UNIVERSITÀ DEGLI STUDI
DI TRENTO

Network Security

AA 2015/2016

Crypto (Review)

Dr. Luca Allodi

Some slides edited from Crispo & Zhauniarovich

Cryptography

- **Cryptography** (or cryptology; derived from Greek κρυπτός *kryptós* "hidden," and the verb γράφω *gráfo* "write" or λεγειν *legein* "to speak") is the practice and study of hiding information.

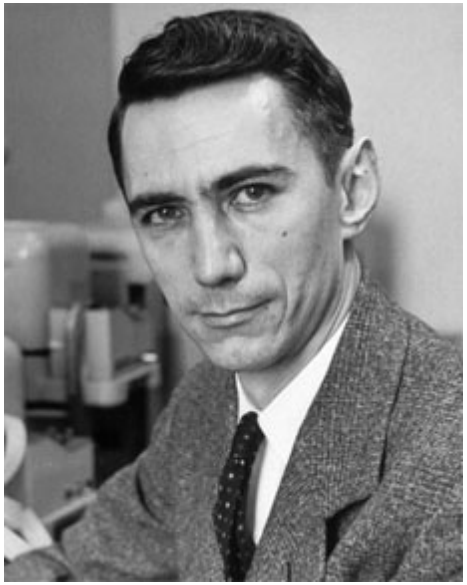
Cryptanalysis

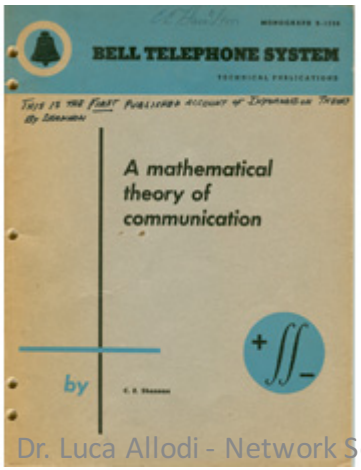
- **Cryptanalysis** (from the Greek *kryptós*, "hidden", and *analýein*, "to loosen" or "to untie") is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required to do so. Typically, this involves finding the secret key

Cryptanalysis

- Attacks:
  - Cryptanalytic attack
  - Brute-force attack
- Cryptanalytic attacks:
  - **ciphertext only** – (enc. alg., ciphertext)*
  - **known plaintext** – (*) + some plain/cipher pairs
  - **chosen plaintext** - (*) + get own plaintext + its corresponding ciphertext
    - e.g. attacker generates ciphertext using a public key, tries to infer secret key
  - **chosen ciphertext** – (*) + get own ciphertext + its corresponding plaintext (adaptive, non-adaptive)
    - attacker can decrypt any ciphertext they wish, tries to infer secret key
      - E.g. first implementations of SSL

Claude Shannon (1916 - 2001)





- *A Mathematical Theory of Communication* (1948), outlining what we now know as Information Theory
- Described ways to measure data using the quantity of disorder in any given system, together with the concept of entropy
- *The Magna Carta of the information age*

Shannon's Theory

- Shannon theory describes
- Confusion (K ⇔ C)
  - Confusion makes the relation between the key and the ciphertext as complex as possible.
- Diffusion (M ⇔ C)
  - Diffusion refers to the property that the statistics structure of the plaintext is dissipated into long range statistics of the ciphertext.
    - In contrast to confusion, diffusion spreads the influence of a single plaintext letter over many ciphertext letters.
- Unconditional Security
  - Unconditionally secure systems can not be broken even if all possible keys could be tried within short time.

Shannon's entropy

- Entropy H (Eta) is a measure of *unpredictability* of a set of codes, or information set
- For example, a coin that is perfectly balanced can assume heads XOR tails with equal probability.
  - In that case we need 1 bit of information to measure this grade of unpredictability → either 1 or 0.
- Question time: In the case the two probabilities are not perfectly balanced (i.e. $P(heads) \gtrless P(tails)$), what should happen to the entropy measure?

Shannon's entropy cntd

- In general, the entropy H of an information set Ω observed with probabilities ζ=[P(x_1), P(x_2),.. P(x_n)] can be computed as

$$H(\Omega) = -\sum_{i=0}^{n} P(x_i) log_2 P(x_i)$$

- Where $x_i \in \Omega \; and \; P(x_i) \in \zeta, \forall \; i \in [0, n]$
- The log need not be base 2 → this base is used to compute the average number of bits representing that information set

Shannon entropy example

- Assume we have a string AABBDCDA
  - How many bits on average are needed to represent it?
- Ω=[A,B,C,D]
- ζ = [P(A)=3/8 ; P(B)=2/8; P(C)=1/8; P(D)=2/8]
- H(AABBDCDA)=1.91
  - On average 1.91 bits of information to represent this information set
- Question time: H(AABBCCDD)=?
- When all values have the same probability, the entropy measure is $\log_2(|\Omega|)$

Network Security

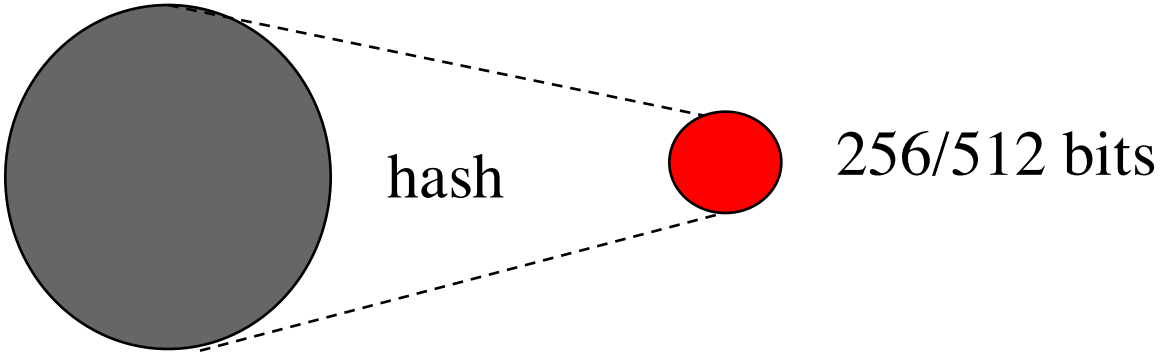
Cryptographic Hash Functions

Hash Functions

- **DEF**: A **hash function** maps a variable-length message into a fixed-length hash value, or message digest.
- **DEF**: The kind of hash function needed for security applications is referred to as a **cryptographic hash function**

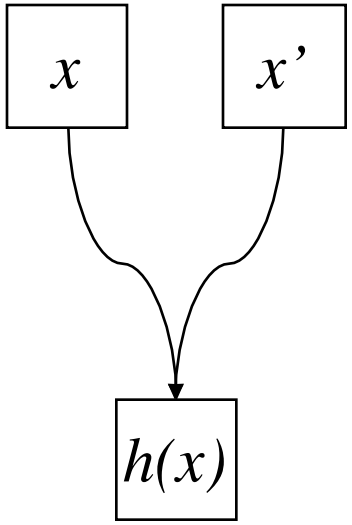
Hash Functions

- Length-reducing function h
  - Map arbitrary strings to strings of fixed length



