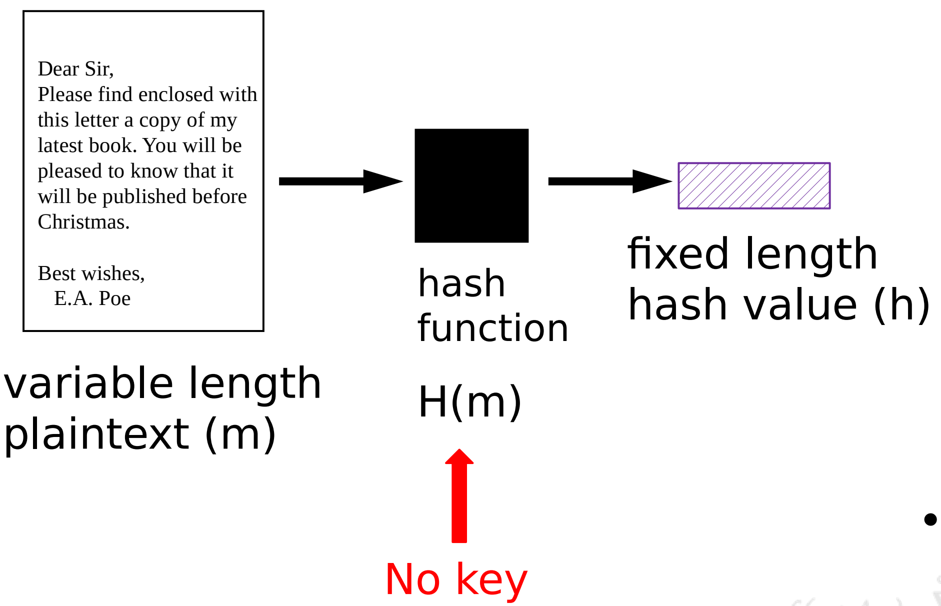
Cryptography II

DTU Compute
Department of Applied Mathematics and Computer Science

Cryptographic Hash Functions



- Cryptographic hash functions must also satisfy:
  1. Given M, computation of h is easy
  2. Given h, it is intractable to compute M such that $H(M) = h$
  3. It is intractable to find M and M' such that $H(M') = H(M)$
- Hash value h is often called a “fingerprint” or “digest” of M

Hash Functions

- The “work horses” of cryptography

- Main uses include:
  - Condensing long strings into short strings (collision resistant hash function)
  - Making an irreversible transformation without a key (one-way function)

- Prominent Examples:
  - MD4, MD5, SHA-0 (badly broken)
  - SHA-1 (broken, still found in standards, but must be retired before 2030)
  - SHA-2 (current standard)
  - SHA-3 (newest standard – since 2015)

Collision Resistance

- Intractable to find random *M* and *M'* such that

$$H(M) = H(M')$$

- Collisions invalidates the use of hash functions in digital signatures
  - Alice creates to contracts M and M', where M is fair, but M' is favourable to the Alice
  - Bob signs M: Sign(H(M), Bob_{Priv})
  - Since H(M) = H(M') Alice can present M' as if it was signed by Bob

Birthday Paradox and Birthday Attack

- Standard statistics problem

  How many people do you need in a room to have better than 50% chance of two persons with the same birthday?

  - someone with your birthday 253
  - any two with the same birthday 23
  - Any two born of the same day of the month 10

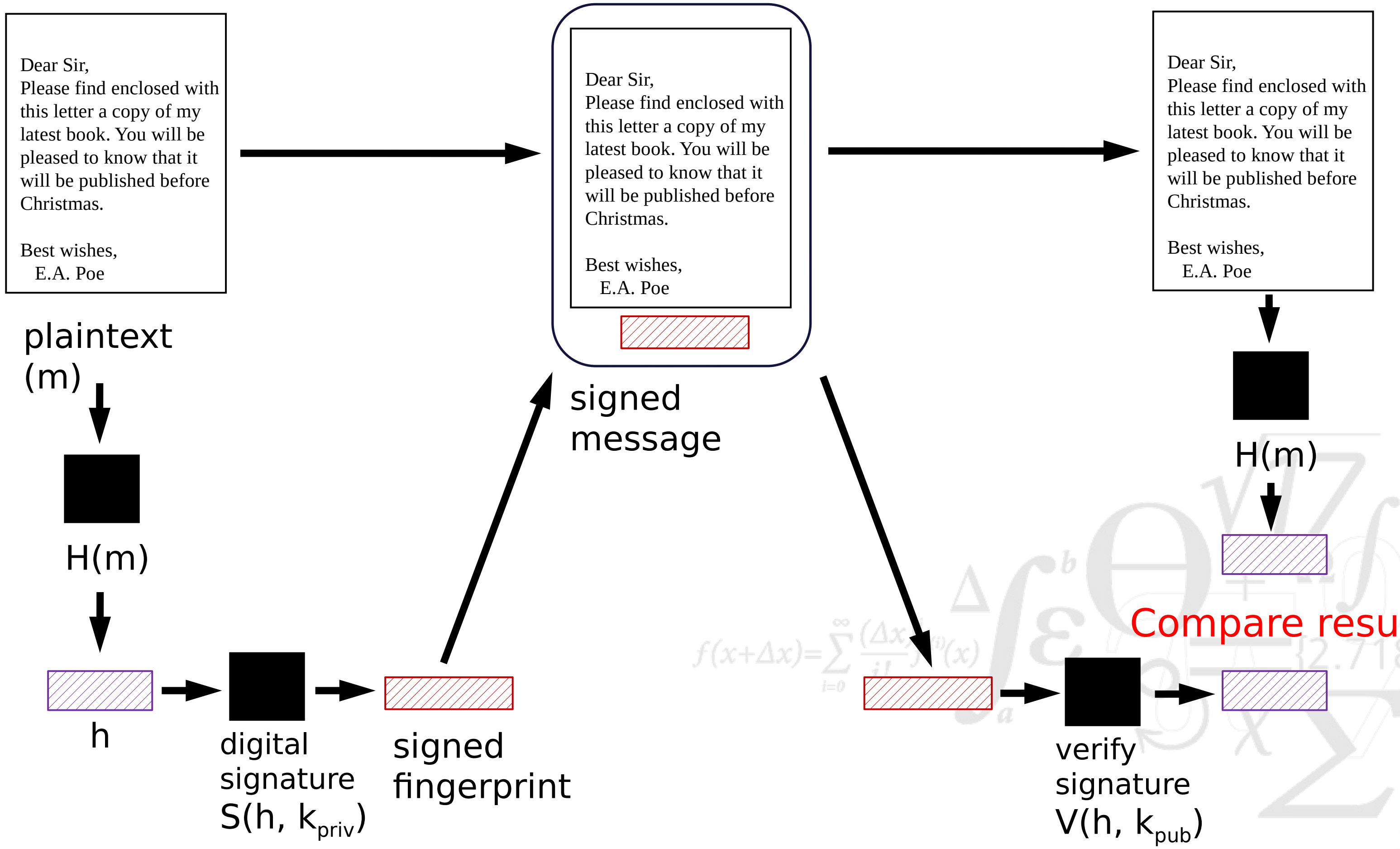
- *m* bit hash function
  - 2^m random hashes needed to find a particular *h*
  - $2^{m/2}$ random hashes needed to find two messages with the same hash value

