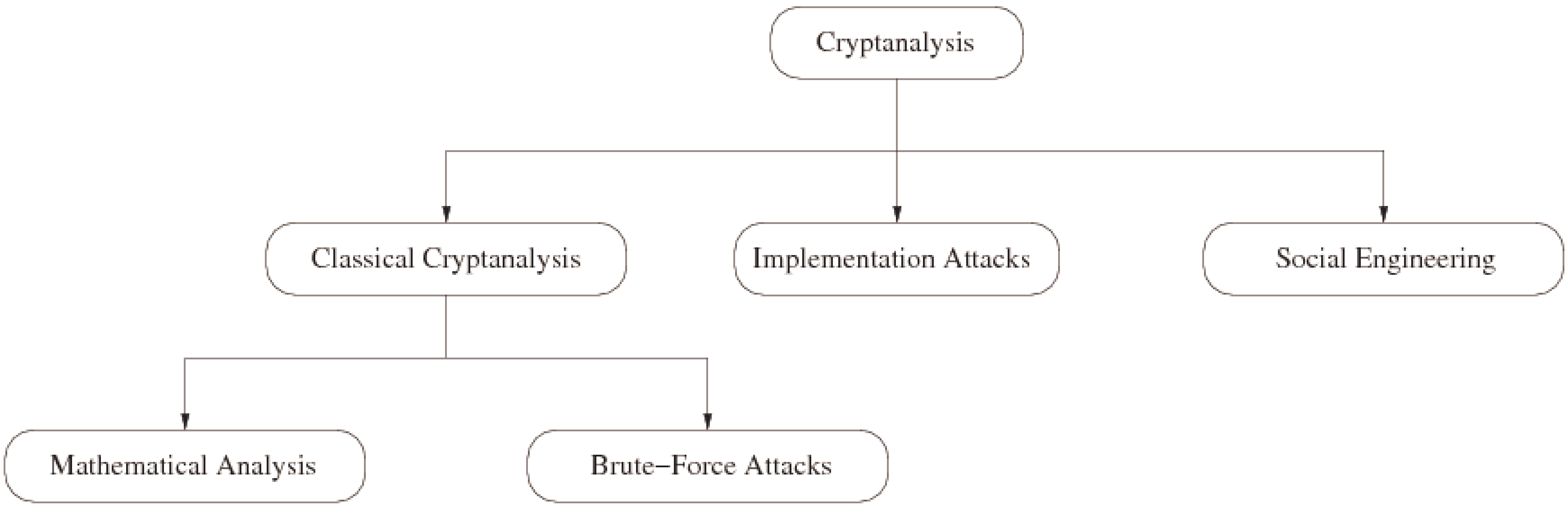
Information Theory

Cryptanalysis

- Reduced entropy means reduced security



- How can you reduce entropy?

Cryptanalysis

- Social Engineering
  - Psychological manipulation of people to perform actions or divulge confidential information
  - Phishing
  - Impersonation on help desk calls
  - Physical access (tailgating, shoulder surfing, dumpster diving)
  - Stealing important documents
  - Baiting - Providing Fake software, Trojans

Cryptanalysis

- What can be gained from a social engineering attack that can reduce the entropy of a cipher?
  - Step 1 – name and describe an attack
  - Step 2 – describe info gained
  - Step 3 – quantify loss of entropy

