Security Introduction

– What you ~~always~~ (perhaps?) wanted to know –

Peter Thomassen <peter@desec.io>

ICANN 85 – GNSO: CPH TechOps – March 8, 2026

What's a "hash"?

Check Digits

ISBN-13 Check Digit

ISBN Number	9	4	5	0	5	0	1	4	7	6	2	1	?
Weight	1	3	1	3	1	3	1	3	1	3	1	3	1
Product	9	12	5	0	5	0	1	12	7	18	2	3	

Check Digit

Current Sum = 74

Sum must be divisible by 10

Hashes

- Like check digit but more generic
  - and much longer
  - SHA-256 hash method: 77 digits
- Any input change causes seemingly random changes in hash output
- Given a hash output, cannot reconstruct input
- Given a hash input, cannot construct another input with same output

→ **Pretty decent “fingerprint” of the input data**

Some Hash Applications

Password storage

- Store hash instead of password. During login, calculate hash + compare stored.
- If stolen, thief cannot reconstruct passwords from stored hashes.
- Problem: Identical passwords have same hash.
  Solution: Internally, add a few letters before hashing (“salt”, different per user).

Indexing: How does a search engine find pages with search term quickly?

- Hash each word, and make a table mapping each hash to pages with the word.
- Table may not fit one machine. Use first hash digit(s) to identify the right one.
  → Looking up from a “hash table” prevents the need for search!

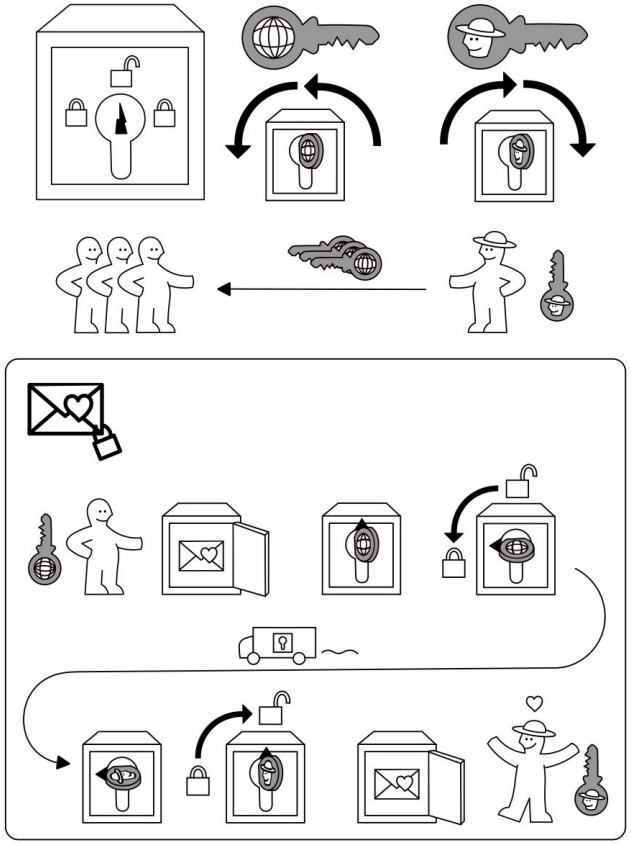
Authenticator Apps (2FA)

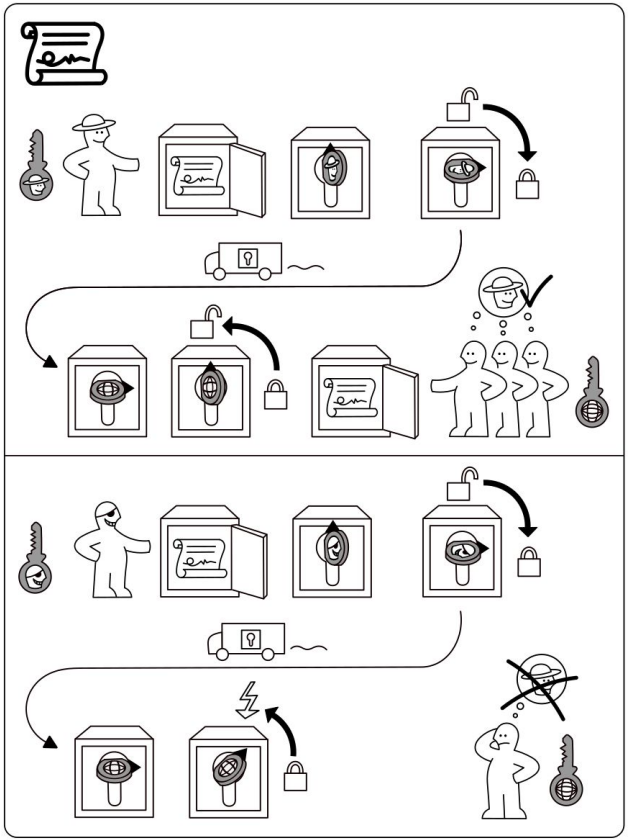
- Secret number stored on server and smartphone, hash along with current time.

Encryption

PUBLIK KEY KRYPTO

idea-instructions.com/public-key/
v1.2, CC by-nc-sa 4.0 IDEA





- Public and private keys are mathematically related
- Constructing a key pair involves randomness
- Computationally infeasible to construct private from public key

Signatures are Very Important

- Even for Encryption
- **How does your web browser know your bank's public encryption key?**

1. Your bank goes to a notary, saying: *"We are example.bank and here's our key"*
2. Notary says: *"Prove it! Put this long random number up on http://example.bank/…"*
3. If confirmed, **notary uses its private key to sign bank's public key**
4. Bank publishes signature received from notary
5. Browsers who have the notary's public key can now trust the bank's public key

This is called the *Web PKI*. The notary is a **Certificate Authority (CA)**. The signature is a **Certificate**. There are many notaries' public keys in the browser's **Trust Store**.