Precomputed tables

- **Time-memory trade-off**: attackers can precompute and store hash values of plaintext
- The first algorithm better than brute force search was invented by Martin Hellman, known as the Hellman tables:
  - Exploits chaining
  - But limited by the risk of collision
- A more complex lookup table was proposed by Oechslin in 2003: rainbow tables
- A mitigation is to salt the passwords and use ”slow” password hashing algorithm such as Argon2:
  - For all new passwords, a unique and fresh salt is concatenated to the password before being hashed
  - Salts are stored together with hashes

Digital Signatures

- Operation is expensive, so we normally sign the fingerprint



Security Level

- If the best known attack is equivalent to running the cryptographic algorithm 2^n times, then we have a security level of n bit

- Typical 80 bit (too low), 128 bit (decent), 256 bit (high)

- Often, the security level is equal to the key length

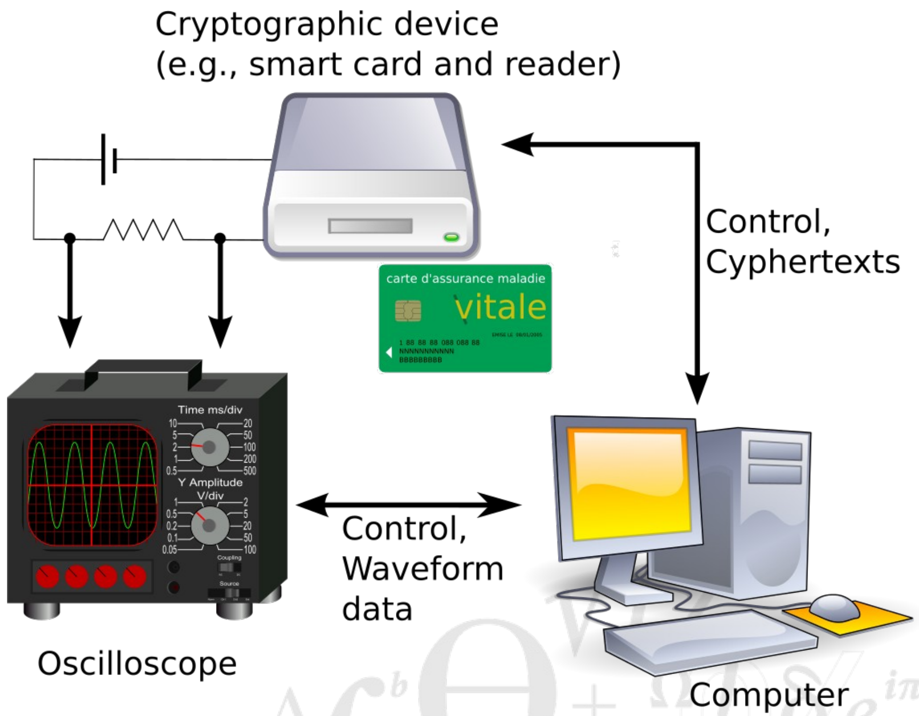
- Important exceptions:
  - Hash functions (hash size $\geq 2 * $ security level)
  - Asymmetric cryptography (e.g. RSA: 1024/2038/4096 bit)

Summary on Building Blocks

- Please remember the following
  - Clarify your security goals
  - Don’t design your own cryptographic primitives and protocols
    - *Always rely on well known (and analysed) standards*
  - Make sure that you understand the primitives and protocols you use:
    - *Security goals*
    - *Security level*
    - *How to apply them*
    - *Known problems and pitfalls*

Side-Channel attacks

- Attackers can exploit cryptosystems by observing their running time, power consumption, etc:
  - Timing attacks
  - Power attacks
  - Electromagnetic attacks
- Implementation needs to take these into account, e.g., by implementing constant-time algorithm



Encrypted Communication

- A protocol defines a sequence of steps involving several parties

- Characteristics of a protocol
  - Everyone knows the protocol in advance
  - Everyone agrees to follow the protocol
  - The protocol must be unambiguous
  - The protocol must be complete

- Typical “actors” in description of cryptographic protocols
  - Alice and Bob (parties who wish to communicate securely)
    - *Charlie and Dave are often added for multiparty protocols*
  - Eve (a malicious agent who wishes to eavesdrop on communication)
  - Mallory (a malicious attacker of any kind)
  - Trent (a trusted arbitrator – Trusted Third Party (TTP))

Cryptographic Protocols

- Cryptographic protocols
  - Describe how multiple parties employ cryptography together
    - *How each party uses the algorithms*
    - *How cryptographic keys are managed*
      - Key generation (not always covered in the protocol description)
      - Key distribution

Encrypted Communication in Practice

- Handshake (taking the initial steps)
  - Agree on cryptographic protocol, including:
    - *Algorithms and Modes*
    - *Cryptographic session keys*
  - Authentication of Communicating Parties
    - *Single-side authentication (typically server)*
    - *Mutual authentication (both parties are authenticated)*
- Communication (steady state operation)
  - Send and receive messages securely
    - *Confidentiality protected through encryption*
    - *Integrity protected through MAC or digital signatures*
- Connection termination
  - Delete all session specific state (keys, cookies, ...)