Implementation attacks

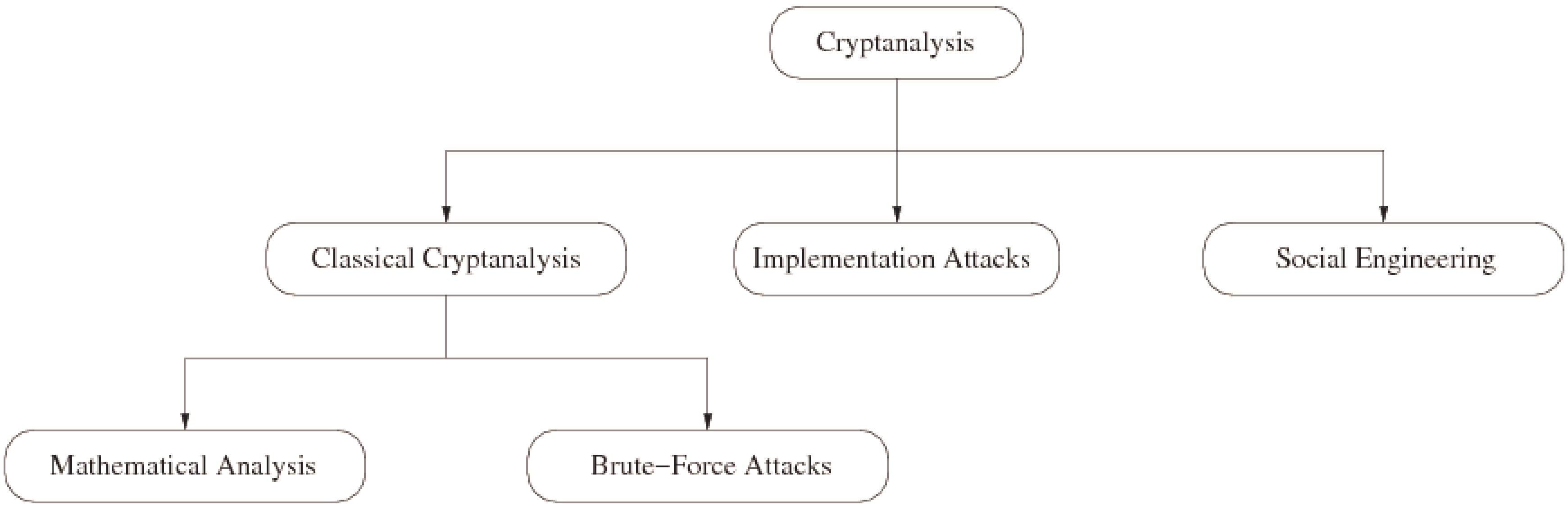
- Side channel attacks
- Timing attacks – study the time of computations
- Power monitoring attacks - study the power of computations
- Remanence/Acoustic/Electromagnetic attacks – study the physical residues of computation
- Differential fault analysis – provide bad data
- Row hammer (rowhammer) – access and change adjacent memory

Implementation attacks

- Countermeasures
  - Eliminate or reduce the release of information from a crypto system
  - Eliminate the relationship between the leaked information and the secret data
- Describe how an implementation attack can reduce the entropy of a cipher?

Cryptanalysis

- Reduced entropy means reduced security



- How can you reduce entropy?

Classical Cryptanalysis Mathematical Attacks

- A large class of attacks that exploit the underlying mathematical properties of keys, ciphertext and plaintext -> ciphertext transformation, including…
  - ***Integer factorization*** – best known modern example, factor primes used in RSA
  - ***Frequency analysis*** - frequency of letters or groups of letters in a ciphertext
  - ***Differential analysis*** - differences in information input can affect the resultant difference at the output
  - ***Linear analysis*** – affine approximations (abstracts the encryption device as a projection on a vector in a hyper plane)

Classical Cryptanalysis
Brute Force Attacks

- ***Brute Force attack***: An exhaustive key search that treats the cipher as a black box and checks all possible keys until condition is fulfilled:

$$d_K(y_0) = x_0$$

- Requires (at least) 1 plaintext-ciphertext pair (x0, y0)

Classical Cryptanalysis
Brute Force Attacks

- Computational Security assumes this attack is the most effective
- Helps think about security in terms of key space and key size

Key length in bit	Key space	Security life time (assuming brute-force as best possible attack)
64	2^{64}	**Short term** (few days or less)
128	2^{128}	**Long-term** (several decades in the absence of quantum computers)
256	2^{256}	**Long-term** (also resistant against quantum computers – note that QC do not exist at the moment and might never exist)

Classical Cryptanalysis Brute Force (BF) Attacks

- Given n operations required to test a key
  - n - dependent on size of input
- Given rate r of testing n operations in a time period
  - *i.e. r = n* operations / Year
- Given $|\mathrm{K}|$ keys (cardinality of the set of all keys)
- Calculate the time, τ_{BF}, required to break a cipher as the worst case time to test every key

$$\tau_{\mathrm{BF}} = (|\mathrm{K}| * \mathrm{n}) / \mathrm{r}$$

Classical Cryptanalysis Brute Force (BF) Attacks

- Calculate the up front cost, C_m , (manufacturing cost) as the per unit cost of a computer (c_c) times the number of computers necessary to achieve the testing rate r

