

## Course Information

**Meets:** Tu and Th, 2:00PM–3:20PM in EBU3B 2154

**Instructor:** Mihir Bellare

**Office:** EBU3B 4244

**E-mail:** mihir@cs.ucsd.edu

**Course Web Page:** <http://www-cse.ucsd.edu/users/mihir/cse207>. Course notes, problem sets and solutions are available here. (Hardcopies of these items will not be provided.)

**Class Mailing List:** The mailing list is compiled from the UCSD roster, so the address that is used for you is whatever appears here. (This is not usually, say, your @cs address, but should be equivalent.) If you think the address UCSD is using is wrong (you should be able to find out what it is via `blink`) send me email quickly with an update.

**TA:** Tom Ristenpart

**Office hours:** See course web page for instructor and TA office hours.

**Contents:** This course is an introduction to cryptography. The viewpoint in this course is “theory brought to practice”, specifically the application of the theory of “provable-security” to the design and analysis of real world cryptographic schemes. We consider tasks like encryption, signatures, authentication, and key distribution. The goal is to instill understanding of fundamentals of cryptographic protocol design.

Beware that although the aim is to end up with practical solutions, the work involved is largely theoretical. We will spend much of our time understanding how to formally define and model various goals, and prove correct protocols for these goals.

This is *not* a computer security course. We will not be covering topics like operating systems security, viruses, and worms.

**Texts:** There is no text that covers all the material of this course, and there is thus no prescribed or required text. The main source of material is lecture and associated course notes, available from the class web page. The web page also has pointers to various books on the subject, parts of which might be useful references.

For the number theory part of the course, Chapter 31 of the Cormen, Leiserson Rivest and Stein book on algorithms is a good reference. I imagine many of you have this book from previous courses. Those who do not, please try to get yourself a copy of this chapter.

**Pre-requisites:** Computer algorithms, probability theory, randomized algorithms, some basic complexity theory (eg. **P**, **NP**, **NP**-completeness, reducibility between problems) and, most impor-

tantly, general “mathematical maturity.” The latter means being comfortable with mathematical definitions and proofs.

There is an undergraduate course on cryptography, CSE107, that overlaps with CSE207. CSE107 is not a pre-requisite for CSE207. Rather it attempts to present the main ideas of cryptography at a level that is less mathematically advanced than CSE207. Students (especially MS) who find CSE207 overly theoretical are encouraged to consider taking CSE107 as an alternative.

**Requirements, policies and grades:** The course requirements are some number of problem sets (also called homeworks). There is no project and there are no exams. Each problem set will be worth a certain maximum number of points. (This number may vary from problem set to problem set, and will be indicated on the problem set.) Your course score will be the sum of your scores (across all problem sets), divided by the sum, over all items, of the number of points the item is worth. (This means that different problem sets are not weighted equally, but rather weighted according to the indicated points.)

Problem sets will be available from the course web page. Solutions will be posted after problem sets are turned in.

**Rules and grading policies:** Points may be lost for failing to adhere to the policies indicated here.

Problem sets are due in class on the day indicated on the problem set. Late problem sets are not accepted. Please do not turn in problem sets at any place other than in class. (If you can't make it to class, give it to someone else to turn in for you.)

On some problem sets you may collaborate, but on others you will be required to work alone. (Which of these is the case will be indicated on the problem set.) If it is a non-collaborative problem set, you are required to not discuss it with anyone other than the instructor or TA. If it is a collaborative problem set, you may work on it with at most one other person, this being a student in the class. However you must *write up solutions on your own*, in your own words, and indicate the name of your partner on your solution. (It is not acceptable to hand in identical solutions.) In either case (whether the problem set is collaborative or not) you may discuss lecture and course-notes material with other students, as long as the discussion does not directly pertain to the problem set in the case the latter is non-collaborative.

External references will not be necessary to solve the problem sets. If you like, you may use books or articles other than the class notes or handouts, but, if you use such a source in a solution to a problem set, you are required to name the source.

In assigning final grades, the instructor reserves the right to be subjective. Of course the course score is a primary indicator. But also taken into account will be discussions with the instructor or TA, class participation, the extent to which help was received on problem sets from the instructor or TA, our perception of contribution in collaborative work as indicated by gaps in scores between collaborative and non-collaborative work, and the extent to which external sources were used. (And this list is not exhaustive.)

You are not allowed to use any material (particularly problem sets and their solutions) from previous years of this course. You are not allowed to use the Internet to find solutions to homework problems.

It will not be possible for me to say to what grade a certain score on a problem set corresponds, for, as you can see, the grade depends on the total course score together with other factors, and

the mapping of that to a grade is not determined until the end of the course. If you are concerned about how you are doing, however, you are encouraged to meet with me personally and I will try to give you some sense of where you stand.

Turn in neat, readable solutions. (Either handwritten or typeset.) If you have more than one sheet, they should be stapled together, not clipped or folded at the corner.

Cheating, including failure to abide by the above course rules, is taken very seriously. Academic dishonesty cases are prosecuted in conjunction with the Dean of Student Affairs and can result in probation or dismissal. Students have been caught cheating in graduate courses in this department in the past, and have been so prosecuted.

**Mathematical writing:** This course involves mathematical abstraction and proofs. Being able to deal with these is one of the important things to learn. In both problems sets and exams, you will be graded based on the correctness, clarity and accuracy of mathematical exposition. Make sure what you write makes sense. Define notation before you use it. Answers that “don’t make sense” will not get much credit. Your solutions should have a logical flow from beginning to end. Remember, you are graded on what you write, not on what you think you “meant.” So make sure you write what you mean. Write top to bottom, left to right on the page. Don’t scatter information all over. Be as concise as possible.

Mathematics is a language. Learn its grammar and semantics, and get used to using it correctly. Like any language, its goal is communication, and when properly used, it is a precise and unambiguous tool to this end. When you mis-use the language, you will not be understood, and you will lose points.

Write top to bottom, left to right on the page. Don’t scatter information all over. Be as concise as possible.

Read through whatever you write before turning it in. Try to make sure there is an argument with a clear flow. If your paper says lots of different things, you are *not* going to get points just because one of them is right; indeed, you will get *less* points for a jumble which sort of includes something right than for something clear even if not the entire answer.

For problem sets, first write a rough draft, then write a new, final draft to actually turn in. Think about it from the point of view of a grader: how are you making sure that person will understand?

Spend time on the writeup. Re-read it after it is written, trying to be in the mindset of someone who does not know what you are thinking, but only has your solution to look at. Try to make sure it can be understood by such a person.

The articles under *Mathematical and technical writing* available via <http://www-cse.ucsd.edu/users/mihir/education.html> provide more information about mathematical exposition. You are encouraged to look at them.