Computer Science Department
University of Crete

CS458: Introduction to Cryptography
Instructor: Harry Manifavas

Project
Crypto Agility Framework for Weak Cryptography and PQC transition

Deadline: 31/01/2025, 23:59
v0.4

Notes:

- You will have approximately **2 months** to complete the project. There will be **no extension**. The project accounts for **30%** of the overall grade. It can be done in **teams of up to 4 people**. You are allowed to use **AI tools** or code found online to build a complete system. At the end, there will be an **oral examination** and similarity checking.
- Due to the workload required, you should **start early** on the necessary research as well as the development. Ensure you **fully understand all the concepts** involved and **design the architecture properly** before starting to write code.

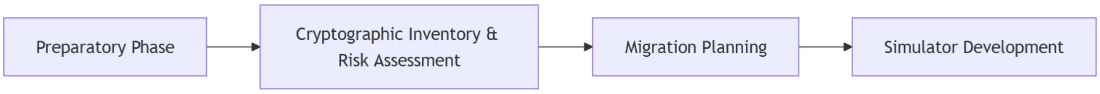
Table of Contents

Introduction

The rise of quantum computing presents a critical challenge to existing cryptographic systems[1]. This project focuses on building a **crypto agility[2] framework** designed to facilitate transition from cryptographically vulnerable to cryptographically strong primitives[3]. That includes the transition of Post-Quantum Cryptography[4] (PQC) algorithms into existing infrastructures. Through this project, students will:

1. Develop an understanding of cryptographic agility
2. Design and **implement** a scanner to compile a cryptographic inventory[5]
3. Risk assess[6] and prioritize cryptography-related vulnerable primitives (not just quantum-vulnerable)
4. Review standards and guidelines to ensure interoperability[7] and compliance[8]
5. Develop a phased migration roadmap[9]
6. Demonstrate vulnerable cryptography and PQC transition strategies
7. Design and **implement** a crypto agility simulation

Project Structure



Part 1: Preparatory Phase

- **Goal**: Understand the basics of cryptographic agility, cryptographic inventories and PQC.
- **Tasks**:
    - Study preparatory material
        - **Terminology** and **concepts**
        - NIST guidelines[10], European guidelines[11], PQC Migration Handbook[12]

[1] https://en.wikipedia.org/wiki/Harvest_now,_decrypt_later
[2] https://en.wikipedia.org/wiki/Cryptographic_agility
[3] https://en.wikipedia.org/wiki/Cryptographic_primitive
[4] https://en.wikipedia.org/wiki/Post-quantum_cryptography
[5] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWP0kj
[6] https://en.wikipedia.org/wiki/Information_security_management
[7] https://en.wikipedia.org/wiki/Interoperability
[8] https://nordlayer.com/learn/regulatory-compliance/cybersecurity-compliance/
[9] https://en.wikipedia.org/wiki/Technology_roadmap
[10] NIST, Migration to Post-Quantum Cryptography fact sheet, Aug 2023, https://www.nccoe.nist.gov/sites/default/files/2023-08/quantum-readiness-fact-sheet.pdf
[11] European Commission Recommendation, A Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography, Apr 2024, https://digital-strategy.ec.europa.eu/en/library/recommendation-coordinated-implementation-roadmap-transition-post-quantum-cryptography
[12] https://publications.tno.nl/publication/34641918/oicFLj/attema-2023-pqc.pdf

  - Identify **vulnerable cryptographic primitives** (not just quantum-vulnerable) – see Appendix (vulnerable crypto)
  - Familiarize with existing open-source **crypto inventory** tools (Part 2)
  - Familiarize with existing open-source **crypto agility** tools (Part 4)
  - Familiarize with compliance standards
- **Deliverables**:
  - A chapter in your report summarizing Part 1 findings and key terms/concepts

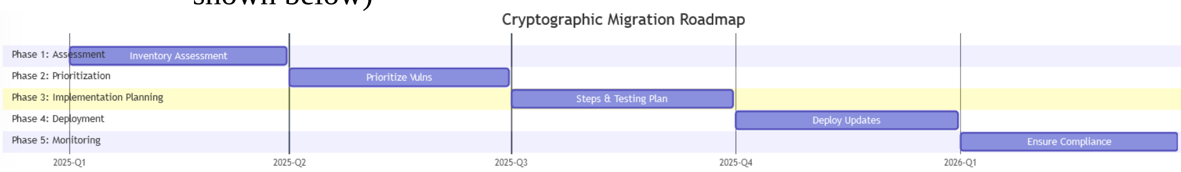
Part 2: Cryptographic Inventory and Risk Assessment