$$C_m = c_c * (\text{number of computers})$$
$$C_m = c_c * (\mathrm{r} / \mathrm{r_c})$$

where r_c is the per computer rate of testing

Assumes BF attack can exploit parallelism

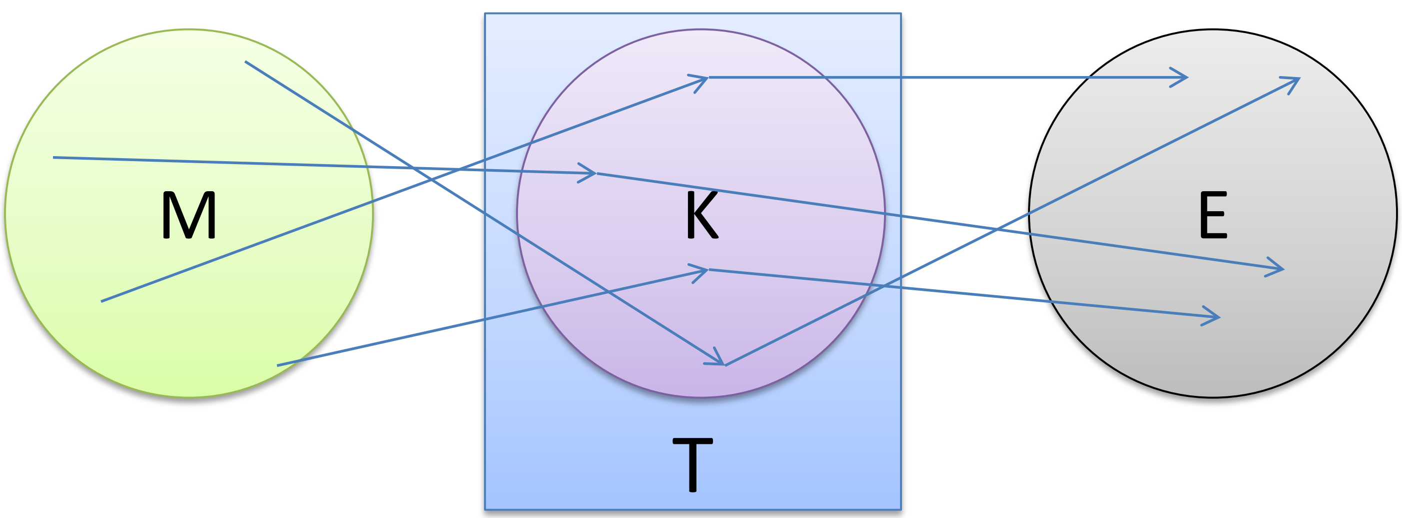
Shannon’s Transformations

- A general secrecy [*crypto*] system is one where

  Message *x from M* and Key k from *K* are independent *RV*