Usage of Cryptographic Keys

- The following is a categorisation of cryptographic keys according to what they are used for:
  - **Data key**: Directly used for cryptographic purposes, e.g. encryption or authentication
  - **Key-encryption key**: Used to encrypt other keys, e.g. in key exchange or key storage
  - **Master key**: Used to generate other keys, e.g. in Key Derivation Function (KDF).
    Example: *Session_Key := KDF(Master_Key, Session_Number)*

Cryptographic Key Management

- Key Generation
- Key Distribution
  - Key Exchange
    - *One party generates the key*
      - Requires key transport to all other parties
  - Key Agreement
    - *All parties influence the generation of the key*
      - Diffie-Hellman algorithm popular for two parties
- Key Storage
- Perfect Forward Secrecy (PFS)
  - Ensures that a session key derived from (a set of) long-term keys cannot be compromised if one of the long-term keys is compromised in the future

Key Generation

- Cryptographic keys must be intractable to guess
  - All possible cryptographic keys should be equally likely

- Typical key generation methods include:
  - Password Based Key Derivation Function
    - *generates a cryptographic key from a readable string*
  - Random number generator (RNG)
    - *Statistical RNG, not cryptographically secure*
      Never use this for Cryptographic purposes!
    - *Pseudo-random number generator (PRNG)*
      Must be seeded correctly, so use with care
    - *True random number generator*
      Uses measurements of physical processes to generate "real" randomness - too expensive for most applications

True random numbers are available online: `https://www.random.org`

Key Exchange
Requirements

- In addition to being generated, the key must also be distributed to all legitimate parties
  - How to prevent others from seeing the key?
  - How to authenticate the legitimate parties (sender and receiver)?
  - How to distributed the key to the legitimate parties?
  - How to ensure that the key is fresh?
  - How to verify that the legitimate parties received the key?

If done remotely: Use cryptography (many different solutions)

Sometimes easier: Personal key exchange (see cryptoparties)

Key Exchange
Techniques

- For key exchange we have the following options:
  - Generated by all parties working together (key agreement)
  - Generated by one party, then published in a public place
    Public Key Infrastructure (PKI)
  - Generated by one party, then sent to the other (key transport)
  - Generated by a trusted third party and sent to all parties
    (TTP key transport)
    - *TTP is often called Key Distribution Centre (KDC), e.g. used in Kerberos*

Key Storage

- Cryptographic keys must be stored *in use* and *at rest*
- In use
  - Keys must be available in memory when they are used
    - *Session keys (Data keys) are required when session is running*
    - *Other keys must only be available when needed*
      - Typically only during the handshake
      - Must be overwritten in memory immediately afterwards
      - Keys in memory was a problem with Heartbleed
- At rest
  - Keys and key generation material must be encrypted at rest
  - Example: Password Manager
- Hardware Secure Modules (HSM)
  - Dedicated hardware for key management and cryptographic operations

Key Expiration

- Keys can (in fact: should) expire sometime. Problems include:
  - How to keep track of key expiration?
  - How to inform all users of key expiration?
  - How to set up a new key?
  - What happens after expiration?
    - *Archive old key material? How?*
    - *Delete old key material? How?*
      - Remember to delete all copies!
      - Verification of deletion in distributed systems is difficult

Key Compromise

- Worst case: Key has been compromised because
  1. An attacker has potentially had access to the key
  2. The corresponding cryptographic algorithm was broken

- What do we have to do?
  - Key must no longer be used in the future
    - *Key expiration (see above)*
  - All concerned parties have to be informed!
    - *Key Revocation*
  - Old data has to be protected
    - *Re-encryption? Re-signing?*
    - *Destruction of all copies of old data*

Symmetric-Key Cryptography Protocol

- The following steps are required to setup communication using SKC
  1) Alice and Bob agree on a cryptosystem
  2) Alice and Bob agree on a key
  3) Alice encrypts her plaintext with the key
  4) Alice sends the ciphertext to Bob
  5) Bob decrypts the ciphertext with the key

How can these steps be compromised?

Problems of SKC

- Keys must be distributed in advance
- Keys must be kept secret from non-participants in the protocol
- A compromised key reveals all information encrypted with that key
  - Keys can be compromised:
    - *Weak keys (e.g. PRNG or KDF is weak) can be guessed*
    - *Man In The Middle (MITM)*
    - *Compromising hosts*
      - Probability of a host is compromised is *p*,
        probability of their shared key is compromised is *2p*
- Both parties have the same key
  - A compromised key can be used to masquerade as either party
    - *SKC cannot be used directly in adjudicated protocols*
- Different keys are needed for every pair of communicating parties
  - *n* users require *n(n -1)/2* keys

Public-Key Cryptography Protocol

- The following steps are required to setup communication with PKC
  1) Alice and Bob agree on cryptosystem
  2) Bob sends Alice his public key
  3) Alice encrypts plaintext with Bobs public key
  4) Alice sends ciphertext to Bob
  5) Bob decrypts ciphertext using his private key

How can these steps be compromised?

