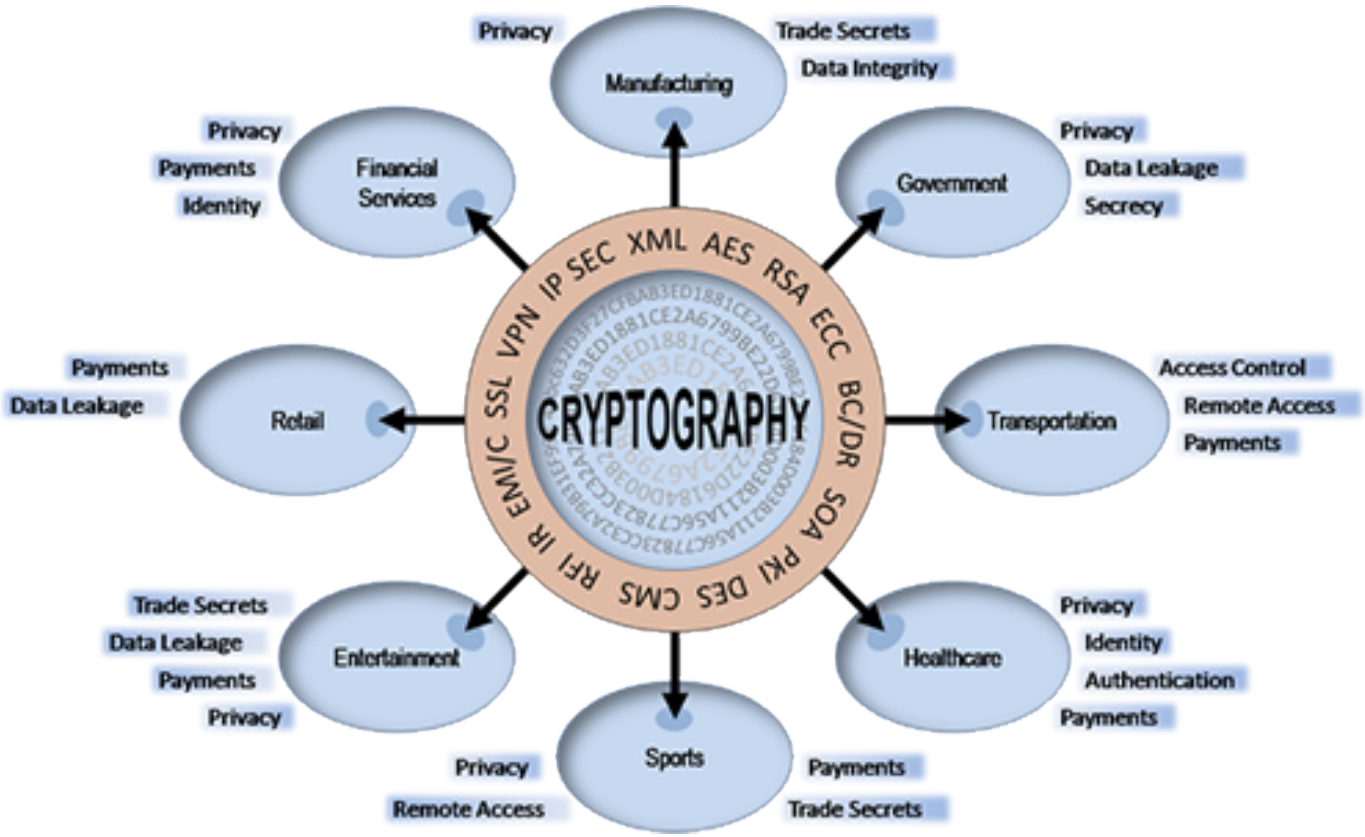
CS4236 Cryptography Theory and Practice Topic 1 - Introduction

Hugh Anderson

National University of Singapore
School of Computing

August, 2022

Crypto...


CRYPTOGRAPHY
XML AES RSA ECC BC/DR SOA PKI DES CMS RFI IR EMV/C SSL VPN IP SEC
Manufacturing
Privacy
Trade Secrets
Data Integrity
Financial Services
Privacy
Payments
Identity
Government
Privacy
Data Leakage
Secrecy
Retail
Payments
Data Leakage
Transportation
Access Control
Remote Access
Payments
Entertainment
Trade Secrets
Data Leakage
Payments
Privacy
Healthcare
Privacy
Identity
Authentication
Payments
Sports
Privacy
Remote Access
Payments
Trade Secrets

Outline

Hugh's coordinates

COM2 #03-24
6516-4262
hugh@comp...

A/P Hugh Anderson (PhD from NUS)

My research interests center on formal methods for analysis.
I have an open-door policy. Please call me Hugh, and contact me at any time if you have any questions...

The difference between CS2107, CS3235, CS4236?

Mainly that CS4236 has a more formal treatment of the notion of "Security": computational constraints (on the attacker's capability), types of information the attacker has (various form of oracles), and goals (indistinguishability). In addition, there are some proofs; some selected construction details (e.g. how to extend fixed size to arbitrary length), and some selected attacks. There are more primitives (secret sharing, commitment schemes), and crypto applications.

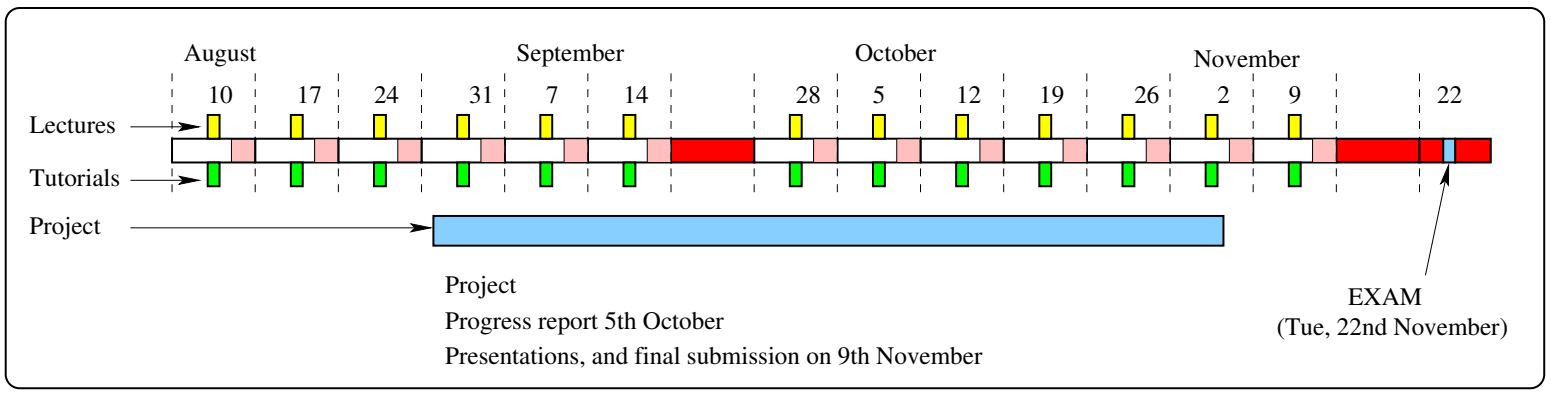
More admin: Assessment and timetable

Assessment: CA 60%, Final exam 40%

- 2 individual writing assignments (Could be mathematical)
- 1 group presentation (Design of a system. Similar for every group)

May vary the assessment in some ways, but I will let you know as we go.

13 weeks...



The due dates for the individual assignments to be announced, and the projects announced in 3rd or 4th week.

Resources and Project

For the course

The textbook is: Introduction to Modern Cryptography, 2nd Ed. Jonathan Katz, Yehuda Lindell.
But also you may make use of

- The 3rd edition, although definition numbers may change...
- “Cryptography and Network Security, Principles and Practice”, by William Stallings.
- Pfleeger, Stinson,

There may be directed readings - all available easily on the Internet, and of course Canvas (Luminus?). For this lecture, please read chapter 1 of the textbook.

Project

Later in the course, we will have a project, where you can choose some topic of interest to you, and use the mathematical structures from this course. I will try to give feedback on this.

The course topics

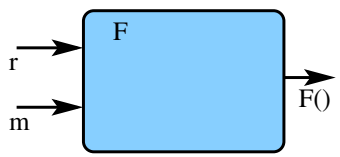
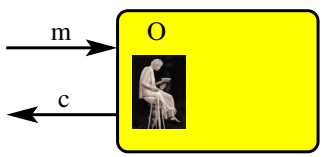
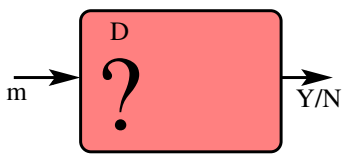
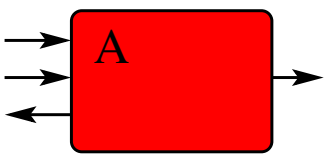
(Approximately) by week:

1	Introduction - administration, context for crypto	Ch 1
2	Perfect Secrecy	Ch 2
3	Computational hardness	Ch 3.1, 3.2
4	Reduction proof	Ch 3.3
5	CPA-indistinguishability, Block ciphers	Ch 3.5, 3.6, 3.7
6	MAC (Information theoretic secure)	Ch 4.1, 4.6
7	Hash, Attack	Ch 5.1, 5.2, 5.4
8	Differential analysis	Ch 6.1, 6.2
9	Number theory	Ch 6
10	RSA	Ch 11.5
11	Key exchange, ElGamal	Ch 11.4
12	Secret sharing, MPC	Ch 13.3
13	...	

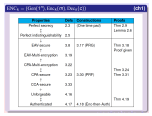
CS4236 concerns

Very different from CS2107, CS3235...

