Final Topics

4440 and 5440 students cover the same material, but 5440 students may find slightly more emphasis on their midterm on definitions, proofs and logical deductions instead of algorithms and computations.

With the exception of the application to Radar **and Bitcoin**, everything we've done follows the textbook. **Boldface means 'since miderm'.** There is an emphasis on material since the midterm.

We have covered:

1. Chapter 1: all
2. Chapter 2: 2.1, 2.2, 2.3, 2.4, 2.7, 2.8, 2.9, 2.10, 2.12
3. Chapter 3: 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, **3.7, 3.8, 3.9, 3.10**
4. Chapter 4: 4.1, 4.2, 4.4, 4.5 (only an overview)
5. Chapter 5: 5.1, 5.2 (only an overview)
6. Chapter 6: 6.1, **6.2 (only 6.2.2, 6.2.3), 6.3, 6.4, 6.5, 6.6**
7. Chapter 7: **7.1, 7.2.2, 7.3, 7.4, 7.5 (7.5.1 no proofs)**
8. Chapter 8: **8.1 pp. 218-220 only, 8.4**
9. Chapter 9: **9.1, 9.2**
10. Chapter 10: **10.1**
11. Chapter 16: **16.1, 16.2 (excluding 16.2.1, 16.2.3), 16.5.1 (in incomplete detail)**
12. Chapter 18: **18.1, 18.2, 18.4 (excluding 18.4.1), 18.5**
13. Chapter 19: **19.1, 19.2, 19.3 (in incomplete detail)**

Ciphers

For each of these, you should be able to encipher or decipher a short message by hand, if given a key. For all of these, you should be able to describe the size of the keyspace in terms of factorials, exponents, parameters etc. (not the full decimal expansions!).

1. Caesar cipher (key = shift)
2. Scytale (key = size of tube to wrap it around; perhaps somewhat impractical on a midterm)
3. Substitution cipher (key = arbitrary permutation of the alphabet)
4. Vigenere cipher (key = word)
5. Affine cipher (key = α and β)
6. Hill cipher (key = $n \times n$ matrix)
7. One-time pad (key = sequence of elements of $\mathbb{Z}/26\mathbb{Z}$ if using alphabet, or $\mathbb{Z}/2\mathbb{Z}$ if using bit strings)
8. Enigma (key = a full set of rotors and reflectors... this is unlikely on your exam but know the way it works)
9. RSA (use of public key and private key)
10. Three-pass protocol (both parties have their own secret key)
11. **ElGamal using Discrete Logs (public and private key)**
12. **Elliptic Curve Encryption (like ElGamal), at least in theory (impractical for hand computations on an exam, but I could ask you to do a small step, like adding two points, or give a large scale description of the process, draw a schematic picture etc.)**

Number Theory Topics

For each of these, you should know the appropriate definitions, and be able to do standard computations by hand efficiently.

1. Modular arithmetic

2. Divisibility
3. GCD (gcd and magic box algorithms)
4. Solving $ax + by = d$ (when is it solvable? how do you solve it? how do you get one solution or all solutions?)
5. Finding an inverse modulo n (when is it possible? how to do it?)
6. Efficient arithmetic modulo n (reducing frequently; successive squaring)
7. Chinese Remainder Theorem (the statement, and the ability to find the solution that it guarantees)
8. Fermat's Theorem (statement and its use to compute exponents)
9. Know the proof of Fermat's Theorem! It's a great proof and I may ask you to prove it on an exam.
10. Euler's Theorem (statement and its use to compute exponents)
11. Inverting Matrices Mod n
12. I could ask you to give small novel proofs using the basic definitions (similar to the problem on your first homework using the definition of divisibility).
13. **Primitive roots. Know the definition, and understand their purpose (to give a new structure to the invertible elements described by exponents).**
14. **Basic properties of primitive roots and their relationship to squares.**
15. **Discrete logs (definition, basic computations).**
16. **How to find square roots mod p if p is 3 modulo 4 – if this is on the exam, I will remind you of the statement needed**
17. **How to use Chinese Remainder Theorem to find roots modulo a composite number by looking at its prime factors**
18. **How to test if an element is a square by taking $(p-1)/2$ power**

19. **How to use 4 square roots of one value to find a nontrivial factor of the modulus**

20. **Legendre and Jacobi symbols and their efficient computation (cheat sheet will be provided on exam)**

21. **Elliptic curves (definition ($y^2 = x^3 + ax + b$), group law, adding two points by hand (excluding special cases like tangency); understand that we are working modulo a prime when doing cryptography)**

Cryptographic 'Hard Problems'

Be able to state each of these hard problems precisely.

