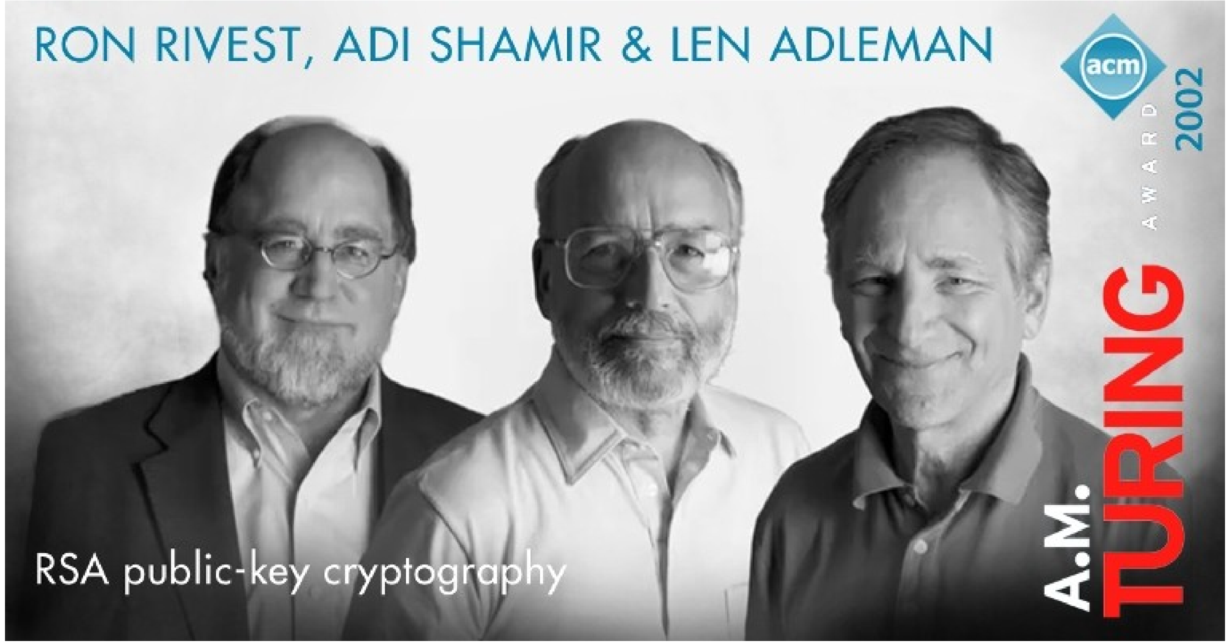
csci54 – discrete math & functional programming
RSA continued, error correction

RSA algorithm

- A very widely used public key encryption algorithm

- Three algorithmic components
  - key generation
  - encryption
  - decryption

- Our plan
  - What is the algorithm?
  - Why does it work?
  - How to implement it efficiently?



