România
Ministerul Apărării Naționale
Academia Tehnică Militară "*Ferdinand I*"

Facultatea de Sisteme Informatice și Securitate Cibernetică
Calculatoare și Sisteme Informatice pentru Securitate si Apărare Națională

Unelte software bazate pe mecanisme de inteligență artificială aplicată în criptografie

Coordonator Științific
Cpt. Subașu Georgiana-Ramona

Absolvent
Std. Sg. Maj. Moldovan Andrei-Gabriel

București
2025

Abstract

Lucrarea propune o platformă integrată care combină diferiți agenți AI specializați ca utilitare de criptanaliză și audit de parole pentru a acoperi un flux complet de securitate modernă din punct de vedere criptografic. Sistemul folosește orchestrare prin microservicii containerizate, momentan, și include: un agent de decizie (SecureBERT) pentru extragere de intenție și de entități relevante; un detector de criptosisteme inspirat din CyberChef/dcode.fr; un orchestrator de spargere de hash-uri cu HashCat și John și generare de parole folosind PassGAN; un agregator de scoruri de parole (rețea neuronală, zxcvbn și HaveIBeenPwned), încă în lucru; un serviciu de testare primalitate/factorizare (YAFU + FactorDB); și un RAG local pentru asistență teoretică în criptografie. Se urmărește realizarea unei platforme centralizate care poate fi folosită local sau în cloud, cu capabilități AI/ML pentru detecție, analiză și suport educațional, punând accent pe modularitate, extindere ulterioară către Kubernetes și aliniere la bune practici de securitate.

Cuvinte cheie: inteligență artificială, criptanaliză, securitate cibernetică, parole, HashCat, John the Ripper, PassGAN, SecureBERT, RAG local, containerizare, microservicii, Kubernetes, FactorDB, YAFU, zxcvbn, HaveIBeenPwned.

Cuprins

Listă de Abrevieri

AEAD ..... Authenticated Encryption with Associated Data

AES ..... Advanced Encryption Standard

AI ..... Artificial Intelligence

API ..... Application Programming Interface

CI/CD ..... Continuous Integration / Continuous Deployment

CLI ..... Command Line Interface

CTF ..... Capture The Flag

DES/3DES .. Data Encryption Standard / Triple DES

ECC ..... Elliptic Curve Cryptography

ECDH ..... Elliptic Curve Diffie–Hellman

ETSI ..... European Telecommunications Standards Institute

FIPS ..... Federal Information Processing Standards

HIBP ..... Have I Been Pwned

LLM ..... Large Language Model

MAC ..... Message Authentication Code

ML ..... Machine Learning

NIST ..... National Institute of Standards and Technology

ONNX ..... Open Neural Network Exchange

PKI ..... Public Key Infrastructure

PQC ..... Post-Quantum Cryptography

RAG ..... Retrieval-Augmented Generation

RFC ..... Request for Comments

RSA ..... Rivest–Shamir–Adleman

TLS ..... Transport Layer Security

WAF ..... Web Application Firewall

Capitolul 1: Introducere

1.1 Context general

Criptografia modernă a evoluat cu pași repezi, de la algoritmi simetrici clasici (DES, 3DES), către standarde robuste precum AES și suitele asimetrice (RSA, ECC), odată cu dezvoltarea infrastructurilor PKI și a protocoalelor TLS, precum și apariției PQC. În paralel, progresul AI a adus atât beneficii considerabile, cât și riscuri: modelele de tip ML/LLM pot accelera detecția anomaliilor și analiza traficului criptat, putând fi folosite și pentru generarea de atacuri (phishing, parole, malware) sau pentru asistarea criptanalizei. Astfel, securitatea cibernetică integrează din ce în ce mai mult tehnici AI precum răspuns rapid, corelare de evenimente și întărirea mecanismelor criptografice.

1.2 Problema abordată

Problema principală urmărește modul în care putem folosi agenți AI specializați pentru a automatiza analiza și răspunsul la amenințări criptografice, de la identificarea algoritmilor și auditul parolelor, până la factorizare pentru testare de securitate, cât și modul în care se poate îmbunătăți educația criptografică. Se observă, de asemenea, lipsa unei platforme integrate care să ofere aceste capabilități într-un mod modular, scalabil și sigur.

