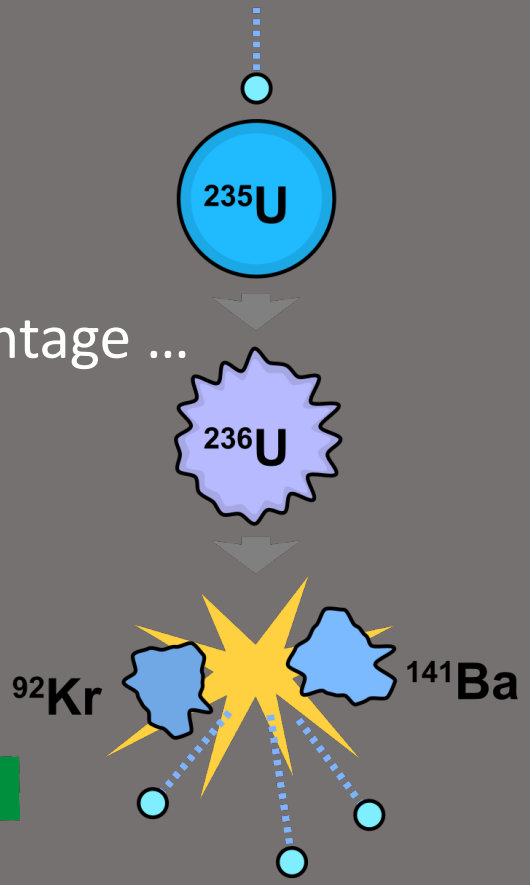
Security & Cryptography

Prof. Darrell Long

CSE 13S

Security Threats

- But why!??
  - For the fun of it …
  - Financial gain …
  - Commercial advantage …
  - Espionage …
  - Terrorism …



Goal	Threat
Data *confidentiality*	Exposure of data
Data *integrity*	Tampering with data
System *availability*	Denial of service

Cryptography

Cryptography

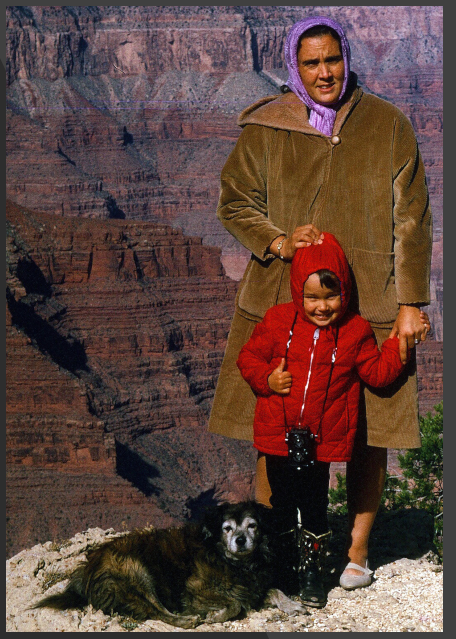
Engineering

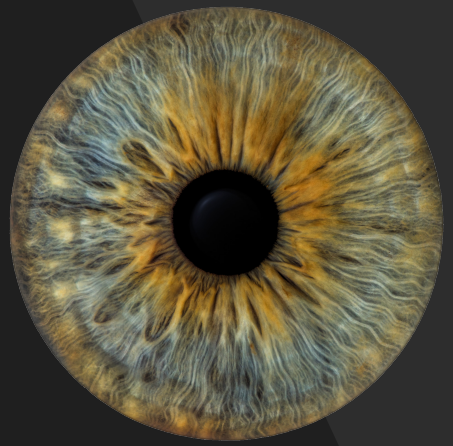
Who are your adversaries?

If they target you, they will get you.

First, you must prove who you are!