Public-Key Distribution

- Public-key distribution must address:
  - Authenticity (Certification) of public-keys
    - *Linking public-keys to named identities*
  - Distribution of public-keys
    - *Obtaining somebody else's public-key*
    - *Distribute own public-key*
  - Revocation of public-keys
    - *Revoking published keys*
    - *Determining validity of a given public-key*

Hybrid Cryptosystems

- Asymmetric cryptography is 3 orders of a magnitude slower than symmetric

- Asymmetric cryptography is vulnerable to chosen plaintext attacks

- If $C = E(P)$, where P is plaintext with entropy n, a cryptanalyst can encrypt all possible plaintexts and compare with the cipher text (thus he learns the plaintext, but not the private-key)
  - The cryptosystem fails to achieve its goals without being broken
    - *Not enough to use strong algorithms and keys, must also use them in a meaningful way*

Hybrid Cryptosystems II

- Asymmetric cryptography can be used to encrypt a randomly generated symmetric key (session key)

- Advantages
  - Randomly chosen symmetric key has entropy close to the key-size
  - The symmetric key is short (compared to the encryption key)
    - *Encrypting symmetric key is shorter than encrypting message*
  - The encrypted symmetric key reveals little about the asymmetric key

Hybrid Cryptosystems III

- The following steps are required to setup communication with hybrid cryptography

  1) Bob sends Alice his public key

  2) Alice generates a random session key, *K*

  3) Alice encrypts *K* with Bobs public key

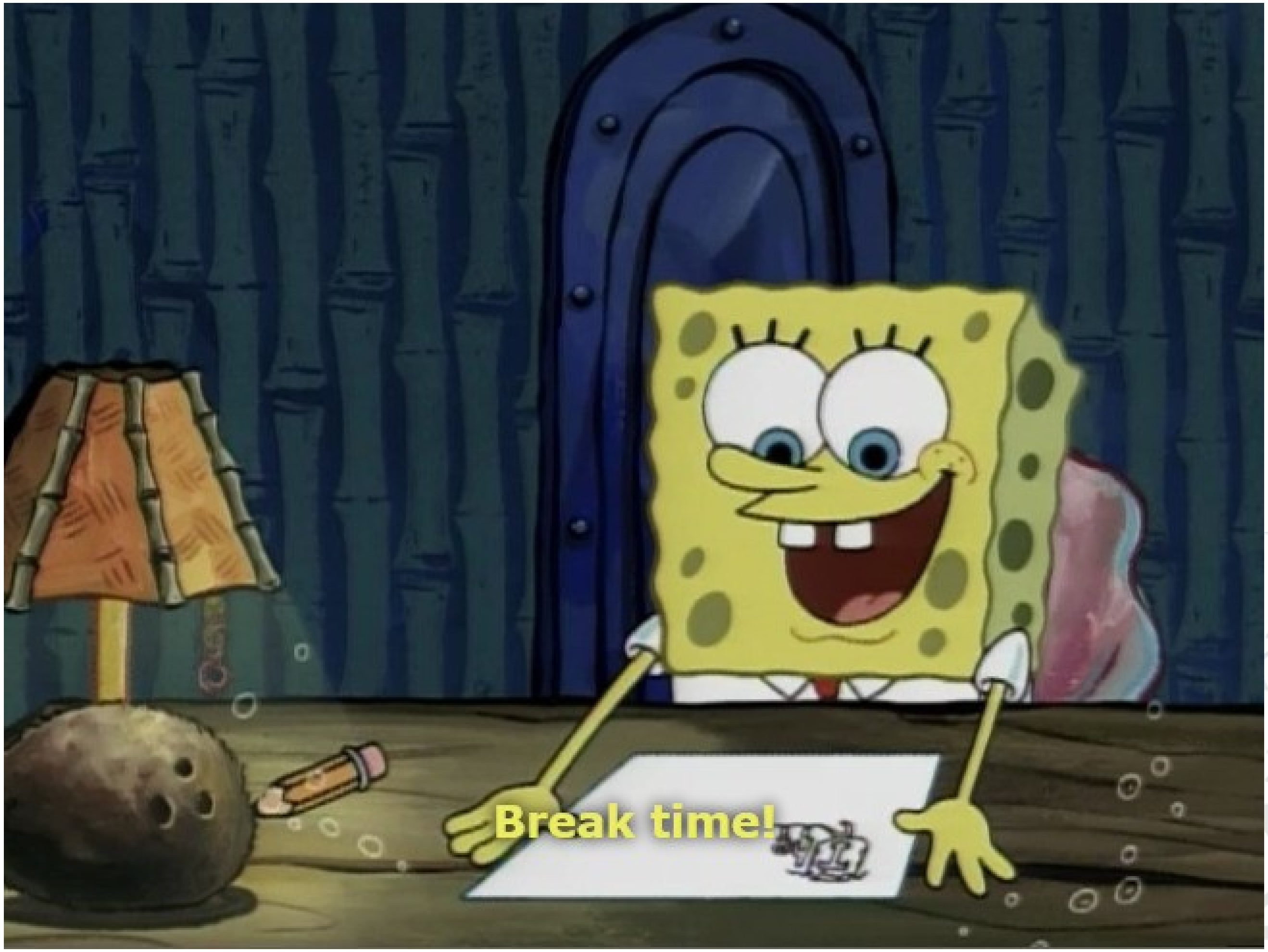
  4) Alice sends the encrypted key to Bob

  5) Bob decrypts *K* with his private key

  6) Alice and Bob exchanges messages encrypted with the session key

These steps still require authenticity of public-key and authentication of end points

Coffee Break



Public Key Distribution

- Public-key cryptography simplifies key distribution
  - secrecy of the encryption key is not required

- Authentication of Bob's public-key is required
  - In-Band (online)
    - *Public Key Infrastructures*
      - Key Distribution Centers (KDC)/Certificate Authorities (CA)
  - Out-of-Band
    - *Build into product*
    - *Published in Sunday Newspaper every week*

