Midterm

(Deadline May 16, 2025)

Problem 1

Arithmetic & Structure in $\mathbb{F}_{2^4}$ (6 %)

Description

Work in the field $\mathbb{F}_{2^4}$ defined by the irreducible polynomial

$$P(x) = x^4 + x + 1.$$

Requirements

a) Compute

$$f_1(x) = x^2 + 1,\ g_1(x) = x^3 + x^2 + 1 \quad \text{and} \quad f_2(x) = x^2 + 1,\ g_2(x) = x + 1$$

then evaluate

- $f_i + g_i$, $f_i - g_i$ and $f_i \cdot g_i \bmod P(x)$ for $i \in \{1, 2\}$.

b)

a. Prove that $P(x)$ is irreducible over $\mathbb{F}_2$.

b. Determine the multiplicative order of x in $\mathbb{F}_{2^4}^{\times}$.

c. Decide whether $P(x)$ is **primitive**. Show all reasoning.

Evaluation

Item	Points	Rubric
Field arithmetic (Req. 1)	1	Correct reductions for each operation
Irreducible proof	2	Exhaustive root test or suitable theorem
Order computation	2	Order found and justified
Primitiveness decision	1	Correct conclusion with explanation

Problem 2

Irreducible and Primitive Polynomials (6 %)

Description

A polynomial $p(x)$ over a finite field $\mathbb{F}_q$ is *irreducible* if it cannot be factored into polynomials of lower degree over the same field. An irreducible polynomial $p(x)$ of degree m is *primitive* if the smallest positive integer n such that $p(x)$ divides $x^n - 1$ is $n = q^m - 1$.

Requirements

a) Consider the polynomial $p(x) = x^4 + x + 1$ over $\mathbb{F}_2$. Prove that it is irreducible.

b) Demonstrate that $p(x)$ is also primitive by showing that:

- $p(x)$ does not divide $x^n - 1$ for any $n < 2^4 - 1 = 15$
- $p(x)$ divides $x^{15} - 1$

c) Explain the relationship between primitive polynomials and maximal-length LFSR sequences. Why does the primitive polynomial $p(x) = x^4 + x + 1$ generate the maximal-length sequence $s = (0, 0, 1, 1, 0, 1, 0, 1, 1)$?

d) List all irreducible polynomials of degree 4 over $\mathbb{F}_2$. Among these, which ones are primitive? Provide a systematic method for identifying primitive polynomials among irreducible polynomials of a given degree.

Evaluation

Item	Points	Rubric
Irreducibility proof	1	Correct proof of irreducibility
Primitivity demonstration	2	Complete demonstration of primitive property
LFSR connection explanation	1	Clear explanation of maximal-length sequence
Polynomial identification	2	Comprehensive list with correct classification

Problem 3

Berlekamp–Massey vs EEA on a Very Short Sequence (7 %)

Description

The Berlekamp-Massey Algorithm (BMA) and Extended Euclidean Algorithm (EEA) are two different methods to find the shortest Linear Feedback Shift Register (LFSR) that can generate a given sequence. While these algorithms approach the problem differently, they should produce equivalent results.

Requirements

Consider the following binary sequence:

$$s = (0, 0, 1, 1, 0, 1, 0, 1, 1). \qquad \text{(Sequence S)}$$

a) Apply the **Berlekamp–Massey Algorithm (BMA)** to sequence S. Show your work step by step and derive the minimal polynomial $C_{\text{BM}}(x)$.

b) Apply the **Extended Euclidean Algorithm (EEA)** to find the minimal polynomial for the same sequence. Form $S(x)$ with the sequence bits as coefficients in ascending powers of x. Use EEA with the pair $\left(x^{2n}, S(x)\right)$ where n is your estimation of the linear complexity. Show your work and derive $C_{\text{EEA}}(x)$.

c) Compare the results from both methods. Are they equivalent? Briefly explain why these two algorithms should yield the same minimal polynomial.

Evaluation

Part	Points	Details
BMA application	3	Correct step-by-step work and final minimal polynomial
EEA application	3	Correct formulation and derivation of minimal polynomial
Comparison of results	1	Accurate comparison and explanation of equivalence

Problem 4

Designing & Auditing a Custom AES S-Box (7 %)

Description

Replace the classical AES irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ with

$$p_2(x) = x^8 + x^5 + x^3 + x + 1.$$

Let the affine-transform matrix M be obtained by *cyclically rotating* the binary vector 11110000 (row-major) and let the affine constant be $0x63$ as in AES.