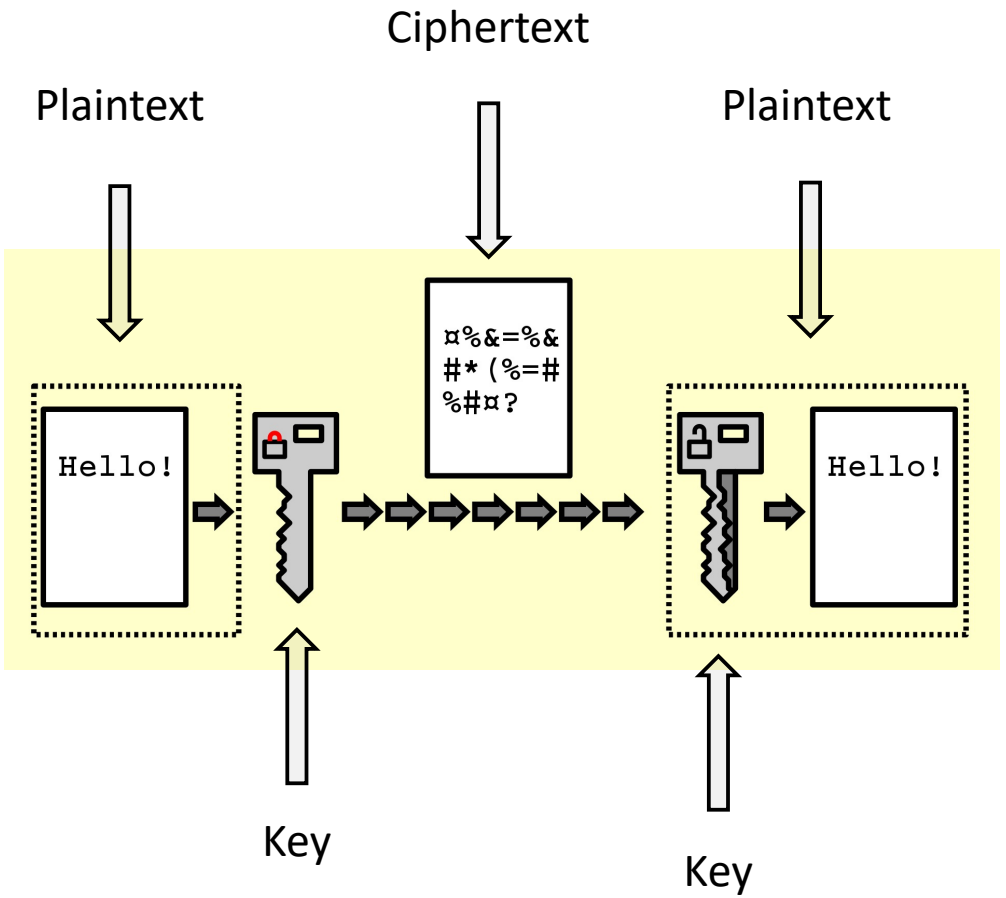
- Something that you *know*,
- Something that you *have*,
- Something that you *are*

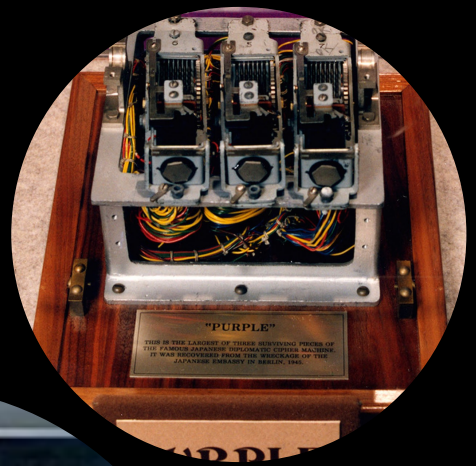




Cryptography

- *Goal*: Render a message incomprehensible to all except the intended recipient.
- *Requirement*: Use a well-known algorithm to encrypt.
  - But why?
  - Relying upon the secrecy of the algorithm is a very bad idea:
    - German *Enigma*
    - Japanese *Purple*
- Algorithm has two inputs: data & key
  - Key is known only to *authorized* users



