *(see Appendix B.1, Appendix B.2, and Appendix B.3)*

Participants also emphasized that future automatic workspace modifications should be used carefully. Automatically changing a workspace without explanation could create additional confusion, especially for beginners. Suggested changes should therefore first be explained in the chat and clearly highlighted before being applied. *(see Appendix B.2)*

Some usability problems were caused by the remote study setup rather than by the agent itself. These included missing question marks, keyboard difficulties, swallowed first letters, and copy-paste issues in TeamViewer. *(see Appendix B.2)*

Despite these issues, participants evaluated the integrated chat positively because it could explain the current CT2 context and provide guidance directly inside the application. This reduced uncertainty and helped participants continue when they did not know which action to take next. Overall, the findings suggest that the agent reduced the entry barrier of CT2 by compensating for discoverability and navigation issues in the interface, especially for first-time users who did not yet know where to find components, templates, or important interface elements. One participant explicitly stated that the agent should remain part of CT2 because it helps new users understand what can be done in the application and how to get started. *(see Appendix B.1)*

6.4.2. Understanding of Components and Workspaces

A recurring pattern across all participants was the need to connect abstract cryptographic concepts with their implementation in CT2. Participants asked about the meaning of the RSA parameters e and d, the functionality of homomorphic encryption, and the difference between homophonic substitution and homophonic encryption. This shows that participants used the agent not only to understand the concepts them-

selves, but also to understand how these concepts are represented in CT2 workspaces. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

Participants also used the agent to understand connectors, component labels, and workspace elements, especially during the DES task. They repeatedly asked about *InputKey*, *InputIV*, *OutputBytes*, mandatory inputs, log messages, and key decoders. However, these interactions also revealed limitations of the agent. In several cases, the agent referred to connector names exactly as they were stored internally in CT2, even though these names were not always visible in the user interface. During the DES task, for example, the agent referred to *InputIV* or *InputKey*, while the visible interface instead used German labels such as "Initialisierungsvektor" or "Schlüssel". Similar problems occurred when the agent referred to "templates" instead of "Vorlagen" or instructed users to press "Play" although the visible button was labeled "Starten". In some situations, the agent also described connector positions incorrectly. These inconsistencies made it more difficult for participants to identify the correct interface elements and connection points in the workspace. *(see Appendix C.1, Appendix C.2, Appendix B.1, and Appendix B.3)*

The integrated access to the current workspace was perceived as especially valuable. Participants reacted positively when they realized that the agent could inspect the current workspace, connections, and logs directly. One participant explained that the integrated agent was more practical than an external chatbot because "I don't have to explain as much to him" and because it "knows where I am and more or less what to do." This illustrates that the perceived usefulness of the agent was closely connected to its contextual awareness within CT2. *(see Appendix B.2 and Appendix B.3)*

Participants also emphasized that the agent could explain aspects of CT2 that were not sufficiently explained by the application itself. This was particularly important when participants encountered unfamiliar connectors, workspace labels, or cryptographic terminology. *(see Appendix B.3)*

Overall, the findings suggest that the agent improved the ability to work with CT2 by helping users understand cryptographic concepts, interpret workspaces, and identify suitable components and workflows. This directly addresses the first three research questions concerning usability, conceptual understanding, and support for concrete tasks inside the application. However, the evaluation also showed limitations. Explanations about connectors, labels, and positions in the workspace were not always reliable, which occasionally caused confusion when precise technical details were required.

6.4.3. Learning Support

Several participants first used the agent to understand basic concepts before working on specific tasks. They asked about the general meaning of cryptography, the purpose of CT2, examples of encryption methods, and the meaning of the RSA parameters e and d. These interactions show that the agent helped participants move from general concepts to more specific questions. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

The agent also supported participants in finding suitable starting points inside CT2. It frequently recommended the wizard, templates, and simpler workspaces before moving on to more advanced examples. This reduced the complexity of the first interaction with the application and helped participants build confidence. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

Homophonic substitution was the clearest example of learning support. All participants initially struggled to understand the concept. The agent explained the basic principle, the encryption and decryption process, and the role of homophones for letters and words. Participants often requested simpler explanations first and then followed up with more specific questions. Participant 2 later stated that they had understood the method after using the agent: "I feel like I understand that." Participant 3 expressed an even stronger dependence on the support of the agent, stating that the homophonic encryption task was something they "could never have done it without" the AI Chat. These statements indicate that the agent did not only help participants complete the task, but also supported their conceptual understanding of the method. *(see Appendix B.2, Appendix B.3, Appendix C.1, Appendix C.2, and Appendix C.3)*

The agent also supported the practical application of cryptographic methods. During the Vigenère task, participants recovered the plaintext without fully understanding the algorithm itself. The agent guided them toward the correct components, explained the required steps, and reformatted the output into readable text. *(see Appendix C.1 and Appendix C.3)*

The agent reduced the need to rely on existing CT2 support mechanisms, such as component documentation or YouTube tutorials linked within the application, by providing explanations directly inside the application and relating them to the current task context. When the integrated resources would not have been sufficient, participants indicated that they would have searched for additional information through Google, forums, or external documentation. *(see Appendix B.1, Appendix B.2, and Appendix B.3)*

Overall, the findings suggest that the agent supported users in exploring, understanding, and applying cryptographic methods in CT2. This directly addresses the research question of whether the agent can support learning inside the application. However, the

evaluation also showed that participants sometimes followed the generated instructions without fully understanding them, especially during the Vigenère task.

6.4.4. Problem Solving and Error Correction

The agent was able to identify several independent problems in the same workspace. These included missing *InputKey* connections, incorrect use of *InputIV*, invalid DES key lengths, unsuitable DES modes, incorrect additional connections to *InputStream*, and cases where plaintext was decrypted instead of ciphertext. This shows that the agent could support debugging scenarios with multiple simultaneous issues. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

Participants used the agent iteratively during debugging. They repeatedly asked whether the current workspace configuration was correct, whether a specific connection was required, or whether a parameter such as an "initialization vector" was necessary. Participants often changed the workspace, asked for validation, and then continued with the next correction step. The agent therefore supported both error detection and validation of repairs. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

The agent was especially helpful when participants did not know whether an issue was caused by the algorithm, the workspace configuration, or the selected parameters. Participants were unsure which DES mode to use, which key length DES expects, or whether a selected Vigenère template was suitable. The agent reduced this uncertainty by explaining the role of parameters, execution modes, and workspace elements. *(see Appendix C.1 and Appendix C.2)*

During the error-correction task, all participants used the agent to inspect the workspace, validate configuration choices, or identify possible causes of errors. Two participants stated that error correction was the area in which the agent helped the most. They reported that they would otherwise have needed much more time, external resources, or would have been overwhelmed by the task, with Participant 1 explicitly noting that solving the task without the agent "takes much longer." Even participants who believed that they could eventually have solved some problems independently still described the agent as important for providing orientation during debugging. *(see Appendix B.1, Appendix B.2, and Appendix B.3)*

Overall, the findings suggest that the agent supported problem solving and error correction in CT2 by helping participants identify configuration problems, interpret error messages, and validate repair steps. This was particularly useful in the DES task, where several independent errors had to be detected and corrected iteratively. However, the evaluation also showed that inaccurate explanations about connectors, parameters, or workspace elements could occasionally lead to additional confusion during debugging.

6.4.5. Response Quality

Participants generally described the responses as useful, reliable, and well adapted to the current task. The agent was perceived as especially helpful because it provided clear next steps and concrete instructions for continuing inside CT2. *(see Appendix B.1, Appendix B.2, and Appendix B.3)*

A central strength of the responses was their structure. The agent frequently provided numbered lists, concrete next steps, and step-by-step instructions. This was especially visible in the DES, Vigenère, and homophonic encryption tasks. Participants appreciated that the agent not only explained concepts but also described which component to choose, what to type into the search bar, or how to continue in the workspace. *(see Appendix C.1, Appendix C.2, and Appendix C.3)*

Another important aspect was the combination of conceptual explanations and concrete CT2 actions. Participants valued that the agent could explain a cryptographic concept and directly connect it to relevant components, templates, or settings. This made the responses more practical than those of a generic chatbot without access to the current workspace. One participant explicitly highlighted that the integrated context awareness was useful because the agent already knew the current workspace and the likely next steps. *(see Appendix B.2, Appendix B.3, and Appendix C.2)*

Participants also appreciated that the agent could adapt its explanations when shorter or simpler answers were requested. However, this adaptation was not always sufficient. Some responses remained too detailed or too technical for beginners, especially when participants wanted quick guidance rather than conceptual background information. *(see Appendix B.2 and Appendix C.2)*

The agent was generally perceived as credible and trustworthy. Some participants followed the generated instructions with little verification because previous suggestions had often worked. At the same time, this creates a risk that incorrect suggestions may also be accepted without further checking. This is particularly relevant because, as discussed in the previous sections, some explanations about connectors in the workspace were inaccurate. *(see Appendix B.1, Appendix B.2, and Appendix B.3)*

One limitation concerned context handling across longer conversations. Reusing the same chat thread across multiple unrelated topics sometimes led to irrelevant or confusing responses. In one case, the accumulated chat history exceeded the available context window of the model, which interrupted the interaction and required starting a new chat thread. This issue was addressed after the evaluation by introducing token-budget-based reduction of conversation history and tool outputs, as described in Section 5.3.5. *(see Appendix B.2)*

Overall, the findings suggest that the agent produced responses of sufficient quality to support navigation, learning, and problem solving in CT2. Its main strengths were the structured presentation, the combination of conceptual explanations and concrete actions, and the use of the current workspace context. However, the evaluation also showed that response quality decreased when the agent referred to very specific technical details or when long conversations mixed unrelated topics.

