

# Cryptography

Mathematical Foundations

Public Key Cryptography

Private Key Cryptography

Hash Functions

Digital Signatures

Elliptic Curve Cryptography

Post-Quantum Cryptography

Blockchain Cryptography

Quantum Cryptography

Homomorphic Encryption

Secure Multi-Party Computation

Zero-Knowledge Proofs

Physical Cryptography

Biometric Cryptography

Cloud Cryptography

Future Trends

# Cryptography

Throughout most of this history:

**cryptography** = “secret writing”:

“Scramble” (**encrypt**) text such that it is hopefully unreadable by anyone except the intended receiver that can **decrypt** it.

**Recurring theme:** (until 1970's)

- Secret code invented
- Typically claimed “unbreakable” by inventor
- Used by spies, ambassadors, kings, generals for crucial tasks.
- Broken by enemy using **cryptanalysis**.

# Examples

**1587:** Ciphers from Mary of Scots plotting assassination of queen Elizabeth broken; used as evidence to convict her of treason.

**1860's (civil war):** Confederacy used good cipher (Vigenere) in a bad way. Messages routinely broken by team of young union cryptanalysts; in particular leading to a Manhattan manufacturer of plates for printing rebel currency.

**1878:** New York Tribune decodes telegram proving Democrats' attempt to buy an electoral vote in presidential election for \$10K.

**1914:** With aid of partial info from sunken German ships, British intelligence broke all German codes.  
Cracked telegram of German plan to form alliance with Mexico and conquer back territory from U.S. As a result, U.S. joined WWI.


**WWII:** Cryptanalysis used by both sides. Polish & British cryptanalysts break supposedly unbreakable **Enigma** cipher using mix of ingenuity, German negligence, and mechanical computation. Churchill credits cryptanalysts with winning the war.

# This Course

## What you'll learn:

- Foundations and principles of the science
- Basic primitives and components.
- Definitions and proofs of security
- High-level applications
- Critical view of security suggestions and products

## What you will *not* learn:

- Buzzwords
- The most efficient and practical versions of components.
- Designing secure systems\* 
- “Hacking” – breaking into systems.
- Viruses, worms, Windows/Unix bugs, buffer overflow etc..
- Everything important about crypto

Will help you avoid designing  
insecure systems.

# This Course

## Modern (post 1970's) cryptography:

### Provable security – breaking the “invent-break-tweak” cycle

- ✎ Perfect security (Shannon) and its limitations
- ✎ Computational security
- ✎ Pseudorandom generators, one way functions

### Beyond encryption – public-key crypto and other wonderful creatures

- ✎ Public-key encryption based on factoring and RSA problem
- ✎ Digital signatures, hash functions
- ✎ Zero-knowledge proofs
- ✎ Active security – Chosen-Ciphertext Attack

### Advanced topics (won't have time for all ☹ )

- ✎ The SSL Protocol and attacks on it
- ✎ Secret Sharing
- ✎ Multi-party secure computation
- ✎ Quantum cryptography
- ✎ Password-based key-exchange, broadcast encryption, obfuscation

# Prerequisites

## Required:

1. **Ability to read and write mathematical proofs and definitions.**

---
2. **Familiarity with algorithms – proving correctness and analyzing running time ( $O$  notation).**
3. **Familiarity with basic probability theory (random variables, expectations – see handout).**

## Helpful but not necessary:

**Complexity.** NP-Completeness, reductions, P, BPP,  $P_{/poly}$

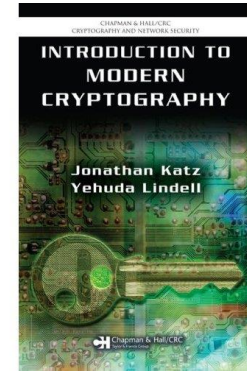
**Probabilistic Algorithms.** Primality testing, hashing,

**Number theory.** Modular arithmetic, prime numbers

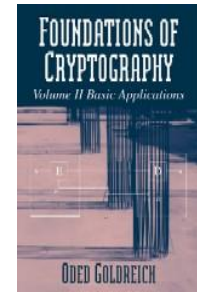
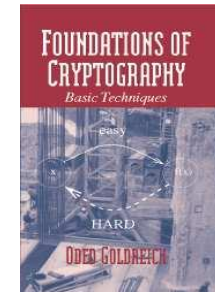
**See web-site for links and resources.**

# Reading

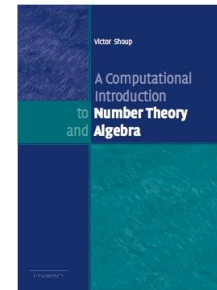
**Introduction to Modern Cryptography / Katz & Lindell**  
Main text used, though not 100% followed



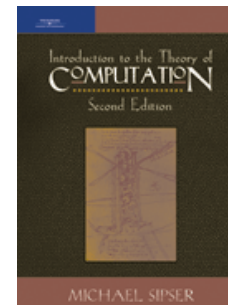
**Foundations of Cryptography / Goldreich.**  
Graduate-level text, will be sometimes used.