RSA - public key algorithm

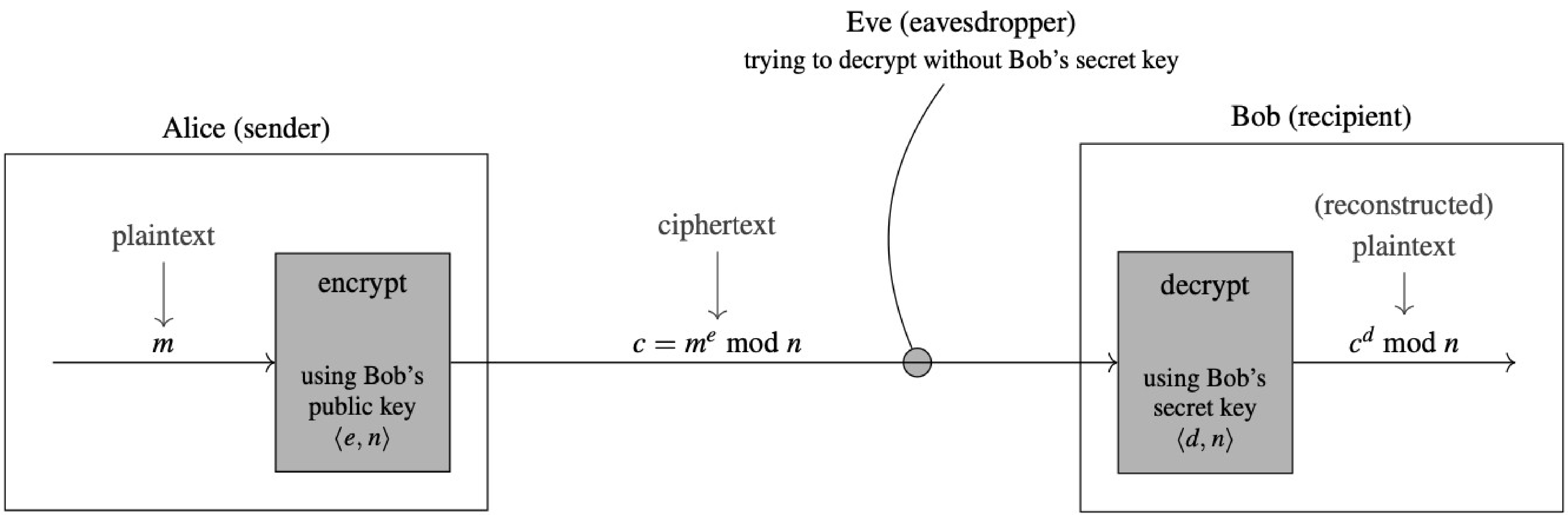


Figure 7.27 A schematic of the RSA cryptosystem, where $n = pq$ and $de \equiv_{(p-1)(q-1)} 1$, for prime numbers p and q.

possibly helpful video on RSA by Art of the Problem:
https://www.youtube.com/watch?v=wXB-V_Keiu8

RSA: implementing efficiently

- public key: (e,n) and private key: (d,n)
- encrypt(m) = $m^e \bmod n$
- decrypt(z) = $z^d \bmod n$

exponentiation

- key generation:
  - Choose a bit-length k
  - Choose two primes p and q which can be represented with k bits

    how to choose?
  - Let $n = pq$ so $\phi(n) = (p-1)(q-1)$
  - Find e such that $0 < e < n$ and $\gcd(e,\phi(n)) = 1$
  - Find d such that $(d*e) \bmod \phi(n) = 1$

    how to find?

Implementing RSA – key generation (part 1)

- computing primes p, q that are k bits long
  - pick a random number and test to see if it's prime
  - how?

Fermat's Little Theorem:

If p is prime and gcd(a,p) = 1, then a^{p-1} 1 mod p

Equivalently, a^{p} a mod p

```
prime-test(num):
     for i = 1:maxIter:
          pick a random number 1 < a <
num-1
          if not ( anum  a mod num )
              return False
        return True
```

Implementing RSA – key generation (part 2)

- finding d, e such that (d*e) mod ϕ(n) = 1

if gcd (a,b) = 1 then:
we say that a and b are <u>relatively prime</u>
there exists an integer c such that (a*c) mod b = 1

in fact, gcd(a,b)=1 if and only if there exists an integer c such that (a*c) mod b = 1

```
compute-de(n):
   pick random e, 0 < e < n
     try to find d such that (d*e) mod ϕ(n) =
1
           if none exists, try another e
           if one exists, we’re done!
```

Euclid's algorithm

What is the algorithm?
Why does it work?
How to implement it efficiently?

How to use it ethically?