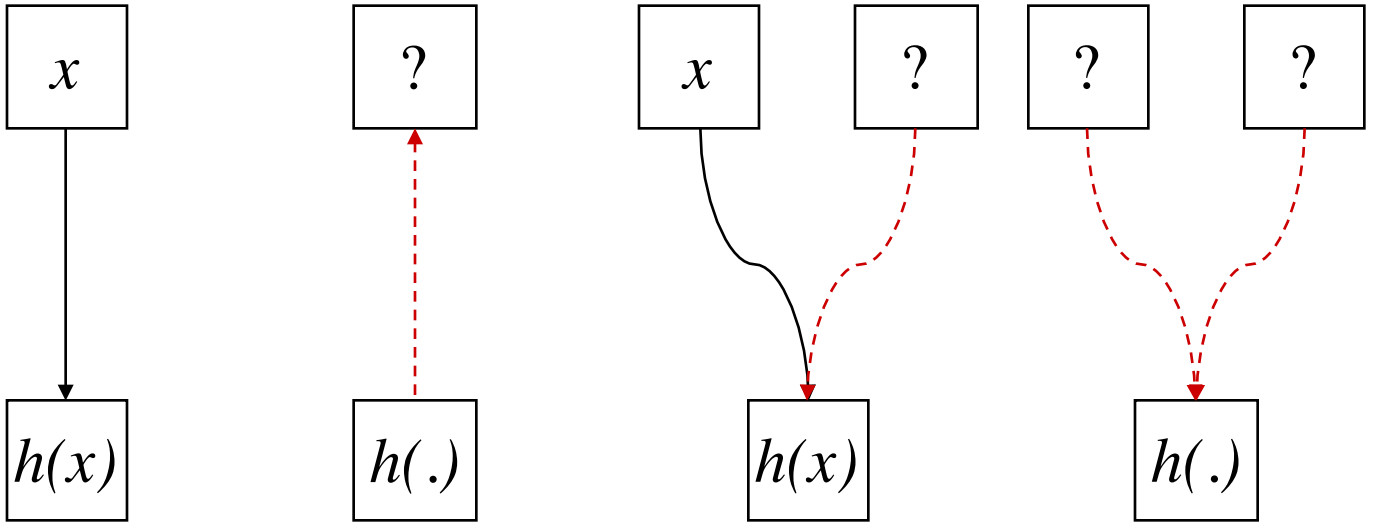
- A change in a bit or number of bits in the input message results, with high probability, in a change to the hash code

Collisions



collision

Security Requirements for Cryptographic Hash Functions



ease of computation

pre-image resistance (one-way function)

2nd pre-image resistance

collision resistance

One-way functions: properties

- A hashing function must be **not invertible**
- For each given hashing function H(), it should be hard to find m_1 and m_2 such that $H(m_1)=H(m_2)$
- If the output of H(.) follows a uniform distribution (i.e. all outputs are equally likely) then no cryptoanalysis will help
- To find a collision you need a brute force attack
- → **Collision resistance**
  - Try all combinations m_i, m_j until $H(m_i)=H(m_j)$
  - Depending on output size, that may take several universe lifetimes to accomplish
  - e.g. SHA-0 is 160bit → 2^{160} equally likely outputs
    - Assume 10B (hash computation+comparison)/sec = approx 4.5×10^{30} years
- 2^{nd} pre-image resistance → hard to find x' s.t. H(x)=H(x')
  - "weak collision resistance"
- Collision resistance → hard to find any tuple (x,x') s.t H(x)=H(x')

Applications of One-way Hash

- **Password files**
  - Rather than storing password in clear, store password hash
- **Data integrity**
  - Compute and securely store hash of some data
  - Check later by recomputing hash and comparing
  - E.g. used in software distribution
- **Digital signatures**
  - Sign hash of message instead of entire message
- **Keyed hash for message authentication**
  - MAC – Message Authentication Code

Hashing password files

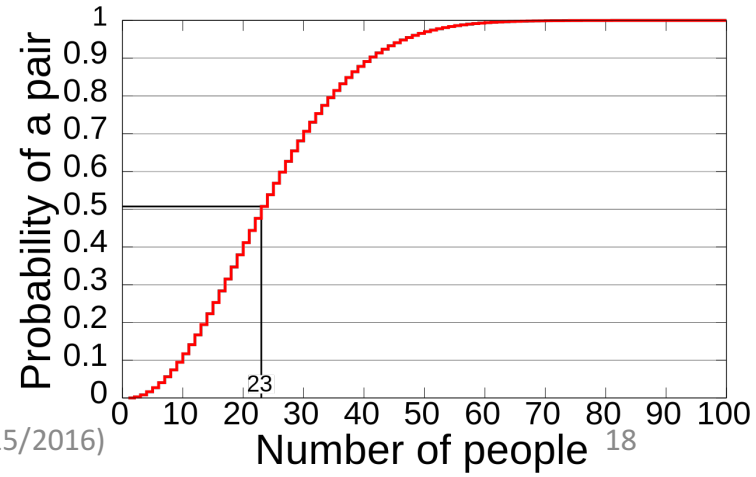
- The system grants access if

  hash(input_password)=hashed_stored_password

- If attacker breaches in, gets list of hashes H(.)
- Hard for the attacker to find password p s.t. H(p) in H(.)
  - → which property of hashing functions is important here, and why?

Collision resistance vs birthday paradox

- Brute force attack requires $O(2^n)$ computation time (size of input)
- But how likely is it to find a match in reasonable time?
- → birthday paradox
  - Likelihood of having a match P'() scales up fast with number of trials
    - Usually faster than assumed
      - (hence the "paradox")



Birthday attack

- Ω = Space of possible hash values *h* of 160 bits
  - Size of Ω is 2^160=n
- At first trial, we have
  - P(1,n) = $1/2^{160}$ ← probability of a collision
  - P'(1,n)= 1-P(1,n)= $(2^{160}-1)/2^{160}$ ← probability of no collision
- At second trial, we have
  - P'(2,n)=(1-P(1,n))(1-P(2,n))=$(2^{160}-1)/2^{160}$ x $(2^{160}-2)/2^{160}$
- At the k^{th} trial we have
  - $P'(K,n) = \prod_{i=0}^{k} \frac{n-i}{n}$
  - $P(K,n) = 1 - P'(K)\ or$
    - $P(K,n) = 1 - \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)..\left(1-\frac{k}{n}\right)$