**Computational Intro to Algebra and Number Theory / Shoup.**  
(Available also on the web)



**Introduction to the Theory of Computation / Sipser.**  
For complexity background



**Lecture notes on web:** (links on web site)

# Requirements

**Exercises:** Weekly from Thursday till Thursday before class.

Submit by email / mailbox / in class to Rajsekar.

**Flexibility:** 4 late days, bonus questions

**Take home final.**

**Final grade:** 50% homework, 50% final

**Honor code.** Collaboration on homework with other students **encouraged**.  
However, write alone and give credit.

Work on final alone and as directed.



# This course is hard

- Challenging weekly exercises
- Emphasis on mathematical *proofs*
- Counterintuitive concepts.
- Extensive use of quantifiers/probability

## But it's not my fault :)

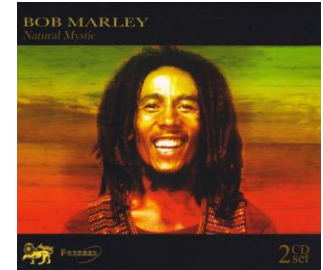
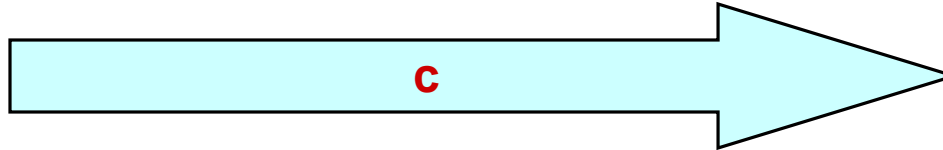
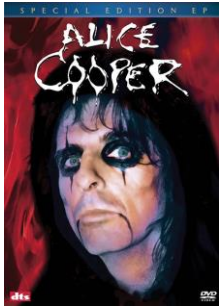
- Good coverage of crypto (meat, vegetables and desert) takes a year.
- Simulation / experimentation can't be used to show security.
- Need to acquire “crypto-intuition”
- Quantifiers, proofs by contradiction, reductions, probability are inherent.

## Mitigating hardness

- Avoid excessive exercises – only questions that teach you something.
- Try best to explain intuition behind proofs
- Me and Rajsekar available for any questions and clarifications.

# Encryption Schemes

**Alice** wants to send **Bob** a secret message.



$$c = E(m, k)$$

$$m' = D(c, k)$$

They agree in advance on 3 components:

- **Encryption** algorithm: **E**
- **Decryption** algorithm: **D**
- **Secret key**: **k**

To **encrypt** plaintext **m**, Alice sends **c = E(m, k)** to Bob.

To **decrypt** a cyphertext **c**, Bob computes **m' = D(c, k)**.

- A scheme is **valid** if **m'=m**
- Intuitively, a scheme is **secure** if eavesdropper can not learn **m** from **c**.

# Example 1: Caesar's Cipher

**Key:**  $k$  = no. between 0 and 25.

**Encryption:** encode the  $i^{\text{th}}$  letter as the  $(i+k)^{\text{th}}$  letter.  
(working mod 26:  $z+1=a$  )

**Decryption:** decode the  $j^{\text{th}}$  letter to the  $(j-k)^{\text{th}}$  letter.

Plain-text:	S	E	N	D	R	E	I	N	F	O	R	C	E	M	E	N	T
Key:	2																
Cipher-text:	U	G	P	F	T	F	K	P	H	Q	T	E	G	O	G	P	V

**Problem:** only 26 possibilities for key – can be broken in short time.

**Kerchoff's Principle (1883):** System should be secure even if algorithms are known, as long as key is secret.

In other words: “**security through obscurity**” does not work.

# Example 2: Substitution Cipher

**Key:**  $k$  = table mapping each letter to another letter

A	B	C	Z
U	R	B	E

**Encryption and decryption:** letter by letter according to table.

**# of possible keys:**  $26!$  ( = 403,291,461,126,605,635,584,000,000 )

However – substitution cipher is still insecure!