6.4.6. Summary

Overall, the evaluation indicates that the main research question can largely be answered positively. The integrated agent improved the usability of CT2 and reduced the entry barrier for first-time users, especially during navigation, workspace exploration, conceptual understanding, and debugging tasks. Across all evaluated tasks, participants were able to complete the assigned tasks with the support of the agent. In particular, the agent helped reduce uncertainty during workspace exploration, supported the identification of relevant components and settings, and provided guidance when participants were unsure how to continue.

Participants also indicated that, without the agent, they would have relied more strongly on traditional support mechanisms such as documentation, tutorials, external search, or trial-and-error. The agent therefore did not replace these resources but complemented them by providing contextual explanations and concrete next steps directly inside CT2.

The integrated access to the current workspace was perceived as a major advantage because it enabled more specific explanations and more targeted guidance than external chatbots without workspace access. However, the evaluation also revealed limitations, including inaccurate connector names, inconsistent terminology, overly detailed explanations, and context problems in longer conversations.

7. Discussion and Limitations

This chapter discusses the evaluation findings presented in Chapter 6. It interprets the observed effects of the agent on usability, learning support, and problem solving in CT2. It then examines limitations of the AI Chat, including terminology mismatches, missing visual context, and context-window issues. Finally, it discusses limitations of the study design and threats to validity.

7.1. Interpretation of the Results

The evaluation indicates that the AI Chat was particularly useful for reducing the entry barrier for first-time CT2 users. Since all participants were unfamiliar with the software, the study mainly reflects the perspective of beginners who needed orientation, explanations, and support while exploring the application for the first time. In this context, the agent helped participants understand both the purpose of CT2 and the next steps required to complete specific tasks.

A central finding is that the agent was most valuable when participants encountered uncertainty. Instead of searching through documentation, menus, or external search engines, participants could ask context-specific questions and receive answers related to their current workspace or task. This reduced the effort required to understand the interface and allowed participants to focus on the actual tasks. This was especially important because CT2 contains many menus, templates, and components that can be difficult to understand without prior knowledge.

This effect was largely enabled by the direct integration into CT2. Unlike an external chatbot, the agent already had access to the current workspace, open tabs, available templates, component settings, and validation messages. Participants, therefore, did not need to describe their current situation in detail before asking a question. Over time, they recognized this contextual awareness and increasingly expected the agent to understand their current state in CT2. This reduced interaction effort and made the conversation more natural.

The results also suggest that the agent supported different user types in different ways. Participants with more technical confidence mainly used it to save time and validate assumptions, while less experienced participants relied on it more strongly for orientation and guidance. This indicates that the system may be useful both as a learning tool and as a productivity tool.

Another important observation is that the agent supported both conceptual understanding and practical task completion. Participants not only solved the assigned tasks but

also developed at least a basic understanding of concepts such as homophonic encryption and the role of different CT2 components. At the same time, some participants completed tasks successfully without fully understanding the underlying algorithms. This suggests that the agent can support both learning-oriented and goal-oriented usage styles.

The DES repair task was the clearest example of this combination of learning and problem solving. In this scenario, the agent identified likely causes of errors, explained them in simple terms, and guided participants step by step toward a solution. Several participants stated that they would otherwise have needed much more time or external help.

One particularly notable result was the performance of Participant 3. Despite having no prior knowledge of cryptography and only moderate confidence in using unfamiliar software, this participant completed all tasks successfully and was the fastest overall. This suggests that the agent can compensate for missing domain knowledge to some extent, although the participant's willingness to experiment with unfamiliar software likely also contributed to this result.

The evaluation also suggests that the agent may be particularly important for users who would otherwise abandon CT2 because of its complexity. Although Participant 3 repeatedly stated being unsure what to do, all tasks were still completed successfully. Without the agent, this participant may have eventually stopped working on the tasks. In contrast, Participants 1 and 2 would likely have completed the tasks without the integrated chat but with substantially more effort and time.

It was not possible to observe whether participants became more independent from the agent over time. Since the individual tasks focused on different goals and required different forms of interaction, changes in usage behavior during the evaluation cannot be interpreted as a learning effect.

7.2. Limitations of the AI Chat

Although the agent was generally perceived as useful, the evaluation also revealed several limitations of the current implementation.

One limitation concerned the use of internal terminology that did not always match the visible CT2 interface. In various cases, the agent referred to internal connector names or English interface terms that were not directly visible to the user. This occasionally made it more difficult to identify the correct interface elements during debugging and workspace modification.

The evaluation also showed that the agent can only work with the information available to it. Although it had access to structured workspace data, it could not directly interpret the visual layout of the interface. As a result, the agent sometimes referred to connectors or buttons without knowing their exact position on screen. During the DES task, this led to situations in which participants searched for the wrong connector because it was located on a different side of the component than expected. This suggests that future versions could benefit from additional visual context, such as screenshots, visible labels, or component positions.

Long conversations remain a limitation of the system. During the evaluation, one conversation exceeded the available context window of the model, and longer chat threads sometimes led to less relevant responses when several unrelated topics were mixed. This issue was partially addressed after the evaluation by introducing token-budget-based reduction of conversation history and tool outputs. Nevertheless, context management remains a limitation because reduced histories may omit older information that could still be relevant when users return to previous topics later in the conversation.

Direct workspace modification also introduces new risks. Although this issue did not appear during the evaluation because direct workspace modification was not yet available at that time, the current implementation can make it difficult to understand which changes the agent has applied to the workspace. Although users can undo modifications through the existing undo functionality and the agent usually explains its actions, incorrect changes can still create additional confusion or temporarily break the workspace. Future versions should therefore make changes more transparent, clearly highlight modified components and connections, and provide simple ways to review or revert changes.

As discussed in the security considerations in Section 4.7, direct workspace modification also increases the potential impact of prompt injection attacks. Malicious prompts, manipulated workspace content, or misleading component labels could influence the generated actions and cause unintended workspace modifications. Even though the current implementation already contains user guides, automatic modifications should therefore remain restricted and transparent.

Finally, the agent still has technical limitations regarding conversation length. During one interaction, the accumulated conversation history exceeded the available context length of the model and caused an error. As described in Section 5.3.5, this issue was addressed after the evaluation by limiting the amount of chat history sent to the model. However, this also means that older messages may no longer be available to the model, even if they are still relevant for the current conversation.

7.3. Limitations of the Evaluation

The evaluation has several methodological limitations that reduce the generalizability of the results.

First, the study included only three participants. Although the selected participants had different educational backgrounds and different levels of confidence with software tools, the sample size is too small to draw strong conclusions about the usefulness of the agent for all CT2 users. In particular, all participants were beginners with no prior CT2 experience. It therefore remains unclear how useful the agent would be for more advanced users, teachers, or students with stronger backgrounds in cryptography.

The study design may also have influenced participant behavior. Participants knew that the evaluation focused on the AI Chat and were encouraged to use it during the tasks. In situations where participants directly asked the interviewer for help, they were often redirected to the agent. As a result, the observed chat usage may be higher than it would be under completely natural conditions.

Another limitation is that the interviewer occasionally intervened when participants became stuck or spent too much time on a task. This was done to ensure that the total interview duration remained manageable and did not exceed roughly 90 minutes. For example, Participant 1 would likely have continued exploring the application for a much longer time before moving on to the actual tasks. Such interventions may have influenced the observed task completion times and user behavior.

The evaluation was conducted only with a single LLM and one specific system configuration. Different results might be obtained with other language models and different prompting strategies. In particular, no local language model was evaluated, so the results cannot be generalized to smaller or self-hosted models with different capabilities and performance characteristics. Since the capabilities of modern AI systems are developing rapidly, the findings of this study may become outdated relatively quickly.

Another limitation is that the evaluation was conducted before direct workspace modification was available. As a result, potential problems caused by automatic changes to the workspace, such as reduced transparency, incorrect modifications, or overreliance on generated actions, could not be observed during the study.

The selected learning tasks also covered only a limited range of cryptographic concepts. For example, homophonic substitution is comparatively easy to explain because it relies on intuitive examples and simple mappings between symbols. It therefore remains unclear whether the agent would be equally effective for more complex topics such as asymmetric cryptography, homomorphic encryption, or advanced protocol analysis.

There were also some technical issues during the evaluation itself. TeamViewer introduced a slight input delay that may have made the software appear slower than in a normal local setup. Participant 2 additionally experienced keyboard issues, including trouble with the Shift key and the inability to enter a question mark. In the third interview, the prepared DES workspace was accidentally not broken in exactly the same way as in the previous sessions and had to be modified again during the interview. These inconsistencies may have influenced the comparability of the results.

7.4. Threats to Validity

Several threats to validity must be considered when interpreting the results of this evaluation.

The think-aloud protocol may have influenced participant behavior. Since participants continuously verbalized their thoughts, they may have worked more slowly, systematically, or consciously than they would under normal conditions. In addition, the interviewer occasionally provided hints or redirected participants to the agent when they became stuck. As a result, participants may have solved problems more quickly than they would have without researcher involvement.

The composition of the participant group is another threat to validity. All participants were relatively young, technically interested, and already familiar with AI chatbots. Users with less technical experience or no familiarity with AI systems may struggle more with the agent and may not benefit from it to the same extent.

Language may also have influenced the findings. The evaluation was conducted entirely in German and used the German version of CT2. Some observed problems resulted from inconsistencies between English technical terms generated by the agent and the German labels visible in the interface. With English-speaking participants and an English version of CT2, some of these issues might not have occurred, while other language-related challenges could have arisen instead.