Birthday attack cntd

- $P(K)$ can be hard to compute depending on values of K, n
- A good approximation is

$$P(K,n) = 1 - e^{-k(k-1)/2n}$$

- Hence to obtain the number of trials k over which one gets a certain probability of finding a collision p' is

$$e^{-k(k-1)/2n} = 1 - P(K,n)$$

$$-k(k-1) = 2n\ln\big(1 - P(K,n)\big), and\ since\ k(k-1) \approx k^2$$

$$k \approx \sqrt{2\mathrm{n}\ \ln((1 - \mathrm{P(k,n)})^{-1})}$$

- E.g. for $n=2^{160}$ and p=0.5 we have $K=1.42\text{x}10^{24} \approx 2^{80}$

Dictionary attacks

- Remember that if x=x' then H(x)=H(x')
- The entropy of a set H(.)=Ω of hashed passwords is the same as that of the passwords
  - Because there is a 1:1 correspondence between password and its hash (ignoring collisions)
- Thus if passwords where sampled over a uniform distribution, we'd have maximum entropy as there is full uncertainty → all passwords and hashes have the same probability of being observed
- But passwords are chosen by humans, and humans do no choose them from a uniform distribution of character sets
- In practice, as some passwords are much more likely than others, the entropy of a typical information set of hashed passwords Ω is less than $\log_2(|\Omega|)$

Dictionary attack cntd

- In practice it's typically easier for the attacker to find a collision in a set of hashed passwords because some passwords are likely to be identical
- Solution → apply a random salt *s* to the password
  - *H(concat(s,x))≠ H(concat(s',x))*
- Set of computed hashes will have a higher entropy than that of the cleartext passwords

Checksums

- The result of applying a hash function is called hash value, message digest, or checksum
- The last term creates frequent confusion
- In communications, checksums often refer to error correcting codes, typically a cyclic redundancy check (CRC)
- Checksums used by anti-virus products, on the other hand, must not be computed with a CRC but with a cryptographic hash function

Secure Hash Algorithm (SHA)

- SHA-0:
  - Developed in 1993
  - Output size: 160 bits
  - Design is similar to MD4
  - Found an undisclosed “significant flaw”
- SHA-1:
  - Substituted SHA-0 in 1995
  - Output size: 160 bits
- SHA-2:
  - NIST proposed a new standard in 2002
  - Output size: 256, 384 and 512 bits (224 bits added in 2008)
- SHA-3:
  - Chosen in 2012 after public competition among non-NSA designers
  - Internal structure significantly differs from the rest of SHA

News on Hash Functions

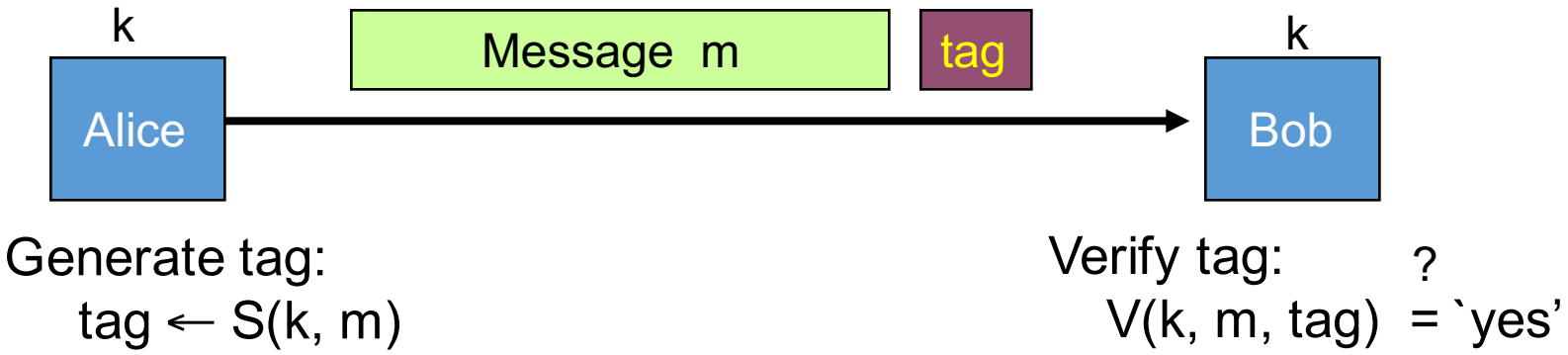
- MD5 hashing function has been found to be completely vulnerable
  - poc finds collision in a few hours on consumer systems
  - cryptographic artefacts enable much cheaper attacks than brute-forcing
- News (early 2005): "Reports that collisions for SHA-1 can be found in 2^{69} "steps".
- For 160-bit hash values, the yardstick is the computation of 2^{80} random hash values.
- Longer hash values are advisable: SHA-256

Message Authentication Code

- **Message authentication** is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid
- A message authentication code (MAC) is an algorithm that requires the use of a secret key. A MAC takes a variable-length message and a secret key as input and produces an **authentication code**. A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.
- It does **not** offer **non-repudiation**

Message Authentication Code: MAC

- Goal: message integrity. No confidentiality.
  - ex: Protecting public binaries on disk.
- Employed on SSL/TLS, IPSec



Network Security

Authenticated Encryption:
Encryption + MAC

Kerckhoffs's Principles

- The security of an encryption algorithm must reside **solely** in choosing the key
  - Algorithm is assumed to be known
  - Encrypted message can be seen
- One of the golden principles of security
- Never assume something is secure only because nobody knows it's there
  - "security by obscurity"

Goals

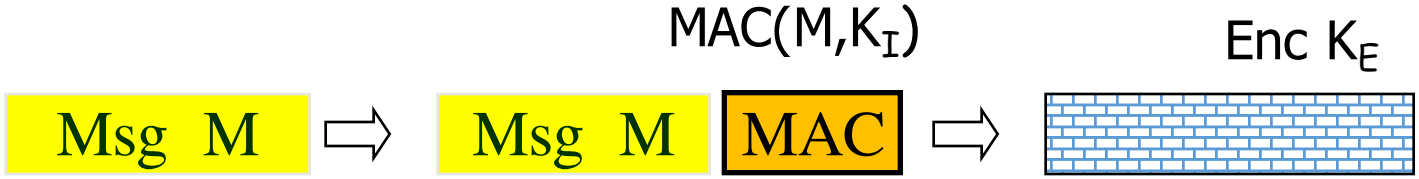
- **Confidentiality**: Encryption
  - Encryption secure against **eavesdropping** only
- **Integrity**: Message Authentication Codes
  - Existential unforgeability under a chosen message attack