Requirements

a) Precisely define M and prove it is invertible over $\mathbb{F}_2$. Is it *mathematically justified* to construct the affine transformation matrix M by cyclically rotating the binary vector 11110000? Reference at least **one** peer-reviewed source.

b) Implement (pseudo-code is fine) the S-box $S(x) = M \cdot x^{-1} \oplus c$ built from

- inversion in $\mathbb{F}_{2^8}$ with modulus $p_2(x)$,
- the new affine layer.

c) Using **your own software** (Python or GoLang), compute the full S-box and verify the seven cryptographic criteria below. Summarise the numerical results you obtain.

#	Criterion	Target for AES-like strength
1	Bijectivity	256 unique values
2	Non-linearity	≥ 112
3	Strict Avalanche (SAC)	$\sim$50% bit flips
4	Differential Uniformity (DU)	≤ 4
5	Linear-approx. bias	≤ 16
6	Algebraic degree	7
7	Fixed-point count	0

d) Discuss whether cyclic rotation of 11110000 is *mathematically justified* as a method to build secure affine layers. Reference at least **one** peer-reviewed source.

Evaluation

Item	**Points**	**Rubric**
Matrix definition & invertibility	2	Mathematical proof with citation
S-box construction	1	Correct inversion and affine layer
Empirical verification	3	Analysis of all cryptographic criteria
Security discussion	1	Critical evaluation with citation

Problem 5

Side-Channel-Aware MixColumns Optimisation (6 %)

Description

AES round-columns are multiplied by the matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

Requirements

a) Explain why a **four-table** T-box implementation merges SubBytes, ShiftRows and MixColumns.

b) Design a constant-time LUT scheme that avoids cache-timing leakage on modern superscalar CPUs.

c) Compare its memory footprint and latency with a bit-sliced implementation. Provide rough cycle counts for an AVX2-capable processor (no assembly required).

Evaluation

Item	Points	Rubric
T-box implementation rationale	2	Clear explanation of operation merging
Constant-time LUT design	2	Effective cache-timing attack mitigation strategy
Implementation comparison	2	Accurate memory and latency analysis with AVX2

Problem 6

Symmetric Round-Structure Sharing (6 %)

Description

In order to make AES encryption and decryption more similar in structure, the MixColumns operation is missing in the last round.

Requirements

Explain how to take **advantage** of this property to share some of the code (for software implementation) or chip area (for hardware implementation) for AES encryption and decryption.

Hint: Denote InvSubBytes, InvShiftRows, and InvMixColumns as the inverses of SubBytes, ShiftRows, and MixColumns operations, respectively. Try to show:

- *The order of InvSubBytes and InvShiftRows is indifferent;*
- *The order of AddRoundKey and InvMixColumns can be inverted if the round key is adapted accordingly.*

Additionally, provide a **quantitative estimate** of gate-count savings when the final MixColumns is removed, for both a micro-coded software AES and a low-area ASIC implementation.

Evaluation

Item	Points	Rubric
Operations ordering	3	Operation commutativity explanation
Gate-count savings	3	Software and ASIC implementation analysis

Problem 7

Irreducible vs. Primitive Polynomials (6 %)

Description

Irreducible and primitive polynomials are key concepts in finite field theory with important applications in cryptography. Irreducible polynomials cannot be factored over their base field, while primitive polynomials generate maximal-length sequences in LFSRs.

Requirements

a) Provide formal definitions for (a) an irreducible polynomial over $\mathbb{F}_2$ and (b) a primitive polynomial over $\mathbb{F}_2$. Clearly explain the relationship between these two concepts.

b) Let $p(x) = x^4 + x^3 + 1$ be a polynomial over $\mathbb{F}_2$.

- Prove whether $p(x)$ is irreducible over $\mathbb{F}_2$ or not.
- If $p(x)$ is irreducible, determine whether it is primitive. Show all your work.

c) Let $q(x) = x^4 + x + 1$ be a polynomial over $\mathbb{F}_2$.

- Prove whether $q(x)$ is irreducible over $\mathbb{F}_2$ or not.
- If $q(x)$ is irreducible, determine whether it is primitive. Show all your work.

d) Explain the connection between primitive polynomials and maximal-length LFSR sequences. If an LFSR is based on a primitive polynomial of degree n, what is its period? How does this differ if the polynomial is irreducible but not primitive?

Evaluation

Item	**Points**	**Rubric**
Irreducibility proofs	2	Proving methods and techniques
Order computations	2	Calculating element orders in field
Primitiveness analysis	1	Determining polynomial properties
Real-world application	1	Practical use case demonstration

Problem 8

Perfect Secrecy and Linear Congruential Generators (7 %)

Description

Perfect secrecy (introduced by Shannon) provides information-theoretic security guarantees, while pseudorandom number generators like Linear Congruential Generators (LCGs) offer computational efficiency but may have security limitations.

