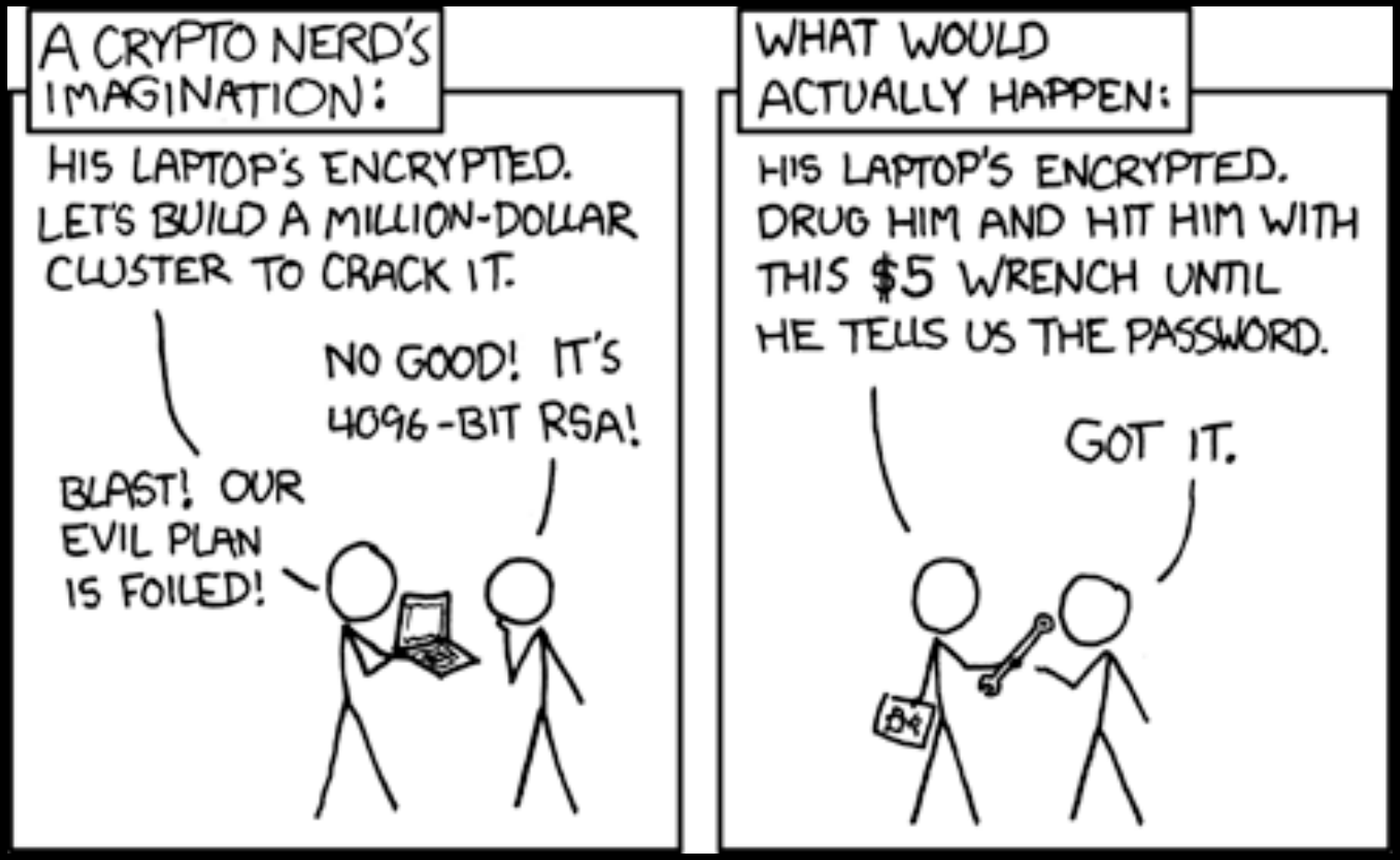


https://xkcd.com/538/

CS 168: Blockchain and Cryptocurrencies

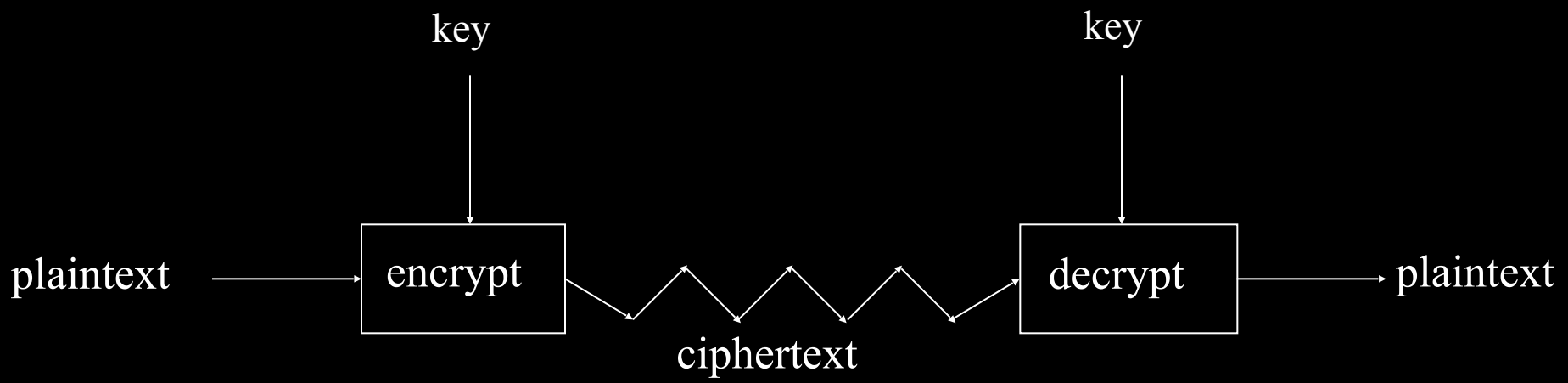
Cryptography Review

Prof. Tom Austin
San José State University

Goals of Cryptography

- Confidentiality
  - Protecting secrets
- Integrity
  - Notice when something has been corrupted

Crypto as Black Box



A generic view of symmetric key crypto

Cryptography Taxonomy

- Symmetric key
  - Stream cipher
  - Block cipher
- Public key
- Cryptographic hashes

Symmetric Key Notation

Encrypt the *plaintext* `P` with the *key* `K` to produce the *ciphertext* `C`.

`E(P,K) = C`

Decrypt the ciphertext `C` with the key `K` to produce the plaintext `P`.

`D(C,K) = P`

Stream Ciphers

- Based on one time pad (OTP)
- Sacrifices provable security for usability
- Often implemented in hardware

One-Time Pad Review

Provably secure!