- Prevent tampering?
  - **Confidentiality + Integrity**

Combining MAC and ENC

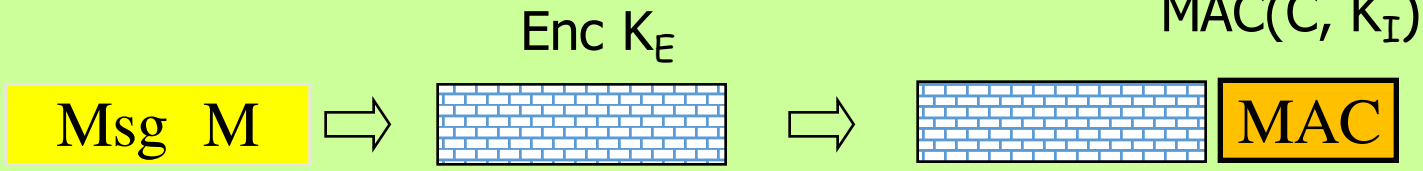
Encryption key K_E MAC key = K_I

Option 1: MAC-then-Encrypt (SSL)

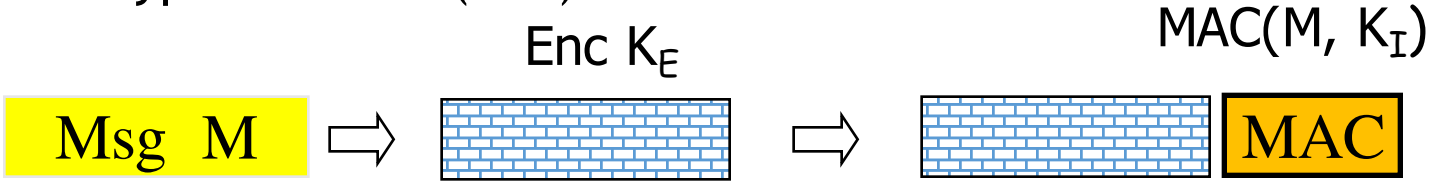


Option 2: Encrypt-then-MAC (IPsec)

Secure on general grounds

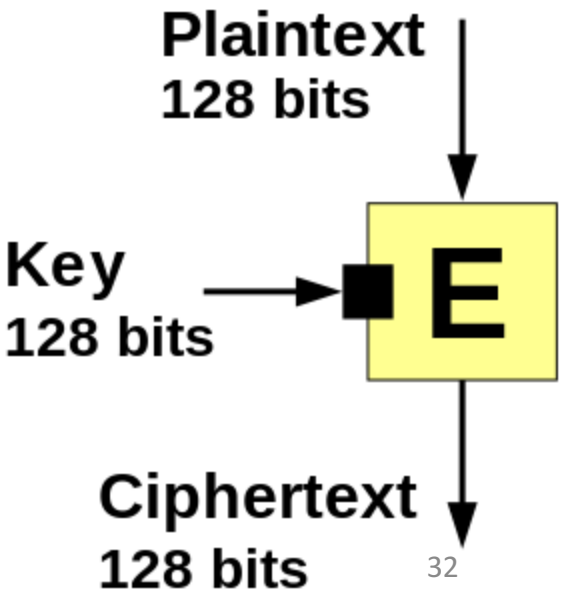


Option 3: Encrypt-and-MAC (SSH)



Block Ciphers

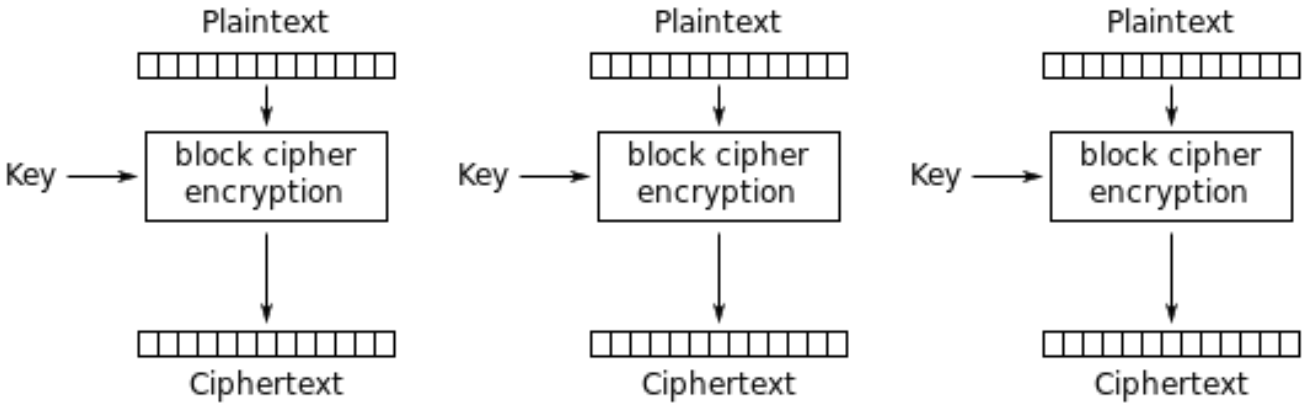
- A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.



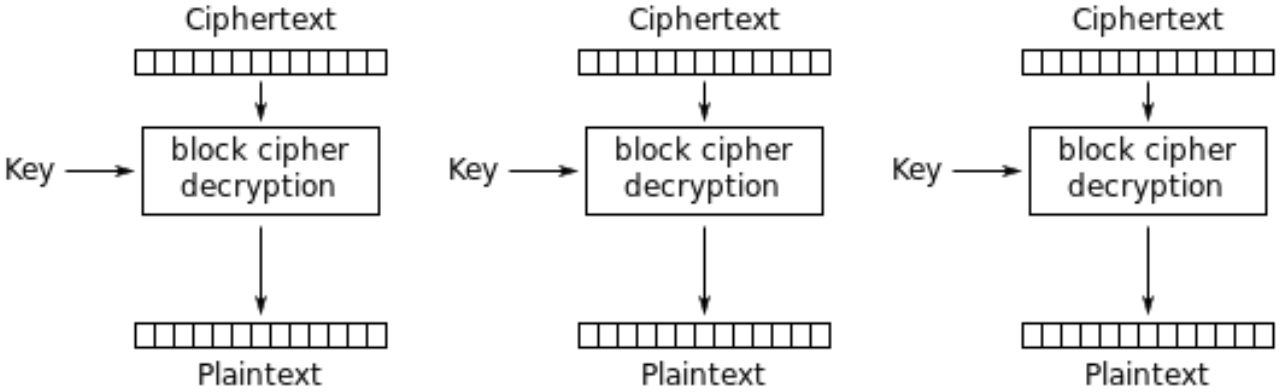
Block ciphers in practice

- Most messages M that need be encrypted are larger than 128 bites
- Two main modes of encryption
  - ECB (Electronic Code Book)
    - Split plaintext into blocks, encrypt each with the key, then concatenate encrypted blocks
  - CBC (Cipher Block Chaining)
    - Each block encrypted with key and XOR of previous encrypted block and cleartext block