Verification of identity is hard

Certification Authorities to the rescue

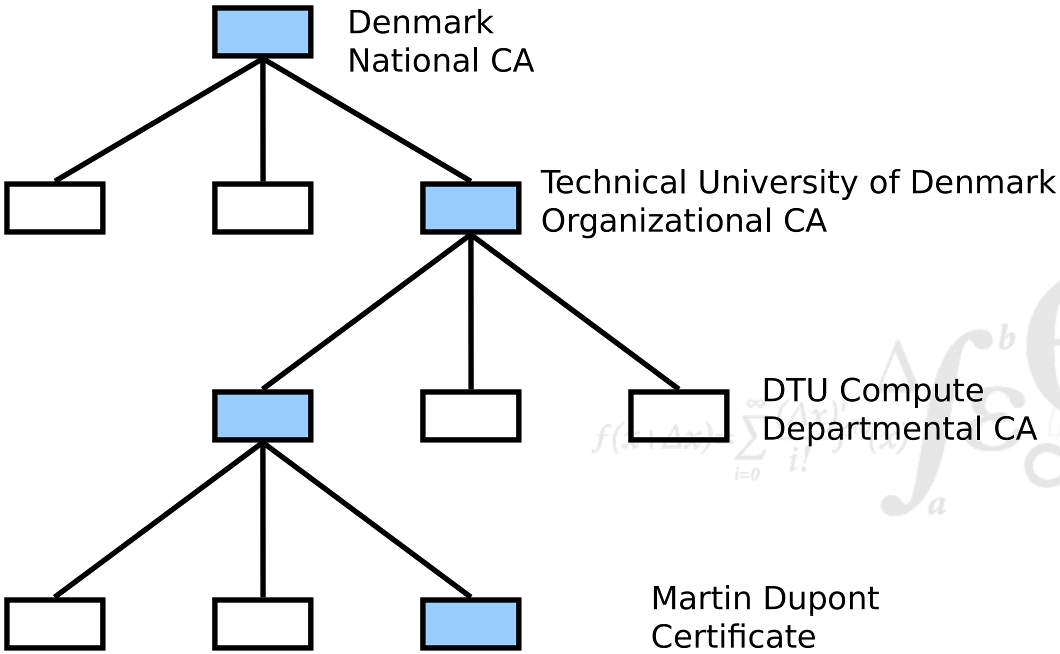
- A certification authority (CA) guarantees that the key belongs to the named principal
- A principal can be:
  - A user
  - An attribute of the user (e.g., her role in within an organisation)
  - An organisation (e.g., a company or another CA)
  - A pseudonym
  - A piece of hardware/software
- Some CAs only allow certain types of principals

Obtaining a Certificate from a CA

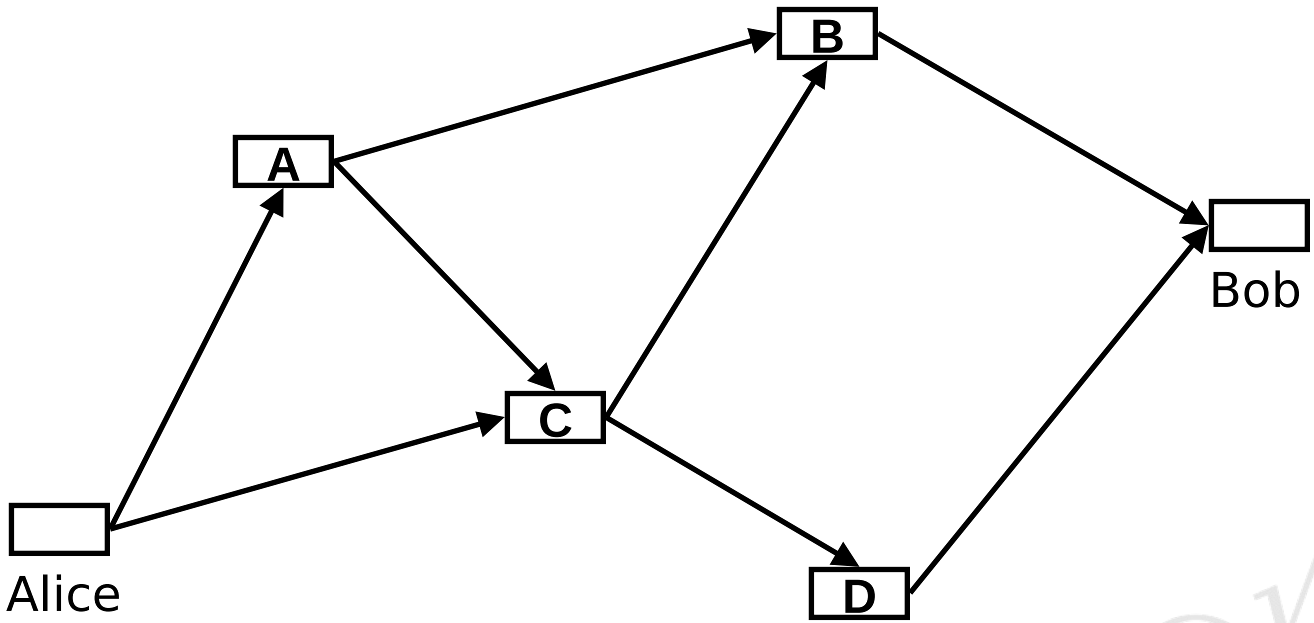
- Alice wishes to obtain a certificate from Charlie the CA
  1. Alice generates a public-/private-key pair and signs the public-key and identification information with the private-key
     - *Proves that Alice knows the private-key*
     - *Protects public-key and identification information in transit to CA*
  2. CA verifies Alice's signature on the public-key and her ID (CSR: Certificate Signature Request)
     - *Verification may be done out-of-band*
       - Email/phone callback
       - Business/Credit bureau records, in-house records
  3. CA signs Alice's public-key and ID with the CA private-key
     - *Creating a certificate that binds Alice's public-key to her ID*
  4. Alice verifies the public-key, ID and CA's signature
     - *Proves that CA didn't substitute Alice's public-key*
     - *Protection of the certificate in transit from CA*
  5. Alice and/or CA publishes the certificate