Plaintext: `0101 1010 0101 1011 0101`

⊕ **Key:** `1011 0010 1101 1001 0001`

Ciphertext: `1110 1000 1000 0010 0100`

One-Time Pad Review

Key is as long as the original message

Plaintext: 0101 1010 0101 1011 0101

⊕ **Key:** 1011 0010 1101 1001 0001

Ciphertext: 1110 1000 1000 0010 0100

Replacing the key with a keystream

Key: 1001 1110

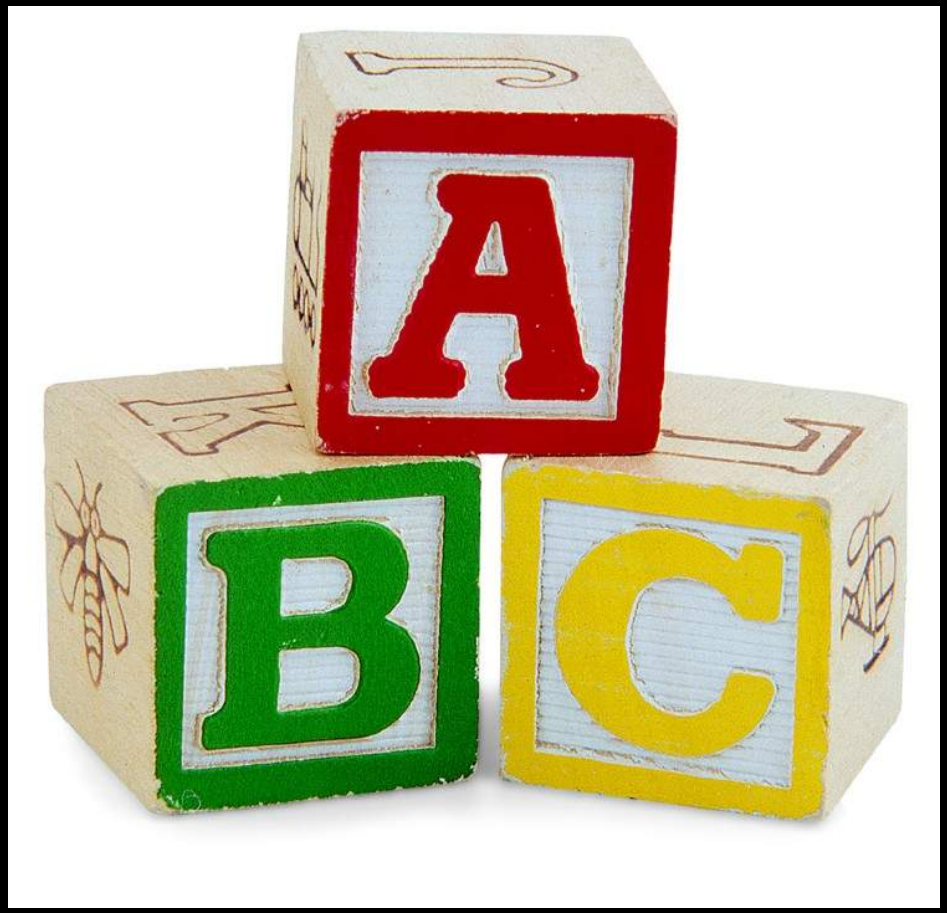
Keystream Generator

Keystream: 1001 0011 1101 1000 …

P: ⊕ 0101 1010 0101 1011

C: 1100 0001 1000 0011

Block Ciphers



Review of codebook ciphers

Word	Codeword
Apple	00123
Banana	11439
Citrus	92340
Cranberry	87642
Durian	58629
Orange	66793
Strawberry	88432
Watermelon	90210

Plaintext:

Apple Durian Orange

Ciphertext:

00123 58629 66793

Block Ciphers: Codebooks of Bytes

Input	Output
…	…
9E	CB
9F	80
A0	4F
A1	ED
A2	62
A3	9A
…	…

OK, they are a bit more complicated than that…

(Iterated) Block Cipher

- Plaintext and ciphertext consist of fixed-sized blocks
- Ciphertext obtained from plaintext by iterating a ***round function***
- Input to round function consists of ***key*** and ***output*** of previous round
- Usually implemented in software
- Also useful for integrity checks (MACs)

Important Block Ciphers

- DES (Data Encryption Standard)
  - Back door included by the NSA!
    (Allegedly, but not likely).
  - Never broken, but key length is too small
- AES (Advanced Encryption Standard)
  - Replacement for DES
  - Public review process
- Others: IDEA, Blowfish, RC6, TEA

Block Cipher Modes

- Many modes: we discuss 3 most popular
- Electronic Codebook (**ECB**)
  - Encrypt each block independently
  - Most obvious, but has a serious weakness
- Cipher Block Chaining (**CBC**)
  - Chain the blocks together
  - More secure than ECB, virtually no extra work
- Counter Mode (**CTR**)
  - Block ciphers acts like a stream cipher
  - Popular for random access

ECB Weakness

- Suppose $P_i = P_j$
- Then $C_i = C_j$ and Trudy knows $P_i = P_j$
- This gives Trudy some information, even if she does not know P_i or P_j
- Trudy might know P_i
- Is this a serious issue?

Alice Hates ECB Mode

- Alice's uncompressed image, and ECB encrypted (TEA)

- Why does this happen?
- Same plaintext yields same ciphertext!