1.3 Scopul lucrării

Scopul lucrării este reprezentat de proiectarea unei platforme integrate, modulare și scalabile care folosește agenți AI pentru identificare de algoritmi, audit de parole, factorizare / teste criptografie, asistență teoretică, astfel încât detecția, analiza și răspunsul la amenințări să fie automatizate, reproductibile și ușor de folosit, atât local, cât și în cloud.

1.4 Obiective specifice

- Realizarea și testarea robustă a unei părți din agenții AI propuși.
- Integrarea tuturor agenților într-o infrastructură comună.
- Crearea unei soluții de deploy și testare.
- Crearea unei arhitecturi de rețea care să asigure comunicația securizată.

Capitolul 2: Fundamente Teoretice

2.1 Elemente de bază ale criptografiei

Criptografia urmărește să asigure confidențialitate, integritate, autentificare și nerepudiere pentru date transmise pe canale nesigure. Se împart două familii majore: criptografia simetrică (aceeași cheie pentru criptare/decriptare) și criptografia asimetrică (pereche publică/privată), completate de funcții hash și coduri de autentificare a mesajelor (MAC/HMAC) [11, 9, 10]. Un criptosistem se definește formal prin spațiul mesajelor, al textelor cifrate, al cheilor și printr-o familie de funcții de criptare/decriptare parametrizate de cheie, iar securitatea se descrie prin modele de adversar și noțiuni riguroase de rezistență la atac. În practică, protocoalele moderne combină aceste primitive cu management de chei (generare, stocare, rotație) și cu mecanisme de transport securizat (ex. TLS), fiind fundamentele pe care se sprijină soluția propusă.

2.1.1 Definiții și clasificare

Din punct de vedere al clasificării, criptografia acoperă o suită vastă de primitive: criptare simetrică, criptare asimetrică, diverse criptosisteme (bloc, flux, hibrid), mecanisme de autentificare, funcții hash, semnături digitale, management al cheilor, generatoare pseudo-aleatoare și protocoale de distribuire a secretelor. Criptografia simetrică utilizează aceeași cheie pentru criptare și decriptare, oferind performanțe foarte bune și fiind preferată pentru volume mari de date, dar presupune existența unui canal sigur pentru distribuirea cheilor (exemplu: AES). Criptografia asimetrică (cu cheie publică) folosește o pereche publică/privată pentru a facilita schimbul securizat de chei, autentificarea și semnăturile digitale (RSA, ElGamal), însă costul computațional este mai ridicat.

Funcțiile hash criptografice produc un digest de lungime fixă dintr-un mesaj de lungime variabilă, fiind concepute să reziste la inversare și coliziuni, motiv pentru care sunt folosite la verificarea integrității și la stocarea sigură a parolelor. Alte primitive importante includ codurile de autentificare a mesajului (MAC/HMAC), care combină o cheie secretă cu o funcție hash sau cu o schemă de criptare pentru a valida autenticitatea și integritatea; scheme de semnătură digitală ce oferă non-repudiere fără partajarea cheii; generatoare de numere aleatoare criptografic sigure; și protocoale de negociere a cheilor, precum Diffie–Hellman sau ECDH. În practica modernă, sistemele combină aceste primitive: asimetria este folosită pentru a stabili secrete, simetria pentru traficul intens, hash-ul și MAC-urile pentru integritate și autentificare, iar managementul cheilor (generare, rotație, revocare) asigură durabilitatea modelului de securitate [7].

2.1.2 Standardizare

Organismele de standardizare în criptografie sunt esențiale pentru a asigura interoperabilitatea între sisteme, validarea algoritmilor criptografici și adaptarea tehnologiei la amenințările emergente, inclusiv cele cuantice. Exemple cheie includ NIST (FIPS 140 pentru module criptografice, FIPS 197/AES și seria FIPS 203–205 pentru algoritmi post-cuanți), ISO/IEC (ISO/IEC 19790 privind cerințele de securitate și ISO/IEC 18033 pentru algoritmi de criptare), ETSI (TS 104 015 dedicat criptografiei cuantice), ITU (familia ITU-T Y.3800 pentru distribuția cuantică de chei), BSI (TR-02102 cu recomandări de algoritmi) și IETF (RFC 5280 pentru PKI X.509). Ele stabilesc cerințele de certificare, mecanismele de testare și ghidurile de implementare care ghidează atât soluțiile civile, cât și cele militare [1].