Jargon/Examples	**Proof**	**Representations**	**Data**
Adversary,CPA	Detailed, skeletons	1^n, $\{0,1\}^n$	0110101110...
Attacks	...suppose not ...	$a \times b \rightarrow c$	
Systems	Proof by construction		

Functions	**Oracles**	**Distinguishers**	**Adversaries**
			

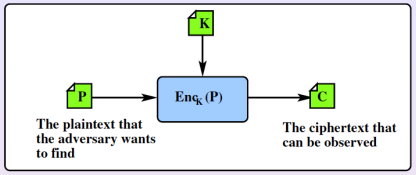
Constructions	**Game/Experiments**	**System/Schemes Π**	**Properties**
ECC, IBE, RSA	$\text{Name}^{\text{context}}_{\mathcal{A},\Pi}(\text{param})$	(Enc, Dec, Gen)	CCA-secure
MPC, OAEP	($\mathcal{A}$ is adversary)	Gen, Mac, Vrfy)	Unforgeable

Randomness	**Probability/Negligable**	**Complexity**	**Math Ecosystem**
true: r, $f(\cdot)$	Bayes/distributions	$\mathcal{O}(n)$	
PRG: $G()$	...there is a negl s.t.	PPT, exp	

Math building blocks 1

Probability theory

Probability theory $\Pr[E_1|E_2] = \frac{\Pr[E_2|E_1]\times\Pr[E_1]}{\Pr[E_2]}$



Relevance to crypto

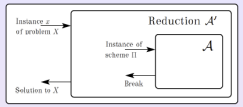
In the context of cryptography, one can imagine throwing two independent dice. One of them is the message P, and the other the key K. Next, a ciphertext is computed from the plaintext and key. Since the key and plaintext are randomly chosen, the ciphertext C is thus also random.

Now, an adversary has observed the ciphertext and thus knows that C=3. The adversary wants to know what are the chances that the plaintext is 5, given that the ciphertext is 3, that is, the probability

$$\Pr[P = 5|C = 3]$$

Proof styles

Proof styles



Outline of reduction proofs (p65)

Proof that some new construction Π is secure if some underlying cryptographic primitive X is secure. We often have this situation - for example Diffie-Hellman and factoring large numbers. We start with "Suppose not".

Assume that X is secure but Π is insecure, so this means that a PPT adversary $\mathcal{A}$ attacks Π with success probability $\epsilon(n)$, not $\mathrm{negl}(n)$.

- Make a PPT algorithm $\mathcal{A}'$ that tries to solve instance x of X using $\mathcal{A}$.
- If $\mathcal{A}$ succeeds in breaking Π, $\mathcal{A}'$ solves x with probability $\frac{1}{p(n)}$.
- $\mathcal{A}$ and $\mathcal{A}'$ solve X with probability $\frac{\epsilon(n)}{p(n)}$. $\mathcal{A}$ and $\mathcal{A}'$ are both PPT.

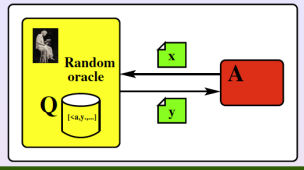
X is solved by PPT $\mathcal{A}'$ with non-negligable probability. A contradiction, so

$$X \text{ is secure} \longrightarrow \Pi \text{ is secure}$$

■

The random oracle

The random oracle (for hashes)



An idealised model, for use in proofs

In the definition of the PRF before, we saw similar oracles. The "random oracle" is an alternative way, a methodology, for formulating hash ideas. It makes proofs (sometimes) a little simpler. However, a random oracle is not a realizable, actual thing!

It is an idealized model, and so any proofs based on an assumption that (say) the underlying hash is a random oracle are theoretically meaningless. We really should not be saying

...xxx is secure, assuming $\mathcal{H}$ is a random oracle.

Statements that are made without assuming a random oracle are called "*Standard model*". (e.g. xxx is secure under standard model).

Euler and Fermat

Euler and Fermat

$\mathbb{Z}_p$ $\mathbb{Z}_N$

Euler's theorem: (composites)

If N is any positive integer and a is any positive integer less than N with no divisors in common with N, then

$$a^{\phi(N)} \bmod N = 1$$

where $\phi(N)$ is the *Euler phi (totient) function*:

$$\phi(N) = N(1 - 1/p_1)\ldots(1 - 1/p_m)$$

and $p_1, \ldots, p_m$ are all the prime numbers that divide evenly into N.

If N is a prime, then using the formula, we have

$$\phi(N) = N(1 - \frac{1}{N}) = N(\frac{N-1}{N}) = N - 1$$

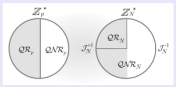
We see that Fermat's result is a special case of Euler's:

$$a^{\phi(N)} \bmod N = a^{N-1} \bmod N = 1$$

Math building blocks 2

Legendre and Jacobi

Identifying quadratic residues


The Legendre and Jacobi "symbol" notation

Legendre: *identifies* the QR in $\mathbb{Z}_p$; a function of a and p: $(\mathbb{Z}_p)$

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod p = \begin{cases} 1 & \text{if } a \text{ is a QR} \\ -1 & \text{if } a \text{ is not a QR} \\ 0 & \text{if } a = 0 \mod p \end{cases}$$

Jacobi: is a generalization of Legendre, extending it to $\mathbb{Z}_N$ (composites). $(\mathbb{Z}_N)$ Easy to compute given it's prime factorization. For example, given $N = p \times q$ with $p \neq q$, then

$$\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$$

Note that there is an algorithm that, given a and N, can find the Jacobi symbol without knowing the factorization of N. From the Jacobi symbol, we can't tell whether a is a QR, but no entry with -1 is a quadratic residue.

Hard and easy

Hard and easy (Root finding, Dlog,...)

The difficult problem in RSA: find e^{th} root in $\mathbb{Z}_N^*$

Given N, e, y, find x s.t. $x^e \mod N = y$.

Root finding can occur in any group, and there are efficient algorithms when N is prime. The hardness of RSA requires N to be a composite and thus relates to factorization, which only makes sense in a ring with two operators $\times, +$. It is not easy to generalize RSA encryption to other algebraic groups or rings.

The difficult problem in other systems: solve discrete log

Given N, x, y, find e s.t. $x^e \mod N = y$.

Such problems also naturally occur in any group. It turns out that there many groups whereby these problems seem to be very difficult, and at the same time, with certain useful properties (e.g. bilinear maps). This flexibility makes it attractive to design crypto systems based on DL, DDH or CDH. (there are many variants, e.g. knowledge of exponent, generalized DDH, etc, and many choices of groups, e.g Elliptic Curve, bilinear map, etc).

Cyclic groups

Cyclic groups

a	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
2	2	4	8	5	10	9	7	3	6	1
3	3	9	5	4	1	3	9	5	4	1
4	4	5	9	3	1	4	5	9	3	1
5	5	3	4	9	1	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1
9	9	4	3	5	1	9	4	3	5	1
10	10	1	10	1	10	1	10	1	10	1

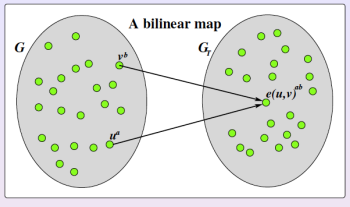
Cyclic groups and subgroups

All elements in $\mathbb{Z}_{11}^*$ can be obtained by successive multiplication by 6. So this is a cyclic group and can be written as $\langle 6 \rangle$. 6 is called a generator.

In general, when p is a prime, for any g in $\mathbb{Z}_p^*$, $\langle g \rangle$ forms a cyclic group or subgroup of $\mathbb{Z}_p^*$, . The order of this group is the number of elements in it, and so for $\langle 6 \rangle = \mathbb{Z}_{11}^*$, we have the order $|\langle 6 \rangle| = 10$. By contrast, $\langle 3 \rangle$ forms a cyclic subgroup of $\mathbb{Z}_{11}^*$ of order 5.

Bilinear maps

Bilinear maps (or pairings)



Pairings ...

A (symmetric style) bilinear map is a pairing between two elements of a group to a second (same order) group: $G \times G \to G_T$. There are also (asymmetric) pairings like $G_1 \times G_2 \to G_T$.

Such a symmetric mapping (if it can be efficiently computed) can be used to solve DDH: $z = ab$ iff $e(g, g^z) = e(g^a, g^b)$. Originally this was the focus of bilinear maps in crypto, and this is why CDH is considered *harder* than DDH.

The other thing is that it can allow you to reduce the discrete log problem in G to a (perhaps simpler) discrete log problem in G_T: $e(g, g^a) = e(g, g)^a$.

The math "ecosystem" in this course

Math ecosystem/framework elements:

- **Systems/schemes** which are the basic elements of the crypto world.
- **Security properties** over schemes defined by **games/experiments**.
- **Constructions**, which are instances of schemes, and
- **Proofs**, to show relations over properties and constructions.

The 9 schemes (with properties) are...

