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 23: Secure channels

Note: Just like the hash functions in Lecture 21, you do ***not*** *need to understand the math behind encrypt/decrypt, MAC, or sign/verify; in 6.1800, we are interested in using them as a cryptographic primitive from which we can build more secure systems.*

- What are examples of some things that an adversary in the network might do?
- Our policy for secure channels is to provide confidentiality and integrity. What do each of those things mean? How do they differ?
- How can we employ encryption to provide confidentiality?
- At a high level, why does encryption alone not also provide integrity?
- Secure channels
    - What is a replay attack? How does adding a sequence number prevent it?
    - What is a reflection attack? How does using different keys, and different sequence numbers, in each direction prevent it?
- What *is* "key exchange"? Why does how Alice and Bob exchange keys matter?
- How does Diffie-Hellman key exchange work?
    - *Example: given p and q, you should be able to perform a Diffie-Hellman key exchange between two parties.*
- How can an on-path attacker (Eve, in our examples) cause problems with Diffie-Hellman key exchange? (E.g., what exactly is Eve doing here, what is she able to determine after the key exchange is complete, etc.)
- How do we use signatures (in addition to encryption and MAC'ing) to provide secure channels?
- What is a certificate authority? A certificate? What problem(s) do certificates solve?

You do not *need to understand the details of the TLS handshake, although you should recognize familiar elements in it (sequence numbers, keys, etc.).*