Finally, the study only evaluated short-term usage. It remains unclear whether the same benefits would still be observed over multiple days or weeks. The agent was especially useful when participants first encountered CT2 and needed orientation. Over time, this benefit may decrease as users become more familiar with the software. At the same time, advanced users may become less tolerant of hallucinations or incorrect answers, which could reduce trust in the agent.

8. Conclusion and Future Work

This chapter summarizes the main findings of the thesis and discusses their implications for CT2 and cryptography education. It then outlines possible directions for future work, including further evaluation, improved context handling, and additional interface support, before concluding with a final assessment of the proposed approach.

8.1. Summary and Main Findings

This thesis presented the design, implementation, and evaluation of an AI-based chat agent integrated directly into CT2. The main contribution is the context-aware integration of an LLM into the application. Instead of acting as a generic external chatbot, the agent can access information about the current CT2 state, including open tabs, active workspaces, available templates, component settings, and validation errors. This allows the agent to provide more specific and relevant answers than a browser-based chatbot without access to the application context.

The evaluation showed that the agent provides clear benefits for first-time CT2 users. It reduced the entry barrier of the software and helped participants complete tasks more quickly. Instead of searching through documentation, menus, or external resources, participants could ask task-specific questions and immediately receive answers related to their current workspace.

Another important finding is that the agent compensated for several usability challenges of the CT2 interface. Participants often described CT as difficult to understand, not intuitive, or overwhelming for beginners. Nevertheless, all participants completed all tasks successfully with only limited intervention by the interviewer. This suggests that the agent can reduce uncertainty and help users remain productive even when they are unfamiliar with the software.

The evaluation also showed that the current prototype is already useful despite its limitations. Although issues such as overly technical responses, limited context windows, and missing visual understanding of the interface still exist, the agent was able to support both conceptual learning and practical problem-solving in CT2.

8.2. Implications for CT2 and Cryptography Education

The results of this thesis suggest that an integrated AI Chat could become a valuable long-term feature of CT2. Since CT2 contains many functions, templates, and compo-

nents, beginners may struggle to understand how to use the software effectively. The evaluation showed that the agent can reduce this entry barrier by providing direct support whenever users encounter difficulty or uncertainty.

At the same time, the agent does not replace the existing strengths of CT2. As discussed in earlier chapters, CT2 already offers a large collection of tools, templates, and educational workspaces that are not available in most other cryptography learning environments. These features remain one of the main strengths of the software. The agent should therefore complement the traditional CT2 experience rather than replace it.

The findings are also relevant beyond CT2. Although the implementation presented in this thesis is specific to CT2, the general concept of integrating a context-aware agent into educational software can likely be transferred to other applications. Software with a steep learning curve, many features, or a complex interface could benefit from similar support.

The strongest educational benefit appears in self-directed learning scenarios. Students can explore CT2 independently, ask questions when they become stuck, and receive immediate explanations without searching through documentation or external websites. This may be especially useful for users who want to learn cryptography outside formal teaching environments.

The evaluation further suggests that the agent mainly reduces frustration and uncertainty. Users can solve small problems more quickly and are less likely to become stuck because of confusing menus, missing connections, or unclear error messages. As a result, they may be more likely to continue working with the software instead of abandoning it.

The results also indicate that the agent supports actual learning rather than only compensating for missing knowledge. Although some tasks could be solved without fully understanding the underlying concepts, participants still developed at least a basic understanding of the relevant cryptographic methods and CrypTool components.

8.3. Future Work

One important area for future work is the evaluation of direct workspace modification. Although the current prototype already supports automatic changes to workspaces, these features were not part of the evaluation conducted in this thesis. Future studies should therefore investigate their effects on usability, trust, transparency, and user control, as well as suitable mechanisms for highlighting, reviewing, confirming, or undoing modifications.

Similar visual support could also improve manual interaction with the agent. During the evaluation, participants sometimes struggled to identify the correct connector or button because the agent used technical names that did not exactly match the visible user interface. Future versions could therefore visually highlight relevant connectors, components, or buttons directly in the CT2 interface when the user chooses not to allow automatic workspace modification.

Context management also remains an important topic. Although the current implementation, added after the evaluation, includes token-budget-based reduction of conversation history and tool outputs, future work could further improve how relevant information is preserved across longer interactions. Possible extensions include more advanced summarization, selective long-term memory, or more fine-grained strategies for deciding which tool results should remain available to the model. This is particularly relevant for locally hosted models with smaller context windows, where both the conversation history and the available tool definitions must be handled carefully.

Another possible direction concerns interoperability with external tools and systems through the MCP. Future versions of the system could use it in two different ways.

First, CT2 itself could expose selected functionality through an MCP server. In this scenario, external LLMs or other software systems could connect to CT2 and directly access its cryptographic components, templates, and workspace functions. This would allow external systems to perform encryption, decryption, cryptanalysis, or workflow generation through CT2 without requiring users to manually create, debug, and execute workspaces themselves.

Second, the integrated agent inside CT2 could connect to external MCP-compatible systems. For example, future versions could access external databases containing cryptographic knowledge, vulnerability information, or attack techniques. Other useful MCP servers could provide access to recent research papers in cryptology, mathematical computation tools, code execution environments, or image analysis systems. This would allow the agent to combine the internal capabilities of CT2 with additional external knowledge and services without requiring tightly coupled integrations.

Future studies should include larger participant groups and more diverse user profiles, including experienced CT2 users, English-speaking users, teachers, and students with different levels of technical knowledge. It would also be useful to compare local models with API-based models in order to investigate how strongly model quality influences the usefulness of the agent.

Future work could also investigate the use of multimodal models that can process both text and images. The evaluation showed that some problems resulted from the lack of visual understanding of the workspace. Models with access to screenshots or inter-

face layouts could potentially identify connectors, labels, and component positions more reliably.

8.4. Final Conclusion

This thesis investigated whether an LLM-based agent integrated into CT2 can provide meaningful support for users in the context of cryptography and cryptanalysis education.

The results show that the implemented agent can reduce the entry barrier of CT2, support users during navigation, learning, and debugging, and provide more relevant answers than an external chatbot without access to the current application context.

Overall, the work demonstrates that context-aware AI integration can be a meaningful extension for CT2. Although the current prototype still has limitations, it already provides practical value for beginners and creates a foundation for future research on AI-supported cryptography education.

Bibliography

[1] Sally Addad. "Implementation and Visualization of Steganography Procedures in CrypTool2". Bachelor's Thesis. Siegen, Germany: Universität Siegen, 2020.

[2] Christian Bender. "Analyse symmetrischer Blockchiffren mittels differenzieller Kryptoanalyse in CrypTool 2". Master's Thesis. Siegen, Germany: Universität Siegen, 2019.

[3] Benjamin Nye, Dillon Mee, and Mark G. Core. "Generative Large Language Models for Dialog-Based Tutoring: An Early Consideration of Opportunities and Concerns". In: *LLM@AIED 2023*. 2023. URL: https://api.semanticscholar.org/CorpusID:262068884.

[4] Doug Bonderud. *Cost of a Data Breach 2024: Financial Industry*. IBM. 2024. URL: https://www.ibm.com/think/insights/cost-of-a-data-breach-2024-financial-industry (visited on 04/06/2026).

[5] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. URL: https://arxiv.org/pdf/2005.14165.

[6] Bundesamt für Sicherheit in der Informationstechnik. *Die Lage der IT-Sicherheit in Deutschland 2025*. Bonn: Bundesamt für Sicherheit in der Informationstechnik (BSI), 2025. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2025_Achtseiter.pdf?__blob=publicationFile&v=7 (visited on 04/06/2026).

[7] Weiru Chen et al. "Exploring Cybersecurity Education at the K-12 Level". In: *Proceedings of SITE Interactive Conference* (2021). URL: https://digitalcommons.odu.edu/itds_facpubs/61.

[8] Yuheng Cheng et al. *Exploring Large Language Model based Intelligent Agents: Definitions, Methods, and Prospects*. 2024. DOI: 10.48550/arXiv.2401.03428.

[9] Zhendong Chu et al. *LLM Agents for Education: Advances and Applications*. 2025. DOI: 10.48550/arXiv.2503.11733.

[10] Continue.dev. *Continue: Open-Source AI Code Assistant*. 2026. URL: https://marketplace.visualstudio.com/items?itemName=Continue.continue (visited on 05/02/2026).

[11] CrypTool Project. *Contributors*. 2026. URL: https://www.cryptool.org/en/contributors/ (visited on 04/27/2026).

[12] CrypTool Project. *CrypTool*. 2026. URL: https://www.cryptool.org/en/ (visited on 04/27/2026).

[13] CrypTool Project. *CrypTool 2*. Version 2.1. 2024. URL: https://www.cryptool.org/en/ct2/ (visited on 03/10/2026).

[14] CrypTool Project. *CrypTool 2 Downloads*. 2026. URL: https://www.cryptool.org/en/ct2/downloads/ (visited on 05/02/2026).

[15] CrypTool Project. *CrypTool 2 GitHub Repository*. 2025. URL: https://github.com/CrypToolProject/CrypTool-2 (visited on 03/06/2026).

[16] Discord Inc. *Discord*. Voice communication software. 2026. URL: https://discord.com/ (visited on 05/10/2026).

[17] Duolingo Team. *Introducing Duolingo Max, a learning experience powered by GPT-4. AI and education make a great duo.* 2023. URL: https://blog.duolingo.com/duolingo-max/ (visited on 03/17/2026).

[18] Duolingo Team. *Video Call lets you have real life conversations with Lily. Now you can practice your new language by having a quick chat with Lily!* 2024. URL: https://blog.duolingo.com/video-call/ (visited on 03/17/2026).