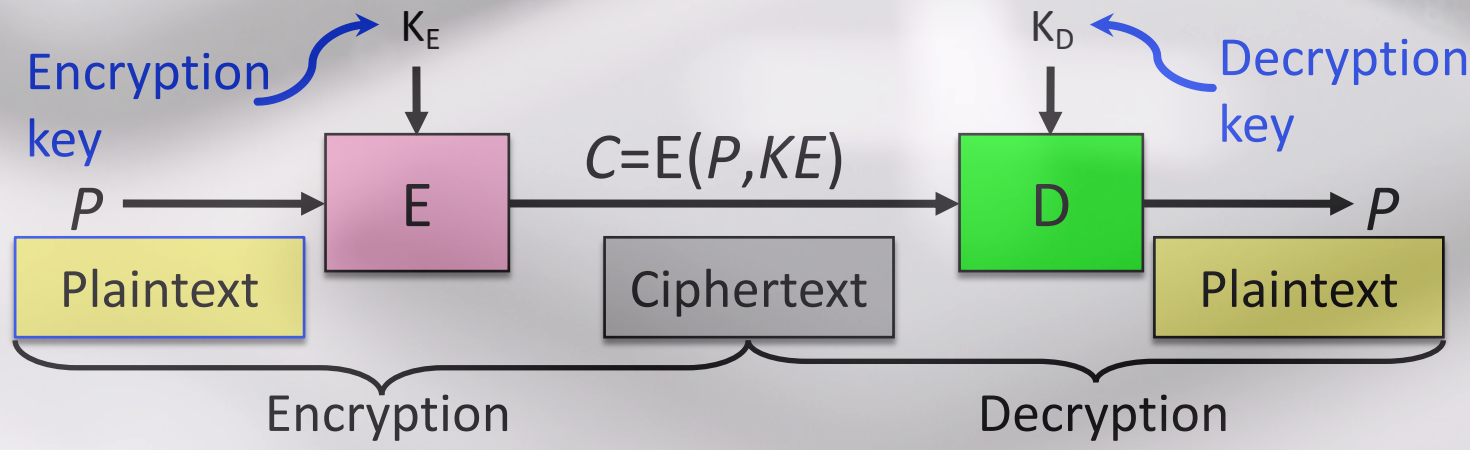


Kerckhoffs's desideratum

- The system must be practically, if not mathematically, indecipherable;
- It should not require secrecy, and it should not be a problem if it falls into enemy hands;
- It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will;
- It must be applicable to telegraph communications;
- It must be portable, and should not require several persons to handle or operate;
- Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

Cryptography Basics

- Algorithms (E, D) are *publicly known.*
- Keys (K_E, K_D) represent a *shared secret*.
- The *ciphertext* should be the only information that's available to the world.
- The *plaintext* is known only to the people with the keys.



Historical Cryptography

- Cryptography has existed since there has been writing:
  - Writing is a cipher to the illiterate and was once considered magical.
- Typically involved one or more of the following:
  - Substituing one symbol for another,
  - Rearranging the order of the symbols, or
  - Using numbers to reference words in a known text.
- An early popular example is *The Gold Bug* by Edgar Allan Poe.

Secret Key Encryption

- Also called symmetric-key encryption
- Monoalphabetic substitution:
  - Each letter replaced by different letter.
- Vignère cipher (polyalphabetic):

  THEMESSAGE

  ↓

  ELMELMELME

  ↓

  XSQQPEWLSI
- These are easy to break.
  - Given the encryption key, easy to generate the decryption key

Caesar Cipher

- Reputed to have been used during the Gallic Wars.
- Replaces each letter with the letter three positions down, thus
  - It uses modular arithmetic.
- Why did they use three?
  - Perhaps ancient Romans were not as good at arithmetic as the Greeks.

```
char *shift(char *s, int k) {
    char *t = strdup(s);
    for (int i = 0; i < strlen(s); i += 1) {
        if (isalpha(s[i])) {
            if (islower(s[i])) {
                t[i] = ((s[i] - 'a' + k) % 26) + 'A'; // ([0, 25] + k) % 26 + 'A'
            } else {
                t[i] = ((s[i] - 'A' + k) % 26) + 'A'; // ([0, 25] + k) % 26 + 'A'
            }
        } else {
            t[i] = s[i];
        }
    }
    return t;
}
```

Caesar Cipher

- Traditionally, the Caesar cipher replaces each letter with the letter three letters down.
- This is a monoalphabetic substitution cipher.

Vigenère Cipher

- In this cipher, the key is a *word* and each letter is used to shift a letter of the clear text by its ordinal number positions.
- To encrypt, a table of alphabets can be used, termed a tabula recta, Vigenère square or Vigenère table.
- Algebraically,
  - $C_i = E_K(M_i) = (M_i + K + i) \bmod 26$
  - $M_i = D_K(C_i) = (C_i - K_i + 26) \bmod 26$

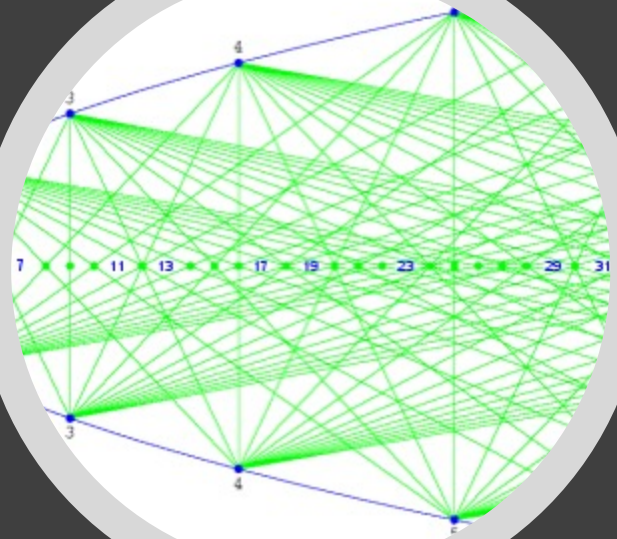
```
#define ASCII 255

// Name: encode
//
// Function: encode a string using the Vigenère cipher algorithm. Shift a letter
// (byte) forward by the ordinal number of corresponding letter in the key.
//
// Inputs: encryption key (key) and cleartext (cleartext)
//
// Outputs: enciphered text (ciphertext)

void encode(char *key, char *cleartext, char *ciphertext) {
    int i, lk = strlen(key), lc = strlen(cleartext);
    for (i = 0; i < lc; i += 1) {
        ciphertext[i] = (unsigned char) (cleartext[i] + key[i % lk]) % ASCII;
    }
    ciphertext[lc] = '\0';
    return;
}
```