- **Goal**: **Implement** a tool to identify and prioritize vulnerable crypto primitives
- **Tasks**:
  - Create a dummy folder with cryptographically vulnerable source code (e.g., Python, C files)
    - Ask AI to create code samples (max 3 different programming languages) incorporating vulnerable functionality – see Appendix (Vulnerable Code Samples)
    - The samples must be functioning / runnable (e.g., actually performing encryption/decryption)
  - Develop a tool (with a database and GUI) to track cryptographic assets
  - The tool should scan the folder with the source code files and search for library calls including weak primitives (like those listed in the Appendix)
  - To further test your code, try it on samples created by another team
  - Implement a risk assessment module that compares your findings to currently accepted best practices
    - Prioritize (and be able to sort in the GUI) your findings in terms of criticality/severity
    - Explain what is the meaning of the risk levels that you chose (e.g., High / Medium / Low)
  - Print scan statistics
    - Number of files scanned (x python, y java, ...)
    - Number of vulnerable files discovered (x python, y java, ...)
- **Deliverables**:
  - A working **cryptographic inventory tool** with a risk assessment component
  - A chapter in your report summarizing Part 2 findings

Part 3: Crypto Migration Planning

- **Goal**: Create a phased roadmap for stronger cryptography.
- **Tasks**:
  - Design a step-by-step migration plan
    - Cryptographic migration involves replacing outdated or weak cryptographic primitives with stronger ones. This process requires identifying crypto vulnerabilities, and ensuring minimal impact on day-to-day operations. In this project, migration planning also involves addressing PQC challenges.

        - Migration takes time and must be implemented as a **structured process**
            - Assessment: Inventory and assess current cryptographic primitives.
            - Prioritization: Rank vulnerabilities based on criticality and impact.
            - Implementation planning: Define steps for replacing weak cryptography.
            - Testing: Test systems with updated cryptography for functionality and interoperability.
            - Deployment: Roll out updated cryptographic systems in phases.
            - Monitoring: Continuously monitor and ensure compliance post-migration.
        - The process should include detailed steps, prioritization criteria, expected timeframes and key milestones. Focus on gradual adoption to ensure stability. Include considerations for business continuity[13], interoperability and backwards compatibility[14].
    - Consolidate the plan with a visual migration roadmap (a sample is shown below)



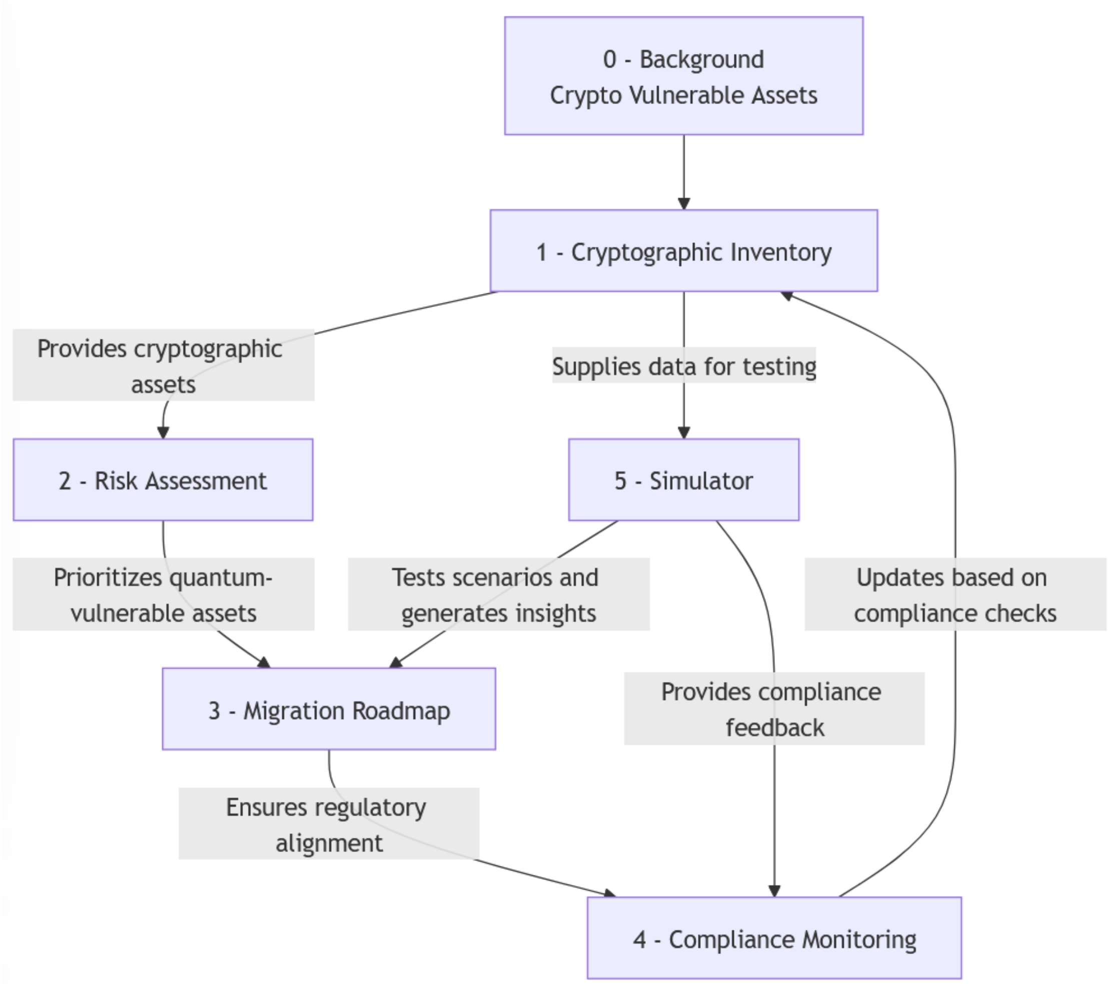
- **Deliverables**:
    - A chapter in your report summarizing Part 3 findings (migration roadmap for weak cryptography replacement).
    - Use a Small and Medium-Size Enterprise (SME)[15] as a case study (a focus of your analysis), considering its typical constraints.
    - Explore how these factors influence the adoption of stronger cryptographic systems and the transition to PQC.