CBC Mode

- Blocks are “chained” together
- A random initialization vector, or IV, is required to initialize CBC mode
- IV is random, but not secret

Encryption

$C_0 = E(IV \oplus P_0, K)$,

$C_1 = E(C_0 \oplus P_1, K)$,

$C_2 = E(C_1 \oplus P_2, K)$,…

Decryption

$P_0 = IV \oplus D(C_0, K)$,

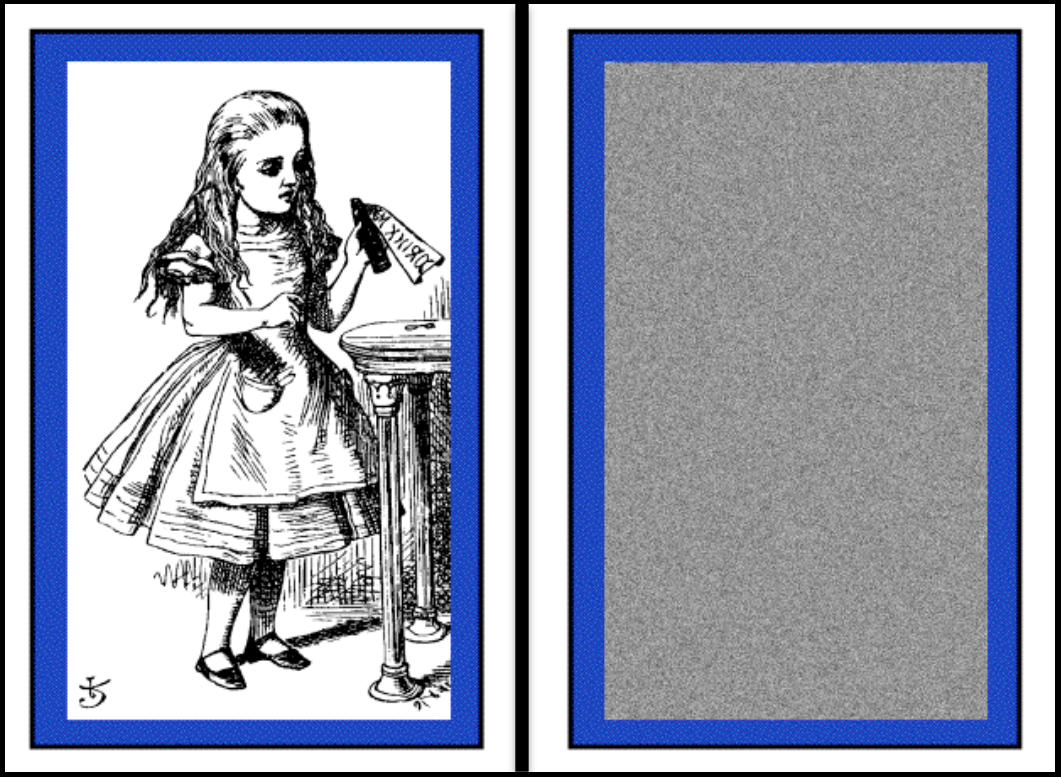
$P_1 = C_0 \oplus D(C_1, K)$,

$P_2 = C_1 \oplus D(C_2, K)$,…

- Analogous to classic codebook *with additive*

Alice Likes CBC Mode

- Alice's uncompressed image, Alice CBC encrypted (TEA)



- Why does this happen?
- Same plaintext yields different ciphertext!

Message Authentication Code (MAC)

- Provide data ***integrity***
  - **Has the data been corrupted?**
  - Unrelated to confidentiality
- Computed as **CBC residue**
  - Compute CBC encryption
  - Save final ciphertext block, the MAC
  - Discard all other ciphertext blocks

Counter Mode (CTR)

- CTR is popular for random access
- Use block cipher like a stream cipher

Encryption

$C_0 = P_0 \oplus E(IV, K)$,

$C_1 = P_1 \oplus E(IV+1, K)$,

$C_2 = P_2 \oplus E(IV+2, K)$,…

Decryption

$P_0 = C_0 \oplus E(IV, K)$,

$P_1 = C_1 \oplus E(IV+1, K)$,

$P_2 = C_2 \oplus E(IV+2, K)$,…

Stream Cipher or Block Cipher

- Stream ciphers
  - Better in hardware
  - Better on noisy channels
  - Confidentiality only
- Block ciphers
  - Better in software
  - Confidentiality or integrity