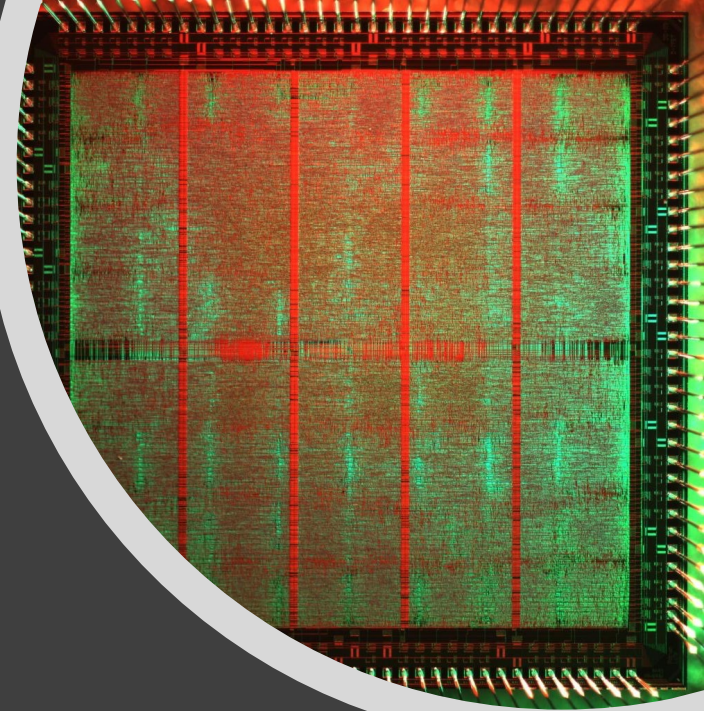
Vigenère Encode
(adapted to ASCII)

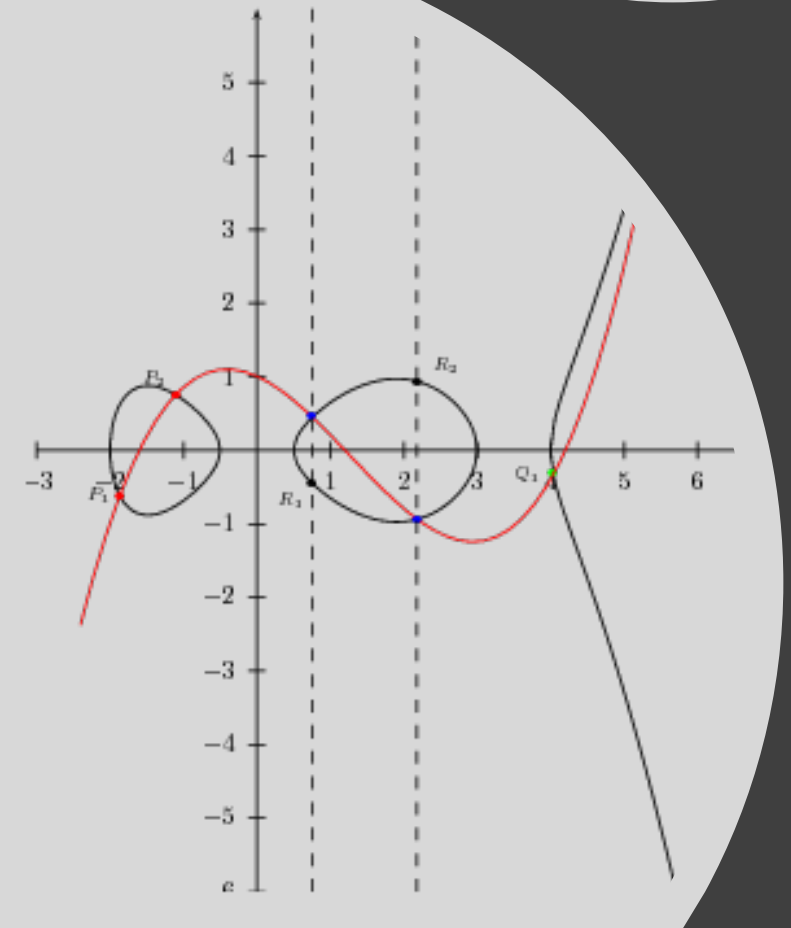
```
// Name: decode
//
// Function: decode a string using the Vigenère cipher algorithm. Shift a lette
// (byte) backward by the ordinal number of corresponding letter in the key.
//
// Inputs: decryption key (key) and ciphertext (ciphertext)
//
// Outputs: deciphered text (cleartext)

void decode(char *key, char *ciphertext, char *cleartext) {
    int i, lk = strlen(key), lc = strlen(ciphertext);
    for (i = 0; i < lc; i += 1) {
        cleartext[i] = (unsigned char) (ciphertext[i] - key[i % lk] + ASCII) %
    }
    cleartext[lc] = '\0';
    return;
}
```