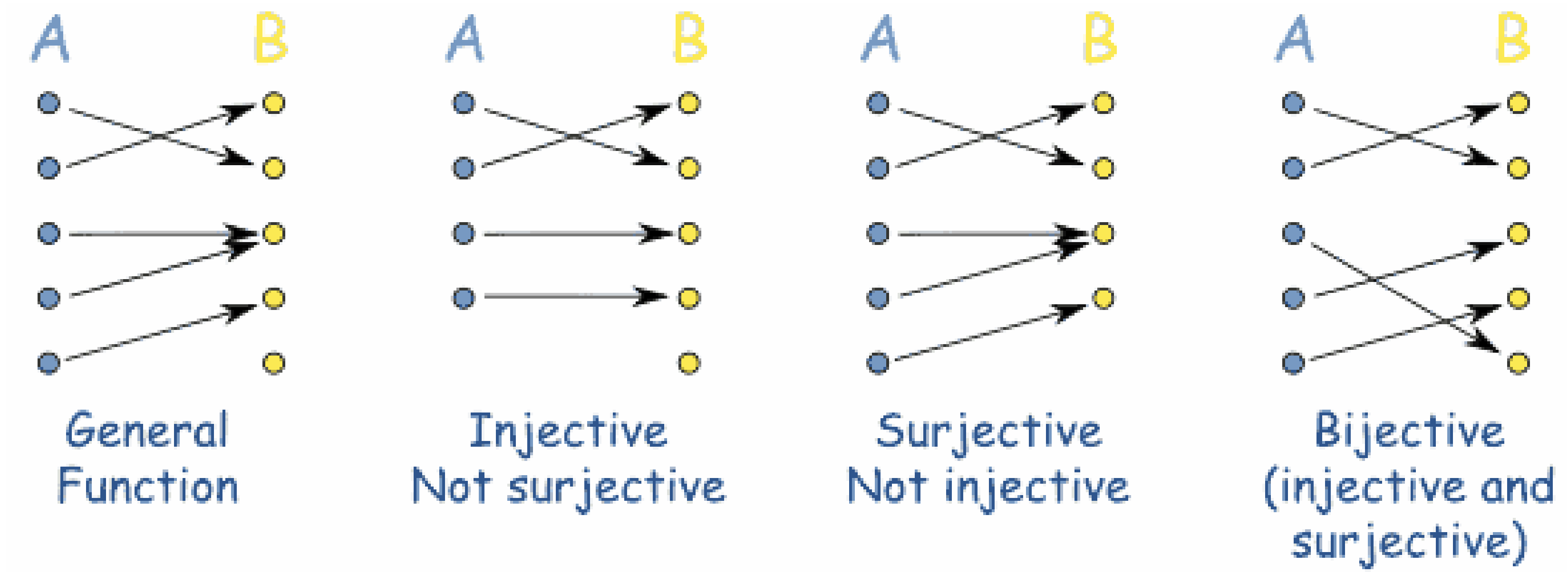
  Ciphertext y ∈ *E* is generated by *E* = *f*(x ∈*M*, k ∈ *K*)

  Transformation *T* maps plaintext to ciphertext



Function Behaviors

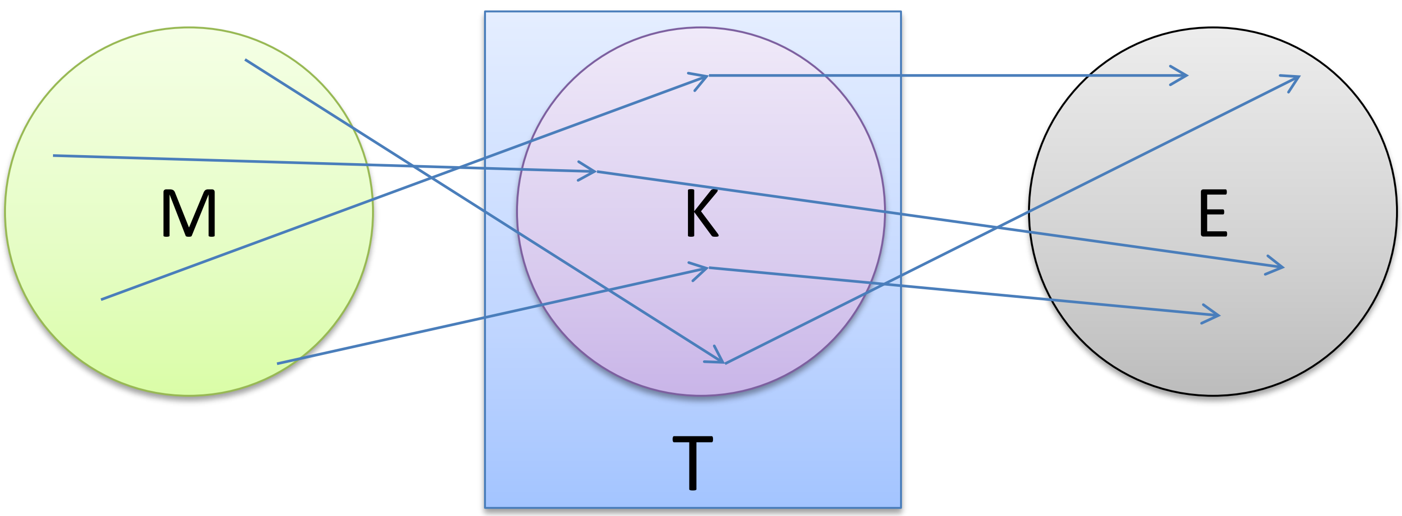
- Key space mapping, T
  - Encryption – typically bijective (one-to-one , onto)
  - Surjective makes decryption impossible
  - Injective limits key space mapping