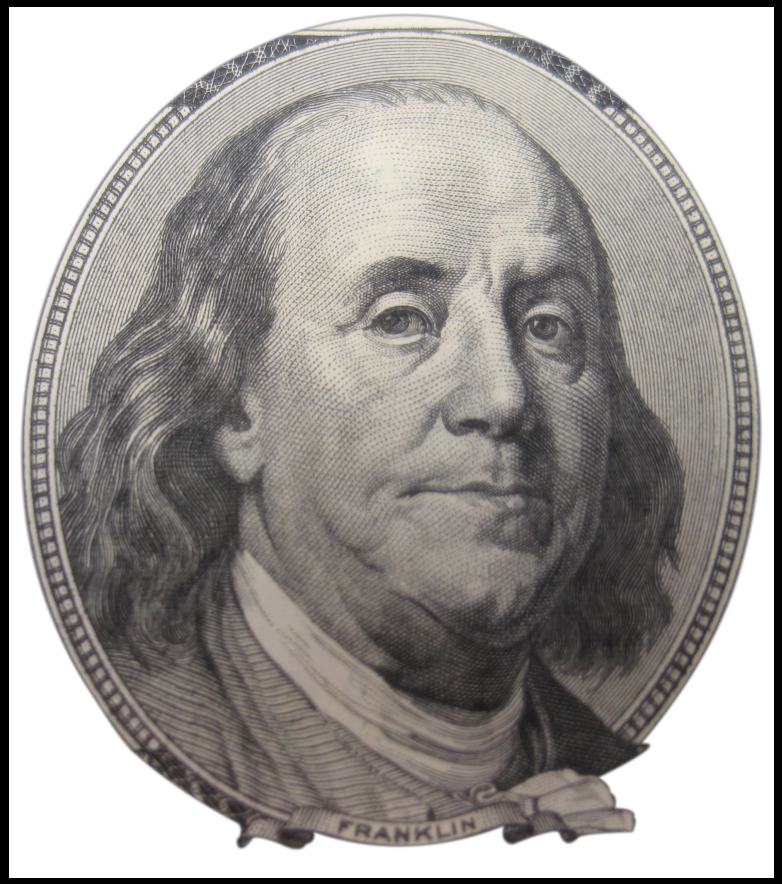
Encrypting a String in Node.js

```
let crypto = require('crypto');

function encryptString(s, key, iv) {
  let c = crypto.createCipheriv('aes-256-cbc',
                                key, iv);
  let ctext = c.update(s, 'utf8','hex');
  ctext += c.final('hex');
  return ctext;
}

let ptext = 'hello world';
let key = crypto.generateKeySync('aes',
                          { length: 256});
let iv = crypto.randomBytes(16);
let ctext = encryptString(ptext, key, iv);
```

Public Key Cryptography



Three may keep a secret, if two of them are dead.
–Ben Franklin

Public Key Encryption

- Relies on 'trap-door' functions
- Uses two separate keys
  - Public key is known by everyone
  - Private key known only to the owner
- Analogy: locked mailbox
  - Anyone can put a letter in the mailbox
  - Only the mail carrier can get them

Digital Signatures

- Reverse process is used for digital signatures:
  - Private key can encrypt a message
  - Public key can decrypt the message
- Analogy: Enclosed bulletin board
  - Anyone can read the messages
  - Only the owner could have put the messages there

Note on Digital Signatures vs. MACs

- Both tools provide integrity
- Only digital signatures offer *non-repudiation*
  - Only Alice can use her private key
  - Alice cannot deny her signature later

RSA

- Discovered by Clifford Cocks (GCHQ)
- Rediscovered by **R**ivest, **S**hamir, and **A**dleman (MIT)
  - RSA is the ***gold standard*** in public key crypto
- Let p and q be two large prime numbers
- Let N = pq be the **modulus**
- Choose e relatively prime to (p–1)(q–1)
- Find d such that ed = 1 mod (p–1)(q–1)
- **Public key** is (N,e)
- **Private key** is d

RSA

- Message M is treated as a number
- To encrypt M we compute

  $C = M^e \bmod N$
- To decrypt ciphertext C compute

  $M = C^d \bmod N$
- Recall that e and N are public
- If Trudy can factor N=pq, she can use e to easily find d since $ed = 1 \bmod (p-1)(q-1)$
- **Factoring the modulus breaks RSA**