ENC_k:	Symmetric encryption - EAV, CPA, CCA, unforgeability
MAC_k:	Existential unforgeability, strongly-secure
HASH:	Collision resistance
COM:	Commitment - Binding+hiding (together secure)
ENC_{p_k}:	Asymmetric encryption - DDH, Factor, RSA, EAV, CPA, CCA
IBE_{p_k}:	Boneh/Franklin (construction only)
SIG_{p_k}:	Existential unforgeability
KEM_{p_k}:	KE-eav, CPA, CCA
SS:	Shamir, Feldman (constructions only)

Example: Symmetric encryption scheme

$\Pi = (\mathrm{Gen}(1^n), \mathrm{Enc}_k(m), \mathrm{Dec}_k(c))$ **(ch1)**

Properties	Defs	Constructions	Proofs
Perfect secrecy	2.3	(One time pad)	Thm 2.9
↕			Lemma 2.6
Perfect indistinguishability	2.5		
↓			
EAV-secure	3.8	3.17 (PRG)	Thm 3.18
↑			Proof given
EAV-Multi-encryption	3.19		
↑			
CPA-Multi-encryption	3.22		
↕			Thm 3.24
CPA-secure	3.23	3.30 (PRF)	Thm 3.31
↑			
CCA-secure	3.33		
+			
Unforgeable	4.16		
↕			Thm 4.19
Authenticated	4.17	4.18 (Enc-then-Auth)	

Examples: MAC and HASH schemes

$$\mathrm{MAC}_k = (\mathrm{Gen}(1^n), \mathrm{Mac}_k(m), \mathrm{Vrfy}_k(m, t)) \quad \textbf{(4.1)}$$

Properties	Defs	Constructions	Proofs
Unforgeable	4.2		
↑			Discussed
Strong	4.3	4.5 (Using PRF)	Thm 4.5
		4.7 (Variable length)	Thm 4.8
		4.11 (CBC-MAC)	Thm 4.12
		4.11(a) (Variable CBC-MAC)	

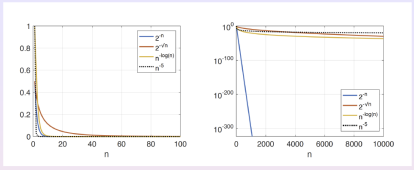
$$\mathrm{HASH} = (\mathrm{Gen}(1^n), \mathcal{H}^s(x)) \quad \textbf{(5.1)}$$

Properties	Defs	Constructions	Proofs
CollisionResistant	5.2	5.3 (Long message)	

Functions, structures, algorithms investigated

Negligable functions

Negligable functions



Definition 3.4 - negligable function definition

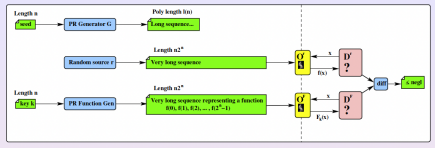
A non-negative function f from natural to real numbers $f : [0, 1 \ldots] \to [0, \infty)$, is *negligible* if there is an N s.t. for all $n > N$, $f(n) < \frac{1}{p(n)}$. It is asymptotically smaller than any inverse polynomial function.

Properties of negl

1. Proposition 3.6. $\mathrm{negl}_1 + \mathrm{negl}_2$ is *negligible*
2. $p(n) \times \mathrm{negl}$, is *negligible* where $p(n)$ is any polynomial.

PRGs and PRFs

Pseudo-random generators and functions



Definition 3.25

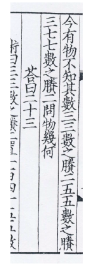
Let $F : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}^*$ be an efficient length-preserving keyed function. F is a pseudorandom function if for all PPT distinguishers D, there is a negl s.t.

$$\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| \leq \mathrm{negl}(n)$$

The first probability is taken over choice of the key k, randomness of D. The second probability is taken over choice of f from Func_n .
(Here, the size of the key is the same as the message. In general, length-preserving only refers to the size of message and ciphertext.)

CRT

CRT (Chinese Remainder Theorem)



Problem posed by Wei/Jin mathematician Sun Tzu 1700 years ago:

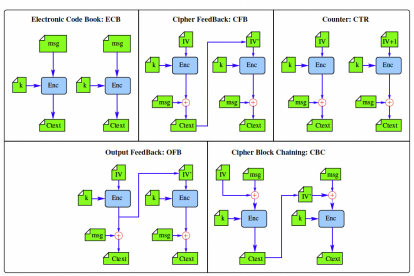
There are certain things whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What will be the number?

The Chinese remainder theorem - nowadays

Two simultaneous congruences $n = n_1 \bmod m_1$ and $n = n_2 \bmod m_2$ are only solvable when $n_1 = n_2 \bmod (\gcd(m_1, m_2))$. The solution is unique modulo $\mathrm{lcm}(m_1, m_2)$.
It demonstrates to us that a number less than the product of two primes can be uniquely identified by its residue modulo those primes. It is useful in RSA.

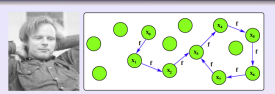
Modes

Modes



Floyd

Floyd - for chains like $x_0 \to f(x_0) \to f(f(x_0)) \to \ldots$



Find loop with $f(x) \stackrel{\text{def}}{=} \mathcal{H}(x)$**; time** $\mathcal{O}(2^n)$**, and memory** $\mathcal{O}(1)$

(Robert W.) Floyd's cycle-detection algorithm: the tortoise t and the hare h. Variables hop through a graph at different rates, stopping when they collide.

Algorithm 5.9: part 1 finds the loop in a (digest/hash) chain

```
def H(x):
    return md5(x.encode('utf-8')).hexdigest()
def f(x):
    return H(x)[:10]

def floyd(x0):
    t = f(x0)
    h = f(f(x0))
    print('Find the loop')
    while (t != h):
        t = f(t)
        h = f(f(h))
```

Extended Euclidean

Extended Euclidean algorithm

An interesting property

If the $\gcd(a, b) = r$ then there exist integers m and n so that $ma + nb = r$.
Use to calculate the multiplicative inverse of an element x modulo n.

The extended Euclidean algorithm

1. We begin by dividing n by x, and as we carry out each step i of the Euclidean algorithm discovering the quotient q_i, we also calculate an extra number x_i. For the first two steps $x_0 = 0$ and $x_1 = 1$.
2. For the following steps, calculate $x_i = x_{i-2} - x_{i-1}q_{i-2} \bmod n$.
3. If the last non-zero remainder occurs at step k, then if this remainder is 1, x has an inverse and it is x_{k+2}.

The inverse of 15 modulo 26, showing the method

0 :	26	=	$1 * 15 + 11$	$x_0 = 0$			
1 :	15	=	$1 * 11 + 4$	$x_1 = 1$			
2 :	11	=	$2 * 4 + 3$	$x_2 = 0 - 1 * 1 \bmod 26$	=	25
3 :	4	=	$1 * 3 + 1$	$x_3 = 1 - 25 * 1 \bmod 26$	=	2
4 :	3	=	$3 * 1 + 0$	$x_4 = 25 - 2 * 2 \bmod 26$	=	21
				$x_5 = 2 - 21 * 1 \bmod 26$	=	7

The (crypto) constructions

The constructions are...

Enc-Auth: Enc-then-Auth - authentic and CCA secure

MD/HMAC: Merkle/Damgård and HMAC for variable lengths

RSA: Encryption, CCA-secure KEM

OAEP: Feistel based padding scheme

DHKE: Diffie/Hellman key exchange

Elgamal: Encryption, signatures

IBE: Boneh/Franklin IBE based on bilinear maps

ECC: Encryption, ECC/DHKE

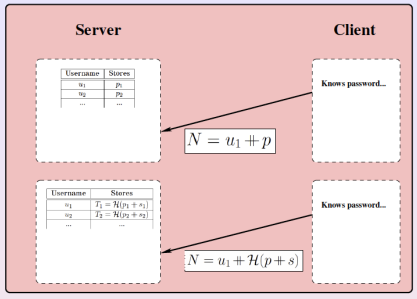
Hash&Sign: For secure signatures

MPC: Multi-party computations

Applications (apart from the obvious)

Passwords and hashing

Passwords and hashing



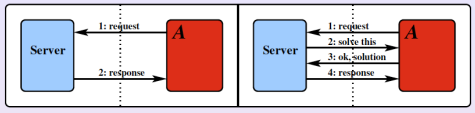
Encryption $\neq$ Hashing... Basic authentication is not much

Assume $\mathcal{H}$ is a cryptographic hash function, p is the password, s is a salt for the password, and $p + s$ as before.

Note that here, a hash is as good as the password.

Proof of work

Proof-of-work



NDSS 1999

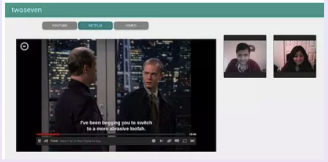
In a typical Denial of Service attack, the attacker invests relatively less resource in issuing a request, but the victim has to spend much more resources in answering the request. The puzzle based method tries to flip this asymmetry. An attacker needs to submit a proof-of-work before the server serves the request.

The puzzles should be easy to construct, easy to verify, but difficult to solve. One way to construct a puzzle might be to choose a random string s, and force the attacker to find a string r s.t. the first (or last) 20 bits of the digest $\mathcal{H}(r + s)$ are all zeros.
The attacker is forced to carried out an expected number of 2^{20} hash operations (?) to find such a choice of r.

https://www.ndss-symposium.org/wp-content/uploads/2017/09/A-Cryptographic-Defense-Against-Connection-Depletion-Attacks-Ari-Juels.pdf

Commitment

Commitment



Movie night

Alice and Bob are talking over a secure channel. They want to decide which movie to watch. Alice prefers *movieA*, but Bob wants *movieB*.

To resolve that, they derive this protocol: each of them is to think of a bit, say a and b. If $(a \oplus b) = 1$, then they will watch *movieA*, otherwise *movieB*.

Now, how to exchange these two bits a and b so that they are unable to cheat? If Alice sends Bob $a = 1$ first, Bob can force Alice to watch *MovieB*.

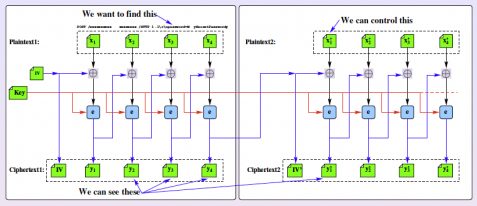
Tossing a coin?

Alice and Bob are not in the same room. It is problematic.

Attacks #1

Rizzo/Duong (CPA)

CPA attack - Rizzo/Duong



BEAST: Browser Exploit Against SSL/TLS

The attack is exploited by Rizzo & Duong on TLS (BEAST).
A malicious site with some javascript/java may convince a browser to send crafted messages (A CPA), and by observing the encrypted messages, perhaps discover an important cookie or token.