Source: http://www.mathsisfun.com/sets/injective-surjective-bijective.html

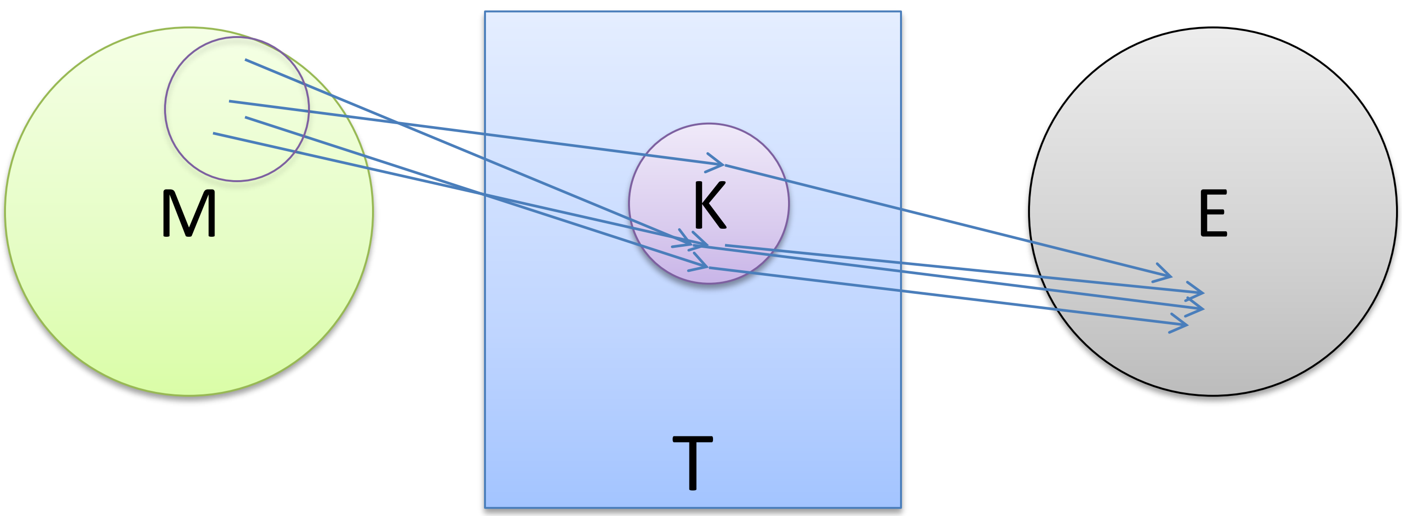
Shannon’s Transformations

- Entropy related to Key Equivocation
- Key Equivocation H(K|E) is
  - the amount of uncertainty of the key K
  - given the observation of the ciphertext E
- Ideal H(K|E) is shown



Shannon’s Transformations

- Key Equivocation H(K|E) in practical systems
  - Observation of E constrains the possible message space since key space is also limited
  - Ex: Shift Cipher: Given y, M can only be 1 of 26 messages



Shannon's Transformations

- Encryption function must completely obscure statistical properties of original message

- Proposed two types of properties that preserve Key Equivocation
  - **Confusion**
  - **Diffusion**

Confusion

- Confusion operation - make relation between statistics of ciphertext and the value of the encryption key as complex as possible
  - Focus on relationship between key and ciphertext
  - Promote independence of ciphertext space and key space
- Examples
  - XOR with random number
  - Complex substitution algorithm (using expansion)

