Why is Modern Cryptography part of a Complexity course?

Short answer:

Because Modern Cryptography exploits results from Complexity theory to guarantee the security of its schemes.
Why is Modern Cryptography part of a Complexity course?

It seems that we want to be able to say something of the form: a scheme is $(t,\varepsilon)$-secure if every attacker running in time at most $t$ succeeds in breaking the scheme with probability at most $\varepsilon$.

e.g. when the cryptographic key has length $n$, an attacker that runs in time $t$ can break the scheme with prob $\leq t/2^n$.

Using a typical key length of $n=128$ we can say that NO adversary running a super-computer for up to 200 years can break the scheme with prob $\geq 10^{-30}$.

Such an approach is difficult to analyze and does not scale well. We can do better…
Why is Modern Cryptography part of a Complexity course?

An better approach is the Asymptotic Approach, which is directly related to Complexity Theory. Note that the Asymptotic Approach is a mere generalization of the first case we considered.

In asymptotic approach we view the running time of the attacker and the his success probability as functions of some parameter \( n \) (which is called the security parameter and usually represents the length of the secret key).

We consider feasible or efficient any process that takes poly\((n)\) time and negligible any probability that is \( 1/\text{poly}(n) \).

The reasoning behind these principles are rooted in Complexity Theory.
A scheme is a secure if a every attacker that runs in polynomial time succeeds in breaking the scheme with only negligible probability.

E.g. an attacker that runs in $n^3$ minutes can succeed in breaking the scheme with probability $2^{40} \cdot 2^{-n} \leq 1/\text{poly}(n)$. For small values of $n$, say $n \leq 40$, the attacker can break the scheme with prob. 1 when running for up to 6 weeks.

But because of the asymptotic relation of the probability with the running time, we can gradually increase the value $n$ until the corresponding running time of a non-negligible success probability is extremely long.

Consider $n=500$, the attacker running for more than 200 years would break the scheme with prob. $2^{-500}$ which is incredibly small, so small we can neglect it.
Modern Cryptography and Complexity

\[ P=NP \implies \text{CRYPTO} \]

\[ P\neq NP \lor \]

\begin{itemize}
  \item A. No Crypto
  \item B. Secret Key Crypto
  \item C. Public Key Crypto
\end{itemize}

\begin{itemize}
  \item A. No One-Way Functions
  \item B. One-Way Functions & NO Public Key Crypto
  \item C. Public Key Crypto
\end{itemize}
Cryptography is concerned with the construction of schemes that can withstand malicious attempts to abuse the scheme. e.g. Alice (Sender) wants to communicate secretly with Bob (Receiver). How can their communication protocol guarantee the security of the conversion?
Cryptography is concerned with the construction of schemes that can withstand malicious attempts to abuse the scheme. e.g. Alice wants to communicate secretly with Bob. How can their communication protocol guarantee the security of the conversion?

A. Demand that is impossible for any attacker (that watches the communication channel) to learn any secrets.

B. Demand that is infeasible for any attacker (that watches the communication channel) to learn any secrets, unless with extremely low probability.
Cryptography is concerned with the construction of schemes that can withstand malicious attempts to abuse the scheme. e.g. Alice wants to communicate secretly with Bob. How can their communication protocol guarantee the security of the conversion?

A. Information Theoretic Security

B. Computational Security
Cryptography is concerned with the construction of schemes that can withstand malicious attempts to abuse the scheme. e.g. Alice wants to communicate secretly with Bob. How can their communication protocol guarantee the security of the conversion?

A. Shannon showed the first negative result and introduced the study of Cryptography.

B. The focus of Modern Cryptography for which, however, we need strong computational assumptions that imply \( \text{NP} \neq \text{P} \).
A scheme is secure if the enemy hasn’t managed to break it.

A scheme is secure if no efficient algorithm can ever* be constructed that will break it with non-negligible prob.

*Complexity theorists have strong reasons to believe that no such algorithms will ever be found, no matter how human ingenuity grows. These are the computational assumptions upon which Modern Cryptography is based.
Shannon’s Impossibility Result[’49]

In an information secure Encryption Scheme the private-key must be longer than the total entropy of the plain texts to be sent using that key.

