Lecture 11: Cryptography

Cryptography is the theory of **codes**. Two important aspects of the field are the **encryption** rsp. **decryption** of information and **error correction**. Both are crucial in daily life. When getting access to a computer, viewing a bank statement or when taking money from the ATM, encryption algorithms are used. When phoning, surfing the web, accessing data on a computer or listening to music, error correction algorithms are used. Since our lives have become more and more digital: music, movies, books, journals, finance, transportation, medicine, and communication have become digital, we rely on strong error correction to avoid errors and encryption to assure things can not be tempered with. Without error correction, airplanes would crash: small errors in the memory of a computer would produce glitches in the navigation and control program. In a computer memory every hour a couple of bits are altered, for example by cosmic rays. Error correction assures that this gets fixed. Without error correction music would sound like a 1920 gramophone record. Without encryption, everybody could intrude electronic banks and transfer money. Medical history shared with your doctor would all be public. Before the digital age, error correction was assured by extremely redundant information storage. Writing a letter on a piece of paper displaces billions of billions of molecules in ink. Now, changing any single bit could give a letter a different meaning. Before the digital age, information was kept in well guarded safes which were physically difficult to penetrate. Now, information is locked up in computers which are connected to other computers. Vaults, money or voting ballots are secured by mathematical algorithms which assure that information can only be accessed by authorized users. Also life needs error correction: information in the genome is stored in a **genetic code**, where a error correction makes sure that life can survive. A cosmic ray hitting the skin changes the DNA of a cell, but in general this is harmless. Only a larger amount of radiation can render cells cancerous.

How can an encryption algorithm be safe? One possibility is to invent a new method and keep it secret. An other is to use a well known encryption method and rely on the **difficulty of mathematical computation tasks** to assure that the method is safe. History has shown that the first method is unreliable. Systems which rely on "security through obfuscation" usually do not last. The reason is that it is tough to keep a method secret if the encryption tool is distributed. Reverse engineering of the method is often possible, for example using plain text attacks. Given a map T, a third party can compute pairs $x, T(x)$ and by choosing specific texts figure out what happens.

The **Caesar cypher** permutes the letters of the alphabet. We can for example replace every letter A with B, every letter B with C and so on until finally Z is replaced with A. The word "Mathematics" becomes so encrypted as "Nbuifnbujdt". Caesar would shift the letters by 3. The right shift just discussed was used by his Nephew Augustus. **Rot13** shifts by 13, and **Atbash cypher** reflects the alphabet, switch A with Z, B with Y etc. The last two examples are involutive: encryption is decryption. More general cyphers are obtained by permuting the alphabet. Because of $26! = 403291461126605635584000000 \sim 10^{27}$ permutations, it appears first that a brute force attack is not possible. But Cesar cyphers can be cracked very quickly using statistical analysis. If we know the frequency with which letters appear and match the frequency of a text we can figure out which letter was replaced with which. The **Trithemius cypher** prevents this simple analysis by changing the permutation in each step. It is called a polyalphabetic substitution cypher. Instead of a simple permutation, there are many permutations. After transcoding a letter, we also change the key. Lets take a simple example. Rotate for the first letter the alphabet by 1, for the second

letter, the alphabet by 2, for the third letter, the alphabet by 3 etc. The word "Mathematics" becomes now "Ncwljshbrmd". Note that the second "a" has been translated to something different than a. A frequency analysis is now more difficult. The **Viginaire cypher** adds even more complexity: instead of shifting the alphabet by 1, we can take a key like "BCNZ", then shift the first letter by 1, the second letter by 3 the third letter by 13, the fourth letter by 25 the shift the 5th letter by 1 again. While this cypher remained unbroken for long, a more sophisticated frequency analysis which involves first finding the length of the key makes the cypher breakable. With the emergence of computers, even more sophisticated versions like the German **enigma** had no chance.

Diffie-Hellman key exchange allows Ana and Bob want to agree on a secret key over a public channel. The two palindromic friends agree on a prime number p and a base a. This information can be exchanged publically. Ana choses now a secret number x and sends $X = a^x$ modulo p to Bob over the channel. Bob choses a secret number y and sends $Y = a^y$ modulo p to Ana. Ana can compute Y^x and Bob can compute X^y but both are equal to a^{xy}. This number is their common secret. The key point is that eves dropper Eve, can not compute this number. The only information available to Eve are X and Y, as well as the base a and p. Eve knows that $X = a^x$ but can not determine x. The key difficulty in this code is the **discrete log problem**: getting x from a^x modulo p is believed to be difficult for large p.
The **Rivest-Shamir-Adleman public key system** uses a **RSA public key** (n, a) with an integer $n = pq$ and $a < (p-1)(q-1)$, where p, q are prime. Also here, n and a are public. Only the factorization of n is kept secret. Ana publishes this pair. Bob who wants to email Ana a message x, sends her $y = x^a \bmod n$. Ana, who has computed b with $ab = 1 \bmod (p-1)(q-1)$ can read the secrete email y because $y^b = x^{ab} = x^{(p-1)(q-1)} = x \bmod n$. But Eve, has no chance because the only thing Eve knows is y and (n, a). It is believed that without the **factorization** of n, it is not possible to determine x. The message has been transmitted securely. The core difficulty is that **taking roots** in the ring $Z_n = \{0, \ldots, n-1\}$ is difficult without knowing the factorization of n. With a factorization, we can quickly take arbitrary roots. If we can take square roots, then we can also factor: assume we have a product $n = pq$ and we know how to take square roots of 1. If x solves $x^2 = 1 \bmod n$ and x is different from 1, then $x^2 - 1 = (x-1)(x+1)$ is zero modulo n. This means that p divides $(x-1)$ or $(x+1)$. To find a factor, we can take the greatest common divisor of $n, x-1$. Take $n = 77$ for example. We are given the root 34 of 1. ($34^2 = 1156$ has reminder 1 when divided by 34). The greatest common divisor of $34-1$ and 77 is 11 is a factor of 77. Similarly, the greatest common divisor of $34+1$ and 77 is 7 divides 77. Finding roots modulo a composite number and factoring the number is equally difficult.

