Foreword and Syllabus

"Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve." Edgar Allan Poe, 1841

Cryptography - the art or science of "secret writing" - has been around for several millenia. For almost all that time, Edgar Allan Poe's quote above held true. Indeed, the history of cryptography is littered with the figurative corpses of cryptosystems believed secure and then broken, and sometimes with the actual corpses of those who have mistakenly placed their faith in these cryptosystems. Yet, something changed in the last few decades. New cryptosystems have been found that have not been broken despite being subjected to immense efforts involving both human ingenuity and computational power on a scale that completely dwarves the "crypto breakers" of Poe's time. Even more amazingly, these cryptosystems are not only seemingly unbreakable, but they also achieve this under much harsher conditions. Not only do today's attackers have more computational power but they also have more data to work with. In Poe's age, an attacker would be lucky if they got access to more than a few ciphertexts with known plaintexts. These days attackers might have massive amounts of data - terabytes or more - at their disposal. In fact, with *public key* encryption, an attacker can generate as many ciphertexts as they wish.

These new types of cryptosystems, both more secure and more versatile, have enabled many applications that in the past were not only impossible but in fact *unimaginable*. These include secure communication without sharing a secret, electronic voting without a trusted authority, anonymous digital cash, and many more. Cryptography now supplies crucial infrastructure without which much of the modern "communication economy" could not function.

This course is about the story of this cryptographic revolution. However, beyond the cool applications and the crucial importance of cryptography to our society, it contains also intellectual and mathematical beauty. To understand these often paradoxical notions of cryptography, you need to think differently, adapting the point of view of an attacker, and (as we will see) sometimes adapting the points of view of other hypothetical entities. More than anything, this course is about this cryptographic way of thinking. It may not be immediately applicable to protecting your credit card information or to building a secure system, but learning a new way of thinking is its own reward.

0.1 SYLLABUS

In this fast-paced course, I plan to start from the very basic notions of cryptography and by the end of the term reach some of the exciting advances that happened in the last few years such as the construction of *fully homomorphic encryption*, a notion that Brian Hayes called "one of the most amazing magic tricks in all of computer science", and *indistinguishability obfuscators* which are even more amazing. To achieve this, our focus will be on *ideas* rather than *implementations* and so we will present cryptographic notions in their pedagogically simplest form – the one that best illustrates the underlying concepts – rather than the one that is most efficient, widely deployed, or conforms to Internet standards. We will discuss some examples of practical systems and attacks, but only when these serve to illustrate a conceptual point.

Depending on time, I plan to cover the following notions:

- Part I: Introduction

  1. **How do we define security for encryption?** Arguably the most important step in breaking out of the "build-break-tweak" cycle that Poe's quote described has been the idea that we can have a *mathematically precise* definition of security, rather than relying on fuzzy notions, that allow us only to determine with certainty that a system is *broken* but never have a chance of *proving* that a system is *secure* .
  2. **Perfect security and its limitations:** Showing the possibility (and the limitations) of encryptions that are perfectly secure regardless of the attacker's computational resources.
  3. **Computational security:** Bypassing the above limitations by restricting to computationally efficient attackers. Proofs of security by reductions.

- Part II: Private Key Cryptography

  1. **Pseudorandom generators:** The basic building block of cryptography, which also provided a new twist on the age-old philosophical and scientific question of the nature of randomness.
  2. **Pseudorandom functions, permutations, block ciphers:** Block ciphers are the working horse of crypto.

3. **Authentication and active attacks:** *Authentication* turns out to be as crucial, if not more so, to security than *secrecy* and often a precondition to the latter. We'll talk about notions such as Message Authentication Codes and Chosen-Ciphertext-Attack secure encryption, as well as real-world examples why these notions are necessary.
4. **Hash functions and the "Random Oracle Model":** Hash functions are used everwhere in crypto, including for verifying integrity, entropy distillation, and many other cases.
5. **Building pseudorandom generators from one-way permutations (optional):** Justifying our "axiom" of pseudo-random generators by deriving it from a weaker assumption.