[19] Bernhard Esslinger. *Das CrypTool-Buch: Kryptografie lernen und anwenden mit CrypTool und SageMath. Kryptografie, Mathematik und mehr mit dem freien E-Learning-Programm CrypTool*. 2nd ed. Berlin: Lehmanns Media, 2025. 899 pp. ISBN: 9783965436107.

[20] GitHub. *GitHub Copilot*. 2026. URL: https://github.com/features/copilot (visited on 05/02/2026).

[21] A. Gorodilova et al. *An overview of the Eight International Olympiad in Cryptography "Non-Stop University CRYPTO"*. 2022. URL: https://arxiv.org/pdf/2204.11502.

[22] Wayne Holmes, Maya Bialik, and Charles Fadel. "Artificial intelligence in education". In: *Data ethics : building trust : how digital technologies can serve humanity*. Ed. by Christoph Stückelberger and Pavan Duggal. Globethics Publications, 2023, pp. 621–653. ISBN: 9782889315246. DOI: 10.58863/20.500.12424/4276068.

[23] Lei Huang et al. *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. 2025. DOI: 10.1145/3703155. URL: https://arxiv.org/pdf/2311.05232.

[24] Dylan Huitema and Albert Wong. *A Case Study in Gamification for a Cybersecurity Education Program: A Game for Cryptography*. 2025. URL: https://arxiv.org/pdf/2502.06706.

[25] IBM. *IBM-Studie: Kosten von Datenlecks erreichen neues Rekordhoch*. IBM Newsroom Deutschland. 2024. URL: https://de.newsroom.ibm.com/2024-07-30-IBM-Studie-Kosten-von-Datenlecks-erreichen-neues-Rekordhoch (visited on 04/06/2026).

[26] James Newton-King. *Json.NET Documentation*. 2026. URL: https://www.newtonsoft.com/json (visited on 04/16/2026).

[27] Ziwei Ji et al. "Survey of Hallucination in Natural Language Generation". In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38. ISSN: 0360-0300. DOI: 10.1145/3571730. URL: https://arxiv.org/pdf/2202.03629.

[28] Stylianos Karagiannis and Emmanouil Magkos. "Adapting CTF challenges into virtual cybersecurity learning environments". In: *Information & Computer Security* 29.1 (2021), pp. 105–132. ISSN: 2056-4961. DOI: 10.1108/ICS-04-2019-0050.

[29] Enkelejda Kasneci et al. "ChatGPT for good? On opportunities and challenges of large language models for education". In: *Learning and Individual Differences* 103 (2023). PII: S1041608023000195, p. 102274. ISSN: 10416080. DOI: 10.1016/j.lindif.2023.102274.

[30] Khan Academy. *Khanmigo*. 2026. URL: https://www.khanmigo.ai/ (visited on 03/17/2026).

[31] Daniel Koch, Andreas Kohne, and Nils Brechbühler. *Prompt Engineering im Unternehmen – eine Einführung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2025. ISBN: 978-3-658-47698-4. DOI: 10.1007/978-3-658-47699-1.

[32] Mojtaba Komeili, Kurt Shuster, and Jason Weston. "Internet-Augmented Dialogue Generation". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Dublin, Ireland). Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Stroudsburg, PA, USA: Association for Computational Linguistics, 2022, pp. 8460–8478. DOI: 10.18653/v1/2022.acl-long.579.

[33] Abdullah Konak. "Experiential Learning Builds Cybersecurity Self-Efficacy in K-12 Students". In: *Journal of Cybersecurity Education, Research and Practice* 2018.1 (2018).

[34] Nils Kopal. "Design und Entwicklung einer schnellen Laufzeitumgebung für CrypTool 2.0". Bachelor. Universität Duisburg-Essen, 2010. URL: https://www.cryptool.org/media/publications/theses/BA_Kopal.pdf (visited on 01/14/2026).

[35] Nils Kopal and Bernhard Esslinger, eds. *CrypTool 2 – Ein Open-Source-Projekt zur Kryptologie für Lehre, Forschung, Selbststudium und Experimentieren*. DACH Security 2018 (Gelsenkirchen). syssec – Austrian Association for Secure Information Systems. Peter Schartner, Norbert Pohlmann, 2018. ISBN: 978-3-00-060424-9. URL: https://www.syssec.at/de/veranstaltungen/dachsecurity2018/papers/DACH_Security_2018_Paper_11A2.pdf (visited on 03/05/2026).

[36] Nils Kopal, Olga Kieselmann, Arno Wacker, and Bernhard Esslinger. "CrypTool 2.0". In: *Datenschutz und Datensicherheit - DuD* 38.10 (2014). PII: 274, pp. 701–708. ISSN: 1614-0702. DOI: 10.1007/s11623-014-0274-7.

[37] LM Studio. *LM Studio*. 2026. URL: https://lmstudio.ai/ (visited on 04/16/2026).

[38] Lumivero. *Citavi: Reference Management Software*. 2026. URL: https://lumivero.com/products/citavi/ (visited on 05/09/2026).

[39] Maryam Mehreen. "Usability Analysis of CrypTool-Online and CrypTool 2". Master's Thesis. University of Siegen, 2022.

[40] Alfred J. Menezes, Scott A. Vanstone, and Paul C. van Oorschot. *Handbook of Applied Cryptography*. 1st. USA: CRC Press, Inc, 1996. ISBN: 0849385237.

[41] Microsoft. *Introduction to Semantic Kernel*. 2024. URL: https://learn.microsoft.com/en-us/semantic-kernel/overview/ (visited on 04/16/2026).

[42] Microsoft. *Microsoft 365 Copilot Overview*. 2026. URL: https://learn.microsoft.com/en-us/copilot/microsoft-365/microsoft-365-copilot-overview (visited on 05/02/2026).

[43] Microsoft. *Semantic Kernel*. 2026. URL: https://github.com/microsoft/semantic-kernel (visited on 03/22/2026).

[44] Microsoft. *Semantic Kernel documentation*. 2026. URL: https://learn.microsoft.com/en-us/semantic-kernel/ (visited on 03/22/2026).

[45] Microsoft. *Visual Studio IDE*. 2026. URL: https://visualstudio.microsoft.com/ (visited on 04/16/2026).

[46] Microsoft. *Windows Presentation Foundation Documentation*. 2026. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/ (visited on 04/16/2026).

[47] Shervin Minaee et al. *Large Language Models: A Survey*. 2024. DOI: 10.48550/arXiv.2402.06196.

[48] Model Context Protocol Project. *What is the Model Context Protocol (MCP)?* LF Projects. 2025. URL: https://modelcontextprotocol.io/docs/getting-started/intro (visited on 03/20/2026).

[49] OBS Project. *OBS Studio*. Open-source software for screen and audio recording. 2026. URL: https://obsproject.com/ (visited on 05/10/2026).

[50] OpenAI. *GPT-5.2*. Large language model used with medium reasoning effort. 2025. URL: https://developers.openai.com/api/docs/models/gpt-5.2 (visited on 05/10/2026).

[51] OpenAI. *Models*. 2026. URL: https://developers.openai.com/api/docs/models/all (visited on 04/16/2026).

[52] Jon Ander Oribe. *The Model Context Protocol (MCP) Emergence, Technical Architecture, and the Future of Agentic AI Infrastructure*. 2025. DOI: 10.5281/ZENODO.17390299.

[53] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. URL: https://arxiv.org/pdf/2203.02155.

[54] Overleaf. *Overleaf: Online LaTeX Editor*. 2026. URL: https://www.overleaf.com/ (visited on 05/09/2026).

[55] Christof Paar, Jan Pelzl, and Tim Güneysu. *Understanding cryptography. From established symmetric and asymmetric ciphers to post-quantum algorithms*. Springer, 2009. ISBN: 978-3-642-04100-6. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=31343018.

[56] Nathan Percival, Pranathi Rayavaram, Sashank Narain, and Claire Seungeun Lee. *CryptoScratch: Developing and evaluating a block-based programming tool for teaching K-12 cryptography education using Scratch*. 2022. DOI: 10.1109/EDUCON52537.2022.9766637. URL: https://arxiv.org/pdf/2302.11606.

[57] Pejman Peykani, Fatemeh Ramezanlou, Cristina Tanasescu, and Sanly Ghanidel. "Large Language Models: A Structured Taxonomy and Review of Challenges, Limitations, Solutions, and Future Directions". In: *Applied Sciences* 15.14 (2025). PII: app15148103, p. 8103. DOI: 10.3390/app15148103.

[58] Changle Qu et al. *Tool learning with large language models: a survey*. PII: 40678. 2025. DOI: 10.1007/s11704-024-40678-2. URL: https://arxiv.org/pdf/2405.17935 (visited on 05/10/2026).

[59] Mohaimenul Azam Khan Raiaan et al. "A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges". In: *IEEE Access* 12 (2024), pp. 26839–26874. DOI: 10.1109/ACCESS.2024.3365742.

[60] Pranathi Rayavaram. *CryptoEL: A Novel Experiential Learning Tool for Enhancing K-12 Cryptography Education*. 2025. URL: https://github.com/PranathiRayavaram/CryptoEL (visited on 04/08/2026).

[61] Pranathi Rayavaram, Ukaegbu Onyinyechukwu, Maryam Abbasalizadeh, Krishnaa Vellamchetty, and Sashank Narain. *CryptoEL: A Novel Experiential Learning Tool for Enhancing K-12 Cryptography Education*. 2024. DOI: 10.48550/arXiv.2411.02143.