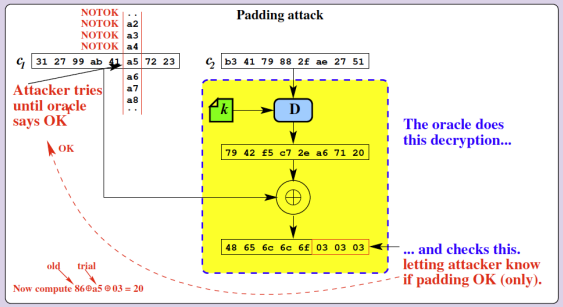
Description

https://www.exploit-db.com/docs/english/15137-practical-padding-oracle-attacks.pdf

Padding oracle (CCA)

CCA attack - padding oracle

Attacker discovers third-to-last byte...

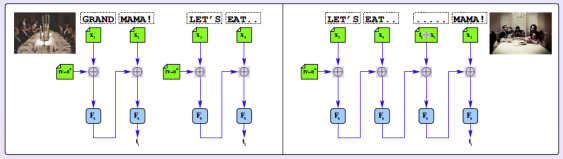


Since we know (now) the last two bytes (02 02), we can set the last two bytes to 03 03.

Now we try all the third-to-last values in c_1 and uncover 20.

Concatenation (CPA)

CPA concatenation attack (on CBC-MAC)



Construction 4.11 - Fixed-length CBC-MAC (p123)

With key $k \in \{0,1\}^n$, PRF F_k, message length $\ell \times n$, $t_0 := 0^n$, then we have:

$\mathrm{Mac}_k(m)$: For $i = 1, ..., \ell$: $t_i := F_k(t_{i-1} \oplus m_i)$. Then output the tag $t = t_\ell$.

$\mathrm{Vrfy}_k(m, t)$: output 1 if and only if $t = \mathrm{Mac}_k(m)$.

Theorem 4.12

If F_k is PRF, and m fixed length, CBC-MAC is secure.

Concatenation attack possible for arbitrary length m

If the adversary has obtained two pairs (m_1, t_1) and (m_2, t_2) - then m_3 can be easily created with a valid tag t_1 or t_2. In the above case (m_3, t_1) passes Vrfy.

Rainbow tables

Rainbow table attack

$$w_0 \xrightarrow{\mathcal{H}} y_0 \xrightarrow{R} w_1 \xrightarrow{\mathcal{H}} y_1 \xrightarrow{R} w_2 \xrightarrow{\mathcal{H}} y_2 \xrightarrow{R} w_3 \xrightarrow{\mathcal{H}} y_3$$

$$w_9 \xrightarrow{\mathcal{H}} y_9 \xrightarrow{R} w_2 \xrightarrow{\mathcal{H}} y_2 \xrightarrow{R} w_3 \xrightarrow{\mathcal{H}} y_3 \xrightarrow{R} w_4 \xrightarrow{\mathcal{H}} y_4$$

Collisions of the reduce function may happen frequently

The pairs $\langle w_0, y_3 \rangle$ and $\langle w_9, y_4 \rangle$ will be stored. This causes two issues:

1. Efficiency in storage: Part of the chain is duplicated. The words w_2 and w_3 appear in the table twice.
2. Efficiency in search: Lead to searches in the wrong chains, before hitting the right chain. For the query y_9, the lookup process would transverse both chains.

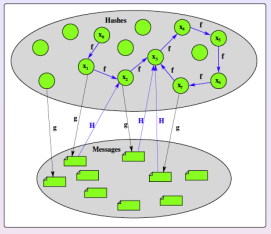
Solution to the problem: the "Rainbow" table

Suppose there are k reduce operations and k hash operations in the chain, instead of using the same function $R()$ in all the reduce operations, the Rainbow table uses k different functions. The first reduce is $R_1()$, followed by $R_2()$ and so on.

Attacks #2

Hash collision

Hash collision attack



Apply Floyd by using $f(x) \stackrel{\text{def}}{=} \mathcal{H}(g(x))$

Consider messages $m \in M$, digests $h \in H$, the functions $g : H \to M$, and $f : H \to H$ defined as $f(x) \stackrel{\text{def}}{=} \mathcal{H}(g(x))$. The elements in the *chain/trail* of digests are mapped to the messages. In the textbook, p168, there is an example with *good*, and *bad* messages about Bob:

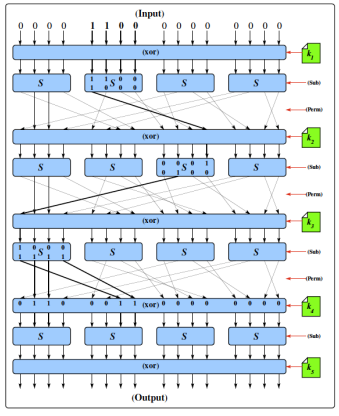
$g(0000)$ = Bob is a good and honest worker.
$g(0001)$ = Bob is a difficult and....

Differential cryptanalysis

Differential cryptanalysis attack

x	$S(x)$
0	0
1	b
2	5
3	1
4	6
5	8
6	d
7	4
8	f
9	7
a	2
b	c
c	9
d	3
e	e
f	a

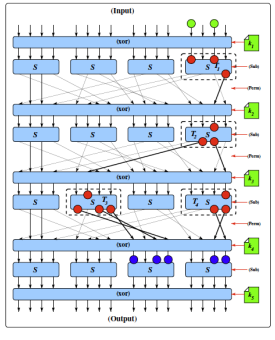
z	$P(z)$
1	7
2	2
3	3
4	8
5	12
6	5
7	11
8	9
9	10
10	1
11	14
12	13
13	4
14	6
15	16
16	15



Linear cryptanalysis

Linear cryptanalysis attack

Note that the bias from the red dots/bits $\varepsilon(T_1 \oplus T_2 \oplus T_3 \oplus T_4)$ is exactly the bias of the green and blue bits.



Linear bias of the whole SPN

To use the piling lemma, we assume that T_1, T_2, T_3, T_4 are independent , but of course they are not. However, it is a good approximation.

Birthday/factorization

Birthday attack (on factorization)

Pollard Rho Algorithm - a birthday attack!

As we saw before, a straightforward factorization algorithm takes $\mathcal{O}(N^{\frac{1}{2}})$ time. By contrast, this algorithm takes an expected $\mathcal{O}(N^{\frac{1}{4}})$ time[a]. Interestingly, it uses a birthday attack, such as we looked at a few weeks ago. Just for curiosity, let's look at the details.

[a] If N is 200 bits, then the security is about 50 bits under this attack.

Pollard Rho Algorithm – improvement to $\mathcal{O}(N^{\frac{1}{4}})$

Given $N = p \times q$, and where $p < q$, an important observation is that for any x, x' s.t. (a) $x \neq x'$ and (b) $x = x' \bmod p$, then $\gcd(x - x', N) = p$. Why?

A birthday attack! If we have a set X of integers (randomly drawn from $\mathbb{Z}_N$) with $|X| \approx 1.177 p^{\frac{1}{2}}$, then by the birthday attack, the chance that there are two elements x, x' s.t. (a) $x \neq x'$ and (b) $x = x' \bmod p$ is more than 50%. Hence, potentially, we have a $\mathcal{O}(p^{\frac{1}{2}}) \approx \mathcal{O}(N^{\frac{1}{4}})$ algorithm to factorize N.

Now we need an algorithm to find two elements in the set X which have the same reminder modulo p. Hmmmmm... sounds like a collision...

Attacks #3

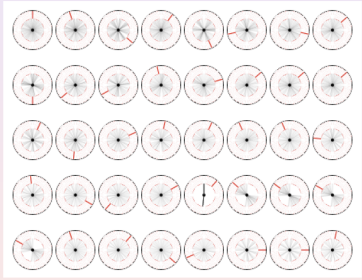
Quantum (Shor)

Quantum computation attack (Shor)

Find repetition values for all powers simultaneously:

A Quantum register holds all possible values at the same time, and we perform an amplifying operation, that only leaves stable repetition values.

Mark the same times every day... In the simulation, only the clock with a repetition rate that we are interested in remains:



Padding attack on RSA (CCA)

CCA padding attack (on RSA)

PKCS#1 padding attack on KEM, from 1998

00	02	padding string	00	data block

In 1998, Daniel Bleichenbacher from Bell labs published a padding oracle attack on PKCS#1 (and hence SSL V3.0). The attack is based on the fact that RSA is homomorphic with respect to multiplication. In addition, some (web) servers can act as padding oracles.

In SSL, during the initial negotiation, critical keys are sent, encrypted using RSA (a KEM). A padding oracle attack can retrieve this "pre"master secret key, in what has been called the “million message” attack. The details of this attack are found in the paper, but it is not dissimilar to the padding oracle attack in Session 5; multiple messages are sent to the server, which responds differently if the format of the padding is correct or incorrect.

“Fixing” PKCS#1

The approach taken to fix this was to ask vendors to add extra countermeasures: primarily returning uninformative server responses.

DDH attack using QR

QR attack (on DDH)

Consider definition 10.1 - eav-security

An adversary wins if it distinguishes between a random key $k' = g^z$ and the real key g^{ab}. It has the transcript `trans`, which in DH contains g^a, g^b, g and p.

This is exactly the DDH problem, distinguishing between $A = (g^a, g^b, g^z)$, and $B = (g^a, g^b, g^{ab})$. In the group $\mathbb{Z}_p^*$, DDH is easy to decide using the Legendre symbol. If a value such as g^a has a QR, then a must be even. We can work out the Legendre symbol for each of A or B above. For A, $\left(\frac{g^z}{p}\right)$ will be random, whereas for B the only possibilities are as follows:

$\left(\frac{g^a}{p}\right)$		$\left(\frac{g^b}{p}\right)$		$\left(\frac{g^{ab}}{p}\right)$	
1	(a even)	1	(b even)	1	(ab even)
1	(a even)	−1	(b odd)	1	(ab even)
−1	(a odd)	1	(b even)	1	(ab even)
−1	(a odd)	−1	(b odd)	−1	(ab odd)

There is an easy way to fix this vulnerability. Instead of using the group $\langle g \rangle$, use the group $\langle g^2 \rangle$. All the elements in $\langle g^2 \rangle$ have a square root.