Cipher	Used for	Difficulty	Attack
Cesar	transmitting messages	many permutations	Statistics
Viginere	transmitting messages	many permutations	Statistics
Enigma	transmitting messages	no frequency analysis	Plain text
Diffie-Helleman	agreeing on secret key	discrete log mod p	Unsafe primes
RSA	electronic commerce	factoring integers	Factoring

The simplest **error correcting code** uses 3 copies of the same information. A single error can be corrected. With 3 watches for example, you know the time even if one of the watches fails. Cockpits of airplanes have three copies important instruments. But this basic error correcting code is not efficient. It can correct single errors by tripling the size. Its efficiency is 33 percent. A cheap way to make it more efficient is to compress the data first and then make three copies. **Data compression** is a topic by itself. Here is a simple example, the **dictionary compression**. Take dictionary with $65'536 = 2^{16}$ words for example. Every word can be encode by two bytes. Assuming an average word length of 6, we can encode every word with 2 bytes instead of 6. There are better error correcting codes using linear algebra or algebraic geometry.

Lecture 11: Cryptology

Cryptology

Cryptology is the science of constructing and breaking codes. It consist of **cryptography**, the creation of codes and **cryptanalysis**, the theory of cracking codes.

A bit outside but still part of information theory is the construction of **error correcting codes**. The purpose of the later is the building of protocols allowing the transmission of information to be more secure. The goal is data corruption or data loss can be reversed by adding redundant information. Already the DNA, encoding the blueprints of life have redundancy built in.

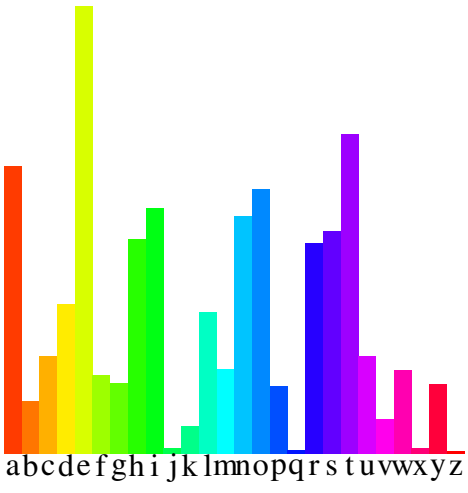
What kind of mathematics is involved? The theory has ties with **probability theory**. Especially in the code breaking part **statistical methods** are useful. Many codes are based on **number theory** like RSA and Diffie-Hellman. **Combinatorial** considerations come into play when looking at the complexity of codes. Especially in code breaking like with plain text attacks. **Algebraic geometry** has entered through examples like **elliptic curve cryptosystems**. In general, **algebra** enters if algebraic objects like number fields are used. New branches like **quantum cryptology** use analysis like Fourier theory.

Substitution ciphers

1) **Cesar ciphers** permute the alphabet. Examples are:

Cesar:	shift three to the left	F becomes C for example
Augustus:	shift to the right	F becomes G.
Atbash:	reflect	B becomes Y and Y becomes B.
Rot13:	move to middle	A Becomes N and N becomes A.

First known attacks using frequency analysis Al Kindi in 9'th century.



2) **Polyalaphabetic ciphers** permute with different alphabets. Examples:

Alberti	Random change of alphabet indicating switch
Trithemius	Deterministic change of alphabet
Viginere	Using key telling which alphabet to use
Enigma	Using key and deterministic alphabet change overlapped with Cesar
Hill Cipher	Use matrices to permute

Block ciphers

Cut text into larger chunks and scramble them. Examples:

DES	Data Encryption Standards 1973
Triple DES	Used for some electronic payments, 1998

Public Key Cyphers

Depends often on Number theoretical mathematical difficulties like factoring integers.

Diffie-Hellman Key exchange	1976 (1974 at GCHQ in England)
RSA encryption	1978 (1973 at GCHQ in England)

The RSA method allows Ana to submit messages to Bob on a public channel which a third party Eve can not read.

Ana publishes a **RSA pair**: (n, a) of integers into the public. The factorization $n = pq$ is secret and $a < (p-1)(q-1)$ is such that there exists b with $ab = 1 \mod (p-1)(q-1)$. Bob sends a secrete message to Ana by transmitting

$$y = x^a \mod n \ .$$

Ana can read the email by computing

$$y^b \mod n \ .$$

The key is public. Still, one can not read the messages unless a hard mathematical problem, the factorization of n is solved.

Diffie Hellman method allows Ana and Bob to exchange keys on a public channel which a third party Eve can not read.

A prime p and primitive root a are given and public. A primitive root is a number for which a^k generates all numbers different from 0 modulo p. Its a number for which the "log" exists.

Ana choses a secret number x and publishes $u = a^x \mod p$.
Bob choses a secret number y and publishes $v = a^y \mod p$.
Ana can compute $v^x = a^{xy}$ and Bob can compute $v^y = a^{yx}$ (always modulo p) But Eve, the eves dropper can not get x, y from u, v.