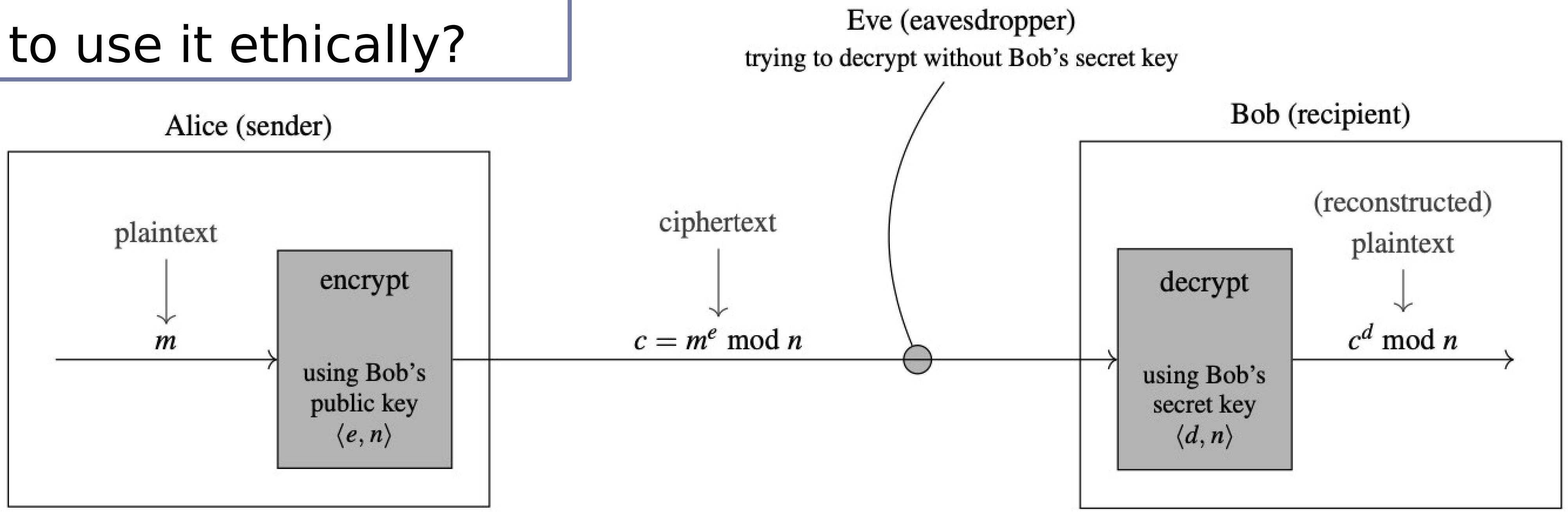


Figure 7.27 A schematic of the RSA cryptosystem, where $n = pq$ and $de \equiv_{(p-1)(q-1)} 1$, for prime numbers p and q.

The Moral Character of Cryptographic Work*

Phillip Rogaway

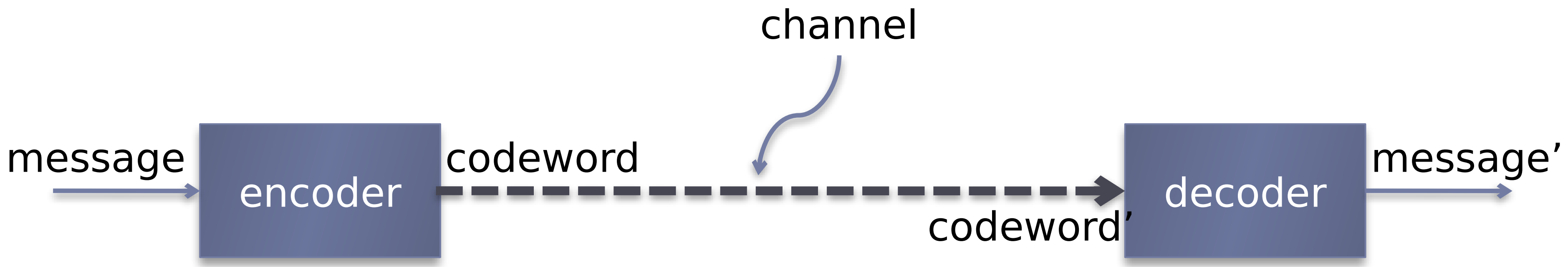
Department of Computer Science
University of California, Davis, USA
rogaway@cs.ucdavis.edu

December 2015
(minor revisions March 2016)

Abstract. Cryptography rearranges power: it configures who can do what, from what. This makes cryptography an inherently *political* tool, and it confers on the field an intrinsically *moral* dimension. The Snowden revelations motivate a reassessment of the political and moral positioning of cryptography. They lead one to ask if our inability to effectively address mass surveillance constitutes a failure of our field. I believe that it does. I call for a community-wide effort to develop more effective means to resist mass surveillance. I plead for a reinvention of our disciplinary culture to attend not only to puzzles and math, but, also, to the societal implications of our work.

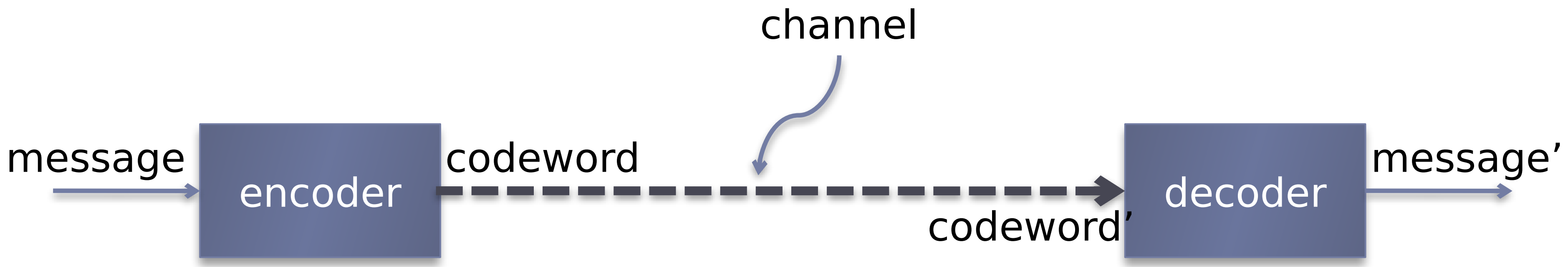
Keywords: cryptography · ethics · mass surveillance · privacy · Snowden · social responsibility

