Introduction to Cryptography, Part I

What is Cryptography?

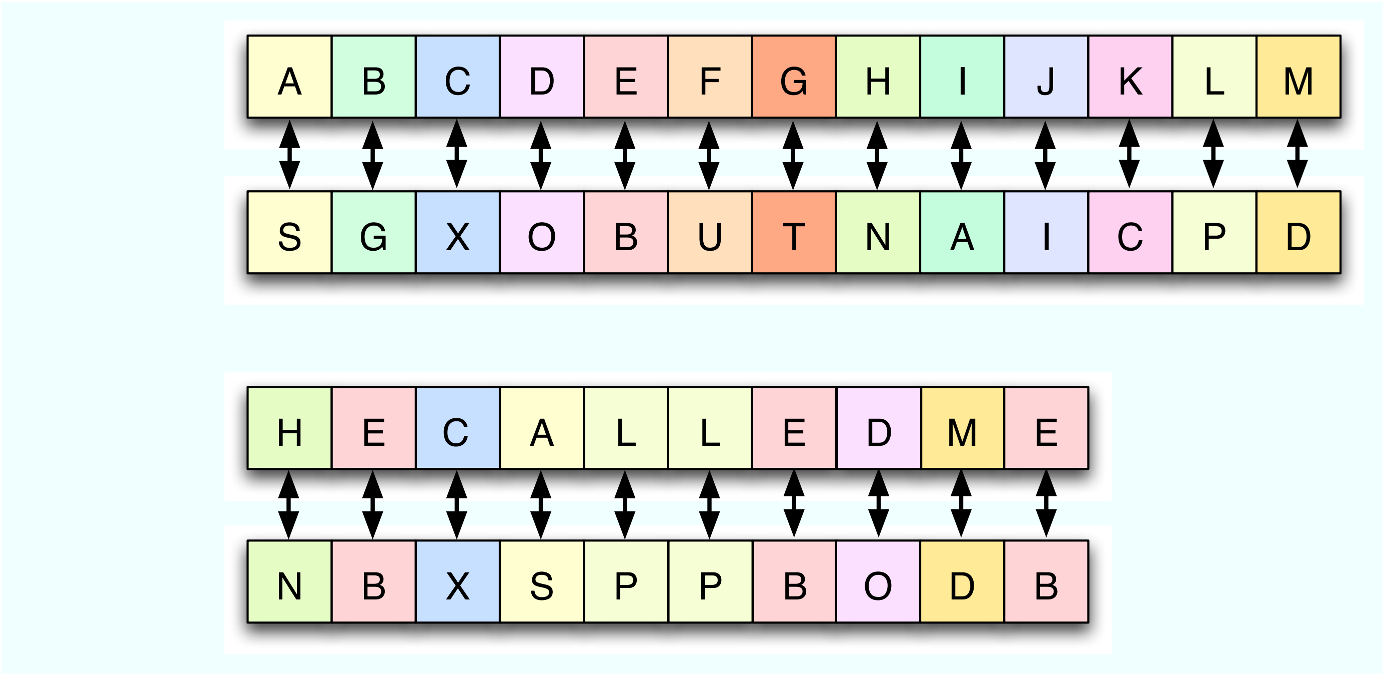
Cryptography – *the art of writing or solving codes*

Goals of modern cryptography – confidentiality, data integrity, authentication, non-repudiation, privacy,. . .