- Part III: Public key encryption

1. **Public key cryptography and the obfuscation paradigm:** How did Diffie, Hellman, Merkle, Ellis even dare to *imagine* the possibility of public key encryption?
2. **Constructing public key encryption: Factoring, discrete log, and lattice based systems:** We'll discuss several variants for constructing public key systems, including those that are widely deployed such as RSA, Diffie-Hellman, and the elliptic curve variants. We'll also discuss some variants of *lattice based cryptosystems* that have the advantage of not being broken by quantum computers and being more versatile. The former's weakness to quantum computers is the reason why the NSA has advised people to transition to lattice-based cryptosystems in the not too far future.
3. **Signature schemes:** These are the public key versions of authentication, though interestingly they are easier to construct in some sense than the latter.
4. **Active attacks for encryption:** Chosen ciphertext attacks for public key encryption.

- Part IV: Advanced notions

1. **Fully homomorphic encryption:** Computing on encrypted data.
2. **Multiparty secure computation:** An amazing construction that enables applications such as playing poker over the net without trusting the server, privacy preserving data mining, electronic auctions without a trusted auctioneer, and electronic elections without a trusted central authority.
3. **Zero knowledge proofs:** Prove a statement without revealing the reason to *why* its true.

4. **Quantum computing and cryptography:** Shor's algorithm to break RSA and friends. Quantum key distribution. On "quantum resistant" cryptography.
5. **Indistinguishability obfuscation:** Construction of indistinguishability obfuscators, the potential "master tool" for crypto.
6. **Practical protocols:** Techniques for constructing practical protocols for particular tasks as opposed to general (and often inefficient) feasibility proofs.
7. **Cryptocurrencies:** Hash chains and Merkle trees, proofs of work, achieving consensus on a ledger via "majority of cycles", smart contracts, achieving anonymity via zero knowledge proofs.

0.1.1 Prerequisites

The main prerequisite is the ability to read, write (and even enjoy!) mathematical proofs. In addition, familiarity with algorithms, basic probability theory and basic linear algebra will be helpful. We'll only use fairly basic concepts from all these areas: e.g. Oh-notation- e.g. $O(n)$ running time - from algorithms, notions such as events, random variables, expectation, from probability theory, and notions such as matrices, vectors, and eigenvectors. Mathematically mature students should be able to pick up the needed notions on their own. See the "mathematical background" handout for more details.

No programming knowledge is needed. If you're interested in the course but are not sure if you have sufficient background, or you have any other questions, please don't hesitate to contact me.

0.2 WHY IS CRYPTOGRAPHY HARD?

Cryptography is a hard topic. Over the course of history, many brilliant people have stumbled in it, and did not realize subtle attacks on their ciphers. Even today it is frustratingly easy to get crypto wrong, and often system security is compromised because developers used crypto schemes in the wrong, or at least suboptimal, way. Why is this topic (and this course) so hard? Some of the reasons include:

- To argue about the security of a cryptographic scheme, you have to think like an attacker. This requires a very different way of thinking than what we are used to when developing algorithms or systems, and arguing that they perform well.

- To get robust assurances of security you need to argue about *all possible attacks* . The only way I know to analyze this infinite set is via *mathematical proofs* . Moreover, these types of mathematical proofs tend to be rather different than the ones most mathematicians typically work with. Because the proof itself needs to take the viewpoint of the attacker, these often tend to be proofs by contradiction and involve several twists of logic that take some getting used to.

- As we'll see in this course, even *defining* security is a highly non-trivial task. Security definitions often get subtle and require quite a lot of creativity. For example, the way we model in general a statement such as "An attacker Eve does not get more information from observing a system above what she knew a-priori" is that we posit a "hypothetical alter ego" of Eve called Lilith who knows everything Eve knew a-priori but does not get to observe the actual interaction in the system. We then want to prove that anything that Eve learned could also have been learned by Lilith. If this sounds confusing, it is. But it is also fascinating, and leads to ways to argue mathematically about *knowledge* as well as beautiful generalizations of the notion of encryption and protecting communication into schemes for protecting *computation* .

If cryptography is so hard, is it really worth studying? After all, given this subtlety, a single course in cryptography is no guarantee of using (let alone inventing) crypto correctly. In my view, regardless of its immense and growing practical importance, cryptography is worth studying for its *intellectual* content. There are many areas of science where we achieve goals once considered to be science fiction. But cryptography is an area where current achievements are so fantastic that in the thousands of years of secret writing people did not even dare *imagine* them. Moreover, cryptography may be hard because it forces you to think differently, but it is also rewarding because it *teaches* you to think differently. And once you pass this initial hurdle, and develop a "cryptographer's mind", you might find that this point of view is useful in areas that seem to have nothing to do with crypto.