2.2 Elemente fundamentale de inteligență artificială

Se folosesc mai multe concepte de IA aplicate:

- învățare supervizată și clasificare semantică;
- embeddings obținute cu arhitecturi Transformer;
- RAG, care combină recuperarea de fragmente, reranking;
- reranking al rezultatelor de căutare pe baza scorurilor ML;
- data augmentation în generatorul de întrebări (questions_generator) pentru a extinde seturile de antrenare cu variații și parafrazări sintetice;
- ensemble și agregare de scoruri (password checker, rețea neuronală, zxcvbn și HIBP);
- inferență optimizată prin ONNX pentru performanță și portabilitate;

2.3 Conexiunea AI–Criptografie

AI-ul este din ce în ce mai implicat în criptografie, atât în analiză, cât și ca obiect al securizării. Modele AI specializate pot automatiza identificarea algoritmilor și clasificarea cererilor tehnice, asistând în alegerea metodelor criptanalitice adecvate, conform tiparelor datelor [4, 3]. Integrarea tehnicilor de RAG permite asistenților AI să furnizeze recomandări fundamentate pe literatura de specialitate și standarde, susținând tranziția către algoritmi post-cuantici și reducând riscul de halucinații [2]. În evaluarea parolelor, se folosesc modele neuronale antrenate pe parole comune combinate cu estimatori euristici (zxcvbn) și baze de date publice precum HIBP, ceea ce îmbunătățește precizia scorării și a analizei de risc [8, 12]. Generarea de scenarii și date sintetice prin modele generative facilitează testarea rezilienței sistemelor, iar sistemele de detecție bazate pe învățare profundă pot identifica comportamente suspecte în trafic sau log-uri [13]. Criptografia contribuie la protecția AI prin tehnici precum criptarea omomorfă, calculul multipartit securizat sau privacy diferențială, folosite pentru protejarea datelor și a modelelor în timpul antrenării sau inferenței [5]. Pentru integritatea modelelor, se propun watermark-uri digitale și semnături criptografice atașate acestora. Totuși, AI introduce riscuri semnificative, precum prompt injection sau exfiltrarea secretelor stocate, motiv pentru care sunt necesare guardrails, monitorizare și evaluare continuă a comportamentului modelului [6].

Capitolul 3: Soluția Propusă

3.1 Arhitectura generală

Arhitectura generală a aplicației este formată dintr-un set de microservicii containerizate, fiecare reprezentând o componentă cheie în funcționarea soluției. Pentru a face posibilă existența unei soluții scalabile, ușor de întreținut și modulare, este nevoie de o orchestrare eficientă a acestor microservicii, prin componente auxiliare. Astfel că am identificat următoarele componente principale:

- Frontend: care poate să fie un UI web, sau o interfață din linie de comandă.
- Backend: care gestionează logica de business, comunicarea între componente și expune API-urile necesare.
- Protection Layer: asigură securitatea datelor în tranzit, soluția exactă de securitate fiind aleasă ulterior.
- Baza de date: pentru stocarea persistentă a datelor relevante.
- Orchestrator: care asigură un flow de lucru bine înrădăcinat semantic, pentru evitarea și minimizarea erorilor.
- Decision Pipeline: un pipeline care să asigure extragerea intenției și a entităților relevante din cererile utilizatorilor.
- AI Agents Pool: un set de agenți AI specializați, fiecare cu rolul său specific în cadrul soluției.

Fiecare dintre agenți la rândul lui poate fi văzut ca un microserviciu containerizat, care poate fi dezvoltat, testat și implementat independent. Aceștia pot comunica între ei prin intermediul unor API-uri bine definite, permițând astfel o flexibilitate și o scalabilitate sporită a întregii soluții. În viziunea mea prezentă, partea de security ar trebui să fie implementată la nivelul microserviciului, și partea de rate-limit și monitorizare a traficului între componente, la nivel centralizat.