Certificate chains

- Certificate Authorities are organized in a hierarchy
  - Only top-level CA's certificate needs to be known by everyone
  - Intermediate CAs have certificates signed by "superiors"
  - Certificate verification is done by verifying certificates from the principal towards the top-level CA (aka. "Root CA")



Alternatives Trust Hierarchies



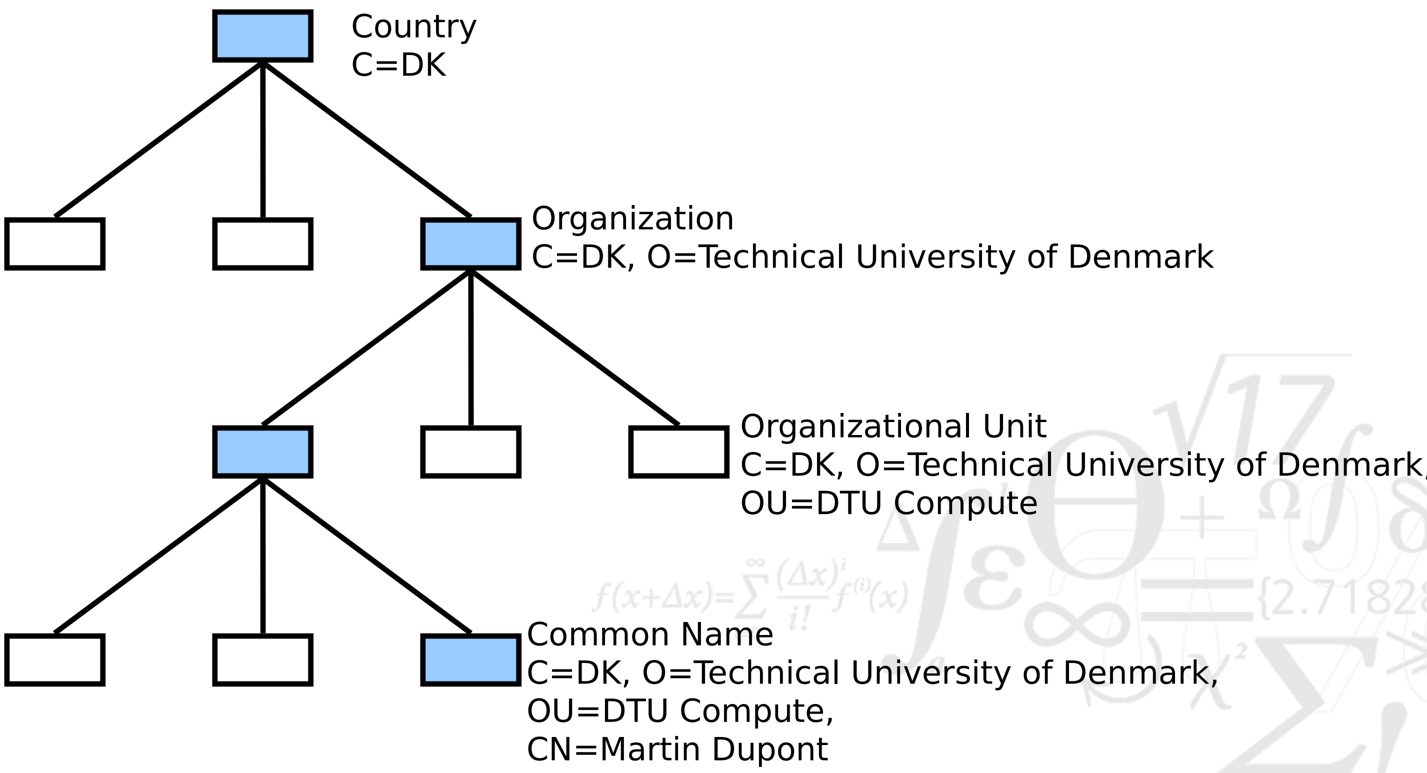
- Bob knows B and D who know A and C who know Alice => Bob knows the key came from Alice
- Web of trust closer to human notions of trust
  - This is the method used in PGP (and GPG)
  - Cryptoparty: cryptohagen (https://cryptohagen.dk/)

What's in a Name

- Alice uses a PKI to find Bob's certificate
  - Which PKI?
  - Which Bob?
- Names are context dependant
  - Bob is personally known by everyone in the village
    - *People have many names: Robert Johnson, Big Bob the sheriff, …*
  - There are many people called Bob (or Robert) in the town
    - *People may not know that Robert Johnson = Big Bob*
  - The ASCII string "Bob" loses meaning in the big city
- Qualifiers can be added to names
  - Passport number
  - Central Person Register (CPR) number
- Are we looking for Bob123 or Bob421?

X.500 Naming

- X.500 defines Distinguished Names (DN) that uniquely names everything on earth



X.500 Distinguished Names

- Typical DN components
  - Country (C)
  - State or Province (SP)
  - Locality (L)
  - Organisation (O)
  - Organisational Unit (OU)
  - Common Name (CN)

- Problems with X.500 Distinguished Names
  - No rules for how the naming hierarchy should be organised
  - Hierarchical naming only works in clearly hierarchical contexts
    - *Governments, national telecoms providers, …*
    - *Cannot accommodate nomadic people, stateless people, people with dual citizenship, …*

What's in a Certificate

- A typical certificate contains
  - Public-key (e.g., 4096bit RSA key)
  - Identification (e.g., X.500 DN)
  - Validity
    - *Not valid before, not valid after*
      - TLS certificates are now valid for at most 13 months
  - Issuer identification
    - *Used to establish certification path back to root CA*
  - Issuer signature
- Extensions may qualify the certificate
  - Certificate can only be used for certain purposes
    - *Especially important for CA certificates*
      - Establish the domain of authority for the CA

Example of a certificate

Certificate

hackertyper.net | R11 | ISRG Root X1

Subject Name

Common Name	hackertyper.net

Issuer Name

Country	US
Organization	Let's Encrypt
Common Name	R11

Validity

Not Before	Wed, 04 Sep 2024 08:22:56 GMT
Not After	Tue, 03 Dec 2024 08:22:55 GMT

Subject Alt Names

DNS Name	hackertyper.net

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	98:98:58:EB:EC:CB:B6:77:81:E8:70:0E:87:22:31:EF:D2:63:63:67:01:9C:90:4E:...