Homomorphic weaknesses

Homomorphic weakness attack (on RSA)

An insecure scheme...

RSA could be used as a signature scheme, where “signing” is encryption with the secret key, and “verifying” is decryption with the public key.

$\text{Sign}_{s_k}(m)$: Given $m, \langle N, d \rangle$, output: $(m, s) \leftarrow (m, m^d \bmod N)$

$\text{Vrfy}_{p_k}(m, s)$: Given $(m, s), \langle N, e \rangle$, if $m = s^e \bmod N$ then valid.

The above is not secure at all. There is a common misconception that a secure signature scheme can be achieved via “encryption” in this way:

1. Signing is done by “encrypting” the message using private key,
2. Verification is done by “decrypting” the signature using public key. [a]

This technique might not be secure. (Some textbooks use the term “encryption/decryption” in describing signature schemes. This is very misleading). Perhaps you can consider the effect of homomorphic properties on this scheme.

[a] RSA has an interesting property that we can use private key for encryption and public key for decryption. This is not true for other PKC.

Attacks #4

Exponentiation timing

Timing attacks (on exponentiation)

Efficient exponentiation

The first example of a timing attack on exponentiation (as in RSA) was introduced by Paul Kocher in 1995, while he was an undergraduate at Stanford. The attack attempts to find the key k, and exploits how efficient exponentiation is carried out.

Note that RSA does repeated multiplication, and Elgamal uses point addition, an analog of exponentiation. The algorithms use doubling and look the same:

RSA

```
To compute c^d mod N in RSA
Q = 1;
foreach bit in key d {
    Q = Q*Q mod N;
    if (bit==1) {
        Q = Q*c mod N;
    }
}
return Q;
```

ECC

```
To compute kP in ECC
Q = O;
foreach bit in key k {
    Q = 2Q;
    if (bit==1) {
        Q = Q + P;
    }
}
return Q;
```

Fault injection

Fault injection attack (in RSA, if using CRT)

What can the attacker do?

Assume the attacker can obtain the output $m = (ae_1 + be_2) \bmod N$.

The attacker can introduce noise/errors during the computation of e_1, and no error during the computation of e_2. Hence the attacker obtains

$$m' = (ae_1' + be_2) \bmod N$$

Now, the attacker computes $\gcd(m–m', N)$. The gcd is either p or q.

Why does it work?

Note that $m - m' = a(e_1 - e_1') \bmod N$, and that $a = 0 \bmod q$ (but is not zero). If $m–m'$ is not divisible by p, then

$$\gcd(m - m', N) = q$$

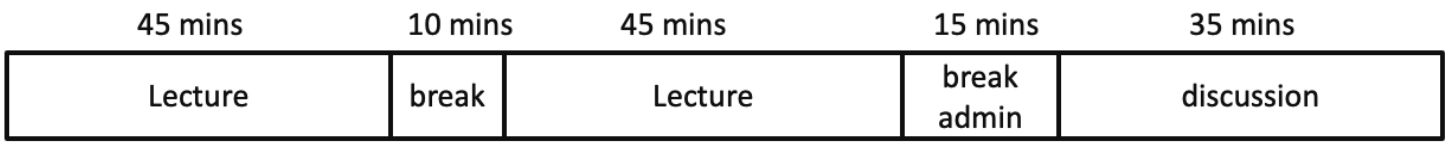
My expectation...

Please, please, please....

Attend classes and tutorials, and Ask if you don't know. Read notes, book, and the readings, and get interested in the subject

Lectures, tutorials and help....

The early parts of the course are a little mathematical, as you can see from the textbook, which we will follow closely.
The lectures will be followed by discussion and assistance - so no specific tutorials.



Quotes to set the scene...

Defense and attack...

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

"The Art of War", Sun Tzu

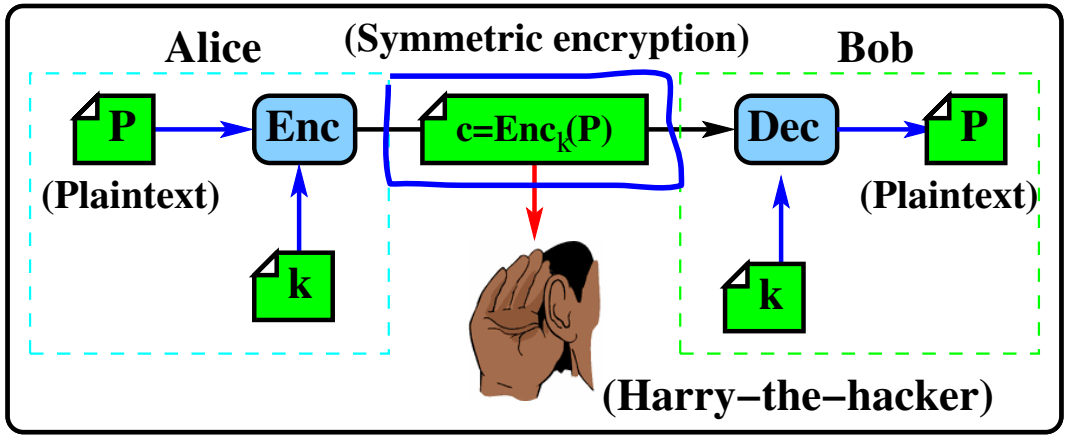
Strength of algorithms, levels of security...

The combination of space, time, and strength that must be considered as the basic elements of this theory of defense makes this a fairly complicated matter. Consequently, it is not easy to find a fixed point of departure..

"On War", Carl Von Clausewitz

Types of communication channels

Bob, Carol, Ted and Alice (and Harry and Darth).

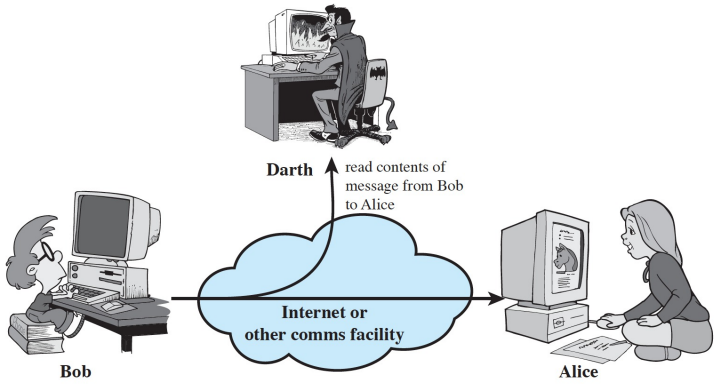


- Private channel: Only Alice and Bob have access to the channel. Safe (wired network between PCs...).
- Public channel: Anyone can send, delete, eavesdrop messages.
- Secure channel: Can be a private channel or a public channel with appropriate protection mechanism (encryption...).
- Broadcast channel: Messages are sent to all entities (facebook...).

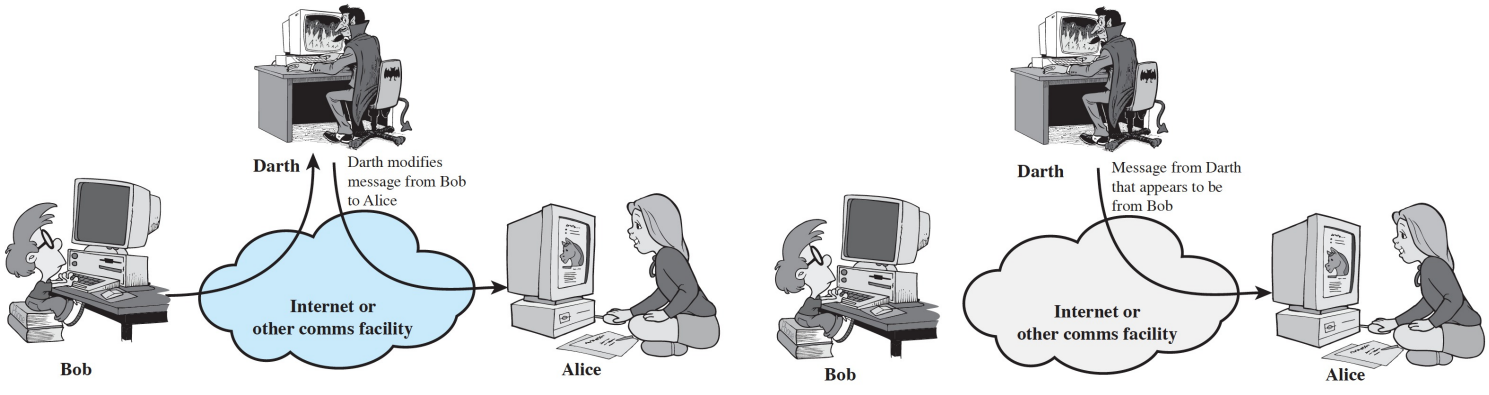
The term “Channel” does not necessarily refer to a computer network. For example, if Alice writes her message on a piece of paper, and the paper is sent to Bob by hand, this is a private channel.

Attacks

Passive...

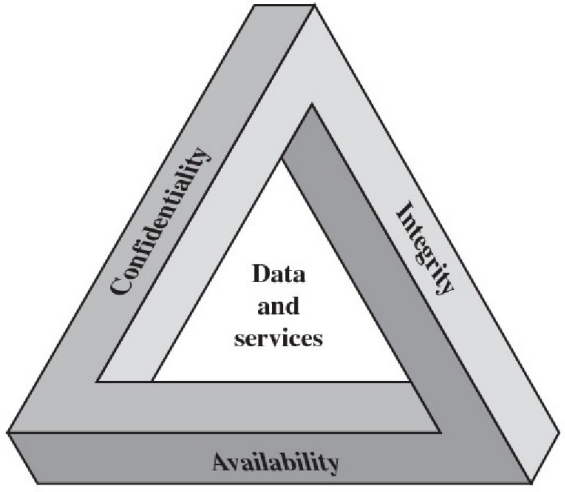


Active...