Vigenère Decode
(Adapted to ASCII)

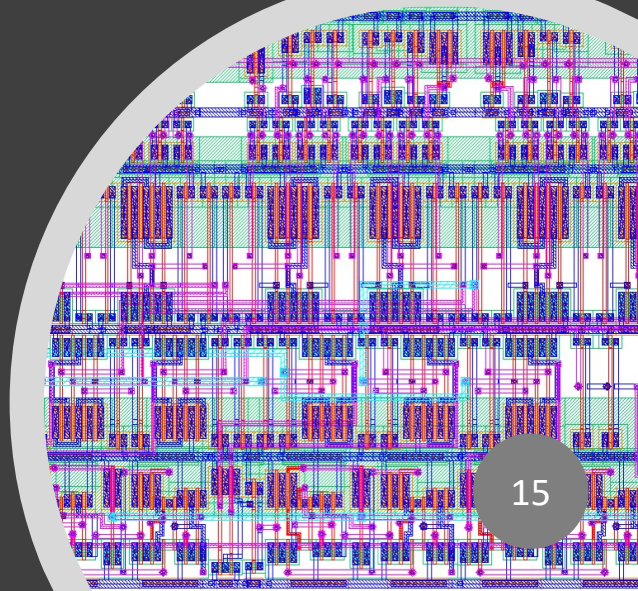
```
_buf2[_i++] = in[_idx - 1];
if( _i ==4 )
{
  for (_i = 0; _i < 4; _i++)
    _buf2[_i] = strchr(_oaes_base64_table, _buf2[_i]) - _oaes_base64_table;

  _buf1[0] = (_buf2[0] << 2) + ((_buf2[1] & 0x30) >> 4);
  _buf1[1] = ((_buf2[1] & 0xf) << 4) + ((_buf2[2] & 0x3c) >> 2);
  _buf1[2] = ((_buf2[2] & 0x3) << 6) + _buf2[3];

  for( _i = 0; (_i < 3); _i++ )
  {
    *(out++) = _buf1[_i];
    (*out_len)++;
  }
  _i = 0;


          _j <4; _j++ )
```







Modern Cryptography



Modern Encryption Algorithms

- Data Encryption Standard (DES) (1977)
  - Uses 56-bit keys
  - Same key is used to encrypt & decrypt
  - Needed to try 2^{55} different keys, on average
    - But… Modern computers can try millions of keys per second with special hardware
- Current algorithms (AES, Blowfish) use at least 128 bit keys
  - Adding one bit to the key makes it twice as hard to guess
  - Must try 2^{127} keys, on average, to find the right one
  - At 10^{15} keys per second, this would require over 10^{21} seconds, or 1000 billion years!
  - Modern encryption isn't usually broken by brute force…

