Each 6.1800 lecture will come with an outline. You can fill this in during lecture, after lecture, or not at all — it's entirely up to you how you use it. The goal of these outlines is to help you understand the main points that you should be taking away from each lecture. In some cases we will also include examples of things you should be able to do after each lecture.

*In the past, these outlines have proved to be an effective tool for studying for the exams. Note that the outlines are **not exhaustive**; there will be topics and nuances in lecture that aren't captured by the outline.*

Lecture 21: Intro to Security + Authentication
*Note: You do **not** need to understand the math behind hash functions; in 6.1800, we are interested in using them as a cryptographic primitive from which we can build more secure systems. However, if you want to read more details about a particular hash functions, here is a popular one.*

- What makes computer security difficult?
- What do we mean by our *policy*? Our *threat model*?
- What is the goal of authentication?
- Why is storing passwords in plaintext on a server a bad idea?
- Hash functions
    - What is a hash function?
    - What do we mean when we say a hash function is *deterministic*? *Collision-resistant*? *One-way*?
        - Why are these properties important? *(As we go through the different approaches to storing passwords, you should connect those approaches back to these properties to answer this question)*
    - There are not many functions that are hash functions, and the ones that exist are well-known. Why does this fact matter?
- Storing passwords
    - Suppose we store the password along with its hash on the server.
        - How does authentication work? I.e., if a user enters their password, how do we check that it's correct?
        - What's the primary way an attacker could thwart this approach?
    - Slow hashes
        - What do we mean by a "slow hash"?
        - Storing slow hashes instead of "normal" hashes improves the security of our system. Why?
        - What's the primary way an attacker could thwart this approach?
    - Salting (slow) hashes
        - What does it mean to "salt" a hash?
        - Why does storing salted hashes improve the security of our system?
        - Why is it safe to store the salt in plaintext on the server?
- Practical concerns
    - What problem(s) do session cookies and challenge-response protocols address?