Requirements

a) Consider a cryptosystem with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and key space $\mathcal{K}$.

- State Shannon's definition of perfect secrecy.
- Prove that if a cryptosystem achieves perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$.
- Explain why one-time pad achieves perfect secrecy but is impractical for most real-world applications.

b) Consider a linear congruential generator (LCG) defined by the recurrence relation $X_{n+1} = (aX_n + c) \mod m$, where $a = 75$, $c = 74$, $m = 2^{16}$, and $X_0 = 12345$.

- Calculate the first five outputs of this generator: X_1, X_2, X_3, X_4, X_5.
- If this LCG is used to generate a keystream for a stream cipher, explain (with proof) at least two security vulnerabilities that would arise.

c) Consider an LFSR-based stream cipher with connection polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$.

- Is this polynomial irreducible over $\mathbb{F}_2$? Is it primitive? Justify your answer.
- If an attacker observes 20 consecutive bits of the keystream, explain how the Berlekamp-Massey algorithm could be used to reconstruct the entire sequence.
- Propose a modification to the LFSR design that would improve its security against such attacks.

Evaluation

Item	Points	Rubric
Perfect secrecy analysis	2	Correct definition and proof of key-length requirement
LCG analysis	2	Complete calculation and security vulnerability identification
LFSR stream cipher analysis	3	Thorough polynomial analysis and security improvement

Problem 9

PhD Qualifying Exam Questions (6 %)

Description

This problem set emulates PhD qualifying examination questions in cryptography, focusing on number theory, abstract algebra, and their applications in cryptographic systems.

Requirements

a) Let $f(x) = ax^2 + bx + c$ be a quadratic function over $\mathbb{Z}_p$, where p is a prime and $a \neq 0 \mod p$. Prove that the equation $f(x) = 0$ has either 0, 1, or 2 solutions in $\mathbb{Z}_p$. Under what conditions does each case occur?

b) Let G be a group and $a, b \in G$ be two elements such that $a^5 = b^3 = e$ (the identity element). If $ab = ba$, what is the order of the element ab?

c) Consider a protocol where Alice selects a random prime p and sends it to Bob. Bob chooses a random quadratic residue $y \in \mathbb{Z}_p^*$ and sends it to Alice.

- Describe how Alice can determine with certainty whether y is indeed a quadratic residue.
- Can Bob cheat in this protocol by sending a non-quadratic residue? Explain.
- Explain how this concept relates to the security of cryptographic schemes.

d) In RSA, the public exponent e and private exponent d satisfy $ed \equiv 1 \pmod{\phi(n)}$, where $\phi(n)$ is Euler's totient function. Using knowledge of number theory, explain why choosing e to be a small prime (like 3 or 65537) while maintaining a large, random d does not compromise security.

Evaluation

Item	Points	Rubric
Quadratic equation	1	Proof with case identification
Group theory	1	Order calculation with justification
Quadratic residue	2	Security and verification analysis
RSA parameters	2	Number-theoretic justification

Problem 10

Enterprise Key-Roll-Over Strategy (7 %)

Description

Key management in enterprise environments requires balancing security needs with operational continuity. When key material needs to be revoked and replaced, organizations must follow structured processes to maintain system availability.

Requirements

If a company has encrypted its most sensitive data with a key held by the chief technology officer and that person was fired, the company would want to change its encryption key. Describe **what would be necessary** to revoke the old key and deploy a new one.

Additionally, provide an incident-response timeline and specify NIST SP 800-57 key-destruction requirements.

Hint: NIST Special Publication 800-57 Part 1

Evaluation

Item	**Points**	**Rubric**
Key revocation process	3	Strategy for key invalidation
New key deployment	2	Secure key distribution approach
Incident response	2	NIST-compliant procedures

Problem 11

Lazy RSA Key Rotation (6 %)

Description

RSA key rotation practices in operational settings sometimes prioritize convenience over security. Reusing cryptographic key components can introduce subtle vulnerabilities that undermine security guarantees.

Requirements

a) Alice just started working at Bitdiddle, and her first public key is (n, e) where n is the product of two safe primes, and $e = 3$.

Whenever a new month starts, Alice (being lazy) changes her public key as little as possible to get by the auditors. What she does, in fact, is just to advance her public exponent e to the next prime number. So, month by month, her public keys look like:

$$(n, 3), (n, 5), (n, 7), (n, 11), ...$$

Explain how Alice's laziness might get her in trouble.

b) The next year, Alice tries a different scheme.