1. **Factoring**

2. **Discrete Logarithm**

3. **Elliptic Curve Discrete Logarithm**

Cryptanalysis

1. The method of frequency analysis as used against a Caesar cipher or substitution cipher

2. Cryptanalysis of the vigenere cipher (the key mathematical observations and how they are used)

3. Cryptanalysis of affine and hill, as done on your homework.

4. Basic outline of cryptanalysis of Enigma (the main important fact about permutations and its usage)

5. RSA: Be able to factor n if you know $\phi(n)$, and be able to compute $\phi(n)$ if you know the factorisation of n.

6. RSA: Be able to use the previous bullet to decipher a message you don't have the private key for, if you know $\phi(n)$ or the factorisation of n.

7. **Primality testing: 4 square roots of an element mod n**

8. **Primality testing: Fermat test**
9. **Primality testing: Miller-Rabin (This algorithm is somewhat involved; I will remind you of it if we use it)**
10. **Primality testing: Solovay-Strassen (I won't put this on the exam)**
11. **Which primes are safe to use in RSA? (i.e. know the existence of various attacks for special types of primes)**
12. **Factoring: Fermat factorization**
13. **Factoring: $p-1$ method (If we use this, I will remind you of the algorithm)**
14. **Factoring: Quadratic Sieve (Know this one!)**
15. **RSA Attack: short plaintext and padding**
16. **RSA Attack: Timing attacks and double-and-add (broad outlines only; don't memorize double-and-add, just know what properties it has that lead to timing attacks)**
17. **Baby-Step Giant-Step for computing a Discrete Log**
18. **Birthday Attack (i.e. know the meaning of this general category of attacks)**

Protocols for Cryptography and Coding

You should be able to implement these by hand, given appropriate parameters.

1. **Sending a secret message with ciphers above**
2. **Treaty Verification – this won't be on the exam unless I remind you of how it works**
3. **Diffie-Hellman Key Exchange**
4. **Digital signatures (RSA only; we will leave aside Discrete Log signatures)**

5. **Bit committment – this won't be on the exam unless I remind you of how it works**

6. **Quantum key distribution**

Error correcting codes

For the codes listed, I will always give you an explicit description of a code if you have to do something with it, but know in a general way how these examples work so it isn't a surprise.

1. **Repetition codes (ex. 1, page 393)**

2. **Parity check codes (ex. 2, page 393 and ex. 3, page 394)**

3. **Hamming codes (ex. 4, page 395)**

4. **Vocabulary (know definitions): alphabet, binary code, ternary code, q-ary code, length of a code, codeword, block codes, Hamming distance, Hamming weight, minimum distance, nearest neighbour decoding, code rate (AKA information rate), linear code, information symbols, check symbols, syndrome, coset leader, coset,**

5. **Meaning of 'metric,' i.e. properties of Hamming distance (top of page 400)**

6. **the usual code parameters: (n, M, d) and $[n, k]$ or $[n, k, d]$.**

7. **determine from minimum distance how many errors can be corrected and detected**

8. **Know what it means for two codes to be equivalent**

9. **Basics of vector spaces: know (in a practical useable sense, not a definition-regurgitation sense) what is a vector space, subspace, dimension, basis.**

10. **Know how to compute the size of a vector space or subspace based on its dimension and the size of the finite field.**

11. **Know that $\mathbb{Z}/p\mathbb{Z}$ is one of a class of things called 'finite fields' (i.e. a field which is finite). You don't need to know other finite fields (which do exist).**

12. **Linear codes: generating matrix, parity check matrix (be able to compute)**

13. **Decoding a linear code using the syndrome and parity check matrix**

14. **Be able to do computations of all the types on the coding theory worksheet from class (the last problem is more challenging and is left as a bonus, but tests the same sorts of skills, so is still relevant).**

Computer Science topics

1. Pseudorandom number generators (know what it is, what a seed is)

2. Know the complexity, i.e. time taken, by the extended gcd algorithm and by successive squaring

3. DES and Rijndael (know the basic large-scale structure but leave out the details)

4. Modes of operation for block ciphers: electronic codebook and cipher block chaining (you can ignore the others)

5. Basics of how passwords are handled by a server

6. The use of chinese remainder theorem to make computations efficient on a computer; the use of chinese remainder theorem in radar.

7. **Runtimes (exponential, subexponential, polynomial)**

8. **Key distribution and certifying trusted authorities (generalities)**

9. **Bitcoin, the basics of how it works – I can ask you comprehension questions, but won't ask you to mine bitcoins on the exam (that might be illegal?)**

Quantum Cryptography

1. **Definition of a qubit**

2. **Polarization experiment**

3. **Quantum Key Distribution**
4. **Discrete Fourier Transform (what it does for you, not the formula)**
5. **Schor's Algorithm (not in full detail, but outlines)**

Miscellaneous

These words should hold meaning for you. I may refer to them on the exam.

1. enciphering, deciphering, plaintext, ciphertext, key
2. block cipher
3. symmetric key cryptography
4. public key cryptography
5. Alice, Bob, Eve, Oscar, Mallory
6. keyspace
7. man-in-the-middle attack
8. **hard problem**
9. **one-way function**
10. **hash function**
11. **quantum computer**