**Key observation:** can recover plaintext using statistics on letter frequencies.

HereUpOnLeGrandAroseWithAGraveAndStatelyAirAndBrought  
LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFEMVEWHKVSTYLX  
MeTheBeetleFromAGlassCaseInWhichItWasEnclosedItWasABe  
ZIXLIKIIXPIJVSZEYPERRGERIMWQLMGLMXQERIWGPSRIHMXQEREKI

**I** – most common letter

**I=e**    **L=h**    **X=t**

**LI** – most common pair

**V=r**    **E=a**    **Y=g**

**XLI** – most common triple

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **statefull** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $n^{\text{th}}$  letter encoded as Caesar w/ key= $k_{(n \bmod m)}$  :  $i \rightarrow I + k_{(n \bmod m)} \pmod{26}$

**Decryption:** In the  $n^{\text{th}}$  letter decoded as Caesar w/ key= $k_2$  :  $i \rightarrow I + k_2 \pmod{26}$

**Important Property:** Can no longer break using letter frequencies alone.  
 $m^{\text{th}}$  letter encoded as Caesar w/ key= $k_m$  :  $i \rightarrow I + k_m \pmod{26}$

‘e’ will be mapped to ‘e’+ $k_1$ , ‘e’+ $k_2$ , ..., ‘e’+ $k_m$  according to location.  
 $m+1^{\text{th}}$  letter encoded as Caesar w/ key= $k_1$  :  $i \rightarrow I + k_1 \pmod{26}$

Considered “unbreakable” for 300 years (broken by Babbage, Kasiski 1850’s)

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **statefull** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $n^{\text{th}}$  letter encoded w/ key= $k_{(n \bmod m)}$  :  $i \rightarrow i + k_{(n \bmod m)} \pmod{26}$

**Decryption:** In the natural way

**Breaking Vigenere:**

LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFEMVEWHKV

**Step 1:** Guess the length of the key  $m$

**Step 2:** Group together positions  $\{1, m+1, 2m+1, 3m+1, \dots\}$

$\{2, m+2, 2m+2, 3m+2, \dots\}$

...

$\{m-1, 2m+m-1, 3m+m-1, \dots\}$

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **statefull** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $n^{\text{th}}$  letter encoded w/  $\text{key} = k_{(n \bmod m)} : i \rightarrow i + k_{(n \bmod m)} \pmod{26}$

**Decryption:** In the natural way

**Breaking Vigenere:**

LIVITC  
SWPIYV  
EWHEVS  
RIQMXL  
EYVEOI  
EWHRXE  
XIPFEM  
VEWHKV

**Step 1:** Guess the length of the key  $m$

**Step 2:** Group together positions  $1, m+1, 2m+1, 3m+1, \dots$   
 $\{2, m+2, 2m+2, 3m+2, \dots\}$   
 $\dots$   
 $\{m-1, 2m+m-1, 3m+m-1, \dots\}$

**Step 3:** Frequency-analyze each group independently.

# Example 4 - The Enigma

A mechanical statefull cipher.

Used by Germany in WWII for top-secret communication.

**Roughly:** composition of 3-5 substitution ciphers implemented by wiring.

Wiring on rotors moving in different schedules, making cipher **statefull**



**Key:**

- 1) Wiring of machine (changed infrequently)
- 2) Daily key from code books
- 3) New operator-chosen key for each message

**Tools used by Poles & British to break Enigma:**

- 1) Mathematical analysis combined w/ mechanical computers
- 2) Captured machines and code-books
- 3) German operators negligence
- 4) **Known plaintext attacks** (greetings, weather reports)
- 5) **Chosen plaintext attacks**



# Post 1970's Crypto

Two major developments:

## 1) Provably secure cryptography

Encryptions w/ mathematical proof that are unbreakable\*

\* Currently use conjectures/axioms,  
however defeated all cryptanalysis effort so far.

## 2) Cryptography beyond “secret writing”

Public-key encryptions

Digital signatures

Zero-knowledge proofs

Anonymous electronic elections

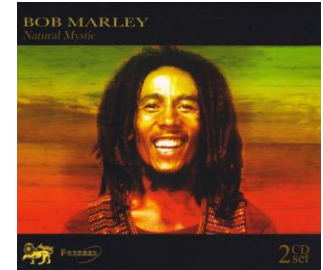
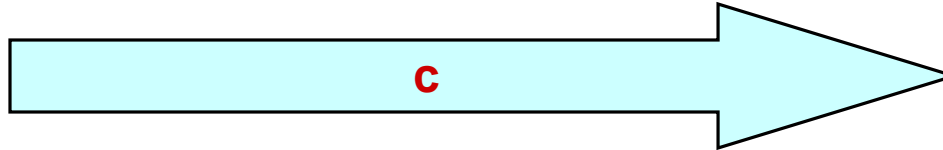
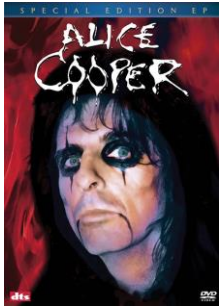
Privacy-preserving data mining

e-cash

...