Transmitting information



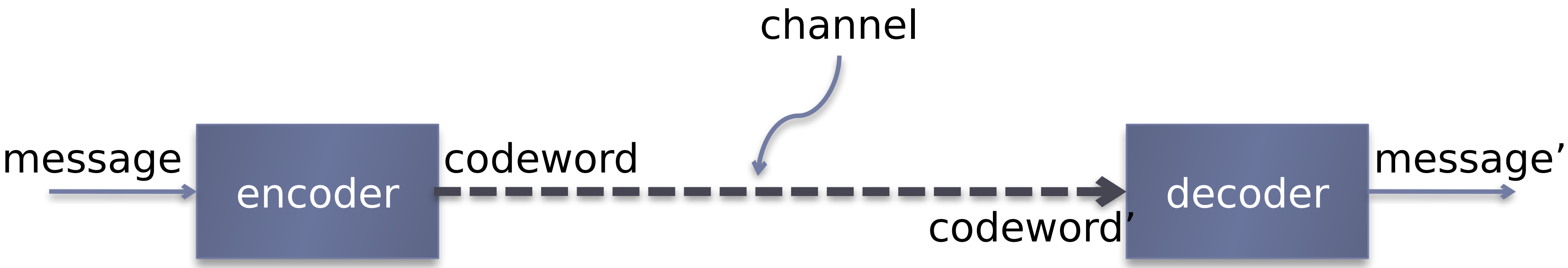
- cryptography
- error correction
- compression

Transmitting information – error correction



- goal is to recover message' = message even if codeword' != codeword
- why?
- how?
- assumptions
  - the message is a string of bits
  - codeword and codeword' have the same length

Error correction



- proposal 1:
  - encode: repeat each bit
  - 1001111   11000011111111
- proposal 2:
  - encode: triple each bit
  - 1001111 111000000111111111111

How do we evaluate?

Error correcting codes

- sender has a message m in $\{0,1\}^k$
- encoding turns m into a codeword c in $C=\{0,1\}^n$

- receiver gets some c' where |c'| = n and c' may not be in C
- decoding maps c' to closest element of C and decodes to m' $\{0,1\}^k$

How different are two strings?

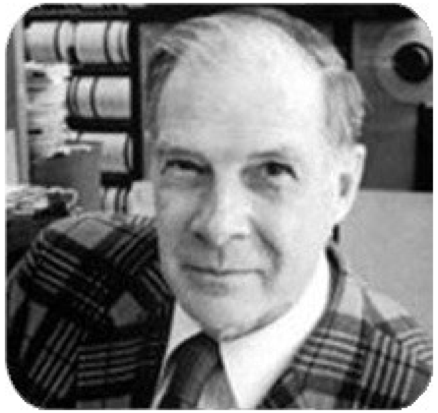
Definition 4.1: Hamming distance.

Let $x, y \in \{0,1\}^n$ be two n-bit strings. The *Hamming distance* between x and y, denoted by $\Delta(x, y)$, is the number of positions in which x and y differ. In other words,

$$\Delta(x,y) = \left|\left\{i \in \{1,2,\ldots,n\} : x_i \neq y_i\right\}\right|.$$

(Hamming distance is undefined if x and y don't have the same length.)

- (110, 000)
- (000111, 010101)



The way mathematics is currently taught it is exceedingly dull. In the calculus book we are currently using on my campus, I found no single problem whose answer I felt the student would care about!