The shared secret must be longer than the message to be sent using that secret.
Theorem: Consider a scheme that is information theoretic secure, where the message space $M$ is and the key space $K$. Then $|K| \geq |M|$

Proof. Information theoretic security means that for any distribution over $M$ and any $m \in M$ it holds that $Pr[M=m|C=c]=Pr[M=m]$.

Assume $|K|<|M|$ and let every message appear with the uniform probability, $Pr[M=m]=1/|M|$.

Let $M(c) \in M$ be the possible decryptions of ciphertext $c$. Clearly $|M(c)| \leq |K|$ (for each key $k$, we can get a new encryption of $m$).

Hence $|M(c)| \leq |K| \leq |M|$, which means that $\exists m' \notin M(c)$ with $Pr[M=m'] \neq 0$.

Hence we found $m'$ s.t. $Pr[M=m'|C=c]=0 \neq Pr[M=m']$, which contradicts the security condition.
Information Theoretic Security

- NO Encryption Scheme
- NO Public Key Cryptography
- NO Pseudorandom Generators
- NO for most Cryptographic Protocols
Modern Cryptography and Complexity

\[ P = NP \implies \text{CRYPTO} \]

\[ P \neq NP \land \]

\begin{itemize}
  \item A. NO Crypto
  \item B. Secret Key Crypto
  \item C. Public Key Crypto
\end{itemize}

\begin{itemize}
  \item A. No One-Way Functions
  \item B. One-Way Functions & NO Public Key Crypto
  \item C. Public Key Crypto
\end{itemize}
1 + 2 + 3 + ... + 99 + 100 = ?

(100 + 1) * 50
I wish I knew a way to humiliate Carl in front of the whole class.

…if only I could find a problem I could solve and he couldn’t no matter how hard he tried…
Prof. Grouse in a Computational Complexity Class
A 3SAT instance?

hmmm....

But is that good enough? Prof. Grouse can't solve 3SAT either…
What Prof. Grouse is looking for is a One-Way Function

\[ x \xrightarrow{\text{easy}} f(x) \xleftarrow{\text{HARD}} \]
Impagliazzo’s Possible Worlds [’95]

1. P=NP
2. P≠NP in some cases*
3. P≠NP and OWF do NOT exist
4. P≠NP and OWF exist, but NO PKC
5. P≠NP and PKC

*There exist hard problems but generating such a problem is a hard problem itself. The complexity of a problem is a function of the time it took to sample an instance of this hard problem.
1. P=NP  Grouse has no chance against Gauss. For any problem that the class can verify its solution, Gauss can find the solution fast.

2. P≠NP in some cases… Grouse is still disadvantaged. He has to work hard, maybe for years, to create a problem that can withstand Gauss’ solving abilities for more than a month.

3. P≠NP and OWF do not exist  Grouse is hopeless. There are hard problems, which Gauss cannot solve, but Grouse cannot solve those either.

4. P≠NP and OWF exist, but no PKC  Grouse finally succeeds. He can create a problem to which, he knows the solution, but Gauss cannot solve.

5. P≠NP and PKC  Grouse’s paradise! He can publicly communicate with his class so that in the end everyone knows the solution except from poor Gauss.
Impagliazzo’s Possible Worlds [’95]

1. \( P = NP \)
2. \( P \neq NP \) in some cases*…
3. \( P \neq NP \) and OWF do NOT exist
4. \( P \neq NP \) and OWF exist, but NO PKC (MiniCrypt)
5. \( P \neq NP \) and PKC (CryptoMania)

*There exist hard problems but generating such a problem is a hard problem itself. The complexity of a problem is a function of the time it took to sample an instance of this hard problem.
One-Way Function

[Open Problem]: Prove that such a function exists unconditionally.

Corollary: $P \neq NP$

Currently we can only conjecture that OWF exist, assuming that certain computational assumptions hold. Most computer scientist believe that OWF exist.
OWF $\Rightarrow$ Crypto (MiniCrypt)

Assuming OWF exists we can construct a lot of cryptographic protocols.

A. Secure Private-Key Encryption
B. Pseudorandom Generators
C. Digital Signatures
D. Non-trivial Zero-Knowledge Proofs
E. Many more

However OWF $\not\Rightarrow$ Public Key Cryptography (CryptoMania). It has been shown that PKC cannot be proven by black-box access to a OWF. For that we need even stronger assumptions*.