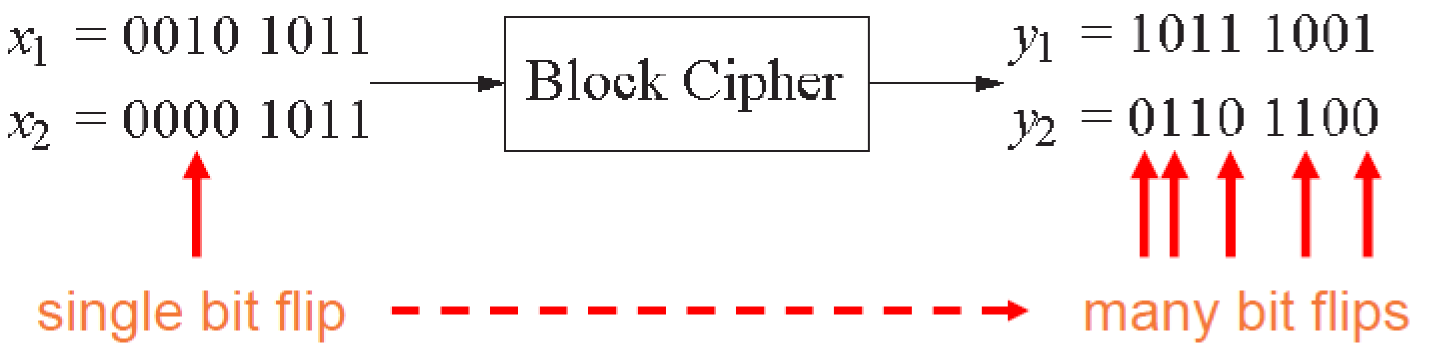
Diffusion

- Dissipates the statistical structure of plaintext over bulk of ciphertext.

- Diffusion operation - dissipate statistical structure of plaintext over bulk of ciphertext
  - Focus on relationship between ciphertext and plaintext
  - One change in the plaintext triggers an average of half of the symbols in the ciphertext to change

Diffusion

- Substitution - plain text character only affected one cipher text character
- Diffusion uses *Permutation to* manipulate the order of bits according to some algorithm.
- Redistributes non uniformity in the ciphertext and makes non-uniformity harder to detect



Avalanche

- ***Avalanche***: a small change yields large effects in the output (averages greater than half of bits)
- Added by Fiestel's to strengthen Claude Shannon's concept of diffusion
- High avalanche impact is desirable in a cryptographic algorithm