Ancient Crypto

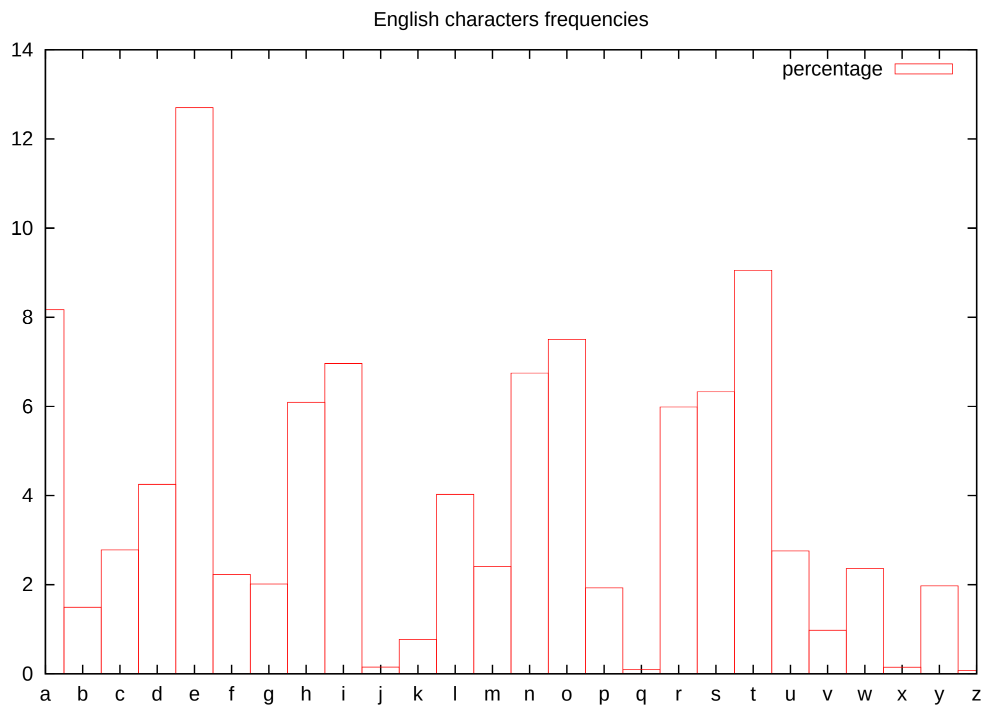
Historical ciphers - transforming alphabet characters to different alphabet characters

- *Mono-alphabetic substitution* – map each character to a different character



Attacks on monoalphabetic cipher:

- Easy to learn patterns
- Frequency analysis



- *Vigenère cipher*

Plaintext:	`tellhimaboutme`
Key:	`cafecafecafeca`
Ciphertext:	`WFRQKJSFEPAYPF`

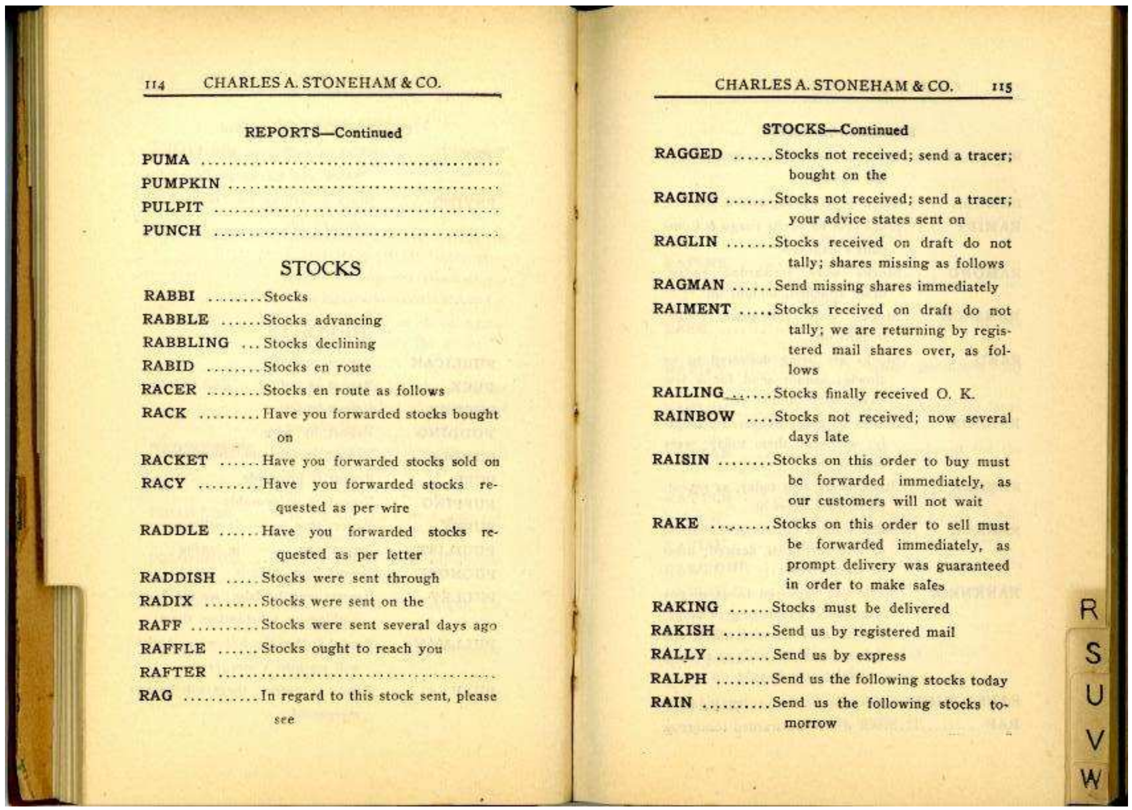
Breaking Vigenère cipher:

- break the ciphertext into pieces of length the length of the key and analyze as shift cipher
- detect patterns of short words (*the, and*)

Codes vs. Ciphers

- Ciphers operate *syntactically*, on letters or groups of letters: $A \longrightarrow D$, $B \longrightarrow E$, etc.
- Most were aimed at economy
- Secrecy from casual snoopers was a useful side-effect, but *not* the primary motivation

A 1910 Commercial Codebook



114 CHARLES A. STONEHAM & CO.

REPORTS—Continued

PUMA ....
PUMPKIN ....
PULPIT ....
PUNCH ....

STOCKS

RABBI ....Stocks
RABBLE ......Stocks advancing
RABBLING ...Stocks declining
RABID ....Stocks en route
RACER ....Stocks en route as follows
RACK ....Have you forwarded stocks bought on
RACKET ......Have you forwarded stocks sold on
RACY ....Have you forwarded stocks requested as per wire
RADDLE ......Have you forwarded stocks requested as per letter
RADDISH .....Stocks were sent through
RADIX ....Stocks were sent on the
RAFF ....Stocks were sent several days ago
RAFFLE ......Stocks ought to reach you
RAFTER ....
RAG ....In regard to this stock sent, please see

CHARLES A. STONEHAM & CO. 115

STOCKS—Continued

RAGGED ......Stocks not received; send a tracer; bought on the
RAGING ....Stocks not received; send a tracer; your advice states sent on
RAGLIN ....Stocks received on draft do not tally; shares missing as follows
RAGMAN ......Send missing shares immediately
RAIMENT .....Stocks received on draft do not tally; we are returning by registered mail shares over, as follows
RAILING.......Stocks finally received O. K.
RAINBOW ....Stocks not received; now several days late
RAISIN ....Stocks on this order to buy must be forwarded immediately, as our customers will not wait
RAKE ....Stocks on this order to sell must be forwarded immediately, as prompt delivery was guaranteed in order to make sales
RAKING ......Stocks must be delivered
RAKISH ....Send us by registered mail
RALLY ....Send us by express
RALPH ....Send us the following stocks today
RAIN ....Send us the following stocks tomorrow

Encryption

$$plaintext \xrightarrow{encryption} ciphertext \xrightarrow{decryption} plaintext$$

- *plaintext, cleartext* – original message
- *ciphertext* – mangled information
- *key* – additional information used for encryption and decryption

How to guarantee security?

Ad-Hoc Approach:

- Build a protocol.
- Let many people try to break it.
- Fix the break.
- Repeat the above two steps.

Bad idea - no security guarantees, adversaries do not report breaks

Modern Cryptography Approach

- Exact Security Definitions
  - What adversarial power is assumed?
  - What is considered to be a break?
- Reliance on precise assumptions – what unproved assumption are made (e.g. factoring is hard)
- Rigorous proofs - if the adversary is limited to his assumed power security break is not possible

Encryption Scheme

(GEN, ENC, DEC)

- GEN(security parameter) $\longrightarrow$ key
- ENC(M, key) $\longrightarrow$ C
- DEC(C, key) $\longrightarrow$ M

What should be kept secret?

- Algorithm should be public.
  - Bad guys will find out about it eventually and will exploit weaknesses.
  - Has to be public to give proofs, higher chance to find and make known weaknesses so that algorithm will not be used.
- Decryption key has to be secret.

Defining Security

Given the ciphertext the adversary should not be able to recover the original message.

Is it OK if the adversary is able to recover half of the message?

Defining Security

No. The adversary should not be able to recover half the message.

What about just one bit of the message?

Defining Security

No. The adversary should not be able to learn any information about the message.