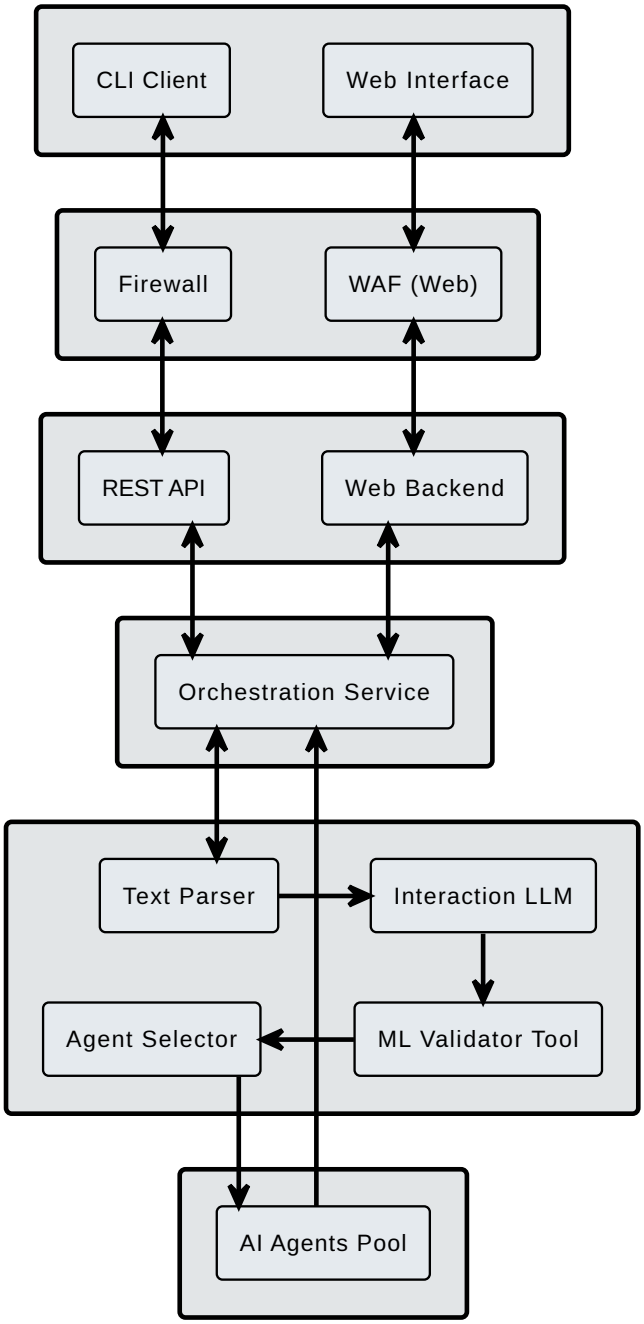

Figure 3.1: Arhitectura generală a solutiei propuse.

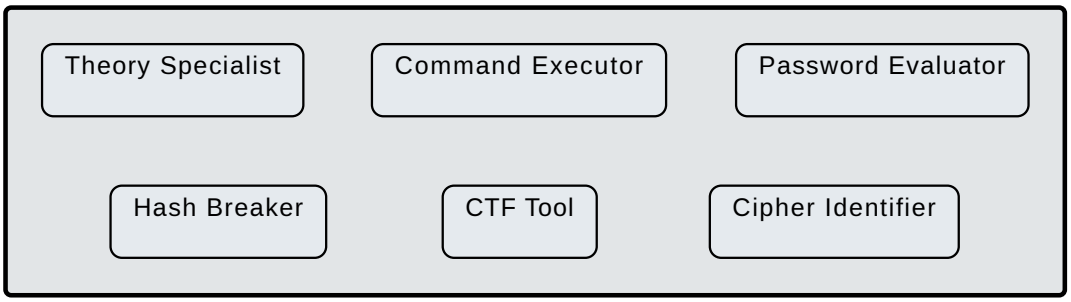

Figure 3.2: Pool-ul de agenți AI specializați.

3.1.1 Selecția tehnologiilor

Alegerea stack-ului tehnologic a fost ghidată de criterii precum **performanță**, **maturitate**, **ușurința integrării** și **compatibilitatea cu containerizarea**. Tabelul 3.1 prezintă principalele tehnologii utilizate.