Part 4: Crypto Agility Simulator Development

- **Goal**: **Implement** a functional simulator to showcase cryptographic agility and transition strategies for addressing weak cryptography and preparing for PQC integration
- **Tasks**:
    - Simulate cryptography and quantum-vulnerable systems, **agility** and **transition** scenarios where cryptographic systems are assessed for vulnerabilities
    - Use the code you developed in **Part 2** (Cryptographic Inventory and Risk Assessment) to identify weak cryptographic implementations and facilitate upgrades

[13] https://en.wikipedia.org/wiki/Business_continuity_planning
[14] https://en.wikipedia.org/wiki/Backward_compatibility
[15] SMEs (e.g., a small retail business or a software development company) face several challenges such as limited resources, lack of dedicated cryptographic expertise, older infrastructure, and budget considerations.

    - Illustrate agility by allowing the simulator to switch between algorithms dynamically and automatically suggest replacements (Do not consider replacements with PQC algorithms)
    - Include **risk** prioritization and **compliance** monitoring[16] in the simulation
    - Output comprehensive statistics
        - Number of vulnerable files discovered (x in python, y in java, ...)
        - Number of files automatically fixed while ensuring they remain functional after cryptographic updates
        - Number of files pending for manual fixes, especially when automation cannot guarantee correctness or compatibility
- **Deliverables**:
    - A **functional simulator**
    - A chapter in your report will be the simulator's user guide



[16] Compliance monitoring must be based on standards. As a reference, use NIST SP 800-57 and NIST SP 800-131A.

Deliverables

1. **Report**: Detailed documentation, as described above
    1. **Chapter 1**: Introduction
    2. **Chapter 2**: Cryptography Inventory tool
    3. **Chapter 3**: Crypto agility and migration planning
    4. **Chapter 4**: Crypto-Agility simulator tool
    5. **Chapter 5**: Conclusions
        - Lessons learned
        - List of tools and references used
2. **Presentation**: Summary of the project for oral defense
3. **Software**: Cryptography inventory tool, simulator
    1. Readme file (installation and configuration instructions)
    2. Source code
    3. Tools reporting functionality (online documentation)
        - Add an "export" button (scan & simulation results) to csv
        - Add a "Help" button to display your pdf report (dummy file in the beginning)

BONUS Tasks: 10%

1. Crypto **vulnerability scanner**
    - Extend your scanner to recursively scan a directory and its subdirectories, ensuring all files within a file system are analyzed
    - Expand the scanner's capabilities to identify a broader range of vulnerabilities beyond those in the appendix. Consider some of the following:

Category	Cryptographic Issue	Details
Symmetric Ciphers	Blowfish (key size < 128 bits)	Short key sizes are inadequate for modern security standards.
	RC4	Known vulnerabilities; susceptible to bias and statistical attacks.
Asymmetric Ciphers	DH KE weak parameters: Short prime modulus or insecure generator	Small modulus sizes (e.g., < 2048 bits) or insecure generator values (e.g., 1, or p−1) make the system susceptible to attacks.
	DH KE quantum threat: vulnerability to Shor's algorithm	Diffie-Hellman is completely insecure against quantum computers; a transition to post-quantum cryptographic alternatives is necessary.
Hash Functions	SHA-224	Too small for modern security; effective security is reduced significantly.
	Whirlpool	Secure but uncommon; improper implementations can introduce vulnerabilities.
Weak Modes	CBC mode with static IV	Predictable IVs make ciphertext vulnerable to chosen-plaintext attacks.

2. Investigation **case management**
    - Every scan must be stored in the database as a separate “case” with a specific name, provided by the user
    - Create case (use the name provided to create a separate database table)
    - Load case (display previously collected results about this case)
    - Delete case
    - Show summary of cases (display summary results, like the statistics, for all stored cases)
3. **Database management**
    - Clear database (delete all contents)
    - Export database (to create a backup or to transfer the contents to another application instance)
    - Import database (e.g., import a backup)
    - Stress test the application and the database by scanning a large file system and importing lots of files
4. Presentation aspects
    - Enhanced statistics: Visualize data in graphs

Indicative Timetable & Rubric

Week	Phase	Criteria	Weight (%)	Description
1 - 2	**Preparatory Phase**	Initial Understanding	5	Demonstrates understanding of cryptographic agility, cryptographic inventory and PQC fundamentals
		Research Quality	5	Includes well-documented findings from preparatory readings.
3 - 4	**Crypto Inventory & Assessment**	Tool Functionality	25	Accurate and functional inventory tool that identifies and tracks cryptography-related assets
		Risk Prioritization	5	Clear, logical, and correct prioritization of cryptography-vulnerable assets
5 - 6	**Migration Planning & Case Study**	Roadmap Clarity	5	Well-structured, actionable, and realistic migration roadmap
		Business Continuity	5	Consider operational and business priorities during migration (e.g., in an SME environment)

		Case Study Insights	10	Comprehensive case study report with real-world applicability.
7 - 8	**Simulator Development**	Simulation Accuracy	25	Valid representation of cryptography (including PQC) transition scenarios and risk assessments
		User Guide Quality	5	Clear and practical instructions for using the simulator
	Overall	Presentation Quality	5	Effective communication of the project through presentations and reports. Evidence of teamwork and fair distribution of tasks.
		Creativity	5	Evidence of original approaches and problem-solving skills
			100	

Instructions on How AI Tools Can Help

AI tools (ChatGPT, Gemini, Windows copilot, GitHub Copilot, code tools) can greatly enhance **efficiency** and **creativity** in this project. Here are phase-specific instructions:

Preparatory phase

- **Literature review**: Use AI tools to summarize key points from reference web pages, documents and standards
- **Understanding concepts**: Ask AI for clarifications on terms like cryptographic agility, cryptographic inventory, vulnerable cryptographic artifacts, quantum-safe algorithms, interoperability, compliance, transition, roadmap, etc.

Cryptography inventory & Assessment

- **Code suggestions**: Use AI coding assistants to write functions for cryptographic asset identification and tracking, database storage and GUI displays. Generated pseudocode for asset scanning and database integration should be tested and refined for accuracy.
- **Database integration**: Seek AI suggestions for designing and optimizing database schemas to track cryptographic assets.
- **Risk analysis**: Generate risk matrices[17] or models based on inputs, such as asset vulnerabilities and threat[18] levels

Migration Planning & Case Study

- **Roadmap creation**: Seek AI input for drafting migration phases and aligning them with business continuity goals (use an SME as a case study)
- **Visualization**: Create timelines or other visual aids to showcase the plan

[17] https://en.wikipedia.org/wiki/Risk_matrix
[18] https://en.wikipedia.org/wiki/Threat_(computer_security)

- **Case study analysis**: Use AI to draft sections of the case study, potentially including cost breakdowns and impact analyses, based on your inputs