Given a ciphertext, the probability of any possible plaintext being encrypted should be the same.

Semantic Security

Experiment $Priv_{\mathcal{A}}^{eav}$:

1. The adversary $\mathcal{A}$ chooses two messages m_0 and m_1.
2. A random key k is generated running GEN and a random $b \leftarrow \{0,1\}$ is chosen. The ciphertext $c = \text{ENC}(m_b)$ is given to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b'. (*The adversary computes b' using all its power to break the encryption algorithm.*)
4. The output of the experiment is $Priv_{\mathcal{A}}^{eav} = 1$ if $b = b'$ and $Priv_{\mathcal{A}}^{eav} = 0$ if $b \neq b'$

An encryption scheme (GEN, ENC, DEC) is semantically secure if $Pr[Priv_{\mathcal{A}}^{eav} = 1] = \frac{1}{2}$.

Chosen Plaintext Security

1. A key k is chosen running GEN.
2. **The adversary $\mathcal{A}$ can choose any text m and obtain ENC$_k(m)$. (oracle access to ENC$_k$)**
3. $\mathcal{A}$ chooses two messages m_0 and m_1.
4. A random $b \leftarrow \{0, 1\}$ is chosen. The ciphertext $c = \text{ENC}(m_b)$ is given to $\mathcal{A}$.
5. **The adversary $\mathcal{A}$ can choose any text m and obtain ENC$_k(m)$.**
6. $\mathcal{A}$ outputs a bit b'.
7. The output of the experiment is $Priv_{\mathcal{A}}^{cpa} = 1$ if $b = b'$ and $Priv_{\mathcal{A}}^{cpa} = 0$ if $b \neq b'$

Note: ENC should not be deterministic, i.e. should use randomness.

Chosen Ciphertext Security

1. A key k is chosen running GEN.
2. **The adversary $\mathcal{A}$ can choose any text m and obtain ENC$_k(m)$. $\mathcal{A}$ can also choose any ciphertext c and obtain DEC(c).**
3. $\mathcal{A}$ chooses two messages m_0 and m_1.
4. A random $b \leftarrow \{0, 1\}$ is chosen. The ciphertext $c =$ ENC(m_b) is given to $\mathcal{A}$.
5. **In CCA1 security the adversary $\mathcal{A}$ can choose any text p and obtain ENC$_k$(m). In CCA2 security $\mathcal{A}$ can further choose any ciphertext $c' \neq c$ and obtain DEC(c').**
6. $\mathcal{A}$ outputs a bit b'.
7. The output of the experiment is $Priv_{\mathcal{A}}^{cca} = 1$ if $b = b'$ and $Priv_{\mathcal{A}}^{cca} = 0$ if $b \neq b'$.

Computational Security

Two relaxations:

- Security is only preserved against *efficient* adversaries that run in feasible amount of time (e.g. cannot enumerate all keys).
- Adversaries can potantially suceed with some *very small probability* (e.g. guess at random and succeed).

An encryption scheme (GEN, ENC, DEC) is semantically secure in the presence of probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ if $Pr[Priv_{\mathcal{A}}^{eav} = 1] \leq \frac{1}{2} + negl$.

Secret Key (Symmetric) Cryptography

Sender and receiver share the **same** key used for encryption and decryption.

$$\begin{array}{ccc} \text{plaintext} & \xrightarrow{\text{encryption}} & \text{ciphertext} \\ & \uparrow & \\ & \text{key} & \\ & \downarrow & \\ \text{ciphertext} & \xrightarrow[\text{decryption}]{} & \text{plaintext} \end{array}$$

Security Uses of Secret Key Cryptography

- Transmitting over insecure channel
- Secure storage in insecure media
- Authentication
- Integrity check

Pseudorandomness

Pseudorandom sequence of strings of length l is indistinguishable from the uniform distribution of strings of length l.

A ***pseudorandom generator*** (PRG) is a determnistic polynomial-time algorithm that takes a short truly random seed and stretches it into a long string that is pseudorandom.