Table 3.1: Principalele tehnologii utilizate în proiect

Tehnologie	**Utilizare**	**Justificare**
FastAPI + Uvicorn	Password Checker, Theory Specialist	Framework **async** rapid, tipizare Pydantic, răspuns JSON facil
Flask	Prime Checker, Choice Maker	Microframework **simplu** pentru endpointuri sync, ușor de containerizat
TensorFlow/Keras	Password Checker (PassStrengthAI)	Model de **clasificare** parole; Keras simplifică inferența
PyTorch + Transformers	Choice Maker (Intent/NER)	Inferență **NLP** cu modele preantrenate, ecosistem matur
ChromaDB + ONNX	Theory Specialist (RAG)	**Vector DB** + embeddings pentru căutare semantică eficientă
LangChain	Theory Specialist	**Orchestrare RAG**: ingestie, recuperare, chain-uri
Docker + Compose	Toate serviciile	**Izolare**, reproducibilitate și orchestrare locală
React 19 + Vite	Interfața web	Stack modern pentru UI **rapid**, build/dev performant
Nginx	Servire SPA	Servire **statică** performantă în container
YAFU + GMP	Prime Checker	Unelte de **factorizare** numerică performante

3.1.2 Principii de rețelistică

Arhitectura de rețea a proiectului se bazează pe principii de **microservicii containerizate** care comunică prin **REST API** peste HTTP. Fiecare agent este izolat într-un container Docker propriu, iar orchestrarea se realizează prin **Docker Compose**, care creează automat o rețea bridge cu **DNS intern** pentru rezolvarea numelor serviciilor. Acest model permite scalare independentă, înlocuire ușoară a componentelor și debugging izolat.

Comunicația între servicii folosește **REST** (Representational State Transfer) – un stil arhitectural bazat pe resurse identificate prin URI-uri și operații standard HTTP (GET, POST). Alegerea REST se justifică prin: interoperabilitate universală, tooling bogat (OpenAPI, Swagger), ușurința testării cu `curl`/Postman și compatibilitate nativă cu frontend-ul React.

Pentru apeluri HTTP interne și externe se utilizează clienți robuști precum **httpx** (Python, async/sync) și **requests** cu retry/backoff, asigurând **toleranță la erori** și timeouts configurabile. Serviciile externe integrate includ **OpenAI API** (generare întrebări), **HaveIBeenPwned** (verificare breșe parole) și **FactorDB** (primalitate). Tabelul 3.2 detaliază elementele de rețea și rolul lor.

Table 3.2: Elemente de rețelistică și comunicație

Element	Rol / Utilizare	Justificare
REST API	Toate serviciile (FastAPI, Flask, Express)	Interoperabilitate simplă, tooling bogat, ușor de testat
Docker Network (bridge)	Rețea virtuală cu DNS per serviciu	DNS automat între containere, trafic izolat de host
Port Mapping	Ex: 9000→8000, 8081→8080, 8100→8000	Acces local de pe host și din browser
CORS Middleware	FastAPI, Flask, Express	Permite apeluri cross-origin din React
httpx (async/sync)	Password Checker, Choice Maker	Timeout/retry robust pentru fan-out intern
Health Endpoints	/health pe toate serviciile	Liveness checks, debugging rapid
OpenAI / HIBP / FactorDB	API-uri externe consumate	Refolosire surse specializate externe
Nginx (reverse proxy)	Servire SPA React	Cache static, fallback index.html pentru routing

Exemple de apeluri REST:

Tabelul 3.3 prezintă două exemple concrete de apeluri REST către microserviciile proiectului.

Table 3.3: Exemple de apeluri REST API

Operație	Serviciu	Descriere
Ingestie document RAG	Theory Specialist (:8100)	Încărcare document PDF în baza vectorială
Intent + Entity Extraction	Choice Maker (:8081)	Extragere intenție și entități din text

1. **Ingestie document RAG** (Theory Specialist, port 8100):

```
curl -X POST http://localhost:8100/ingest \
  -H "Content-Type: application/json" \
  -d '{"document_path": "/app/documents/crypto.pdf",
       "document_type": "pdf"}'
```

2. **Intent + Entity Extraction** (Choice Maker, port 8081):

```
curl -X POST http://localhost:8081/predict \
  -H "Content-Type: application/json" \
  -d '{"operation": "entity_extraction",
       "text": "Encrypt payroll with AES-256-GCM"}'
```

3.1.3 Principii de securitate

După terminarea implementării funcționalităților componentelor, voi **ranforsa securitatea** acestora prin mai multe măsuri stratificate:

Autentificare și autorizare: adaug autentificare (**API key/JWT**) și restricționez **CORS** doar la origin-urile front-end legitime; implementez **rate limiting/throttling** și logare cu request-id fără date sensibile.