Simulator Development

- **Scenario modeling**: Use AI to help you simulate system responses to weak cryptography threats versus vulnerable assets and develop interactive scenarios. While AI can assist with basic scenario modeling, you must critically evaluate the simulated outputs for correctness.
- **Practical examples/prompts**:
  - "Generate sample datasets for testing quantum-vulnerable cryptographic systems"
  - "Suggest algorithms for automating risk prioritization"

General use

- **Experiment** with various AI tools to identify those best suited to specific phases of the project
- **Collaboration**: You may utilize AI tools for sharing notes, task delegation, tracking progress, and maintaining team communication.
- **Presentation preparation**: Create slide content or draft talking points for the final presentation (e.g., based on the contents of your report as well as the implemented code).
- **Document formatting**: Automate the creation of professional-looking documents (e.g., for the roadmap) using timelines, flowcharts, tables, etc.
- **Debugging**: Leverage AI to troubleshoot crypto inventory and simulation code
- **Code checking**: Validate or optimize code using AI tools for enhanced efficiency.

Guidelines for ethical use of AI

1. **Transparency**: Always cite AI-generated content, whether text, code, or visualizations. As already stated, there is no penalty for AI use.
2. **Supplement, not replace**: Use AI as a helper but *ensure original understanding and critical thinking.*
3. **Collaboration**: Share AI findings within the team to foster mutual learning and avoid over-reliance by individuals

References

- Cryptosense, Cryptographic Inventory (6 short videos)
  - https://www.youtube.com/watch?v=91dMLnCv5hQ&list=PLA-8aGQm6tkL6PPTbdg6cy74x7TWFFU3V&ab_channel=Cryptosense
- Cryptographic Agility
  - https://www.youtube.com/watch?v=8pGJVTekDyM&ab_channel=RSAConference

Appendix

Sample Vulnerable Crypto Primitives

Category	Cryptographic Issue	Details
Symmetric Ciphers	DES	Outdated; 56-bit key size is insufficient for modern security.
	3DES with 1 key	Operates like DES; no additional security; insecure.
	3DES with 2 keys	Vulnerable; only provides ~80 bits of security, which is inadequate by modern standards.
	3DES with 3 keys	Deprecated; provides ~112 bits of security, insufficient against quantum attacks.
	AES-128	Secure against classical attacks but not quantum-safe (vulnerable to Grover's algorithm).
	AES-192	Offers slightly better security than AES-128 but also vulnerable to quantum attacks.
	AES-256	Secure against both classical attacks and quantum attacks.
Asymmetric Ciphers	RSA with short keys (512, 1024 bits)	Easily breakable with classical attacks; completely broken with quantum computing (vulnerable to Shor's algorithm).
	RSA with 2048, 3072+ bits	Secure against classical attacks; completely broken with quantum computing (vulnerable to Shor's algorithm).
	RSA without proper padding	Vulnerable to padding oracle attacks, irrespective of key length.
Hash Functions	MD5	Obsolete; Broken due to collision vulnerabilities; insecure under both classical and quantum attacks.
	SHA-1	Obsolete; vulnerable to collision attacks under classical and quantum contexts.
	SHA-256	Secure under classical conditions; Grover’s algorithm reduces effective security to ~128 bits.
Weak Modes	DES with ECB mode	Insecure mode; leaks patterns in plaintext due to lack of diffusion.
	3DES with ECB mode	
	AES with ECB mode	

Sample Vulnerable Code

ChatGPT prompt: “Create three functioning Python scripts, each demonstrating a crypto vulnerability, like the ones listed in the project description.”

Example: DES (code, output)

```python
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes

def des_encryption_demo():
    key = b'8bytekey'  # DES requires an 8-byte key
    cipher = DES.new(key, DES.MODE_ECB)  # ECB mode is insecure
    plaintext = b"SensitiveData"

    # Padding plaintext to be a multiple of the block size
    padded_plaintext = plaintext.ljust(16, b'\x00')

    # Encrypt
    ciphertext = cipher.encrypt(padded_plaintext)
    print(f"DES Encrypted Data: {ciphertext}")

    # Decrypt
    decrypted = cipher.decrypt(ciphertext).rstrip(b'\x00')
    print(f"DES Decrypted Data: {decrypted}")

if __name__ == "__main__":
    des_encryption_demo()
```

```
DES Encrypted Data: b'\x1aD\\=t\x0e\xbc$-\x98\xc4y\x1f\xd5\xb7*'
DES Decrypted Data: b'SensitiveData'
```