ECB



Electronic Codebook (ECB) mode encryption

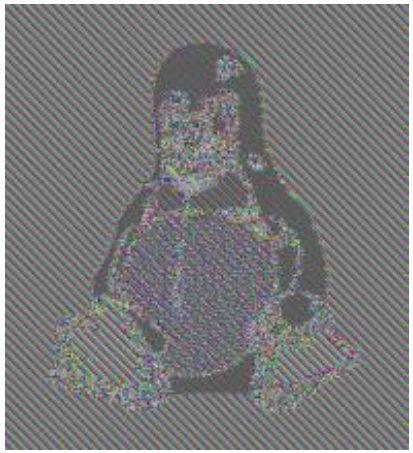


Electronic Codebook (ECB) mode decryption

Pictures from wikipedia

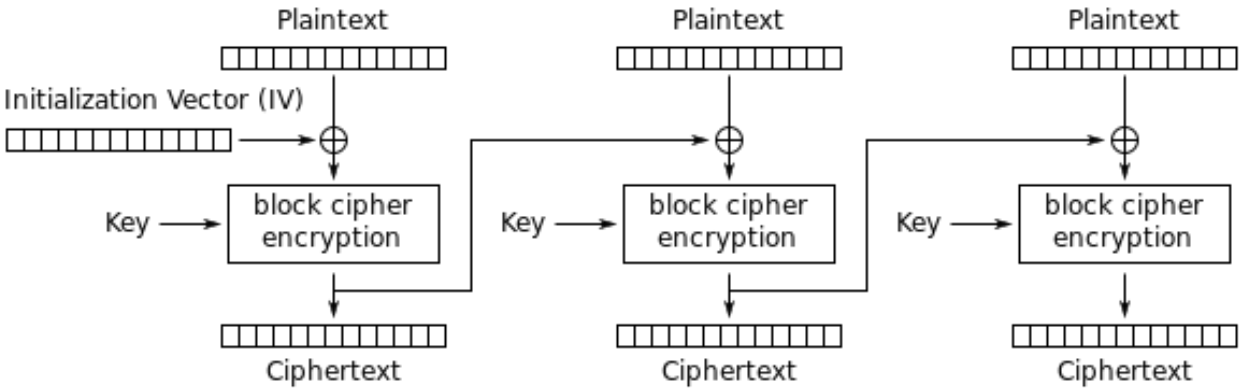
ECB - visible patterns

Encrypted in
ECB mode

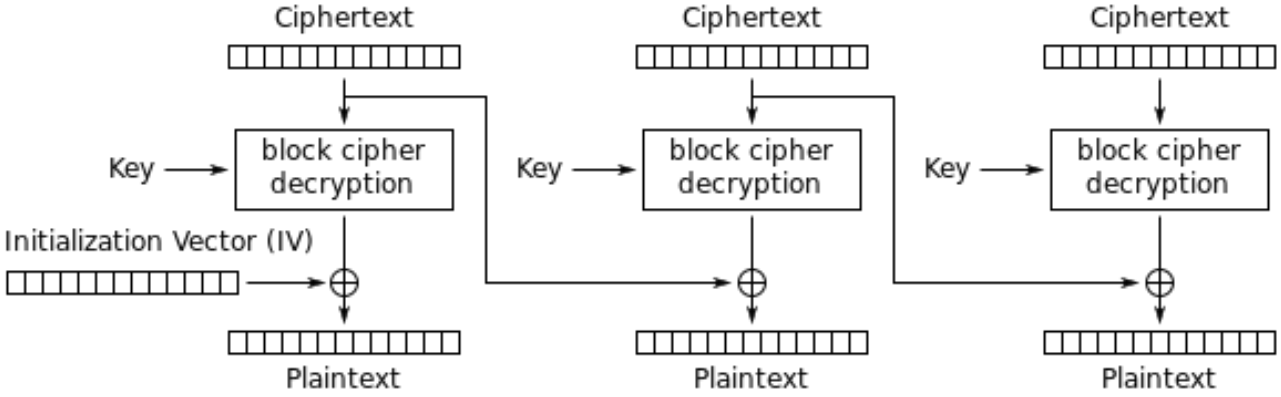


Pictures from wikipedia

CBC



Cipher Block Chaining (CBC) mode encryption

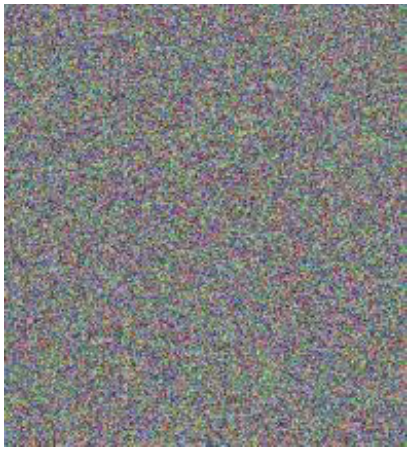


Cipher Block Chaining (CBC) mode decryption

Pictures from wikipedia

CBC – random patterns

Encrypted in
CBC mode



Pictures from wikipedia

Modern Block Ciphers

DES, 3-DES, Blowfish, IDEA, AES,…..

Applications:

- Encryption of file systems
- Encryption of the data on a credit-card
- Encryption of the digital data on your passport
- … and many more…

- How can the two ends of an encrypted communication share a *secret* key?