Transport securizat: termin **TLS** printr-un reverse-proxy (Nginx/Traefik) și păstrez traficul intern pe rețeaua **Compose izolată**, fără expunere directă a porturilor interne.

Validare input: limitez `/ingest` la surse și tipuri de fișiere aprobate, cu verificare **mărime/MIME** și scan antivirus, pe volume cu permisiuni minime; forțez **body-size mic** și `Cache-Control: no-store` pe rutele cu parole.

Protecție apeluri externe: protejez apelurile externe (HIBP, OpenAI, Ollama, FactorDB) cu **timeouts stricte**, **circuit breaker** și stocare chei doar în secret manager/env la runtime.

Hardening containere: rulez containerele cu **user non-root**, resurse limitate și volume **read-only** unde se poate; elimin `host-gateway` dacă nu e necesar.

Observabilitate: reduc log-level la WARN/INFO în producție, expun doar `/health` necesare și adaug **audit/metrics** pentru monitorizare continuă.

3.2 Descrierea componentelor

3.2.1 Sistem RAG pentru asistență teoretică

Această componentă oferă suport teoretic în criptografie printr-un sistem RAG. Utilizează o bază de date vectorială (ChromaDB) pentru stocarea embeddings-urilor documentelor relevante, generate cu un model Transformer. La primirea unei întrebări, sistemul recuperează fragmente relevante, le rerankează și le integrează într-un răspuns generat de un model LLM, asigurând astfel acuratețea și relevanța informațiilor furnizate. Permite functionalități de ingestie a documentelor noi, actualizarea bazei de cunoștințe și optimizarea continuă a procesului de recuperare și generare a răspunsurilor.

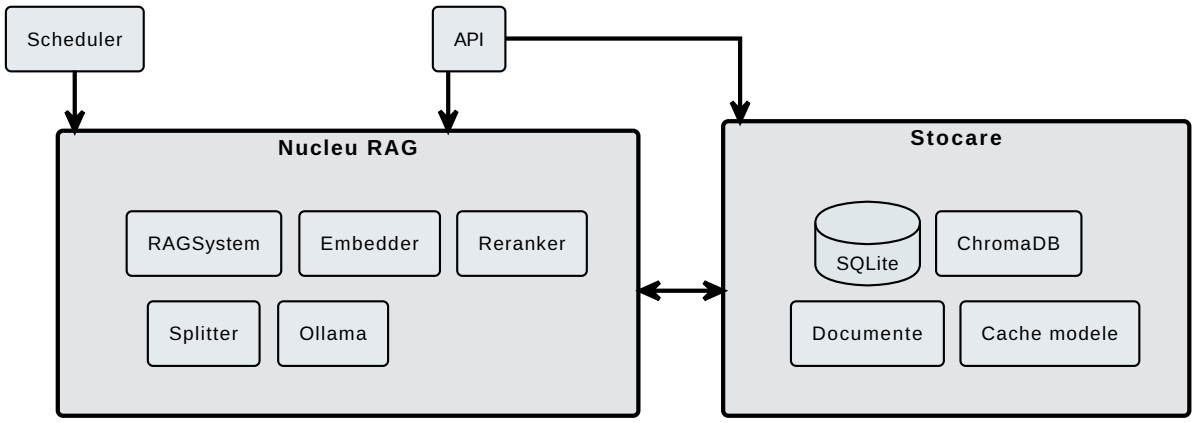

Figure 3.3: Diagrama arhitectură Sistem RAG pentru asistență teoretică

3.2.2 Modul verificare parole

Această componentă evaluează puterea parolelor folosind un model de rețea neuronală antrenat pe un set divers de parole, combinat cu algoritmul zxcvbn și verificarea în baza de date HaveIBeenPwned. Această abordare hibridă oferă o evaluare precisă a securității parolelor, identificând atât complexitatea lor, cât și dacă au fost compromise în breșe cunoscute.