The CIA triad...

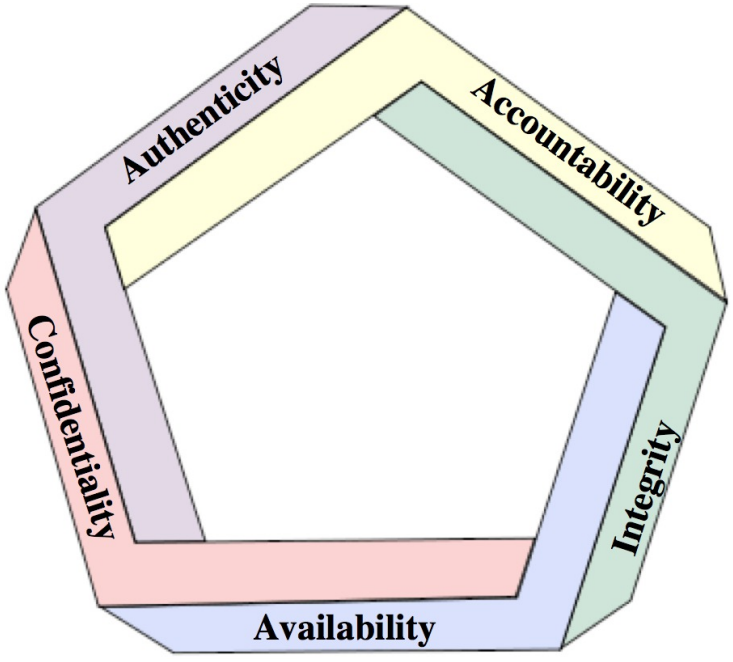
FIPS specify three objectives/goals:



confidentiality: concealing information - resources may only be accessed by authorized parties;
integrity: trustworthiness of data - resources may only be modified by authorized parties in authorized ways;
availability: preventing DOS/denial-of-service - resources are accessible in a timely manner.

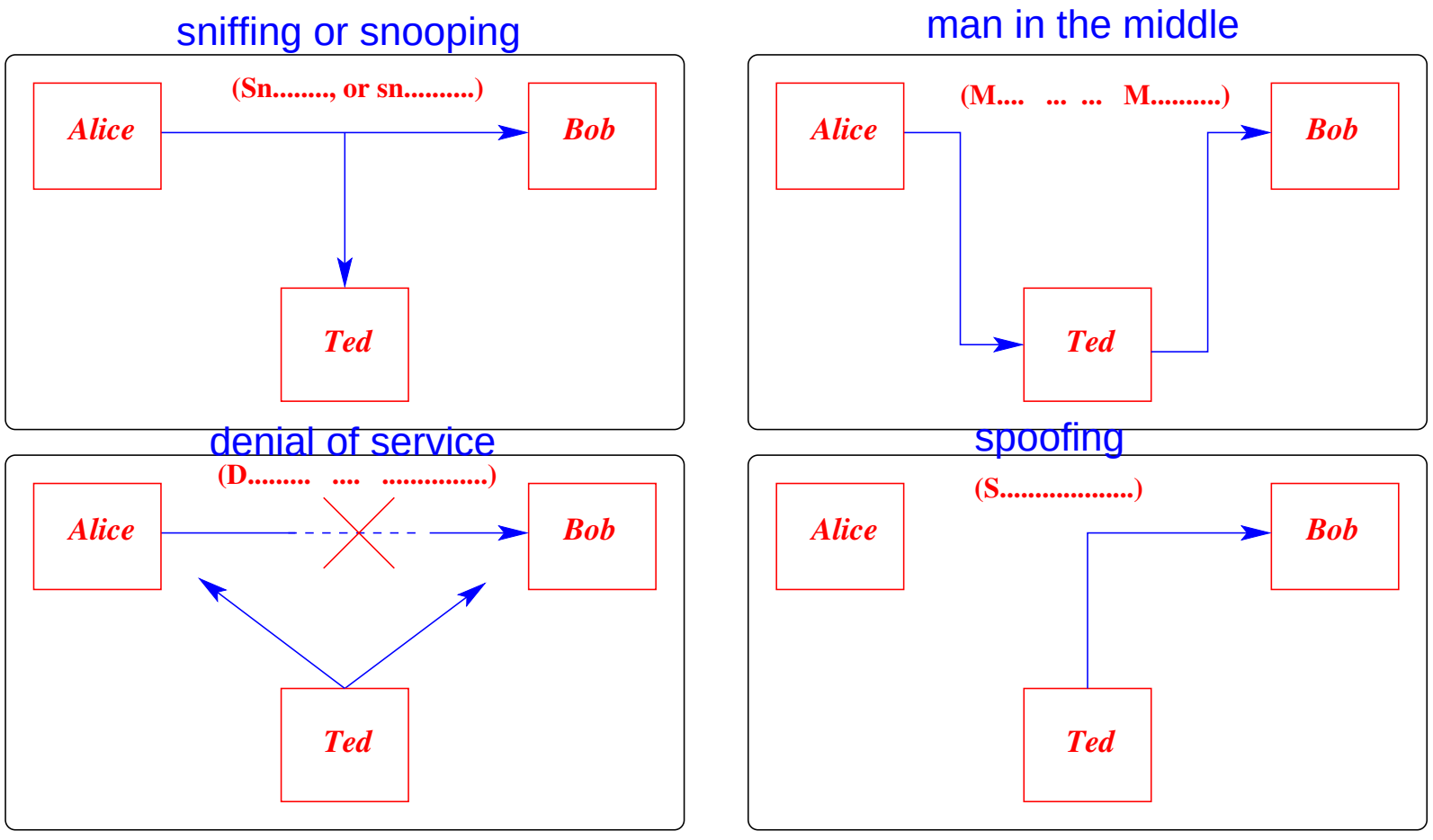
The CIAAA gang-of-five...

Many observers identify more...



Authenticity: logins, password checks
Accountability: non-repudiation of a prior commitment

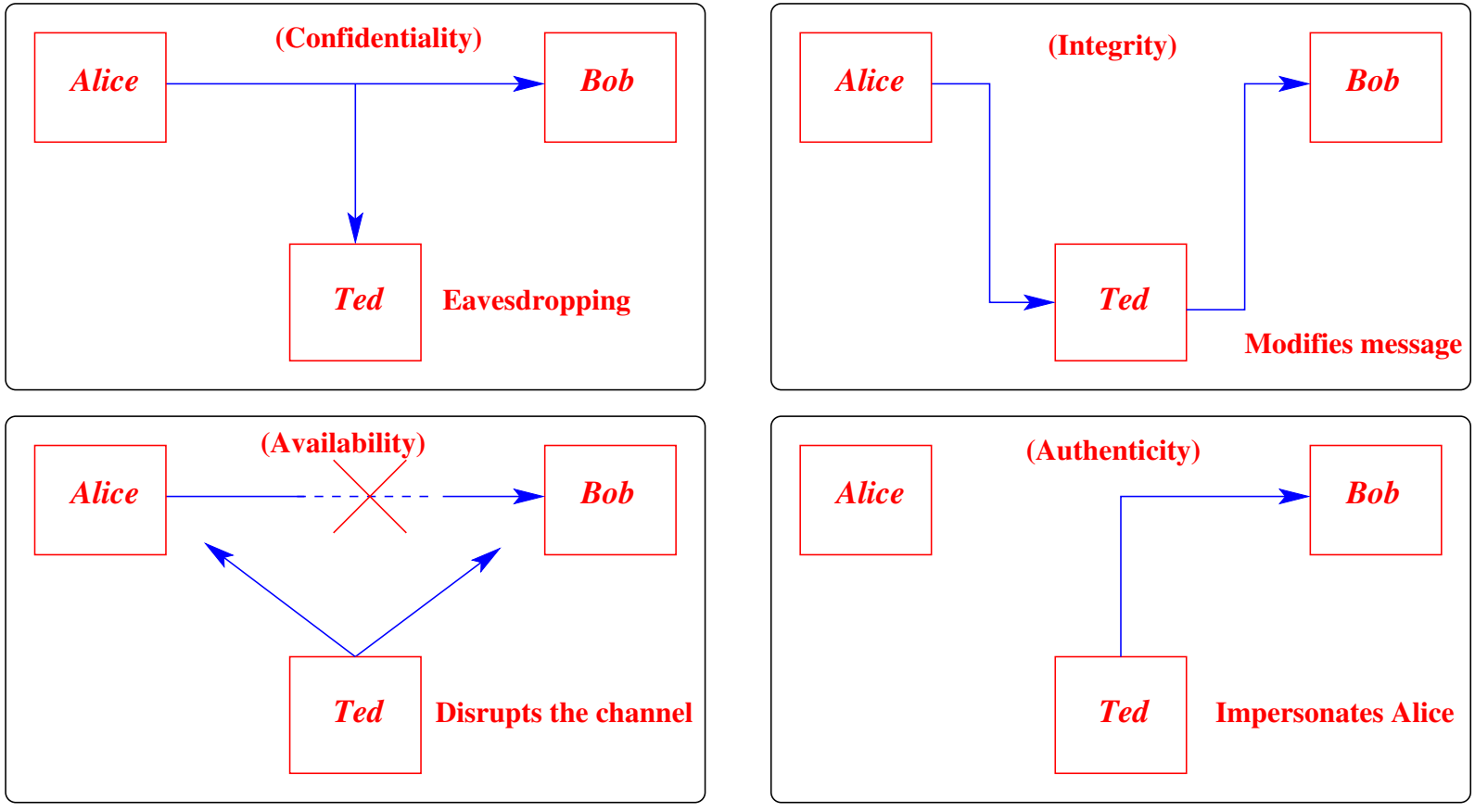
The “attack” point of view



(... or ... Interception, Modification, Interruption, Fabrication)

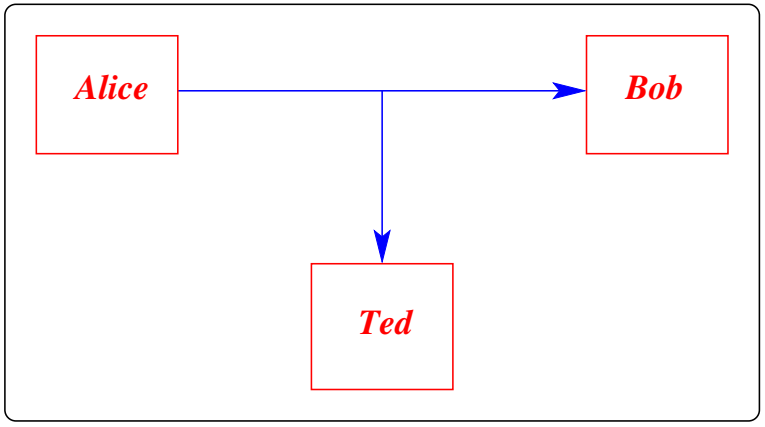
The “security requirements” point of view

the security requirement is trying to achieve



(... the other side of the fence...)