[62] Pranathi Rayavaram et al. *Designing a Visual Cryptography Curriculum for K-12 Education*. 2022. URL: https://arxiv.org/pdf/2302.11655.

[63] Salesforce. *Agentforce: AI Agents for Business*. 2026. URL: https://www.salesforce.com/de/agentforce/ (visited on 05/02/2026).

[64] SAP SE. *Joule: AI Assistant for Business*. 2026. URL: https://www.sap.com/germany/products/artificial-intelligence/ai-assistant.html (visited on 05/02/2026).

[65] Timo Schick et al. *Toolformer: Language Models Can Teach Themselves to Use Tools*. 2023. URL: https://arxiv.org/pdf/2302.04761.

[66] sciebo. *sciebo: The Campus Cloud*. 2026. URL: https://hochschulcloud.nrw/ (visited on 05/09/2026).

[67] Murray Shanahan. *Talking About Large Language Models*. 2022. DOI: 10.48550/arXiv.2212.03551.

[68] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. "A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs)". In: *Preprints* (2025). April. DOI: 10.20944/preprints202504.0245.v1.

[69] William Stallings. *Cryptography and network security. Principles and practice*. 6th ed. Boston: Pearson, 2014. 731 pp. ISBN: 9780133354690.

[70] TabbyML. *Tabby: Open-source, Self-Hosted AI Coding Assistant*. 2026. URL: https://www.tabbyml.com/ (visited on 05/09/2026).

[71] TeamViewer Germany GmbH. *TeamViewer*. Remote access and screen sharing software. 2026. URL: https://www.teamviewer.com/ (visited on 05/10/2026).

[72] TeX Users Group. *TeX Live*. 2026. URL: https://www.tug.org/texlive/ (visited on 05/09/2026).

[73] The LaTeX Project. *LaTeX: A Document Preparation System*. 2026. URL: https://www.latex-project.org/ (visited on 05/09/2026).

[74] UNESCO. *Guidance for generative AI in education and research*. UNESCO, 2023. ISBN: 9789231006128. DOI: 10.54675/EWZM9535.

[75] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/arXiv.1706.03762. URL: https://arxiv.org/pdf/1706.03762.

[76] Lei Wang et al. "A survey on large language model based autonomous agents". In: *Frontiers of Computer Science* 18.6 (2024). ISSN: 2095-2228. DOI: 10.1007/s11704-024-40231-1.

[77] Tianfu Wang et al. "LLM-powered Multi-agent Framework for Goal-oriented Learning in Intelligent Tutoring System". In: *Companion Proceedings of the ACM on Web Conference 2025*. WWW '25: The ACM Web Conference 2025 (Sydney NSW Australia). Ed. by Guodong Long et al. New York, NY, USA: ACM, 2025, pp. 510–519. ISBN: 9798400713316. DOI: 10.1145/3701716.3715244.

[78] Niklas Weimann. "Implementierung und Visualisierung verschiedener kryptografischer Handverfahren und Codes in CrypTool 2". Bachelor's Thesis. Universität Siegen, 2021.

[79] Beverly Park Woolf. *Building intelligent interactive tutors. Student-centered strategies for revolutionizing e-learning*. Amsterdam and Boston: Morgan Kaufmann Publishers/Elsevier, 2009. xii, 467. ISBN: 978-0-12-373594-2.

[80] Huaiyuan Yao, Wanpeng Xu, Justin Turnau, Nadia Kellam, and Hua Wei. *Instructional Agents: Reducing Teaching Faculty Workload through Multi-Agent Instructional Design*. 2025. URL: https://arxiv.org/pdf/2508.19611 (visited on 05/10/2026).

[81] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2022. URL: https://arxiv.org/pdf/2210.03629.

[82] Olaf Zawacki-Richter, Victoria I. Marín, Melissa Bond, and Franziska Gouverneur. "Systematic review of research on artificial intelligence applications in higher education – where are the educators?" In: *International Journal of Educational Technology in Higher Education* 16.1 (2019). PII: 171. DOI: 10.1186/s41239-019-0171-0.

[83] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. 2021. URL: https://arxiv.org/pdf/2102.09690.

[84] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2023. DOI: 10.48550/arXiv.2303.18223. URL: https://arxiv.org/pdf/2303.18223.

List of Abbreviations

AES	Advanced Encryption Standard
AI	Artificial Intelligence
API	Application Programming Interface
BSI	Bundesamt für Sicherheit in der Informationstechnik
CT	CrypTool
CTF	Capture-the-Flag
CT2	CrypTool 2
DES	Data Encryption Standard
DPAPI	Windows Data Protection API
ECC	Elliptic-Curve Cryptography
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ITS	Intelligent Tutoring System
JSON	JavaScript Object Notation
LLM	Large Language Model
MCP	Model Context Protocol
ONNX	Open Neural Network Exchange
RSA	Rivest-Shamir-Adleman
UI	User Interface
UML	Unified Modeling Language
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language

List of Tables

1. Overview of the main Microsoft Semantic Kernel plugins and their tools 55
2. Analytical dimensions used to structure the qualitative evaluation. 63
3. Participant profiles. 64
4. Self-assessed experience ratings of the participants on a scale from 0 to 5. 64
5. Overall chat usage and task duration per participant. 66
6. Overview of AI tools used during the preparation of this thesis. 115

List of Listings

1. Simplified implementation of a Semantic Kernel tool function. 56

2. Simplified excerpt of the serialized workspace representation returned to the LLM. 57

3. Default system prompt 93

List of Figures

1. CrypTool 2 Startcenter providing access to workflow templates, main functions, and learning resources. . . . 16
2. Screenshot of the CrypTool 2 workspace showing the template "Homophonic Substitution Cipher and Nomenclature – Encryption" with highlighted interface elements. . . . 18
3. Conceptual structure of a CrypTool 2 workspace showing components, their connectors, and their connections. . . . 19
4. Simplified architecture of the CrypLLM integration into CT2. . . . 31
5. Layered architecture of CrypTool 2 with the separate CrypLLM AI layer. 34
6. UML component diagram of the CrypLLM subsystem illustrating the main classes, tool plugins, and communication with the CrypTool 2 host application. The *UiTextCatalogPlugin* was omitted for readability. . . . 49
7. Execution workflow of a user request in CrypLLM, including optional tool calls through Semantic Kernel. . . . 52
8. Embedded AI Chat interface in CrypTool 2. . . . 58
9. AI Chat during response generation. . . . 59
10. Error message in the AI Chat. . . . 60
11. First page of the AI Chat settings. . . . 61
12. A prepared DES workspace with intentionally inserted configuration errors used during the evaluation. . . . 65

Appendix

A. Default System Prompt

Listing 3 shows the default system prompt used by the integrated agent in CT2. It defines the agent's role, tone, scope of support, and interaction constraints, thereby guiding its behavior during user interactions.

Code Listing 3: Default system prompt

```
ROLE

You are the built-in chat agent of the CrypTool 2 application.
You support users in understanding cryptography and working effectively
   within CrypTool 2.

TONE & ADAPTATION

Maintain a clear, supportive, and encouraging tone.

The user's knowledge level is unknown. Infer it from their questions and
    dynamically adapt:
- Depth of explanation
- Terminology
- Level of abstraction

Avoid over-correcting minor inaccuracies to prevent discouragement.
   Clarify misconceptions tactfully when necessary.

SCOPE OF SUPPORT
1. Always use the tool get_open_tabs
2. Depending on the open tab always use the tool:
   get_..._description_text
3. Try to answer the user's question as best you can using the tools and
    your knowledge.

TOOL USAGE

- Call up explanatory tools whenever the context is relevant.
- Use available tools whenever possible to retrieve workspace or
   component information before asking the user.
- Only ask concise clarification questions if required information
   cannot be accessed via tools.
```

```
- Treat tool output as internal reasoning data.
- Never expose internal IDs, raw tool data, or technical metadata unless
    explicitly requested.
- If access to a required function is denied, inform the user that it
   can be activated in the settings -mention this only once per
   conversation unless the user asks again.

CRYPTOOL 2 CONTEXT

- When the user refers to something "on the screen" or "here" they mean
   the currently active editor and/or the active workspace (if open).

LIMITATIONS

- Users cannot upload images to you, so do not ask the user to upload
   images.
- You are not able to search on the internet
- If a feature is unsupported, clearly and briefly state that it is not
   supported.

FORMAT REQUIREMENTS

All responses MUST:
- Use Markdown
- Use headings and lists when helpful
- Use tables when appropriate
- Use fenced code blocks with proper language hints for code
- Be concise, structured, and focused
```

B. Think Aloud Protocols and Interview Transcripts

This section presents three-page excerpts from transcripts covering the think-aloud task phase, followed by the corresponding interviews conducted during the evaluation. The full version is included with the digital edition. The transcripts were generated automatically and translated from German into English using AI-supported tools. Minor deviations in wording may therefore be present.

Each transcript is structured into four columns. The first column identifies the speaker, the second shows the timestamp in minutes and seconds, the third contains the original German statement, and the fourth contains the English translation.