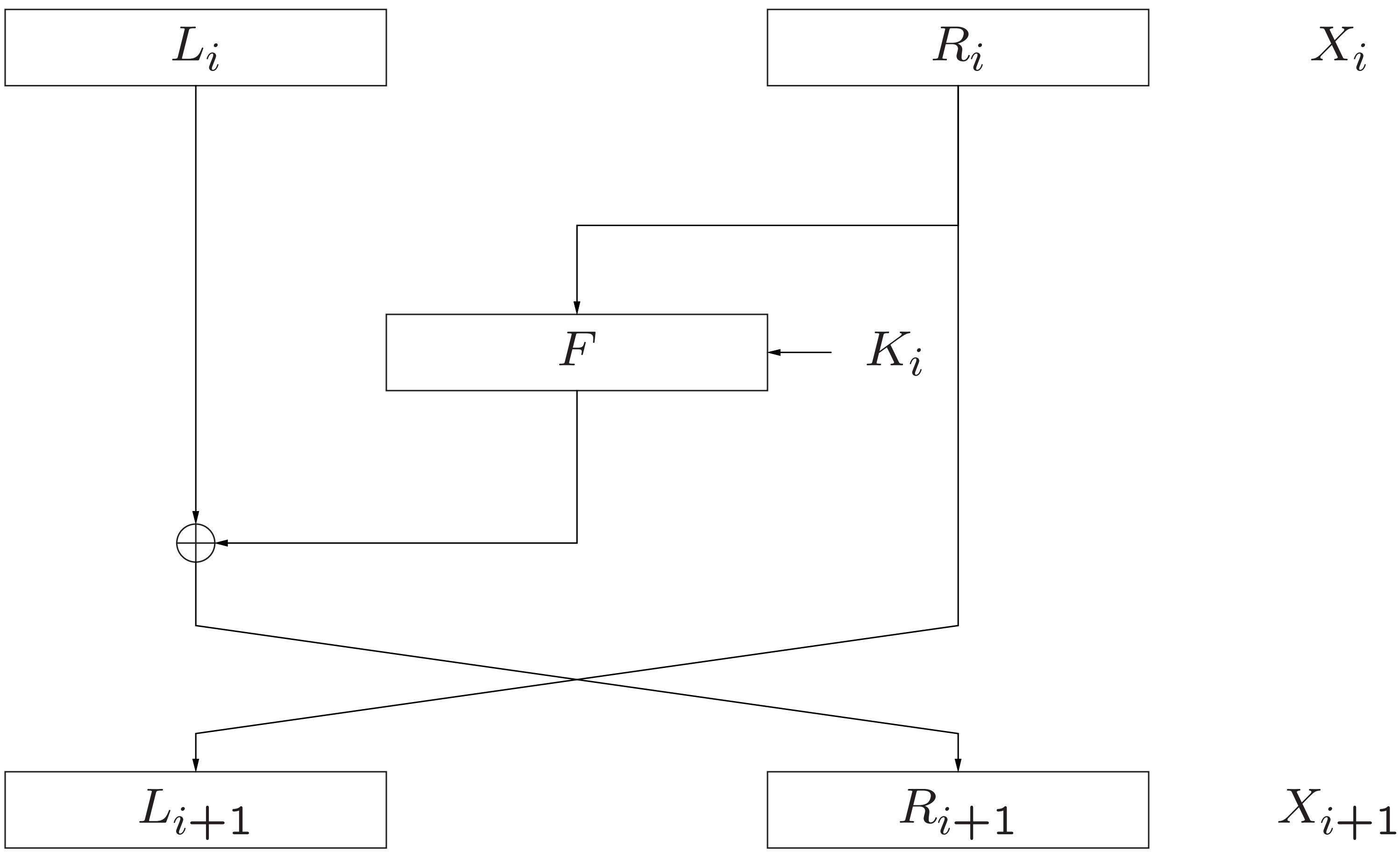
Stream Ciphers

- Generate a stream of pseudorandom bits and XORs it with plaintext.
- A common error - multiple encryptions using a stream cipher (appears in an implementation of Microsoft Word and Excel)
- Secure multiple encryption using a stream cipher:
  - *Synchronized mode* – computing parties use different parts of the stream output to encrypt
  - *Unsynchronized mode* – the PRG takes as inputs a ***seed*** and an ***initializing vector*** (IV); pseudorandom even when IV is known
  - RC4; the first first few bytes of its output are known to be biased (this weakness can be used to break WEP encryption in 802.11 wireless networks); the first 1024 bits should be discarded