Simple RSA Example

- Example of RSA
  - Select "large" primes p = 11, q = 3
  - Then N = pq = 33 and (p – 1)(q – 1) = 20
  - Choose e = 3 (relatively prime to 20)
  - Find d such that ed = 1 mod 20
    - We find that d = 7 works
- **Public key:** (N, e) = (33, 3)
- **Private key:** d = 7

Simple RSA Example

- **Public key:** (N, e) = (33, 3)
- **Private key:** d = 7
- Suppose message M = 8
- Ciphertext C is computed as

  $C = M^e \bmod N = 8^3 = 512 = 17 \bmod 33$
- Decrypt C to recover the message M by

  $M = C^d \bmod N = 17^7 = 410{,}338{,}673$
  $= 12{,}434{,}505 * 33 + 8 = 8 \bmod 33$

Who manages the keys?

Public-key cryptography requires *public key infrastructure* (**PKI**).

- Issues new *certificates*
  - Identity
  - Public key
  - Possible more
- Issues *certificate revocation lists* (**CRLs**).
- Serves as *trusted third party* (**TTP**).

What type of cryptography is better?

Symmetric key crypto

- Useful for
  - Confidentiality
  - Integrity (MACs)
  - …
- Requires shared secret
- No PKI required
- **Faster**
  - By orders of magnitude

Public-key crypto

- Useful for
  - Confidentiality
  - Integrity (digital signatures)
  - *Non-repudiation*
- No shared secret required
- Trusted PKI needed
- Slow

Lab, part 1: Sign an object

Starter code is on course website.

The 'sign' function takes in an object and a private key. Sign the "message" field of the object and store the signature in a "sig" field on that object.

The 'verify' method takes in an object signed with your 'sign' function. The signer's ID is stored in the 'id' field of the object. Look up the public key from the certificate authority and return true if the signature is valid.

Cryptographic Hash Functions

or, *Why can't they tell me my password?*

Cryptographic hash functions

Encrypt data so that it can never be decrypted.
Why is this useful?

- Efficient signatures
- Safely storing passwords
- "Proof of work" protocols

WARNING! Not all hash functions are *cryptographic* hash functions.

Hash functions in action

h("secret") = 5ebe2294ecd0e0f08eab7690d2a6ee69

Username	PasswordHash
Alice	5ebe2294ecd0e0f08eab7690d2a6ee69
Bob	4bbfbb9beab959cc431ec4eed504cde5
Charlie	5f202e7ab75f00af194c61cc07ae6b0c
David	3feb2d8fe13b4e9c3c81de0734257103

Crypto Hash Function Properties

- Crypto hash function h(x) must provide
  - **Compression** – output length is small
  - **Efficiency** – h(x) easy to compute for any x
    - but not *too* efficient
  - **One-way** – given a value y it is infeasible to find an x such that h(x) = y
  - **Weak collision resistance** – given x and h(x), infeasible to find y ≠ x such that h(y) = h(x)
  - **Strong collision resistance** – infeasible to find ***any*** x and y, with x ≠ y such that h(x) = h(y)
- Lots of collisions exist, but hard to find ***any***

Avalanche Effect

- Desired property: **avalanche effect**
  - Change to 1 bit of input should affect about half of output bits
- Crypto hash functions consist of some number of rounds
- Want security and speed
  - Avalanche effect after few rounds
  - But simple rounds
- Analogous to design of block ciphers

Avalanche Effect

```
Tiger("better call saul") =
    0201b60356a7eca259ff4d71
    ea910b83a316ceaed29f9d0a

Tiger("better call paul") =
    a9c6722a7a338cb292787d74
    2474839dd9338a116fafd17c
```

Lab, part 2

Starter codes is available on the course website.

PasswordManager stores

- map of username->hashes
- map of username->salts

The storePassword method takes in a username and a password. Store the password by hashing the username with a unique salt value. You can choose whatever salt value you like, though it should be unique for every user.

Next, update the verifyPassword. Given a username, test whether the specified password is correct.

Additional Reading/Videos

- "Bitcoin and Cryptocurrency Technologies" (prepub version),
  - Sections 1.1, 1.3, and 1.4
  - lecture 1: https://youtu.be/fOMVZXLjKYo
- "Mastering Bitcoin", 2nd ed.
  - P. 55-64
- Mark Stamp's textbook
  - Chapters 2-5