More “security requirements”...



Covert channel?	Anonymity
What if Alice is concerned that Ted should not know even if a message was sent?	In most scenarios, eavesdroppers, and Bob know WHO sent the message. What if we wanted to hide the source (Alice) of the message?

Kerckhoff's-principle vs Security-by-obscurity

Two approaches:

Security by obscurity: To hide the design of the system in order to achieve security. An adversary might reverse-engineer a system to get the details of the system design.
Kerckhoff's principle: A system should be secure even if everything about the system, except the secret key, is public knowledge.

Consider these 2 examples against "obscurity":

1. https://en.wikipedia.org/wiki/RC4 RC4 was introduced in 1987 and its algorithm was a trade secret. In 1994, a description of its algorithm was anonymously posted in a mailing group.
2. MIFARE Classic, a contactless smartcard widely used in Europe. It uses a set of proprietary protocols/algorithms. However, they were reverse-engineered in 2007. It turns out that the encryption algorithms are already known to be weak (using only 48bits) and breakable https://en.wikipedia.org/wiki/MIFARE.

Kerckhoff's-principle vs Security-by-obscurity

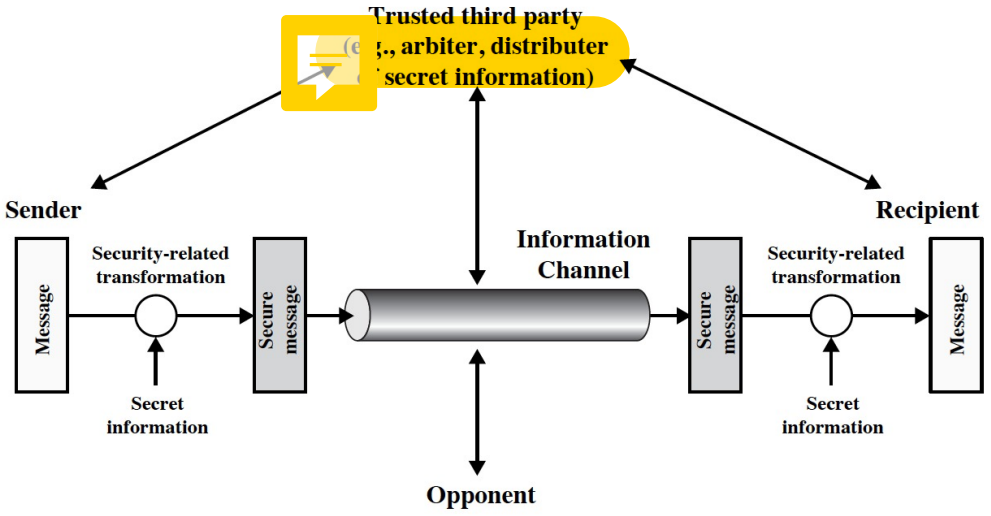
Arguments for "obscurity":

1. It is advisable not to reveal the network structure and settings (for example, what rules are used by the firewall), although these are not "secrets", and people in the organization may already know them. Why? - with this info, adversary may be able to identity vulnerability.
2. It is advisable not to publish the actual program used in a smart-card. Why? - with this info, adversary may be able to identity vulnerability that was previously unknown, or carry out side-channel attacks[a].
3. It is good to hide the administrator's username, although the usernames are not necessarily supposed to be a secret. Why? - without it, increases search space for adversary - we can buy time and deter adversaries with limited resources. Even if we use obscurity, system should still be secure under Kerckhoff's principle.

More examples from
`https://en.wikipedia.org/wiki/Security_through_obscurity`.
`https://technet.microsoft.com/en-us/magazine/2008.06.obscurity.aspx`.

[a] Note that a sophisticated adversary may still do reverse-engineering.

Models



Model for network security requires us to...

...design a suitable algorithm for the security transformation, generate the secret information (keys) used by the algorithm, develop methods to distribute and share the secret information, and specify a protocol enabling the principals to use the transformation and secret information for a security service.

Adversary/attack model

It is secure!

The statement “the encryption scheme is secure” is ambiguous. What does “secure” mean? For example

1. Suppose an attacker knows the ciphertext of the string “yes”, could the attacker determine whether a ciphertext in question correspond to the plaintext “yes”?
2. Suppose an attacker simply wants to know the size of the plaintext, could the attacker derive it from the ciphertext?

Generally, it is easier to describe the “security” of a scheme from the perspective of an attack. If the scheme can prevent the attack, then it is “secure” w.r.t. to that attack. An adversary model describes:

1. the adversary’s goal.
2. capability of the adversary + information the adversary has.

We will discuss various models:

- for encryption, Perfect secrecy, semantic security, etc,
- for hash, collision, one-way, pseudorandomness, etc
- for keyed-hash, forgery.

Definition of (symmetric) encryption...

A symmetric encryption system/scheme $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$**:**

... comprises 3 algorithms:

1. $\mathrm{Gen}(1^n)$: Typically the input is the size of the key
2. $\mathrm{Enc}_k(m)$: Input is the key k and the message. For presentation, the key k is put as a subscript.
3. $\mathrm{Dec}_k(c)$: Input is the key k and the ciphertext.

Note that we use this notation:

$\mathcal{K}$: set of all keys.
$\mathcal{M}$: set of all messages
$\mathcal{C}$: set of all ciphertexts

For correctness, we see that for all k, m, $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$.
Note also that $\mathrm{Gen}()$ is probabilistic, $\mathrm{Enc}_k()$ could be probabilistic, $\mathrm{Dec}_k()$ is deterministic.

Julius Cæsar cipher...

The Cæsar cipher

Cæsar (rotation) cipher over Roman letters: Key is "+3".

							I		
		C	L	A	V	D	I	V	S
A	B	C	D	E	F	G	H	I	K
D	E	F	G	H	I	K	L	M	N
L	M	N	O	P	Q	R	S	T	V
O	P	Q	R	S	T	V	X	Y	Z
M									
F	O	D	Z	G	M	Z	X		

Can define the transformation mathematically:

$$\begin{aligned} c &= \mathrm{Enc}_k(p) = (p+k) \mod 23 \\ p &= \mathrm{Dec}_k(c) = (c-k) \mod 23 \end{aligned}$$

Cryptanalysis of rotation ciphers:

In the above example - we only have 22 possible useful ciphers! So an attacker can try each in turn: a brute force search

Examples of rotation ciphers

Union (North) and Confederate (South) ciphers

Used in the American Civil war, they can be used as simple rotation cipher.

The Confederate one has a keyspace of 26, and a useful keyspace of only 25.

Substitution

Substitution cipher systems encode the input stream using a substitution rule. The Cæsar cipher is an example of a simple substitution cipher system.

A (random) monoalphabetic substitution cipher

Code	Encoding
A	Q
B	V
C	X
D	W
...	...

If the mapping was more randomly chosen it is called a monoalphabetic substitution cipher, and the keyspace for encoding 26 letters would be $26! - 1 = 403,291,461,126,605,635,583,999,999$.

Substitution cipher...

Cryptanalysis of substitution cipher

How safe is this cipher? (Not at all!)

If we could decrypt $1,000,000$ messages in a second, then the average time to find a solution by trying decryptions would be about $6,394,144,170,576$ years!

We might be lulled into a sense of security by these big numbers, but of course this sort of cipher can be subject to frequency analysis...

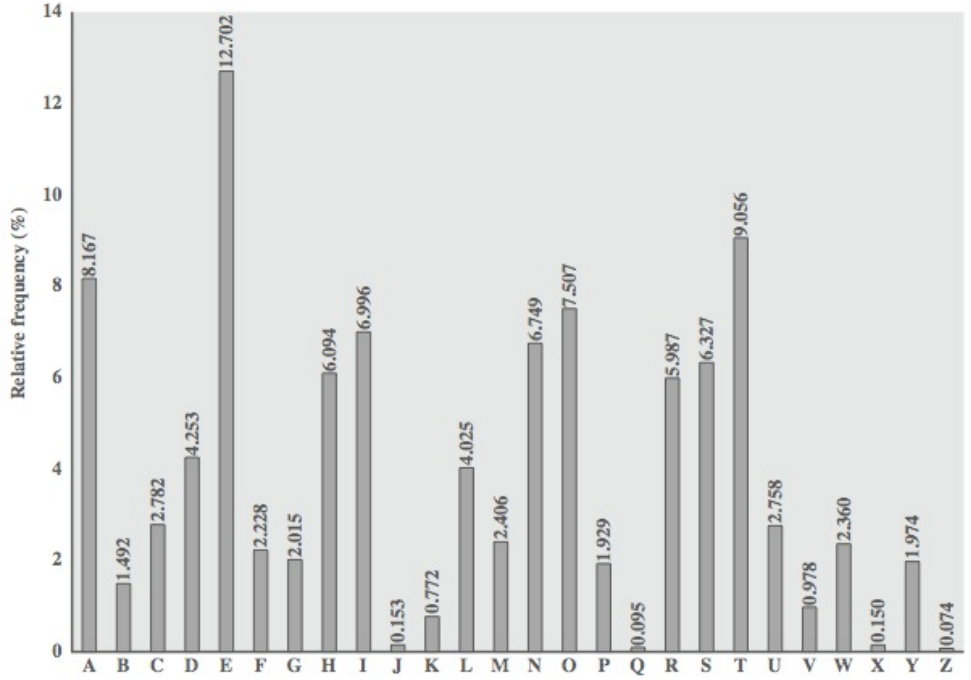
The problems are that:

1. letters are not equally common: ETAOINSHRDLU!
2. human languages have high levels of redundancy: (fr xmpl, hgh ndrsn s tchng cs4236 ths smstr).