Other Use Cases

- **ToR network ("dark web"):** public key embedded in domain name, like https://www.facebookwkhpilnemxj7asaniu7vnjjbiltxjqhye3mhbshg7kx5tfyd.onion/
- **Crypto currencies:** wallet identifiers are their owner's public key (anonymous!)
  - Send money: **publish a signed message** containing **(1) amount, (2) recipient's public key** (wallet ID), and **(3) hash of the last record** on the books
  - **This *append-only storage* is called the "Blockchain".** Anyone can have a copy and follow along.
  - Anyone can verify each wallet balance from the first transaction. (Expensive!)
  - How does the first coin come about?
- **Proof of work:** find a hash input whose output starts with 1, 2, 3, … zeroes
  - Has 50% chance, 25% chance, 12.5% chance, …
  - **For many zeroes, a lot of work is needed.** Expensive!
  - If you found one, publish a signed message containing that hash input and your wallet ID!
    → "Mining"

Some more Thoughts

- **Hashes are very (!) fast**
  - 100 million/second on a modern computer, reaching billions with GPU acceleration (miners!)
  - Password storage therefore uses *hash iterations*
  - Should increase over time (today: ~1,200,000)
- Operations with **public/private keys are slow**
  - ~50k/second (for short messages)
- **For signing a (long) message**, there's a neat solution: **sign its hash instead**
  - Much shorter → less signature storage, less computation time
  - Recipient has both message and signature → easy to compute message hash during validation
- **How to speed up encryption?**
  - Can't encrypt the hash instead, because recipient can't reconstruct the original message!

~~Asymmetric Encryption~~

Symmetric Encryption

Only One Key (shared between two parties!)

- Move all letters by some number of steps (“Caesar cipher”)
  - Vulnerable to statistical attack (letter frequencies)
- One-time pad (OTP). Example: **CPH** in ASCII representation

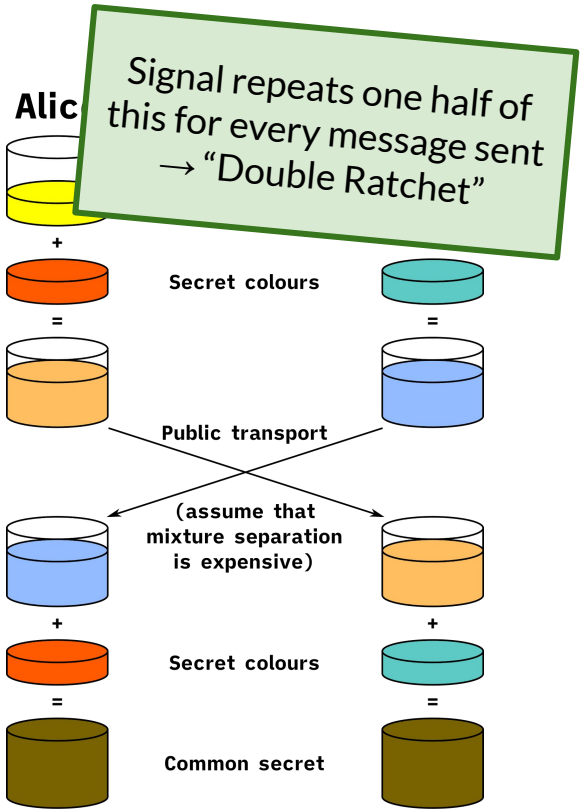
	C	P	H	
Encrypt:	01000011	01010000	01001000	plaintext
⊕	10101100	01011011	11100010	secret key
	11101111	00001011	10101010	ciphertext – random!
Decrypt:	11101111	00001011	10101010	ciphertext
⊕	10101100	01011011	11100010	secret key
	01000011	01010000	01001000	plaintext

OTP reuse is fatal. How to agree on a secret long enough to encrypt streams?
  - Not necessary: Agree on 256 random bits and repeatedly apply hash for “key stretching”.

What Happens in an Encrypted Connection

1. Verify server key (certificates etc.)
2. Use asymmetric cryptography to exchange symmetric secret key
   - Can be done directly (one party chooses one-time pad, encrypts it with other party's public key, and sends it)
   - But: does not prevent decryption after the fact when private key is leaked
   - For "forward secrecy", use **Diffie-Hellman key exchange**
3. Perform actual encryption of connection content

➢ Still at risk from quantum computers



Quantum Computers

- Lets nature work out some calculations instead of doing the steps
- Example: how to find amplitude (maximum height) of a water wave?
  - Either, take a picture (slicing the wave into chunks), then go through them, measure heights
  - Or, let the wave run against a wall and check where it's wet! ("wait and measure")
- Has nothing to do with processor clock speed
- Most computational questions cannot be "physically posed to nature"
- But: it works for reconstructing a private key from a public key
  - Specifically, for the RSA and elliptic-curve algorithms

➢ New asymmetric crypto needed for *classical* computers! But quantum-resilient.

  - They call it "post-quantum cryptography" (PQC) 🤷

deSEC

Peter Thomassen
peter@desec.io