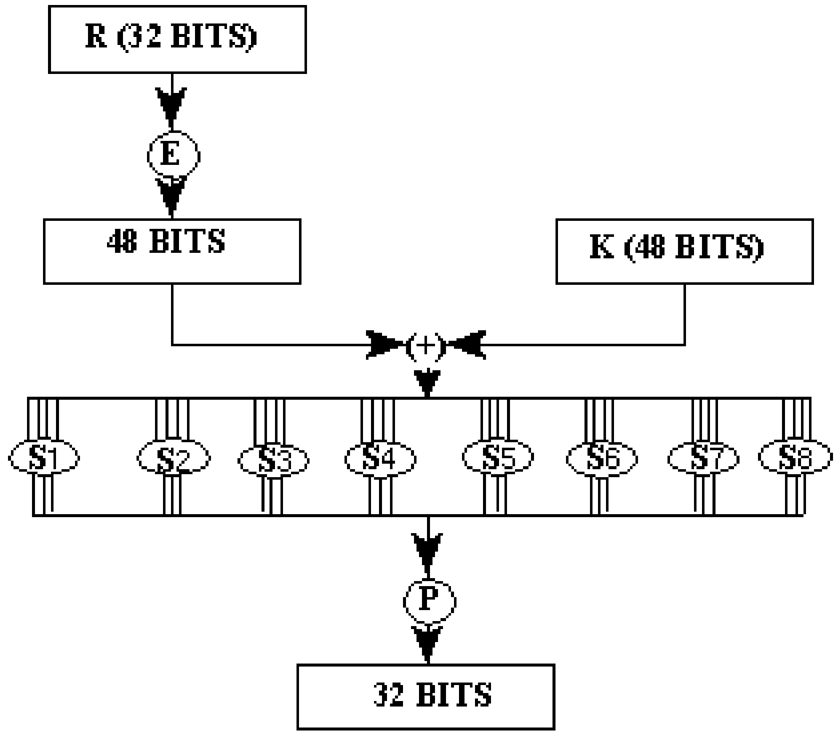
Block Ciphers

- Strong pseudorandom permutations:
  - $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
  - For each k $F_k(\cdot)$ is one-to-one.
  - For each k chosen uniformly at random $F_k(\cdot)$ is indistinguishable from a random permutation.
- Operate on a fixed-length set of bits
- Output blocksize generally the same as input blocksize
- Optional key scheduling — convert supplied key to internal form
- Multiple *rounds* of combining the plaintext with the key
- Examples: DES (56-bit keys; 64-bit blocksize; 16 rounds); AES (128-, 192-, and 256-bit keys; 128-bit blocksize, 9-13 rounds, depending on key length)

DES Round Structure



The Feistel (F) Funciton



How DES Works

For each round:

1. Divide the input block in half. The right half of each round becomes the left half of the next round’s input.
2. Take the right half, pass it through a non-linear function of data and key, and exclusive-OR the result with the current input’s left half.
3. The output of that function becomes the right half of the next round’s input.
4. This is known as a *Feistel network*

Decryption

- Run the rounds backwards
- In the example, L_{i+1} is passed unchanged to the previous round (as R_i)
- Accordingly, it can be fed into $F(K_i)$ to be XORed with R_{i+1} to produce L_i

What's Wrong with DES?

- The key size is too short — a machine to crack DES was built in 1998.
- (Charges that NSA could crack DES were leveled in 1979. But the claim that NSA designed in a back door are false.)
- The blocksize is too short.
- It depends on bit-manipulation, and is too slow in software

Selecting the Advanced Encryption Standard

- NIST issued an open call for submissions
- 15 ciphers were submitted, from all over the world
- Several open conferences were held (and the NSA did its own private evaluations)
- 5 ciphers were eliminated as not secure enough
- 5 more were dropped for inefficiency or low security margin
- Of the 5 finalists, Rijndael — a Belgian submission — was chosen because of good security and very high efficiency across a wide range of platforms

How Does Rijndael Work?

- Input block viewed as a byte array; key viewed as a two-dimensional matrix
- Each round consists of a series of simple, byte-oriented operations: ByteSubstitution, ShiftRow, MixColumn, AddRoundKey.
- The key is mixed with the entire block in each round
- The basic operations are individually reasonably tractable mathematically, but are combined in a hard-to-invert fashion.