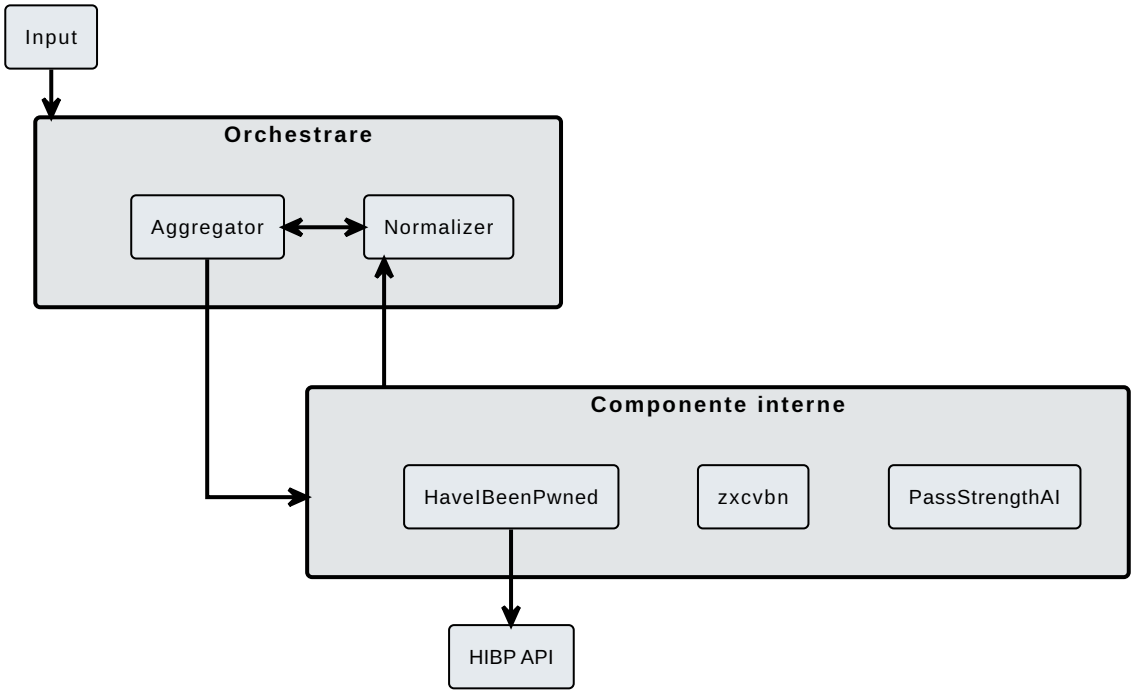

Figure 3.4: Diagrama arhitectură modul verificare parole

O implementare viitoare va include un sistem de feedback care să permită utilizatorilor să înțeleagă mai bine riscurile asociate parolelor slabe și să primească recomandări personalizate pentru îmbunătățirea securității acestora.

3.2.3 Modul detectare intenție și entități

Acest modul utilizează un model NLP bazat pe arhitectura Transformer pentru a extrage intenția și entitățile relevante din cererile utilizatorilor. Acesta ajută la direcționarea corectă a cererilor către agenții specializați, asigurând o interpretare precisă a nevoilor utilizatorului.

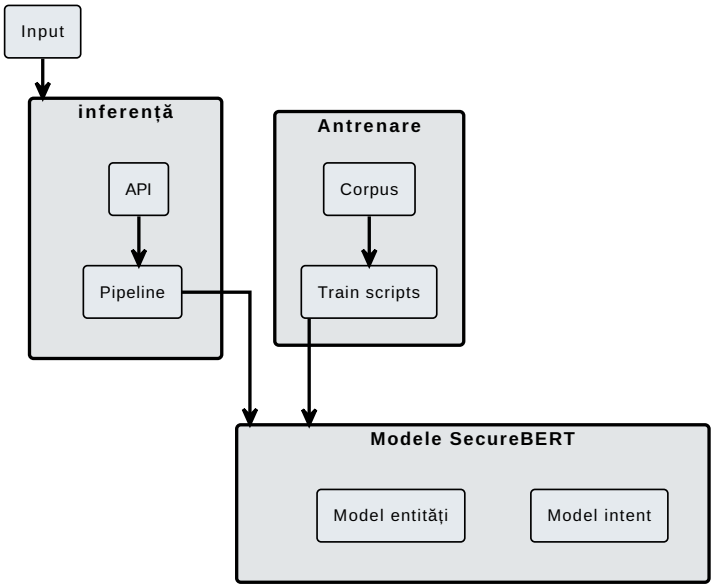

Figure 3.5: Diagrama componentă make_choice pentru detectarea intenției și entităților

Pentru a îmbunătăți performanța modelului, am implementat tehnici de data augmentation, generând variații și parafrazări sintetice ale setului de antrenare. Acest lucru ajută la creșterea robustetei modelului în fața diversității limbajului natural.