B.1. Participant 1

Participant 1	[00:46]	Boah, 3 oder 3,5	Wow, 3 or 3.5
Participant 1	[00:49]	Wenn du Zwischenbewertungen hast.	If you have interim assessments.
Interviewer	[00:51]	Ja, ja, kann ich mir ja noch aussuchen. Genau, Sicherheit, wie sicher fühlst du dich im Umgang generell mit Software-Tools, mit irgendwelchen Anwendungen, wenn du die neu verwendest, quasi,	Yes, yes, I can still choose. How confident do you feel when using new software tools in general? With any applications, when you're using them for the first time, so to speak?
Participant 1	[01:06]	ähh, skalarmäßig oder einfach nur beschreiben?	Er, in terms of scale, or just describe it?
Interviewer	[01:08]	Wieder von 0 bis 5	Again from 0 to 5
Participant 1	[01:09]	Hm, gut, kannst du jetzt nicht so leicht sagen, weil es kommt immer auf die Anwendung an, aber ja, ich würde so sagen, auch nehme ich mal eine 2,5. Manche sind einfacher, manche— da brauchst ein bisschen Zeit, um reinzukommen, aber wenn, dann geht's.	Well, it's hard to say, because it always depends on the application, but yes, I would say so, I'd give it a 2.5. Some are easier, some—you need a little time to get used to them, but once you do, it's fine.
Interviewer	[01:18]	das geht	that's possible
Interviewer	[01:22]	Ja, sehr gut. Dann fangen jetzt gleich die Aufgaben an. Ich sag noch mal kurz, ja, bitte laut denken dabei.	Yes, very good. Then let's start with the tasks right away. I'll say it again briefly: yes, please think out loud while you do them.
Interviewer	[01:35]	Und ja, genau, ich habe mir da so eine kleine Zeit beigeschrieben bei die einzelnen Aufgaben. Ich glaube nicht, dass du die Zeit brauchst, aber ich würde mir— würde immer kurz auf die Uhr gucken, dass es so so ungefähr passt, dass wir nicht zu viel darüber sind, damit es nicht so lange dauert.	And yes, exactly, I've noted down a little time for each of the individual tasks. I don't think you need the time, but I would— I would always glance at the clock to make sure that we're not going overboard, so that it doesn't take too long.

Participant 1	[01:49]	OK	Alright

Tasks

Interviewer	[01:51]	Genau, dann fangen wir mit der ersten Aufgabe an. Also die erste Aufgabe ist quasi, sich erstmal in CrypTool 2, ja, umzugucken, zu gucken, was es gibt, was kann man da machen. Und dann kannst du, wenn du dich erstmal so ein bisschen eingefunden hast, versuchen herauszufinden, was so ein typischer Arbeitsablauf sein könnte in CrypTool 2. Wenn man zum Beispiel eine neue kryptografische Methode lernen möchte.	Right, then let's start with the first task. So, the first task is basically to take a look around CrypTool 2, see what's there, what you can do with it. And then, once you've familiarised yourself a little, you can try to figure out what a typical workflow might look like in CrypTool 2. For example, if you want to learn a new cryptographic method.
Participant 1	[02:17]	Ja, OK	Yes, OK.
Interviewer	[02:19]	Genau, da hab ich jetzt ein paar Minuten für aufgeschrieben. Wir gucken einfach mal.	Exactly, I've just written down a few minutes for that. Let's just see.
Participant 1	[02:23]	Ja, ich darf schon anfangen, ne?	Yes, I can start now, can't I?
Interviewer	[02:24]	Klick einfach mal rum, guck dir die Anwendung an. Versuch den Chat zu finden und was man sonst so tun kann.	Just click around, take a look at the application. Try to find the chat function and see what else you can do.
Participant 1	[02:31]	So, hast du hier normale YouTube-Videos? Ich nehme mal an, das ist hier irgendwo der Chat, zumindest bei Fragen. Kann ich da drauf drücken? Na, guck mal, dass ich mir noch nach, nach ein KI-Chat aus— ja, da ist er doch. So, den habe ich schon mal.	So, do you have normal YouTube videos here? I assume this is the chat somewhere, at least for questions. Can I press that? Well, let me see if I can find an AI Chat— yes, there it is. Right, I've got that sorted.
Interviewer	[02:46]	Sehr gut.	Very good.
Participant 1	[02:48]	Hast du verschiedene Versionen? Okay. Und das ist wahrscheinlich dann, neuen	Do you have different versions? Okay. And that's probably then, create new

		Chat erstellen, ne? Ja, okay. Aha, der ist löschen, minimieren. Funktioniert. So, das wahrscheinlich dann, die ganzen Sachen zu den verschiedenen, Modellen. So, klassisch, ja, gibt's ja ein paar von. Ah, guck mal, die hier Schrift, kann ich da drauf klicken? Ja. Oh. Aha, dann hast du noch mal hier die Verfahren, kannst hier wechseln, und dann hast du hier die Erklärung dafür. Texteingabe, kann ich hier eigentlich was verändern, oder kann ich wirklich, sogar wirklich Darf ich dir Fragen stellen, so, so rein theoretisch gesehen? Also ob ich irgendwas zerstören könnte, wenn ich irgendwas wegklicke?	chat, right? Yes, okay. Aha, that's delete, minimise. Works. So, that's probably then, all the stuff about the different models. So, classic, yes, there are a few of those. Ah, look, this font, can I click on it? Yes. Oh. Right, then you have the procedures here again, you can switch here, and then you have the explanation for it here. Text input, can I actually change anything here, or can I really, really... May I ask you some questions, just theoretically speaking? Like, could I destroy something if I click something away?
Interviewer	[04:06]	Ne, du kannst da alles, alles machen da, machst du nichts.	No, you can do anything there, anything at all, but you're not doing anything.
Participant 1	[04:10]	OK, aber. Auch wenn ich auf das auf— einfach wegmache?	OK, but. Even if I just remove that?
Interviewer	[04:14]	Ja, kannst du auch machen, ist kein Thema.	Yes, you can do that too, no problem.
Participant 1	[04:19]	Es geht auch wahrscheinlich so zurück, ne? Ja, okay. was passiert denn, wenn ich das wegmache? Der speichert das. Hallo? Aha, das funktioniert nicht. Hier komme ich komplett zurück. Wie mache ich den Spaß denn weg?	It probably goes back like this, right? Yes, okay. What happens if I remove this? It saves that. Hello? Oh, that doesn't work. Here I'm back to square one. How do I get rid of this thing?
Interviewer	[05:16]	Also, ich muss sagen, du hast jetzt schon zwei Funktionen entdeckt, die ich in meinem Leben noch nicht gesehen hab	Well, I must say, you've already discovered two functions that I haven't seen in my life yet.
Participant 1	[05:20]	Ich bin verwirrt, was das hier überhaupt sein soll. Also ich bin ja in der Hill-Chiffre drin,	I'm confused about what this is supposed to be. I'm in the Hill cipher, but I actually

B.2. Participant 2

		okay.	Solitaire, okay.
Interviewer	[04:59]	Ja, da gibt's einige, mhm.	Yes, there are a few, mhm.
Participant 2	[05:05]	Moderne Verfahren, das AES.	Modern methods, the AES.
Participant 2	[05:26]	Muss es wahrscheinlich alles eingeben	Probably have to enter everything
Participant 2	[05:44]	weird, dass es diese Symbole gibt. Hm, ja, Verschlüsselung,	It's weird that these symbols exist. Hm, yes, encryption,
Participant 2	[05:56]	Stegonographie, Hashfunktion.	Steganography, hash function.
Participant 2	[06:15]	Ist das besonders hier für Studienzwecken gedacht oder auch für die Anwendung später im Beruf?	Is this intended specifically for study purposes or also for later use in your career?
Interviewer	[06:22]	Ich denk, das ist hauptsächlich gedacht für, ja, Studierende oder die Leute, die sich das beibringen wollen, die ein bisschen mit Kryptografie arbeiten bestimmt auch. Kann bestimmt auch hilfreich sein. Ja, ich denke mal hauptsächlich für Studenten.	I think it's mainly intended for, yes, students or people who want to teach themselves, and certainly also for those who work with cryptography a bit. It can definitely be helpful. Yes, I think mainly for students.
Participant 2	[06:48]	So wie meine Vorstellung ist, ich habe ja keine Ahnung von diesem Programm, das kann ich vielleicht auch fragen. Ja, nur wahrscheinlich für dieses Programm gemacht worden, dieses, dieses dieser Chatbot, ne?	As I understand it, I have no idea about this program, so perhaps I could ask. Yes, but it was probably made for this program, this chatbot, right?
Participant 2	[07:08]	meine Frage wieder vergessen, egal. Zurück zum Start, wie komme ich dahin— Startcenter. Und ich guck das mit dem Wizzard.	I forgot my question again, never mind. Back to the start, how do I get there— Start Centre. And I'll take a look at the wizard.
Participant 2	[07:25]	Wählen Sie diesen Punkt, um einen Klartext zu verschlüsseln	Select this option to encrypt plain text or decrypt

		oder einen Geheimtext zu entschlüsseln.	ciphertext.
Participant 2	[07:40]	Ja, hm. Hier lern ich vielleicht was. Gestehen Sie hier— stellen Sie hier Fragen zur Anwendung. Das ist der Chat.	Yes, hm. I might learn something here. Ask questions about the application here. That's the chat.
Interviewer	[07:48]	Ja, das Fenster ist schon offen. Is halt der Chat, genau.	Yes, the window is already open. It's just the chat, exactly.
Participant 2	[07:53]	Ehh, ja, es hat ja auch das gleiche Symbol, deswegen	Ehh, yes, it has the same symbol, that's why.
Participant 2	[07:58]	Ehhm, sehen Sie sich auch in das Tutorial	Um, please also take a look at the tutorial.
Participant 2	[08:04]	wieder zu— damit man sich mehr informiz— informieren kann.	again— so that one can obtain more information.
Participant 2	[08:10]	Vorlagen. Okay, die ganzen Verfahren, die ich wahrscheinlich eben schon gesehen hab.	Templates. Okay, all the procedures I've probably already seen.
Participant 2	[08:25]	Mhm. Interessant — Passwort	Mhm. Interesting — Password
Participant 2	[09:10]	Go to live widget tree.	Go to live widget tree.
Interviewer	[09:13]	Das ist, nur wegen der Entwicklungsumgebung. Das sind keine Knöpfe aus der Anwendung, da musst du nicht drauf drücken.	That's just because of the development environment. They're not buttons from the application, so you don't need to press them.
Participant 2	[09:17]	Okay. Hehe.	Okay. Hehe.
Participant 2	[09:28]	Ich denke, ich habe alles gefunden. Ich drück mal hier drauf. Ah, ja, ja, so viel Text.	I think I've found everything. I'll press here. Ah, yes, yes, so much text.
Participant 2	[09:38]	Okay	Alright
Interviewer	[09:39]	Ja, das muss ich nicht durchlesen.	Yes, I don't need to read that.