In January, she generates a fresh public key (n, e) where n is the product of two primes, p and q. In February, she advances p to the next prime p_l after p, and q to the next prime q_l after q, and sets her public key to (n_l, e_l) for a suitable e_l. Similarly, in March, she advances p_l to p_2 and q_l to q_2, and so on.

Prove a concrete lattice attack that recovers one month's factors given two adjacent moduli.

Evaluation

Item	Points	Rubric
First scheme vulnerability	2	Identification of modulus weakness
Lattice attack proof	4	Mathematical derivation of attack

Problem 12

Inverse MixColumns Hardware Constant-Time Design (6 %)

Description

Hardware implementations of AES must balance performance with side-channel resistance, particularly for operations like InverseMixColumns that involve complex finite field arithmetic.

Requirements

a) Given the matrix for the Inverse MixColumns operation:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

Describe how the matrix multiplication in the Inverse MixColumns step is performed within AES decryption. Your answer should include a **brief explanation of the arithmetic** used in the finite field $GF(2^8)$ and **how it differs from standard matrix multiplication**.

b) Multiplications in $GF(2^8)$ can be complex. However, they can be simplified using repeated application of simpler operations. Explain how multiplication by 9, 11, 13, and 14 in $GF(2^8)$ can be implemented using the simpler multiplication by 2 and addition (XOR). Provide the **mathematical expressions** that represent these multiplications.

c) Discuss how the use of lookup tables (LUTs) can further simplify the implementation of Inverse MixColumns in AES decryption. What are the **advantages** and potential **drawbacks** of using LUTs for this purpose?

d) Derive logical depth and area when coefficients 9, 11, 13, 14 are implemented with a shared sub-multiplier network.

Evaluation

Item	Points	Rubric
Matrix multiplication	2	Explanation of $GF(2^8)$ arithmetic
Efficient techniques	2	Application of simpler operations
LUT analysis	1	Advantages and drawbacks
Logical depth/area	1	Circuit complexity calculation

Problem 13

Plaintext Block Chaining Attack Formalisation (6 %)

Description

Block cipher modes of operation can introduce security vulnerabilities if not carefully designed. Plaintext Block Chaining (PBC) is an example where seemingly minor modifications to established constructions can undermine security.

Requirements

Show how an attacker who knows IV, C_0, C_1, C_2 (which are public) and also knows that $m_1 = m_2 = x$ (for a known x) can easily compute m_0.

Additionally, provide the adversary's distinguishing advantage $\varepsilon(n)$ as a function of block length.

Evaluation

Item	Points	Rubric
Attack demonstration	3	Derivation of plaintext recovery
Distinguishing advantage	3	Function of block length analysis

Problem 14

Existential Forgery Against CBC-MAC-like Construction (6 %)

Description

Message Authentication Codes (MACs) provide integrity and authenticity verification in secure communications. CBC-MAC constructions must be carefully implemented to avoid vulnerabilities that could allow forged authentication tags.

Requirements

Describe how to produce an existential forgery against this MAC scheme.

Generalise the attack to variable-length messages and prove insecurity under chosen-message queries.

Hint 1: Start with two messages M_1 and M_2 (not to be confused with the individual blocks of a message in the diagram above) for which you know the outputs (IV_1, T_1) and (IV_2, T_2). Produce another message M_3 for which (IV_1, T_2) will be the MAC. M_3 will be close to the concatenation $M_1||M_2$, but with one block altered.

Hint 2: There is also a way to produce a forgery with only one known block if you look closely.

Caution: The blocks $m_0, m_1, \ldots$ in the diagram are distinct from the complete messages $M_1, M_2, \ldots$

Evaluation

Item	Points	Rubric
Existential forgery attack	3	Attack strategy implementation
Variable-length generalization	3	Security proof for chosen-message

Problem 15

RSA Broadcast Attack (6 %)

Description

RSA encryption with the same public exponent across multiple recipients can lead to security vulnerabilities when the same message is sent to multiple users.

Requirements

Remember that an RSA public-key is a pair (n, e) where n is the product of two primes.

$$RSA_{(n,e)}(m) = m^e \mod n$$

Assume that three users in a network Alice, Bob and Carl use RSA public-keys $(n_A, 3)$, $(n_B, 3)$ and $(n_C, 3)$ respectively. Suppose David wants to send the same message m to the three of them. So David computes

$$y_A = m^3 \mod n_A,\ y_B = m^3 \mod n_B,\ y_C = m^3 \mod n_C$$

and sends the ciphertext to the relative user.