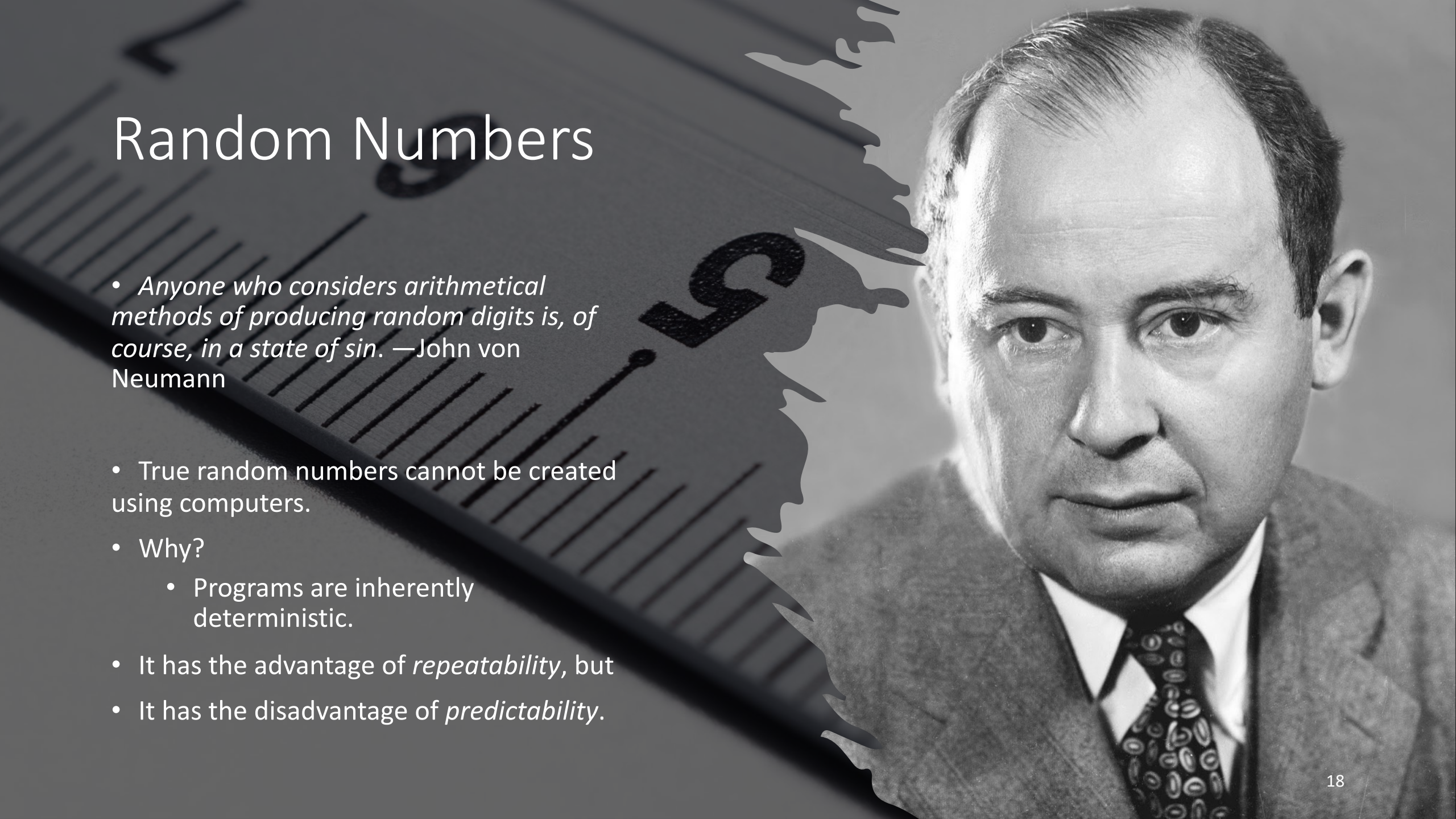
Unbreakable Codes

- There is such a thing as an unbreakable code: It's called the *one-time pad.*
  - Use a *truly random key as long as the message* to be encoded.
    - XOR the message with the key a bit at a time.
- Code is unbreakable because
  - Key could be any string of bits.
  - The message could be *any message* given the appropriate key.
- *Difficulty*: distributing the key is as hard as distributing message.
  - May be easier because of timing
- *Difficulty*: generating truly random bits.
  - What does it mean to be random?

Random Numbers

- *Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin*. —John von Neumann

- True random numbers cannot be created using computers.
- Why?
  - Programs are inherently deterministic.
- It has the advantage of *repeatability*, but
- It has the disadvantage of *predictability*.



One-way Functions

- Function such that
  - Given formula for f(x), easy to evaluate y = h(x)
  - Difficult to find “collisions”: two values with the same hash function
    - Weak collision resistance: given y, hard to find x such that f(x) = y
    - Strong collision resistance: hard to find x and $x' \neq x$ such that f(x) = f(x')
- Often, operate similar to encryption algorithms
  - Produce fixed-length output rather than variable length output
  - Similar to XOR-ing blocks of ciphertext together
- Common algorithms include
  - MD5: 128-bit result
  - SHA-1: 160-bit result
  - SHA-256: 256-bit result

Attacking AES

- Suppose that we have an infinity fast computer,
  - But such a computer must still consume energy.
- There is a fundamental limit on the energy required to flip a bit.
- The Landauer principle says that the energy to flip a bit:
  - $E > k\,T\,\ln n.$
  - $k = 1.38 \times 10^{-23}$
  - $T \sim 293$
- 128-bit AES would require all the electric power of the USA for more than a year
- 192-bit AES would require 1000 times annual solar flux
- 256-bit AES would require more energy than a *billion* supernovae

Public-key Cryptography

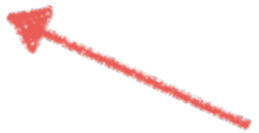
- Instead of using a single shared secret, keys come in pairs:
  - The *public key* K_U.
  - The *private key* K_R.
  - $K_U \neq K_R$.
  - These keys are typically—but not always—inverses of one another.
  - Encryption & decryption are the *same algorithm*, so, typically:
    - $E\big(K_U, E(K_R, M)\big) = K\big(K_R, E(K_U, M)\big) = M$
- Public key cryptography is usually *slow*.
- Typically used for:
  - Encrypting small amounts of data, or
  - Establishing a shared key for symmetric encryption algorithms.
- Currently, the most popular method, RSA, involves *prime rings* and *exponentiation*.
  - Security relies on the *difficulty of factoring large composites*.
- Other methods involve *discrete logarithms* and *elliptic curves*.