Miscellaneous

Serial Number	04:1E:2C:25:8D:CD:65:C1:68:55:99:11:8B:18:9B:9C:4F:F8
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

Fingerprints

SHA-256	C7:9C:73:FE:50:B5:F1:3E:B5:41:5B:78:CE:AB:D7:7F:3E:05:F3:58:C8:4D:F1:C0:...
SHA-1	DB:35:5A:B0:43:C1:63:2F:DC:A7:18:04:E4:48:7E:A4:6B:98:16:B5

Basic Constraints

Certificate Authority	No

Key Usages

Purposes	Digital Signature, Key Encipherment

Extended Key Usages

Purposes	Server Authentication, Client Authentication

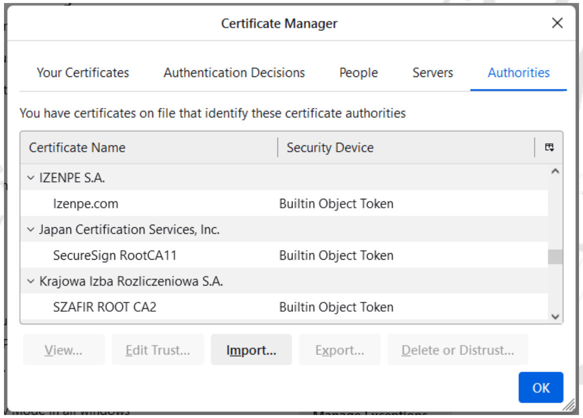
Subject Key ID

Key ID	58:C9:B2:AA:68:E6:A5:48:CC:D8:2B:E8:42:B2:BF:7F:BE:45:66:68

Authority of a CA

- What is the root of authority for a CA?
  - TLS certificate binds a public-key to a business' web-server
    - *CA has no authority to register businesses*
    - *CA has no authority to register domain names*

- How do we get the CA root certificate?
  - Built into most browsers
  - Firefox comes with around 100 root-certificates pre-installed, incl.:
    - *Digicert*
    - *Entrust, Inc.*
    - *Google Trust Services LLC*
    - *Microsoft*
    - *Unizeto Sp. z o.o.*
    - *Verisign, Inc.*



Trust in PKI

- Root CA can compromise the security of everyone
  - Root CA must be infallible
  - No single authority in the world is trusted by everyone
- Trust in root CA is diluted by the certification path
  - Problem exposed by Ueli Maurer’s model
  - 90% in CA gives 60% trust in certificate with 5 levels of CAs
- Adding attributes magnifies this problem
  - Credential containing permissions to access a particular resource lends authority from a root CA who knows neither principal nor resource
  - Further down the credential chain permissions become more specific, but the authority of the root CA more diluted

Certificate Revocation

- Certificates must be revoked whenever *(not if)* a private-key is compromised
- Revocation is measured by:
  - Speed of revocation: maximum delay from the compromise is discovered and the last use of the certificate?
  - Reliability of revocation: is it acceptable that someone sometimes may not have learned about the revocation?
  - Number of revocations: how many revocations can the system handle at the time?
- Revocation is either
  - Automatic, e.g., using short expiration times
  - Manual, e.g., using Certificate Revocation Lists (CRL) or the Online Certificate Status Protocol (OCSP)

Quantum Computer

- Quantum computers use quantum physics in order to run different kind of algorithm.
- Quantum computers are not super-fast computers: they cannot trivilize brute force search or NP-complete problems.
- Classical computers operate on bits, quantum computers operate on quantum bits (qbits) that can be both 0 or 1, and simultaneously (this state is called superposition).
- Quantum computers rely on entanglement: if two particles are entangled, observing the value of one gives the value of the other.

Post-Quantum Computer

- **Quantum speed-up**: if a quantum computer can solve a problem faster than a classical computer.
- **Exponential speed-up**: going from exponential complexity on classical computer to polynomial complexity on a quantum computer.
- **Shor's algorithm**: "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.", Peter Shor, 1995, provides exponential speed-up for:
  - Factoring
  - Discrete Logarithm (DLP)
  - Eleptic Curve Discrete Logarithm (ECDLP)

PQC and engineering challenges

- Quantum computer are not for tomorrow, but we do not know for when, so we need to prepare
- 4 types of post-quantum algorithm:
  - **Code-based**: based on error-correcting codes and decoding of linear code (BIKE, Classic MCEliece, HQC);
  - **Lattrice-based**: based on closest-vector problem (CRYSTALS-Kyber), in practice it can be hard to translate into a secure cryptosystem;
  - **Multivariate**: based on multivariate quadratic equations, but no multivariate PQ algorithms has been standardized yet;
  - **Hash-based**: based on the security of hash functions rather than mathematical hard problems (SPHINCS+)
- This raises substantial engineering challenges: security levels, secure implementations, updated security requirements, technical debt…

Online elections are appealing...

- Ease of use:
  - easy to tally complex voting systems (e.g., Condorcet);
  - easy to deploy widely.
- Elections can last for several days or weeks.
- But it is difficult to offer the same guarantees that traditional paper ballots:
  - Attackers torpedo mail ballots for months;
  - Attackers demand to stop the count because “They are finding votes all over the place”;
  - Citizen are doubtful about whether voting machines give out the right results.