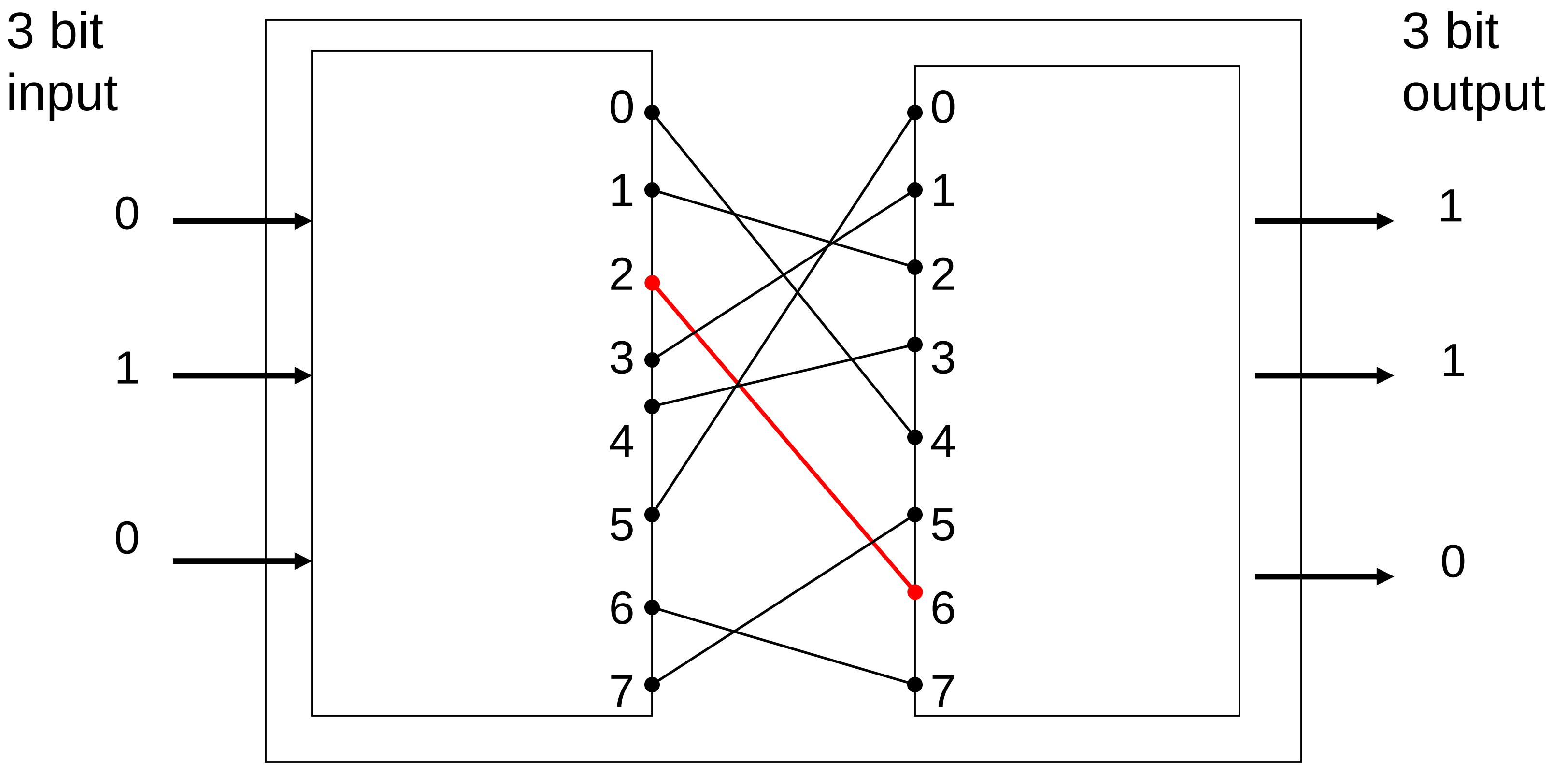
Shannon’s building blocks

- Shannon proposed two components
  - S-Boxes -- *substitution*
    - providing confusion of input bits
  - P-Boxes -- *permutation*
    - providing diffusion across S-box inputs

Substitution Ciphers

- Symbols are replaced by other symbols according to a key.
- Substitution adds *confusion*
- To escape frequency analysis, we can use a expanding substitution cipher
  - Map symbols to multiple symbols.
  - e.g 0 -> {01, 10}, 1->{00,11}
  - 011010010 becomes: 011100101101011110
  - Advantage: frequencies hidden
  - Disadvantage: message and key are longer

S-box (substitution)