RSA algorithm for public key encryption

- Private, public key pair consists of $KR = (d,n)$, $KU = (e,n)$
  - $n = p \times q$ (p and q are large prime numbers)
  - e is a randomly chosen integer with GCD $(e, (p-1)\times(q-1)) = 1$
  - d is an integer such that $(e \times d) \equiv 1 \pmod{(p-1)\times(q-1)}$
- p & q aren't published, and it's hard to compute them: factoring large numbers is "difficult"
- Public key is published, and can be used by anyone to send a message to the private key's owner
- Encryption & decryption are the same algorithm:
  $E(KU,M) = M^e \text{ MOD } n$ (similar for KR)
  - Methods exist for doing the above calculation quickly, but...
  - Exponentiation is still very slow
  - Public key encryption not usually done with large messages

Attacking RSA

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA2DjbcSuCNn3Cluzg0D9eooJCFWJFo0G+OWl2
PCXXt6Vlw1NW8HP/sbdp70+Q00ws5HnTPfKibMljv8MrU+Ht94iVrOkJpdKOM+jVgz6Gx7li
9LnbtqhZtNWdXnmEeDXWU6/3cBCMow1oH4vPcx8YXRgL4a32+8l5FhBNkY52ephrSiNYL2In
U7uPzhbQLXJ0rXaNO5TYaDUbydwh5/aRCYP++qOXLY8gd31MGM2YY7dXJ0ZXqeN2OMecRshz
8G+z9P+WybfmFqbGmugxh6O7y+AK5D2JLNhzuwgdVOL6OwLy0awuUJSNFMp6pExB4ff7Pk+g
aTwI3OzMUOeL2nE1vw== darrell@Matamoros.local