Participant 2	[10:05]	Fragezeichen.	Question mark.
Participant 2	[10:19]	Noch habe ich nicht viele Gedanken, noch bin ich bisschen verwirrt.	I don't have many thoughts yet, and I'm still a bit confused.
Interviewer	[10:23]	Ja, ist gut, ich, gib dir ein irgendwie Tipps, wenn das nicht— wenn wir nicht weiterkommen, oder wenn ich sehe, dass du irgendwo stecken bleibst.	Yes, that's fine, I'll give you some tips if that doesn't work—if we get stuck, or if I see that you're stuck somewhere.
Participant 2	[10:35]	Ich habe jetzt so ein Ziel, dass ich mich bisschen durchprobiere, aber ich denke, wenn hier jetzt so schon so ein Chatbot ist, dann kann ich mir ja ein bisschen durchleiten lassen. Ja, ich tue so, als hätte ich absolut gar keine Ahnung.	I've set myself a goal to try things out a bit, but I think that if there's already a chatbot here, then I can let it guide me a little. Yes, I'm pretending that I have absolutely no idea.
Participant 2	[10:51]	Fertige Lernvorlagen, Templates und den Wizard zu lernen. Du musst nicht sofort selbst Workspaces bauen. Bereits mehrere Tabs offen, oder kann sogar sehen, was ich offen habe.	Ready-made learning templates, templates and the wizard to learn. You don't have to build workspaces yourself right away. Already have several tabs open, or can even see what I have open.
Participant 2	[11:03]	Mhm. Mhm.	Mhm. Mhm.
Participant 2	[11:05]	Du hast bei dir bereits mehrere Tabs. Empfohlener Start: Wizard führt dich hinter— durch Themen und erklärt dabei die Schritte. So, öffne Tab Wizard, wähle ein Einsteigerthema: Klassicher Chiffre.	You already have several tabs open. Recommended start: Wizard guides you through topics and explains the steps. So, open Tab Wizard and select a beginner's topic: Classic cipher.
Participant 2	[11:32]	Hier	Here
Interviewer	[11:34]	Das ist leider ein bisschen versteckt, unten rechts kommst	Unfortunately, it's a bit hidden, but you can continue

B.3. Participant 3

		Weil mir diese Oberfläche überhaupt nichts sagt.	understand this interface at all.
Interviewer	[04:04]	Alses klar	All right
Interviewer	[04:50]	Kannst auch ruhig in dem Programm bisschen rumklicken und dir versuchen, selber da irgendwie zu gucken, was man da machen kann. Du musst nicht unbedingt für jede Sache den Chat benutzen.	You can also click around in the program a bit and try to figure out for yourself what you can do there. You don't necessarily have to use the chat for everything.
Participant 3	[05:00]	Das ist mir schon klar.	I realise that.
Participant 3	[05:21]	Aha	I see.
Participant 3	[05:41]	Davon habe ich schon	I've already heard about that.
Interviewer	[05:48]	Caesar-Verschlüsselung	Caesar cipher
Participant 3	[06:16]	Wow, das gibt voll Sinn. Mhm. Jetzt kann man gar nicht mehr auf weiter klicken	Wow, that makes perfect sense. Hmm. Now you can't even click on 'Continue' anymore.
Interviewer	[06:30]	Wenn du willst, kannst du oben rechts auf den WM+ drücken— Button drücken. Da kommst du quasi in die Haupt— in die Hauptanwendung, also in das Hauptelement von CrypTool 2.	If you want, you can press the WM+ button in the top right corner. This will take you to the main application, i.e. the main element of CrypTool 2.
Participant 3	[06:51]	Ich seh oben rechts leider nicht das, was du mir gesagt hast.	Unfortunately, I don't see what you told me in the top right corner.
Interviewer	[06:54]	Von deiner Maus genau rechts. Kurz vorm KI-Chat. Dieser komische Button.	Right next to your mouse. Just before the AI Chat. That strange button.
Participant 3	[07:00]	Aha	I see.
Participant 3	[07:06]	Aha	I see.
Participant 3	[07:54]	Hat sich jetzt was verändert?	Has anything changed now?

		Tatsache.	Actually, yes.
Participant 3	[08:10]	Ich versuche gerade zu verstehen, wozu diese komische Linie gestrichelt wird, wenn ich draufgehe. Scheint sich mir aber nix zu verändern.	I'm trying to understand why this strange line appears when I click on it. But nothing seems to change.
Participant 3	[08:26]	Dann kann ich, kann ich irgendwo draufgehen?	Then can I, can I go somewhere?
Participant 3	[08:38]	Ich versuche irgendein anderes Verfahren da reinzuschieben.	I'm trying to insert some other procedure in there.
Interviewer	[08:42]	Achso, ja, den Tipp gebe ich immer noch, den habe ich den anderen auch gegeben. Du kannst an dieser Oberfläche nichts verändern, solange der Algorithmus da ausgeführt wird. Also wenn du oben auf Stopp drückst, kannst du erst wieder was verändern	Oh yes, I still give that tip, I gave it to the others too. You can't change anything on this interface as long as the algorithm is running. So if you press Stop at the top, you can only change things again once it has finished.
Participant 3	[08:54]	Aha	I see.
Participant 3	[09:08]	Ein Fenster ist da.	There is a window.
Participant 3	[09:27]	Den hab ich jetzt da reingeschoben. Dat bringt mir irgendwie— gänzlich wenig	I've now put it in there. That brings me somehow— absolutely nothing.
Participant 3	[09:45]	Ah, groß machen.	Ah, make it big.
Participant 3	[11:29]	Sehr ungewohnt, dass der Text rechtsbündig ist.	It is very unusual for the text to be right-aligned.
Interviewer	[11:35]	Was nochmal?	What was that again?
Participant 3	[11:38]	Sehr ungewohnt, dass der Text in dem Bot hier rechtsbündig angezeigt wird	It is very unusual that the text in the bot is displayed right-aligned here.
Interviewer	[11:45]	Ah, okay, ja. Das ist ein bisschen kurios, in der Tat. Wie wär das normal?	Ah, okay, yes. That is a bit strange, actually. How would that be normal?

Participant 3	[11:58]	Ich weiß nicht, wie es normal ist, aber das sieht auf jeden Fall komisch aus.	I don't know what's normal, but that certainly looks strange.
Participant 3	[12:21]	Kann es sein, dass der KI-Chat weiß, wo ich gerade bin?	Could it be that the AI Chat knows where I am right now?
Participant 3	[12:25]	Ah, das ist ja voll praktisch.	Ah, that's really handy.
Participant 3	[12:55]	Hm. Füge ein Komponente zu, die blablabla macht.	Hm. Add a component that does blablabla.
Participant 3	[13:30]	Mm-hmm.	Mm-hmm.
Participant 3	[13:50]	Also meine Aufgabe war jetzt herauszufinden, was ein typischer Workflow in diesem Programm wäre	So my task now was to find out what a typical workflow in this program would be.
Interviewer	[14:05]	Genau. Muss das nicht unbedingt, durchführen. Du kannst mir das auch einfach erklären, das würd mir auch schon reichen	Exactly. You don't necessarily have to do it. You can just explain it to me, that would be enough for me.
Participant 3	[14:12]	Ja, wenn ich wüsste, was ich	Yes, if I knew what I
Interviewer	[14:23]	Kannst auch noch ein bisschen ausprobieren.	You can also try it out a little.
Participant 3	[14:24]	Ich geh davon aus, dieses Programm ist dafür da, mir irgendwas zu erklären	I assume this program is there to explain something to me.
Participant 3	[14:32]	Und zwar diese— Kryptografieverfahren. Ja, und dann kann man hier irgendwie damit rumspielen und irgendwas versuchen zu modellieren, wie diese Verfahren funktionieren. Und dazu gibt's dann auch Erklärungen. Und verschiedener Tools, die ich mich ungeeignet fühle auszuführen.	Namely, cryptographic methods. Yes, and then you can play around with them and try to model how these methods work. There are also explanations for this. And various tools that I do not feel qualified to use.

C. Chat Histories

The following excerpts present the first two user questions and their corresponding AI responses from each chat history, recorded during the task phase of the evaluation. The full version is included with the digital edition. The chat histories were converted from Markdown into PDF format using custom Python scripts and translated into English with AI-supported tools. Due to this conversion and translation process, the visual appearance and wording may differ slightly from the original chats shown in the application.

C.1. Participant 1

Entry 1

Time: 9:13

Question: Name three different encryption methods

Tools: get_open_tabs, get_workspace_description_text

Answer:

Three Different Encryption Methods

1. **Caesar Cipher (classical, symmetric)**
    - Shifts each letter in the alphabet by a fixed number of positions (e.g., +3).
    - Simple, but easy to crack today.
2. **AES (Advanced Encryption Standard, symmetric)**
    - Modern standard for "same key for encryption and decryption".
    - Very widely used (e.g., file/hard drive encryption, TLS internally).
3. **RSA (asymmetric)**
    - Uses a **key pair**: public key for encryption, private key for decryption.
    - Often used for key exchange and signatures, less so for large datasets directly.