Network Security

Asymmetric Key Cryptography

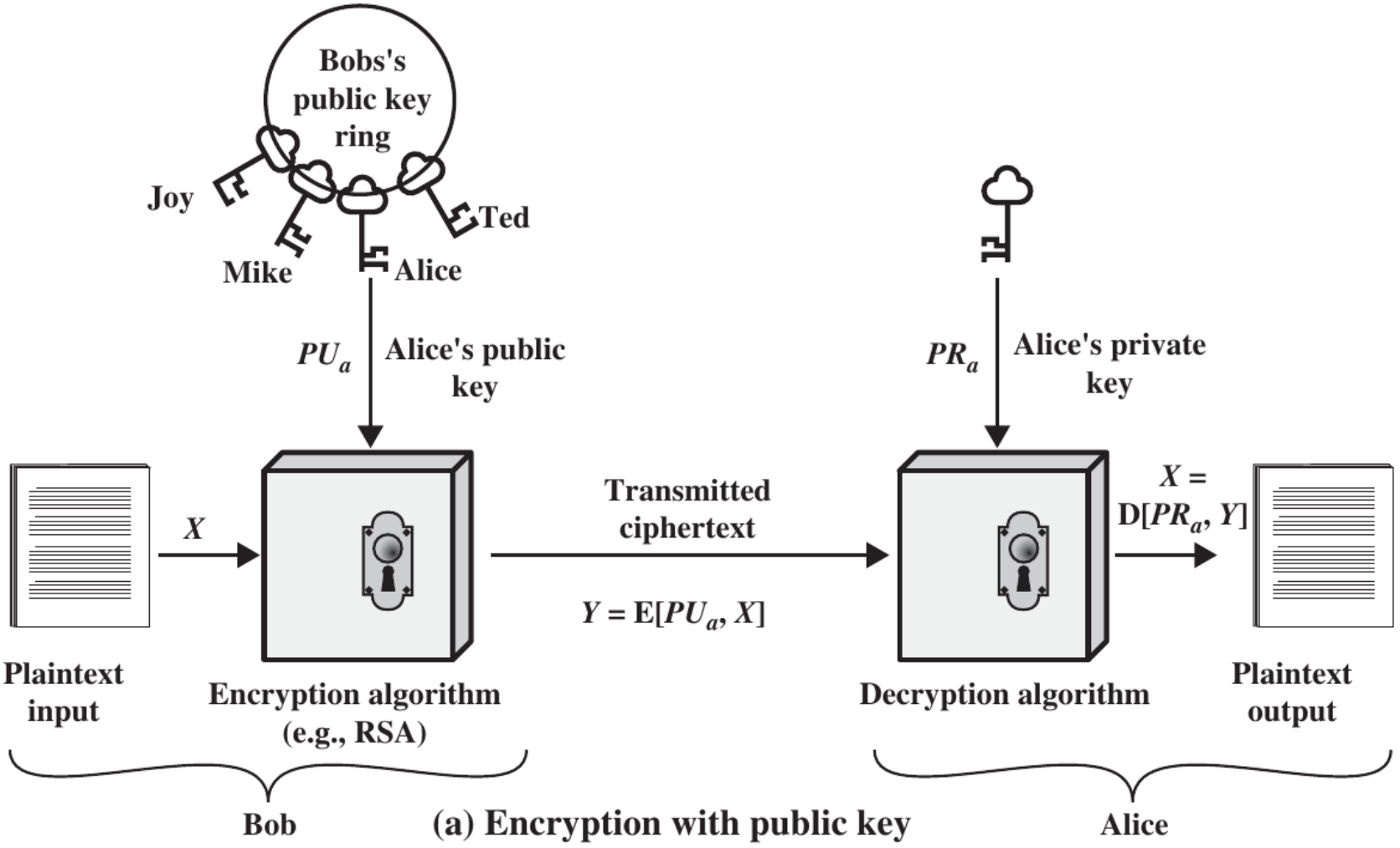
Public Key Encryption

- Proposed in the open literature by Diffie & Hellman in 1976.
- Each party has a public encryption key and a private decryption key.
- Computing the private key from the public key should be computationally infeasible.
- The public key need not be kept secret but it is not necessarily known to everyone.
- There exist applications where access to public keys is restricted.
- RSA is the most widely used public-key encryption algorithm

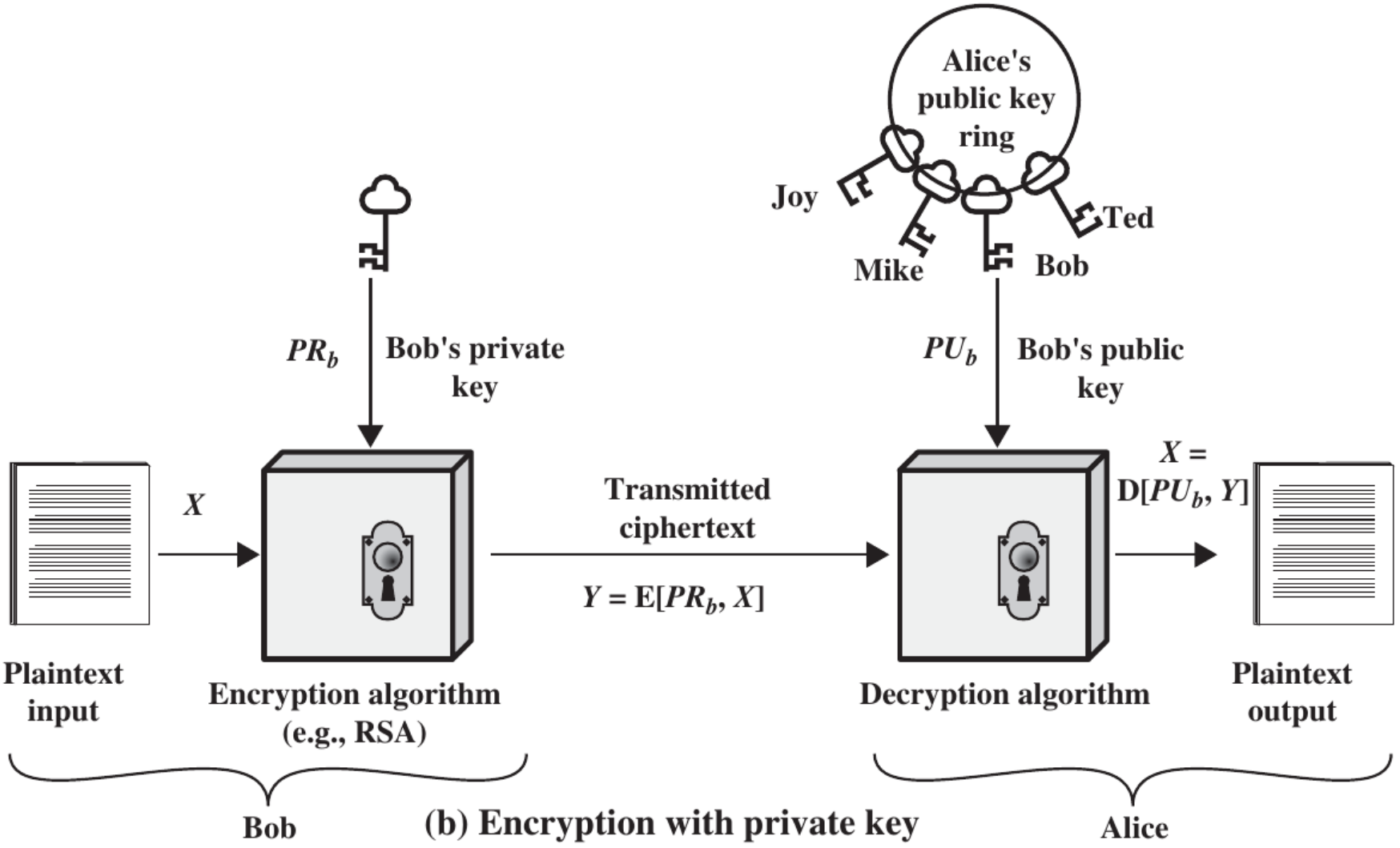
Public Key Encryption

- Encryption protects documents on the way from *A* to *B*.
- *B* has a public encryption key and a private decryption key.
- A procedure is required for *A* to get an authentic copy of *B*'s public key (need not be easier than getting a shared secret key).
- For *n* parties to communicate, *n* key pairs are needed.