```



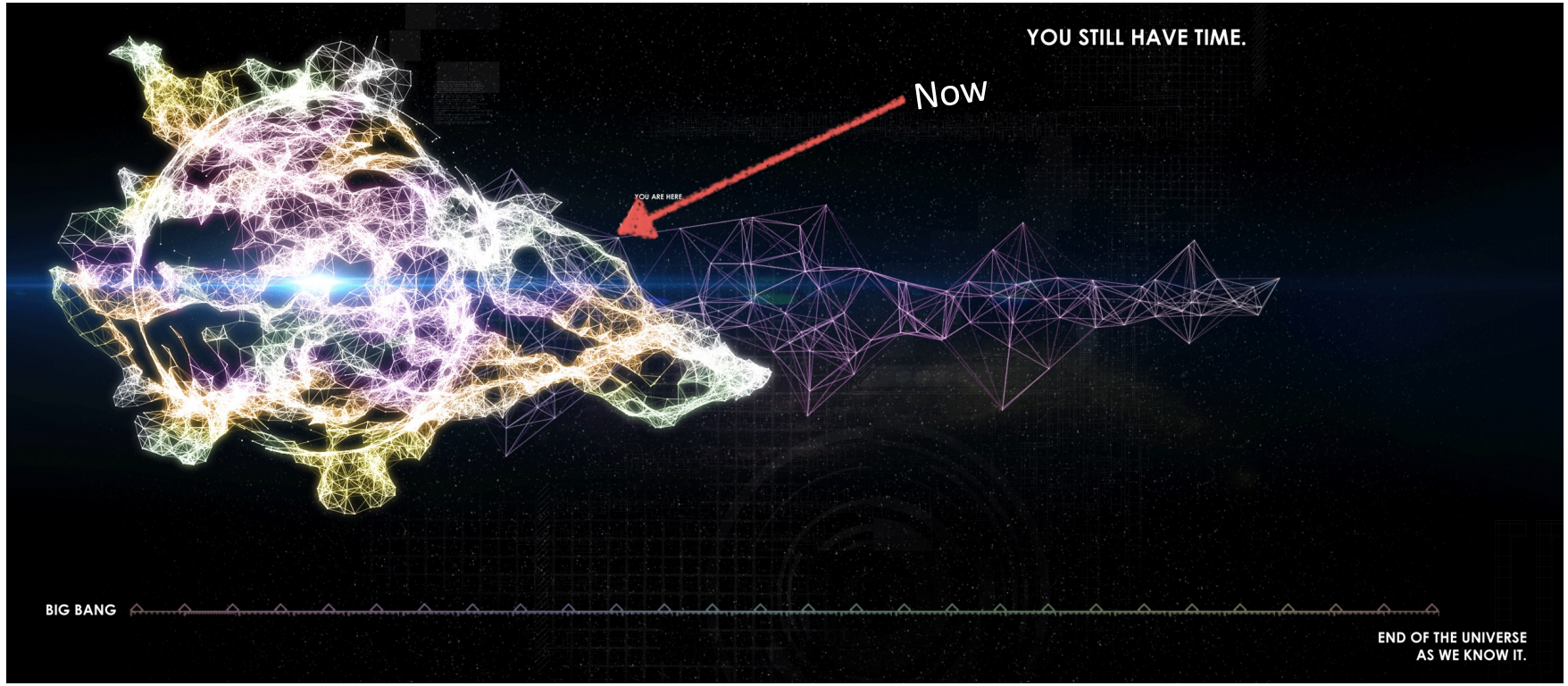
My public key

$n = p \times q$
$p, q > 2^{1024}$ and *prime*
$\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$
$1 < e < \varphi(n)$
$\gcd(e, \varphi(n)) = 1$
$d \times e \equiv 1 \pmod{\varphi(n)}$
$c = m^e \pmod{n}$
$m = c^d \pmod{n}$

You can safely ignore those equations, except noting how simple they are.

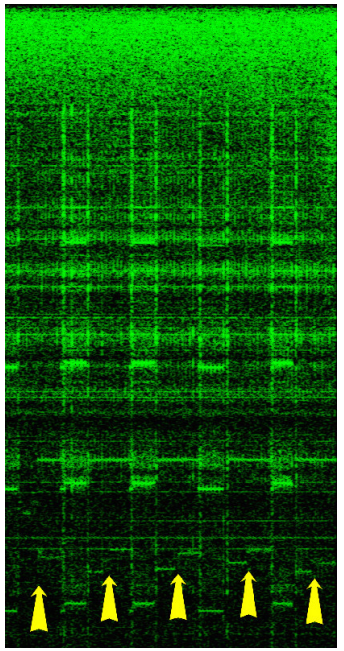
Attacking RSA

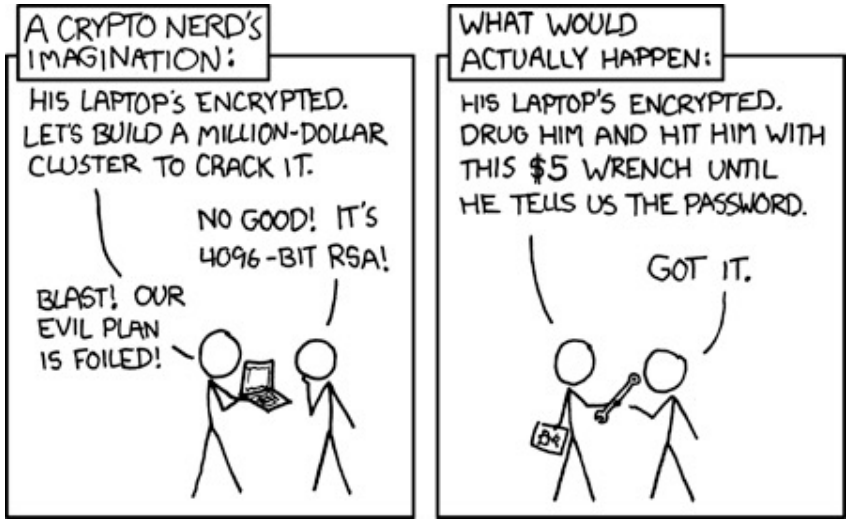
If you began factoring at the Big Bang using the best algorithm and using all of the computers in the world…



How do you attack modern cryptography?

You Cheat!





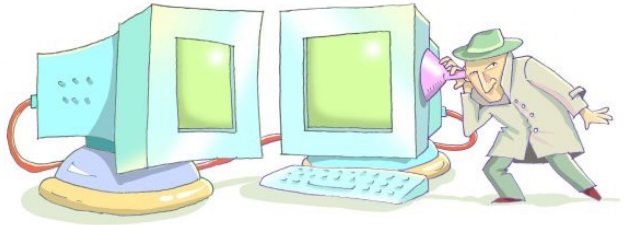
http://xkcd.com/538/

RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis

Daniel Genkin — Technion and Tel Aviv University — danielg3@cs.technion.ac.il

Adi Shamir — Weizmann Institute of Science — adi.shamir@weizmann.ac.il

Eran Tromer — Tel Aviv University — tromer@cs.tau.ac.il



When you can't win playing by the rules, you *change the rules!*

- You lie, you cheat
- You look for mistakes
- You look for bad assumptions
- You look for the weak spots
  - The weakest spot is often the human …