Show how an eavesdropper Eve can now compute the message m even without knowing any of the secret keys of Alice, Bob and Carl.

Evaluation

Item	Points	Rubric
Approach to broadcast attack	2	Approach to broadcast attack
CRT application and proof	2	CRT application and proof
Plaintext extraction procedure	2	Plaintext extraction procedure

Problem 16

Perfect Forward Secrecy (6 %)

Description

Forward secrecy is a property where compromise of long-term keys does not compromise past session keys, protecting previously exchanged messages from retrospective decryption.

Requirements

Suppose two parties Alice and Bob want to communicate privately. They both hold public keys in the traditional Diffie-Hellman model.

An eavesdropper Eve stores all the encrypted messages between them and one day she manages to break into Alice and Bob's computer and find their secret keys, correspondent to their public keys.

Show how using only public-key cryptography we can achieve perfect forward secrecy, i.e., Eve will not be able to gain any knowledge about the messages Alice and Bob exchanged before the disclosure of the secret keys.

Evaluation

Item	**Points**	**Rubric**
PFS protocol specification	2	PFS protocol specification
Forward secrecy proof	3	Forward secrecy proof
Real-world implementation evaluation	1	Real-world implementation

Extra-Credit Problems (each +8 %)

Extra Credit 1

Perfect-Secrecy Multiplicative Cipher & Predictable Generators

Description

Perfect secrecy ciphers provide unconditional security but often require impractical key management, while predictable generators create exploitable patterns.

Requirements

Combine Problems 1 & 2 from the instructor's private notes: formally prove perfect secrecy for $c = mk \bmod p$ and fully break the linear congruential generator $x_i = ax_{i-1} + b \bmod p$ as a keystream source.

Evaluation

Item	Points	Rubric
Perfect secrecy proof	4	Rigorous mathematical proof of information-theoretic security
LCG cryptanalysis	4	Complete attack demonstration with key recovery

Extra Credit 2

Strengthening LFSR-Based Stream Ciphers

Description

Linear feedback shift registers (LFSRs) are efficient but vulnerable to linear algebraic attacks like the Berlekamp-Massey algorithm.

Requirements

Propose two concrete **non-linear** post-processing layers that raise the linear complexity beyond practical BM-attacks. Analyse gate count vs. security.

Evaluation

Item	Points	Rubric
First non-linear design	4	Complete design with security analysis
Second design & comparison	4	Enhanced design with comparative evaluation

Extra Credit 3

Irreducible vs Primitive Polynomials Deep-Dive

Description

Polynomial selection in finite field constructions significantly impacts cryptographic properties and performance.

Requirements

Using $f_1(x) = x^4 + x + 1$ and $f_2(x) = x^4 + x^3 + 1$:

- prove irreducibility,
- compute element orders in $\mathbb{F}_{2^4}$,
- decide primitiveness,
- and present one real-world use-case where primitiveness is essential.

Evaluation

Item	Points	Rubric
Irreducibility proofs	2	Proving methods and techniques
Order computations	3	Calculating element orders in field
Primitiveness analysis	2	Determining polynomial properties
Real-world application	1	Practical use case demonstration

Submission Guidelines

1. Upload a **single ZIP file** named `<student_id>.zip` to the new e3 platform, with the following structure:
   - `<student_id>.zip`
     - `<student_id>.pdf` (Your main solution document)
     - `code/` (Directory for all implementation files)
       - `problem3/` (For Berlekamp-Massey vs EEA implementation, if any)
       - `problem4/` (For AES S-box implementation)
         - `main.py` (or main.go)
         - `README.md` (Brief explanation of your approach)
       - `problem5/` (For constant-time LUT implementation, if any)
       - `bonus/` (For any extra credit implementations)

   (**Incorrect file structure or incorrect file names will result in a 10% deduction**)
2. **Main Solution Document**:
   - Single PDF file named `<student_id>.pdf`
   - Present solutions in **numerical order**
   - Show **all essential steps** for partial credit
3. **Code Submission**:
   - Problem 4: Include well-commented implementation
   - Code must be runnable with expected outputs
4. **Extra Credit Problems**:
   - Mark extra credit solutions on separate pages
   - Place code in the `bonus/` directory

Grading Policy

1. Regular problems: 6-7% each (Total: 100%)
2. Extra credit problems: 8% each (Additional: up to 24%)
3. **No Late Submissions:** Late submissions will receive 0 points
4. Exam is **mandatory** to pass the course
5. Partial credit requires showing your work

Good luck — and remember: *show every essential step; partial credit is generous but only when we can see your work.*