If you like, I can also briefly tell you which ones in CrypTool 2 are good to try out (e.g., Caesar cipher analysis, AES/RSA demos).

Entry 2

Time: 9:50

Question: Where can I find RSA (asymmetric) in the application?

Tools: get_open_tabs, get_workspace_of_active_tab, get_available_workspace_components

Answer:

Finding RSA in CrypTool 2

You can find RSA in the **Workspace Editor** (CWM) on the left side of the **Component Browser**:

Option A: Via the category (the easiest way)

1. Open/switch to a **Workspace** (e.g., via *File* → *New* → *Workspace* or your "Caesar Brute-Force Analysis" tab).
2. On the left side of the component browser, go to:
   - **Modern Ciphers → Asymmetric** (or *Modern Asymmetric*)
3. There you will find:
   - **RSA** ("RSA Encryption/Decryption")
   - **RSA Key Generator**

Option B: Using the Search

1. Click in the **search field** on the left in the component browser
2. Enter `RSA`
3. Then drag **RSA** into the workspace via **double-click** or **drag & drop** into the workspace.

Mini-Workflow (for Trying Out)

- Select **RSA Key Generator** → generates public/private key
- Connect the output to **RSA** → there you can encrypt/decrypt

If you tell me whether you want to test **text** or **files** with RSA, I can outline the appropriate workspace setup in 3–4 steps.

C.2. Participant 2

Entry 1

Time: 03:25

Question: what can you do

Tools: get_open_tabs, get_workspace_description_text

Answer:

What I can do for you in CrypTool 2

I am the **Chat Assistant in CrypTool 2**. I will help you **understand cryptography** and **build/debug workspaces** in the workspace editor.

Currently, only the **"Start Center"** tab is open.

---

1) Help for your current workspace (if one is open)

If you have a workspace open, I can, for example:

- **Identify components** that you have placed (and what they are for)
- **Check wiring** (missing connections, incorrect data types)
- **Explain settings** (what do parameters mean, which ones are useful?)
- **Support troubleshooting** (why is it not working / why is there no output?)
- **Give step-by-step instructions** on how to set something up in CT2

---

2) Explain cryptography (practical)

I can explain concepts to you in an understandable way, e.g:

- Classical methods: **Caesar, Vigenère, Enigma**, frequency analysis
- Modern cryptography: **AES, RSA, ECC**, hashes (**SHA-2**), **HMAC**
- Protocols & topics: **Signatures, key exchange, KDFs**, randomness, padding, modes (CBC/CTR/GCM)

And show how to recreate this in CT2.

---

3) Find components & suggest suitable building blocks

- Which **component** you need for a purpose (e.g. “Hash”, “Base64”, “RSA Encrypt”)
- How to correctly **link** components (input/output types)

---

4) Recommend templates/workflows

I can give you suitable **CT2 templates** (if you tell me what you have in mind), e.g. for:
- Encryption/decryption - Cryptanalysis demos - Encoding/decoding pipelines

---

What would you like to do?

Tell me briefly your goal, e.g:

- “I want to encrypt **AES**”
- “I want to understand **RSA**”
- “I have a workspace that is **not running**”
- “Which components do I need for **Base64 → Hash → HMAC**?”

Then I will guide you directly through the steps in CrypTool 2.

Entry 2

Time: 10:06

Question: i have never had anything to do with cryptography, what is the best way to get started with the tool to learn something?

Tools: get_open_tabs, get_startcenter_description_text, get_available_templates

Answer:

How to get started with CrypTool 2 (without prior knowledge)

The best way to learn CrypTool 2 is via **ready-made learning templates** and the **Wizard**. You don't have to build your own workspaces straight away.

You already have several tabs open - including **"Wizard"**, **"Workspace"** and currently active **"The World of Prime Numbers"** (this is already a learning module).

---

1) Recommended start: Wizard

The **Wizard** guides you interactively through topics and explains the steps.

This is how you proceed: 1. Open the **"Wizard"** tab 2. Choose a beginner theme (classic ciphers are great) 3. Work through the steps and try out values

Why this is good: You get **explanation + direct application**.

---

2) Second step: open Templates in the Start Center

In the **Startcenter** there is a **Templates** area (tree structure). You can open examples there by double-clicking.

C.3. Participant 3

Entry 1

Time: 02:35

Question: What is cryptography?

Tools: get_open_tabs, get_workspace_description_text

Answer:

Cryptography - what is it?

Cryptography is the science and technology of **protecting** or **making verifiable** information in such a way that it can be used securely despite attackers. This typically involves working with **algorithms** and **keys**.

Main objectives of cryptography

- **Confidentiality:** Only authorized persons can read a message (e.g. through *encryption*).
- **Integrity:** Manipulations become recognizable (e.g. through *hash functions* or *MACs*).
- **Authenticity:** It is possible to check who originated something (e.g. *digital signatures*).
- **Non-repudiation:** The sender cannot credibly deny the sending later (also *signatures*).

Important areas (with typical examples)

- **Symmetric cryptography:** same key for encryption and decryption *Examples:* AES, ChaCha20
- **Asymmetric cryptography:** Key pair consisting of *public key* and *private key* *Examples:* RSA, Elliptic Curves (ECC)
- **Hash functions:** "Fingerprint" of data *Examples:* SHA-256
- **Protocols:** Rules on how several procedures interact *Examples:* TLS (HTTPS), signal protocol

Mini example (intuitive)

You want to send “Hello”: - With **encryption** it becomes apparent “random text”, which only someone with the **key** can turn back into “hello”. - With **Signature** the recipient can check: *This really came from you and has not been changed*

Would you like to try this out in CrypTool 2?

If you like, tell me if you want to see **encryption (e.g. AES)**, **hashing (SHA-256)** or **signatures (RSA/ECC)** first-then I’ll briefly describe how to set up a workspace for it.

Entry 2

Time: 03:52

Question: what is cryptool 2 good for?

Tools: get_open_tabs, get_startcenter_description_text

Answer:

What is CrypTool 2 good for?

CrypTool 2 (CT2) is a **learning and experimental environment for cryptography and cryptanalysis**. You can use it to **understand procedures**, **assemble them yourself** and **test them** without having to program everything.

Typical areas of application

- **Learn & demonstrate cryptography**
  - Classic methods (e.g. Caesar, Vigenère, Enigma)
  - Modern methods (e.g. AES, RSA, ECC)
  - Hashes, signatures, protocol components
- **Understanding cryptanalysis**
  - Understanding attacks/methods (e.g. frequency analysis, brute force, key searcher)
  - Change parameters and see when attacks work
- **Experiment with data flows**
  - Connect components in the **workspace** using drag & drop (like "visual programming")
  - Observe inputs, intermediate steps and outputs directly (data/log view)

What makes CT2 special

- **Graphical workspace editor:** You build algorithms as **component chains** (Input → Cipher → Output).
- **Many ready-made templates:** In the **Startcenter** you will find examples by category (cryptography, cryptanalysis, hash functions, codes, protocols, steganography …).
- **Didactics instead of productive security:** CT2 is primarily intended for **understanding, teaching and trying out** (not as a "security tool" for everyday use).

Use of Artificial Intelligence

AI Tool	Provider	Version	Usage
ChatGPT	OpenAI	GPT-5.2 / GPT-5.3	Support for literature exploration, summarization of sources, translation, formulation, text revision, grammar correction, and LaTeX formatting.
GitHub Copilot	OpenAI	GPT-5.2 / GPT-5.2-Codex / GPT-5.3-Codex	Support for programming tasks, debugging, and code commenting in Visual Studio 2022 / 2026.
GitHub Copilot	Gemini	Gemini 2.5 Pro / Gemini 3.1 Pro	Code analysis and explanation in Visual Studio 2022 / 2026.
Claude	Claude	Sonnet 4.5 / Sonnet 4.6	Summarization of sources
DeepL	DeepL SE	Current version	Translation of short text passages.
Custom Python Translation Script	DeepL SE	Free API v2	Translation of interview transcripts and AI Chat threads used during the interviews into English.
VideoTranscriber.ai	VideoTranscriber.ai	Current online version	Automatic transcription of interview recordings.

Tab. 6: Overview of AI tools used during the preparation of this thesis.

Note: As new models became available during the development period, they were gradually adopted and replaced older versions in subsequent work phases.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die schriftliche Ausarbeitung selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach (inkl. Übersetzungen) anderen Werken entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der Quelle (einschließlich des World Wide Web sowie generativer KI und anderer elektronischer Datensammlungen) deutlich als Entlehnung kenntlich gemacht. Dies gilt auch für angefügte Zeichnungen, bildliche Darstellungen, Skizzen und dergleichen. Ich nehme zur Kenntnis, dass die nachgewiesene Unterlassung der Herkunftsangabe als versuchte Täuschung gewertet wird. Die Paragraphen der für mich geltenden Prüfungsordnungen, die etwaige Betrugsversuche betreffen, habe ich zur Kenntnis genommen.

Der Speicherung meiner Abschlussarbeit zum Zweck der Plagiatsprüfung stimme ich zu. Ich versichere, dass die elektronische Version mit der gedruckten Version inhaltlich übereinstimmt.

_____ _____

(Date) (Signature)

Content of the USB stick

- Master thesis as PDF
- Complete source code of the application
- Complete interview transcripts
- Complete chat histories from the interviews