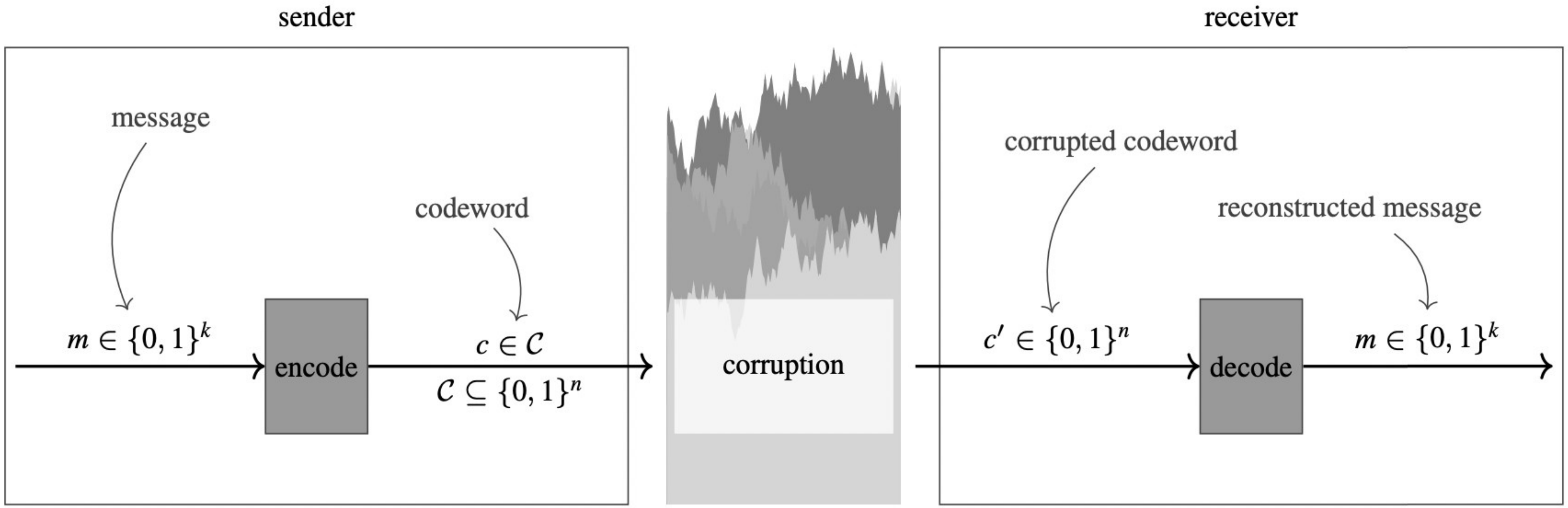
https://amturing.acm.org/award_winners/hamming_1000652.cfm

Error correcting codes

- sender has a message m in $\{0,1\}^k$
- encoding turns m into a codeword c in $\{0,1\}^n$

- receiver gets some c' where $|c'| = n$
- decoding maps c' to closest element of C and decodes to m'
- Definitions:
  - A code is a set $\{0,1\}^n$ for some integer $1 \leq k \leq n$
  - Any element of $\{0,1\}^k$ is called a message and the elements of the code are called codewords.

Error correcting codes



- sender has a message m $\{0,1\}^k$
- encoding turns m into a codeword
  c $\{0,1\}^n$ where c and || = 2^k

- receiver gets some c' where |c'| = k and c' may not be in
- decoding maps c' to closest element of and decodes to m' $\{0,1\}^k$

An example

- Consider the following code:

message	codeword
00	000000
01	000111
10	100001
11	101010

- What is k? What is n?
- How would you encode 10?
- How would you decode 111110?

Error detecting and correcting codes

- Let $\{0,1\}^n$ be a code and let n be a positive integer.

- We say that C can detect errors if, for any codeword c and for any number of up to k errors applied to c, we can correctly report error or no error.

- We say that C can correct errors if, for any codeword c and for any number of up to k errors applied to c, we can correctly identify that the original codeword was c.

Practice problem

- Consider the following code:

message	codeword
00	000000
01	000111
10	100001
11	101010

- What is k? What is n?
- How would you encode 10?
- How would you decode 111110?
- How many errors can this code detect? Correct?

Example

- Let's think about our original repetition codes

- proposal 1:
  - encode: repeat each bit
  - 1001111   11000011111111
- proposal 2:
  - encode: triple each bit
  - 1001111   111000000111111111111

- Assume messages have length 7. What is n? What is C? And how many errors can C detect? Correct?

Formally

- The <u>minimum distance</u> of a code C is the smallest Hamming distance between two distinct codewords in C.

- The <u>rate</u> of a code is the ratio between the message length (k) and the codeword length (n).

- Example:
  - code in which you repeat each of the n bits in a message
    - minimum distance: 2
    - rate: 1/2

Practice

- The minimum distance of a code C is the smallest Hamming distance between two distinct codewords in .
- The rate of a code is the ratio between the message length (k) and the codeword length (n).

- What is the minimum distance? What is the rate?
  1. triple each of the k bits in the message
  2. practice code

message	codeword
00	000000
01	000111
10	100001
11	101010

Relating the minimum distance and the rate

- Let t be any positive integer. If the minimum distance of a code C is 2t+1, then C can detect 2t errors and correct t errors.

- Some questions:
  - How can you design a code that detects as many errors as possible?
  - How likely are you to hit the worst case?