Encryption with Public Key



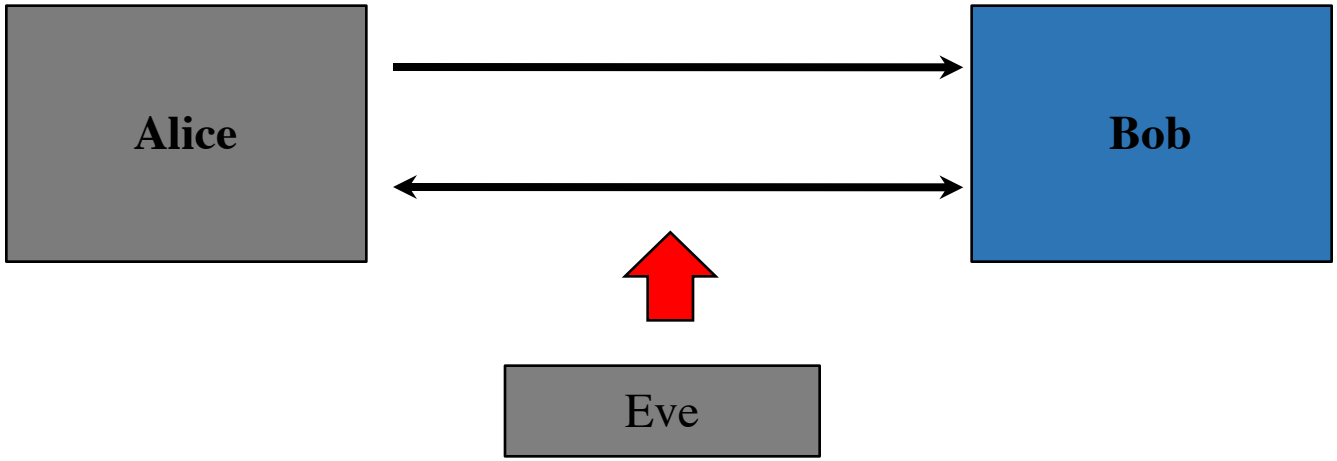
Encryption with Private Key



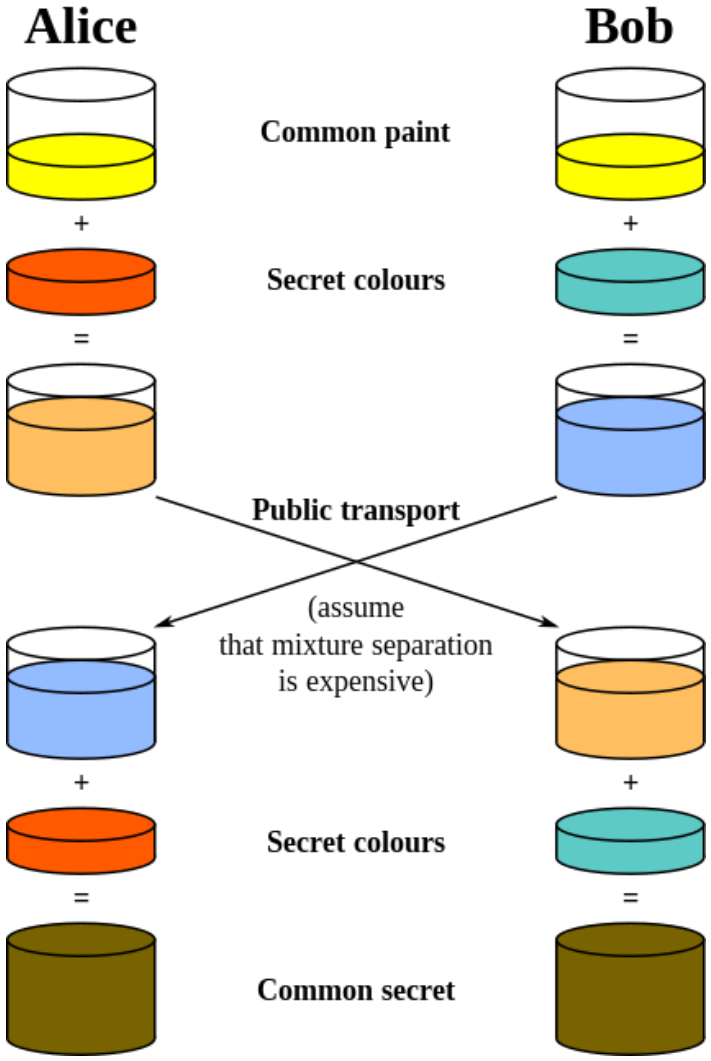
(b) Encryption with private key

Diffie-Hellman Protocol

- Simple public-key algorithm for key exchange
- Based on Discrete Logarithm Problem
- Secure against eavesdropping only



Diffie-Hellman Protocol: Idea



Diffie-Hellman Protocol

- p large prime and g is primitive root module p
  - $g^x \equiv a \pmod{p}$
    - a=coprimes of p
  - (g, p) is public
- Alice
  - chooses random, secret number a → $A=g^a \bmod p$
  - Sends A to Bob
- Bob
  - chooses random, secret number b → $B=g^b \bmod p$
  - Sends B to Alice

$$A^b \bmod p = B^a \bmod p = g^{ab} \bmod p = g^{ab} \bmod p$$

- Shared key $g^{xy} \bmod p$
  - Alice → Bob: $E[g^{xy} \bmod p, msg_1]$
  - Bob → Alice: $E[g^{xy} \bmod p, msg_2]$

Diffie-Hellman: Example

- Alice and Bob agree to use a prime number p = 5 and base g = 2 (which is a primitive root modulo 5).
- Alice chooses a secret integer a =3, then sends Bob A = g^a mod p
  - A = 2^3 mod 5 = 3
- Bob chooses a secret integer b = 2, then sends Alice B = g^b mod p
  - B = 2^2 mod 5 = 4
- Alice computes s = B^a mod p
  - s = 4^3 mod 5 = 4
- Bob computes s = A^b mod p
  - s = 3^2 mod 5 = 4
- Alice and Bob now share a secret (the number 4).
  - $A^b \bmod p = B^a \bmod p = g^{ab} \bmod p$
- Note that there is no authentication here. Bob shares p and g with whom he presumes is Alice
- To find s, attacker must solve $s = 4^a \bmod 5 = 3^b \bmod 5$
  - A,B, p, g are public

Diffie-Hellman: Example (2)

- Alice and Bob agree to use a prime number p = 23 and base g = 5 (which is a primitive root modulo 23).
- Alice chooses a secret integer a = 6, then sends Bob A = g^a mod p
  - A = 5^6 mod 23 = 8
- Bob chooses a secret integer b = 15, then sends Alice B = g^b mod p
  - B = 5^{15} mod 23 = 19
- Alice computes s = B^a mod p
  - s = 19^6 mod 23 = 2
- Bob computes s = A^b mod p
  - s = 8^{15} mod 23 = 2
- Alice and Bob now share a secret (the number 2).
- Note that there is no authentication here. Bob shares p and g with whom he presumes is Alice
- To find s, attacker must solve $s = 19^a \bmod 23 = 8^b \bmod 23$