Word size of 3 bits => mapping of $2^3 = 8$ values

Note: mapping must be reversible

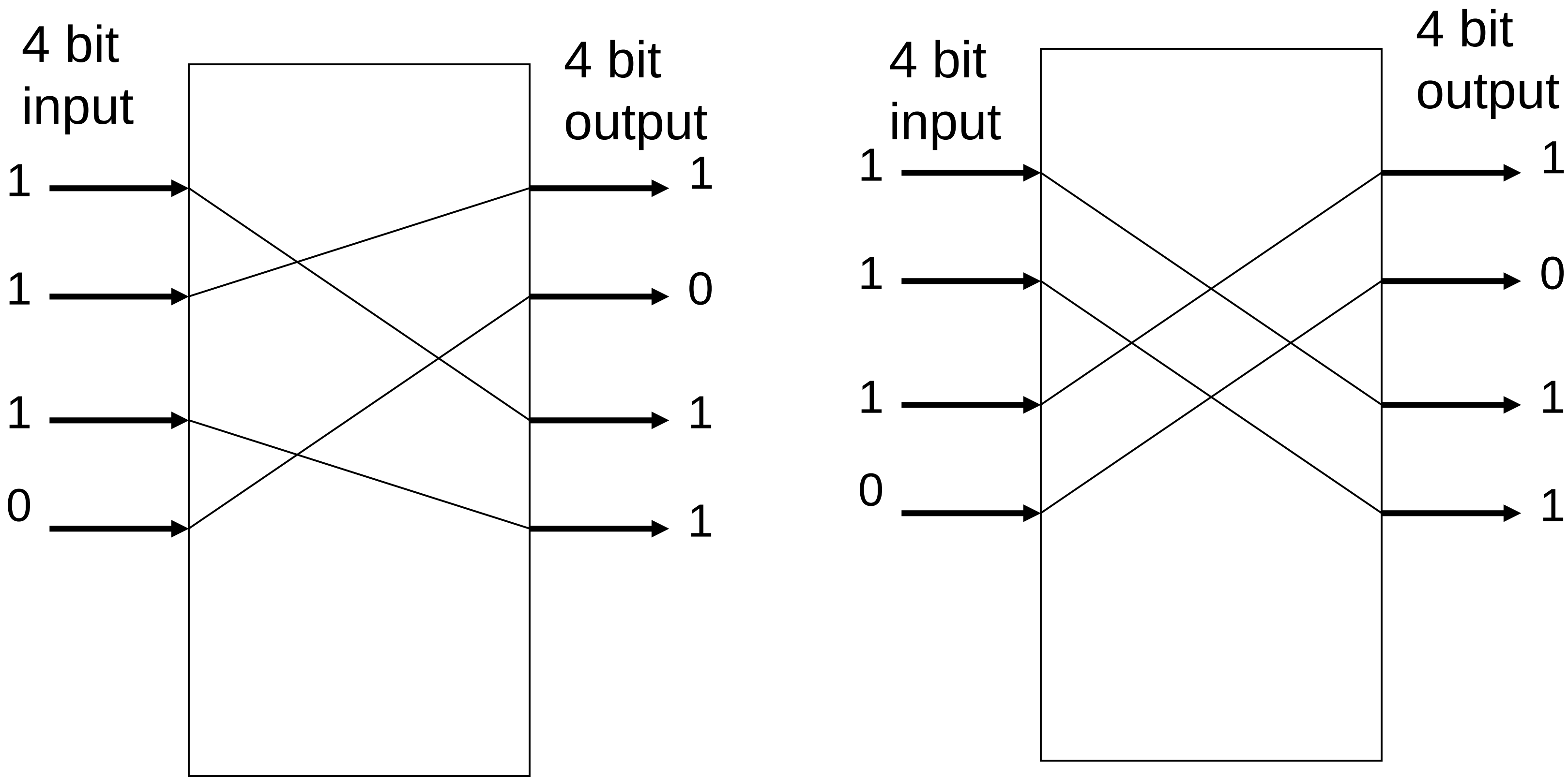
Permutation Ciphers

- A permutation cipher is one that rearranges the symbols of the message according to a preset pattern.
  - “Attack at dawn” becomes “cda tka wan tat”
- Permutation adds *diffusion*
- Helps avoid detection of symbols based on correspondence. (‘q’ followed by ‘u’)

- **Definition (*permutation*) –** Let S be a finite set of elements, a ***permutation p*** is a bijective function from S to itself

$$p(x): S \rightarrow S$$

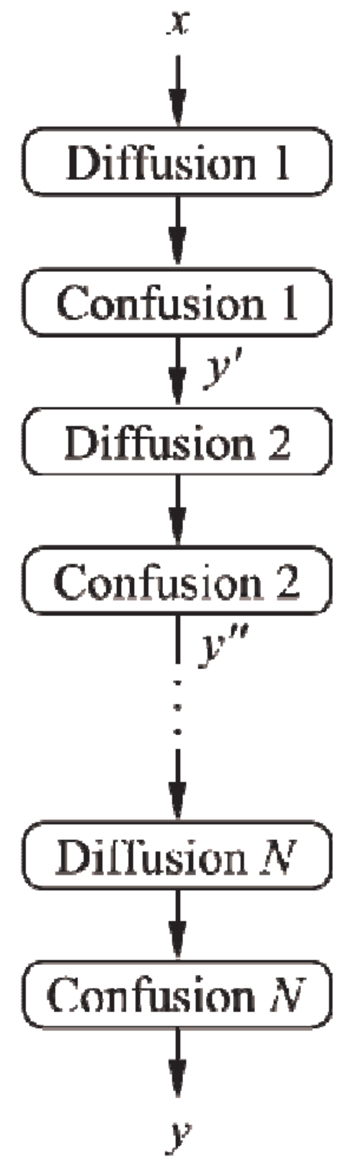
P-box (permutation)



Two examples of permutations. The second one switches halves of the input in the output.

Product Cipher

- By themselves, substitution and permutation ciphers are relatively insecure.
- By combining these operations, we can produce a secure cipher.
- M -> Sub_k(M) -> $Perm_k$(Sub_k(M)).
  - Might go through multiple rounds.



Product Cipher

- Alternate S and P boxes
- BUT, we must also decrypt
- So define the sequence of boxes so that precisely the same system will decrypt as well as encrypt
- Just run it backwards

$$x_i = e^{-1}{}_k(y_i) = e^{-1}{}_k(e_k(x_i))$$

Product Cipher Example

- S-box (substitution):
  - Invertible mapping of the input block into an output of the same or different size but with low correlation
- P-box (permutation):
  - Re-arranges the input bits