… but security requirements are complex

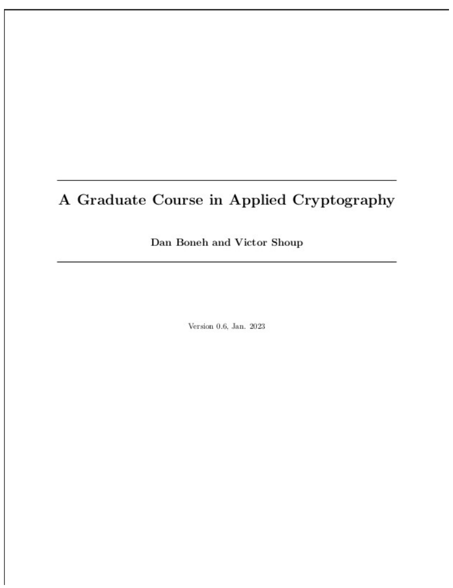
- Privacy properties:
  - **Voting Privacy**: no one should learn more about voters and votes than their numbers and the final result of the election;
  - **Receipt Freeness**: no voter has a way to prove how they voted;
  - **Coercion Resistance**: no voter can be force to vote a certain way.
- Example in a paper ballot election:
  - The envelope ensures voting privacy;
  - No receipt on the content of voters’ envelope: receipt-freeness;
  - Voting booths that voters enter alone ensure coercion-resistance.

… but security requirements are complex

- Verifiability (End-to-End Verifiability)
  - **Individual verifiability**: a voter can check that their vote has been properly counted;
  - **Universal verifiability**: everyone can check that the result corresponds to the ballots on the public bulletin board;
  - **Eligibility verifiability**: ballots come from legitimate voters.
- Example in a paper ballot election:
  - Transparent ballot boxes ensure individual verifiability;
  - All can participate in and observe the tallying: universal verifiability;
  - Everyone can be a polling station clerk: eligibility verifiability.

To learn more

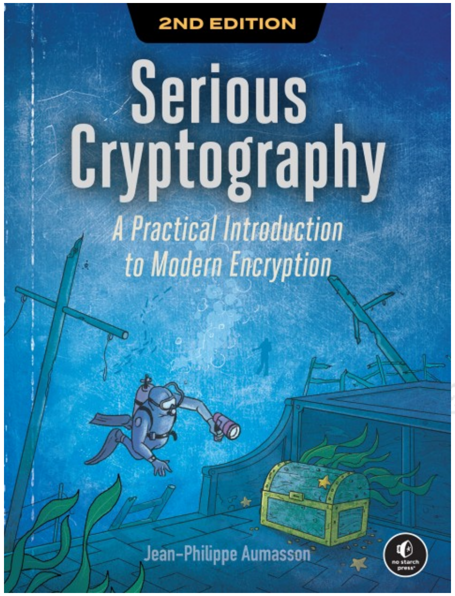
- A Graduate Course in Applied Cryptography by Dan Boneh   and Victor Shoup: https://toc.cryptobook.us/
- Serious Cryptography, 2nd Edition by Jean-Philippe Aumasson
- The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography, Simon Singh

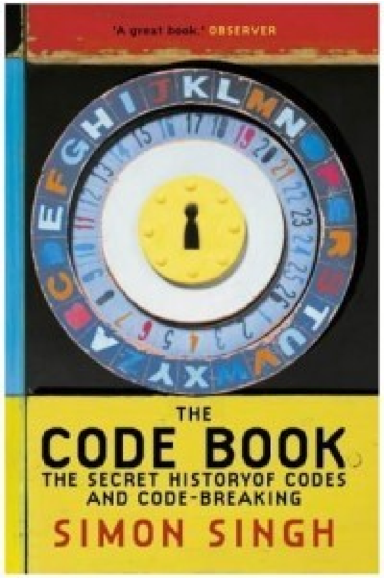


A Graduate Course in Applied Cryptography

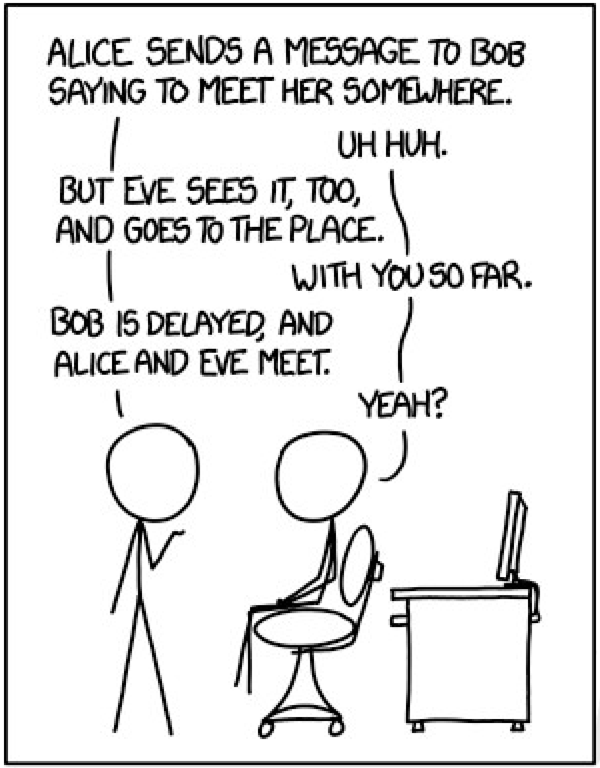
Dan Boneh and Victor Shoup

Version 0.6, Jan. 2023





Have fun with security protocols next week!



I'VE DISCOVERED A WAY TO GET COMPUTER SCIENTISTS TO LISTEN TO ANY BORING STORY.