*This assumptions tend to be less well understood. However, one sufficient assumption is the existence of Trapdoor One-Way Functions. The RSA function is conjectured to be such a function. Another equivalent assumptions is the existence of a Key-Exchange protocol.
Conversely most cryptographic protocol require the existence of OWF functions.

A. Secure Private-Key Encryption [IL '89]

B. Pseudorandom Generators [L '87]

C. Digital Signatures [R '90]

D. Non-trivial Zero-Knowledge Proofs [OW '93]

E. Various Basic Protocols [IL '89]
OWF give rise to Cryptography.

OWF ↔ Crypto

P ≠ NP
OWF give rise to more Complexity theoretic constructions with scope outside Cryptography.

\[
\text{OWF} \iff \text{Crypto} \iff \text{Pseudo-randomness}
\]

\[
\text{P} \neq \text{NP} \quad \text{Derandomization}^* \\
\]

*It is possible to achieve weaker notions of Pseudo-randomness, and hence Derandomization without assuming the existence of OWF.
Before

A string is random if it passes a certain statistical test, that is taken as the norm.

After

A string is pseudorandom if no efficient algorithm will ever* be able to distinguish it from a truly random string.

*Complexity theorists have strong reasons to believe that no such algorithms will ever be found, no matter how human ingenuity grows. These are the computational assumptions upon which Modern Cryptography is based.
OWF give rise to more Complexity theoretic constructions with scope outside Cryptography.

\[ \text{OWF} \Leftrightarrow \text{Crypto} \Leftrightarrow \text{Pseudo-randomness} \]
\[ \Leftrightarrow \text{Probabilistic Proofs} \]

\( \text{P} \neq \text{NP} \)  

Derandomization
So let’s study One-Way Functions

i. See the definition of OWFs and at equivalent notions

ii. Look at a candidate function that is believed to be one-way.
One-Way Functions

A function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ is called one-way if

- easy direction: there is an efficient algorithm which on input $x$ outputs $f(x)$.

- hard direction: given $f(x)$, where $x$ is uniformly selected, it is infeasible to find with non-negligible probability, a preimage of $f(x)$. That is, any feasible algorithm which tries to invert $f$ may succeed only with negligible probability taken over the choices of $x$ and the algorithm’s internal randomness.
Hard Problem Generator

Consider a language $L \in \text{NP}$ (a language where membership can be efficiently verified). Then $G$ is a Hard Problem Generator if it can efficiently find $(a_1, a_2) \in L$ and at the same time $f(a) = \{b : (a, b) \in L\}$ is a OWF. This means that given the first component $a_1$ it's easy to find the second $a_2$ s.t. $(a_1, a_2) \in L$, but given the second component $a_2$ it's hard to find the first $a_1$ s.t. $(a_1, a_2) \in L$. 
Pseudorandom Generator

Let \( \ell : \mathcal{N} \rightarrow \mathcal{N} \) be so that \( \ell(n) > n \) \( \forall n \). A pseudorandom generator, with stretch function \( \ell \), is an efficient (deterministic) algorithm which on input a random \( n \)-bit seed outputs a \( \ell(n) \)-bit sequence which is computationally indistinguishable from a uniformly chosen \( \ell(n) \)-bit sequence.

If \( X_n \) and \( Y_n \) are two probability ensemble of equal length \( n \), we say that \( X \) and \( Y \) are computationally indistinguishable if for every feasible algorithm \( A \) the difference

\[
| \Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1] |
\]

is negligible function in \( n \).
Candidate One-Way Functions

The following functions are believed to be one-way and are heavily used in modern cryptographic protocols:

- **Discrete Logarithm**: Let $G=\langle g \rangle$ be a cyclic group of size $p$. Then $f(a)=g^a$ is easily done in time $O(|p|^3)$. However, as of today there is no known poly-time algorithm that can calculate $f^{-1}=\log(x)$ where $x \in G$.

- **Factoring**: Choose two large primes $p$ & $q$ of equal length and output $N=pq$ which can be done efficiently. However, no known poly-time algorithm exists that can factor a given $N$.

- **Rabin Function**: If $N=pq$ is as above, it is easy to calculate $f(x)=x^2 \mod N$, but no known poly-time algorithm exists that can find a square root of $y \mod N$. 

LE FIN