# Review of Encryption Schemes

**Alice** wants to send **Bob** a secret message.



$$c = E(m, k)$$

$$m' = D(c, k)$$

- Encryption algorithm: **E**
- Decryption algorithm: **D**
- Secret key: **k**

To **encrypt** **m**, Alice sends  **$c = E(m, k)$**  to Bob.  
To **decrypt** **c**, Bob computes  **$m' = D(c, k)$** .

**Q:** Can Bob send Alice the secret key over the net?

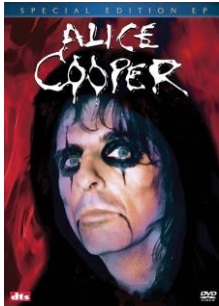
**A:** Of course not!! Eve could decrypt **c**!

**Q:** What if Bob could send Alice a “crippled key”

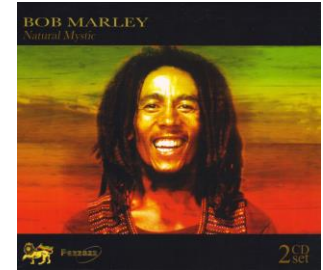
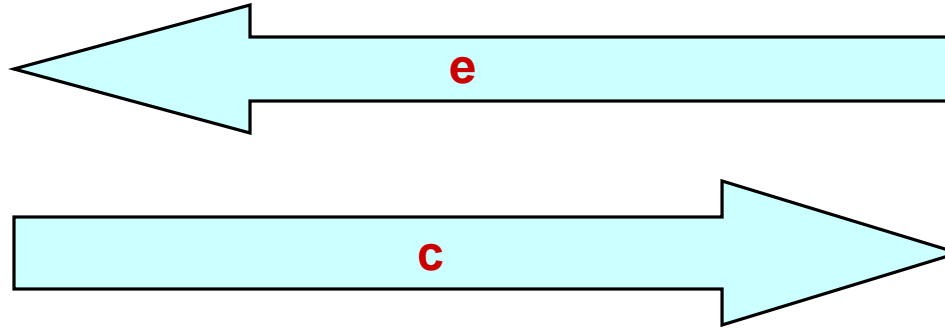
useful only for **encryption** but no help for **decryption**

# Public Key Cryptography [DH76,RSA77]

Alice wants to send Bob a secret message.



$c = E(m, e)$



choose  $d, e$   
 $m' = D(c, d)$

- Encryption algorithm:  $E$
- Decryption algorithm:  $D$
- Key: Bob chooses two keys:  Secret key  $d$  for decrypting messages.  
 Public key  $e$  for encrypting messages.

To encrypt  $m$ , Alice sends  $c$ .

Should be safe to send  $e$  “in the clear”.

Even if Eve knows the key  $e$ !

- A scheme is valid if  $m' = m$
- Intuitively, a scheme is secure if eavesdropper can not learn  $m$  from  $c$ .

# Other Crypto Wonders

**Digital Signatures.** Electronically sign documents in unforgeable way.

**Zero-knowledge proofs.** Alice proves to Bob that she earns  $< \$50K$  without Bob learning her income.

**Privacy-preserving data mining.** Bob holds DB. Alice gets answer to one query, without Bob knowing what she asked.

**Playing poker over the net.** Alice, Bob, Carol and David can play poker over the net without trusting each other or any central server.

**Distributed systems.** Distribute sensitive data to 7 servers s.t. as long as  $< 3$  are broken, no harm to security occurs.



**Electronic auctions.** Can run auctions s.t. *no one* (even not seller) learns anything other than winning party and bid.

# Cryptography & Security

**Prev slides:** Have provably secure algorithm for every crypto task imaginable.

**Q:** How come nothing is secure?

**A1:** Not all of these are used or used correctly:

- Strange tendency to use “home-brewed” cryptosystems.
- Combining secure primitives in insecure way
- Misunderstanding properties of crypto components.
- Strict efficiency requirements for crypto/security:
  -  The cost is **visible** but benefit **invisible**.
  -  Many provably secure algs not efficient enough
- Easy to get implementation wrong – many subtleties
- Compatibility issues, legacy systems,

# Cryptography & Security

**Prev slides:** Have provably secure algorithm for every crypto task imaginable.

**Q:** How come nothing is secure?

**A2:** Cryptography is only **part** of designing secure systems

- Chain is only as strong as weakest link.
- A “dormant bug” is often a security hole.
- Many subtle issues (e.g., caching & virtual memory, side channel attacks)
- Security is hard to “modularize”  
(hard to add to existing system, changes in system features can have unexpected consequences)
- Human element
- Key storage and protection issues.