We have tables of single, double & triple letter frequencies for languages.

Cryptanalysis of mono-alphabetic ciphers

Frequencies of english text:



These ciphers do not change relative letter frequencies

The central concept of this was discovered and described by Arabian scientists in the 9th century. An attacker can calculate the frequencies for ciphertext. The most common ciphertext letter might translate to an E.

Cryptanalysis of mono-alphabetic ciphers

Example - first step of decoding:

```
EV YQS CVV MIWK FRPC FRQF FRV IQFV WM
FIQSCKPCCPWS ACPSN IVJVFPFPWS RQC FW
QJJIWQYR ZVIW FW QYRPVDV KWIV QSB KWIV
IVTPQXTV FIQSCKPCCPWS. RWEVDVI EV LSWE
FRQF FRV FRVWIVFPYQT IQFV CRWATB ...
```

V occurs most often, F next and so on, so replace V with E...

Example - first step of decoding:

```
EV YQS CVV MIWK FRPC FRQF FRV IQFV WM
-E -A- -EE F-O- THI- THAT THE -ATE OF

FIQSCKPCCPWS ACPSN IVJVFPFPWS RQC FW
T-A---I--IO- --I-- -E-ETITIO- HA- TO

QJJIWQYR ZVIW FW QYRPVDV KWIV QSB KWIV
A---OA-H -E-O TO A-HIE-E -O-E A-- -O-E
```

Polyalphabetic ciphers

Polyalphabetic substitution ciphers improve security:

There are more alphabets to guess and hence a flatter frequency distribution.
We use a key to select which cipher is used for each letter of message.

Vigenère (1520) uses a tableau, and a key:

	A	B	C	D	E	F	G	H	...
A	A	B	C	D	E	F	G	H	...
B	B	C	D	E	F	G	H	I	...
C	C	D	E	F	G	H	I	J	...
D	D	E	F	G	H	I	J	K	...
E	E	F	G	H	I	J	K	L	...
F	F	G	H	I	J	K	L	M	...
G	G	H	I	J	K	L	M	N	...
H	H	I	J	K	L	M	N	O	...
...	...	...	...	...	...	...	...	...	...

Vigenère

Keyword is BAD, so encoding HAD A FEED results in:

Key	B	A	D	B	A	D	B	A
Text	H	A	D	A	F	E	E	D
Cipher	I	A	G	B	F	H	F	D

If we can discover the length of the repeated key (in this case 3), and the text is long enough, we can just consider the cipher to be a group of interleaved substitution ciphers and solve accordingly.

Cryptanalysis of Vigenère cipher:

Multiple ciphertext letters for each plaintext letter, and so letter frequencies are obscured (but not totally lost).
We start with letter frequencies to see if it is monoalphabetic or not. If not, then we need to determine the number of alphabets.

Example of a polyalphabetic substitution cipher

The M-94 cipher

Used by the US army from 1922 to 1942. It had 25 disks, each containing a random sequence of the letters A-Z around the outside.

US M209 Rotor machine

WWII Mechanical encryption machine

What is inside? A Beaufort cipher

Reversed tableau...

	A	B	C	D	E	F	G	H	...
0	Z	Y	X	W	V	U	T	S	...
1	A	Z	Y	X	W	V	U	T	...
2	B	A	Z	Y	X	W	V	U	...
3	C	B	A	Z	Y	X	W	V	...
4	D	C	B	A	Z	Y	X	W	...
5	E	D	C	B	A	Z	Y	X	...
6	F	E	D	C	B	A	Z	Y	...
7	G	F	E	D	C	B	A	X	...
...	...	...	...	...	...	...	...	...	...

H encoded with key 5 is X.
More material at
`http://users.telenet.be/d.rijmenants/en/m209sim.htm`

Cryptanalysis: Kasiski method

Method developed by Babbage (1854) and Kasiski (1863):

Repetitions in ciphertext give clues to the period; so find some plaintexts an exact period apart (of course, could also be a random fluke).
Then attack each monoalphabetic cipher individually using the same techniques as before.

Despite this - systems used into 20th century:

The Zimmermann Telegram (or Zimmermann Note; German: Zimmermann-Depesche; Spanish: Telegrama Zimmermann) was a 1917 diplomatic proposal from the German Empire to Mexico to make war against the United States. The proposal was declined by Mexico, but angered Americans and led in part to the declaration of war in April [Wikipedia].

Transposition ciphers

Transposition/permutation ciphers:

- Hide the message by rearranging letter order.
- Have the same frequency distribution as the original text

Rail-fence cipher:

Write message letters out diagonally over a number of rows then read off cipher row by row eg. write message out as:

```
m   e   m   a   t   r   h   t   g   p   r   y
  e   t   e   f   e   t   e   o   a   a   t
```

giving ciphertext

M E M A T R H T G P R Y E T E F E T E O A A T

Row transposition cipher

Row transposition - more complex:

Write letters of message out in rows over a specified number of columns then reorder the columns according to some key before reading off the rows:

Key:	4 3 1 2 5 6 7
Plaintext:	a t t a c k p
	o s t p o n e
	d u n t i l t
	w o a m x y z
Ciphertext:	TTNAAPTMTSUOAODWCOIXKNLYPETZ

Cryptanalysis of transposition cipher:

- Easily recognized because it has the same letter frequencies as the original plaintext.
- For the transposition just shown, cryptanalysis is easy and involves laying out the ciphertext in a matrix and playing around with column positions.
- Digram and trigram frequency tables can be useful.

Transposition

Detecting a transposition cipher

Detect a transposition cipher with the frequencies of the letters, and letter pairs.
If the frequency of single letters in ciphertext is correct, but the frequencies of letter pairs is wrong, then the cipher may be a transposition.
This sort of analysis can also assist in unscrambling a transposition ciphertext, by arranging the letters in their letter pairs.

Product ciphers

Substitution plus transposition:

Is double encryption a thing? i.e. $c = \mathrm{Enc}_{k_1}(\mathrm{Enc}'_{k_2}(x))$ where k_1 is the key for the first encryption Enc and k_2 is the key for the second encryption Enc'.

- If both Enc and Enc' are transposition ciphers, then there is no advantage of doing so. The combined cipher is still a transposition.
- If Enc and Enc' are both substitution ciphers, then the final encryption is simply another substitution cipher.

But doing transposition and sustitution form an important class of ciphers: the Substitution-Permutation Networks (SPN). A bridge from classical to modern ciphers.

Cryptographics and adversary's goals

Cryptographic systems are characterized by:

- **Type of encryption operations used:** substitution, transposition, product,
- **Number of keys used:** single-key or private two-key or public
- **Ways in which plaintext is processed:**block or stream

Attacker's goals:

- If an attacker is able to find the key, we call this a total break.
- The attacker may be satisfied with a partial break. For instance, the adversary can determine some specific information about the plaintext (for example, the first bit).
- The attacker may be satisfied with distinguishability of ciphertext: with some non-negligible probability more than $\frac{1}{2}$, the attacker is able to distinguish between encryption of two given plaintext, or between an encryption of a given plaintext and a random string.
- Total break is the "strongest" goal in the sense that, if an adversary is able to achieve total break, he is also able to achieve partial break and distinguishablility of ciphertext.

Cryptanalysis

Attack models, based on information known to attacker:

- Ciphertext only: The adversary has a collection of ciphertext c.
- Known plaintext: The adversary has a collection of plaintext m and their corresponding ciphertext c.
- Chosen plaintext: The adversary has temporary access to a black box. She can choose a plaintext m and obtain the corresponding ciphertext c from the black box. She can access the black box for a reasonably large amount of time.
- Chosen ciphertext: same as chosen plaintext attack, but here, the adversary chooses the ciphertext and the blackbox gives the plaintext.
- Chosen text: select plaintext or ciphertext to en/decrypt

Brute-force/exhaustive search

The attack:

Given a ciphertext whose plaintext is an English sentence, one way to attack is as follows:

> *Try all possible keys. If the decrypted message is a proper English sentence (words appear in dictionary), then output the key.*

Thus, if the key is 64 bits, then the above is expected to try 2^{63} possible keys, and in the worst case, has to try all the 2^{64} keys.

Key size	**Number of keys**	**Time at** $1\ \mathrm{dec}/\mu S$	**Time at** $10^6\ \mathrm{dec}/\mu S$
32 bits	$2^{32} = 4.3 \times 10^9$	35.8 mins	2.15 *mS*
56 bits	$2^{56} = 7.2 \times 10^{16}$	1142 yrs	10.01 hrs
128 bits	$2^{128} = 3.4 \times 10^{38}$	5.4×10^{24} yrs	5.4×10^{18} yrs
168 bits	$2^{168} = 3.7 \times 10^{50}$	5.9×10^{36} yrs	5.9×10^{30} yrs
26 chars	$26! = 4 \times 10^{26}$	6.4×10^{12} yrs	6.4×10^6 yrs

.