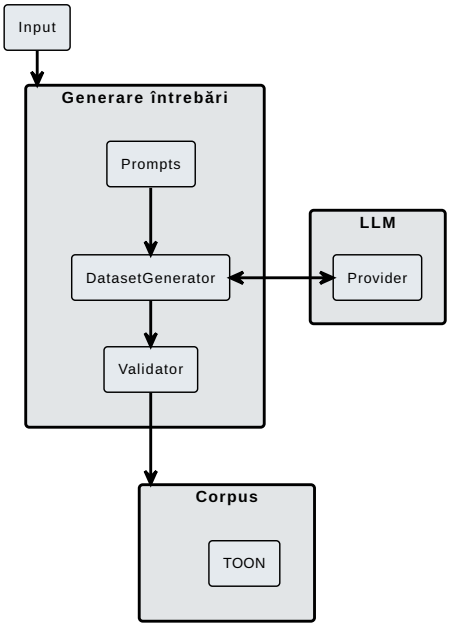

Figure 3.6: Diagrama componentă generate_questions pentru generarea corpusului TOON

Capitolul 4: Implementare

4.1 Mediu de dezvoltare

4.2 Framework-uri și biblioteci utilizate

4.3 Containerizare și infrastructură

4.4 Măsuri de securitate aplicate

4.5 Automatizare

Capitolul 5:
Testare și Evaluare

5.1 Metodologia de testare

5.1.1 Teste unitare

5.1.2 Teste de integrare

5.1.3 Teste de performanță

5.2 Metrici și criterii de evaluare

5.3 Validarea rezultatelor

Capitolul 6: Rezultate și Discuții

6.1 Benchmark-uri comparative

6.2 Analiza performanței

6.3 Beneficii și limitări

6.4 Direcții de îmbunătățire

Capitolul 7:
Concluzii

7.1 Gradul de atingere a obiectivelor

7.2 Contribuții

7.3 Impact și dezvoltări viitoare

Bibliografie

[1] *Cryptographic Standardization*. `https://prism.sustainability-directory.com/term/cryptographic-standardization/`. Accesat 2025.

[2] Y. Dong et al. "ChatIoT: Large Language Model-based Security Assistant for Internet of Things with Retrieval-Augmented Generation". In: *arXiv preprint arXiv:2502.09896* (2025).

[3] H. Hu and K. Yuan. "Identification of Cryptographic Algorithms Based on CNN". In: *Proc. 4th Int. Conf. on Computer, Artificial Intelligence and Control (CAIC)*. 2025.

[4] B. D. Kim et al. "Cryptanalysis via Machine Learning Based Information Theoretic Metrics". In: *arXiv preprint arXiv:2501.15076* (2024).

[5] J.-W. Lee et al. "Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network". In: *IEEE Access* 10 (2022), pp. 30039–30054.

[6] M. Q. Li and B. C. M. Fung. "Security Concerns for Large Language Models: A Survey". In: *arXiv preprint arXiv:2505.18889* (2025).

[7] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[8] J. Mo, H. Kuang, and X. Li. "Password Strength Detection via Machine Learning: Analysis, Modeling, and Evaluation". In: *arXiv preprint arXiv:2505.16439* (2025).

[9] Bruce Schneier. *Applied Cryptography*. Wiley, 1996.

[10] William Stallings. *Cryptography and Network Security*. Pearson, 2017.

[11] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 2005.

[12] Daniel L. Wheeler. "zxcvbn: Low-Budget Password Strength Estimation". In: *Proc. 25th USENIX Security Symposium*. 2016, pp. 157–173.

[13] Z. Xu et al. "Deep Learning-based Intrusion Detection Systems: A Survey". In: *arXiv preprint arXiv:2504.07839* (2025).

Capitolul 8:
Anexe

8.1 Listă de diagrame suplimentare

8.2 Configurații tehnice

8.3 Scripturi și manual de utilizare