Public-key crypto

- Scales well as opposed to symmetric crypto
  - One only needs to know public key of destination to send secure message
- RSA (by Rivest, Shamir & Adleman of MIT in 1977)
  - best known & widely used public-key scheme
  - based on exponentiation in a finite (Galois) field over integers modulo a prime
- The whole security of public key cryptography relies on three aspects
  - Secrecy of private key
  - Authenticity of public key
  - Factorisation is hard, i.e. computationally intractable ☾ is P=NP? unproven

Network Security

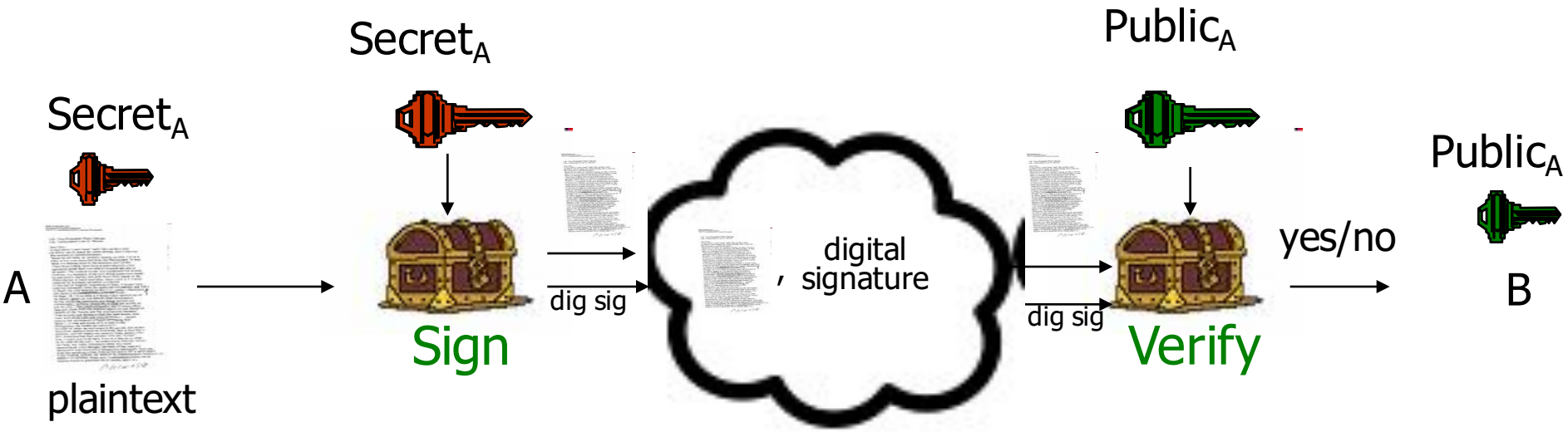
Digital Signatures

Digital Signature

- A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. The signature guarantees the source and integrity of the message.
- **Why?**
  - **MAC problems:**
    - **Alice may forge a different message and claim that it has come from Bob**
    - **Because parties share the key it is impossible to prove that someone has sent a message**

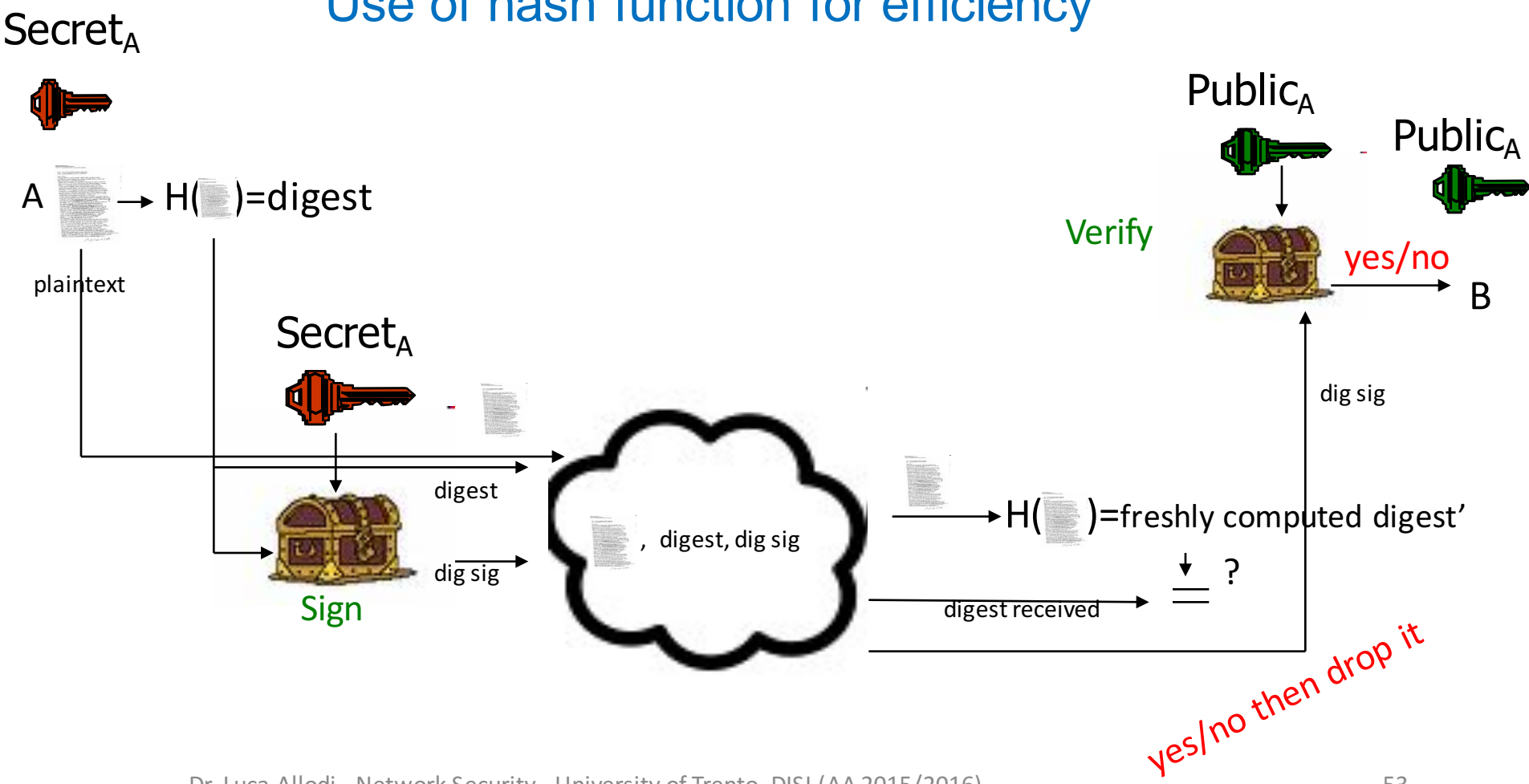
Digital signatures

Digital Signature for authentication and integrity

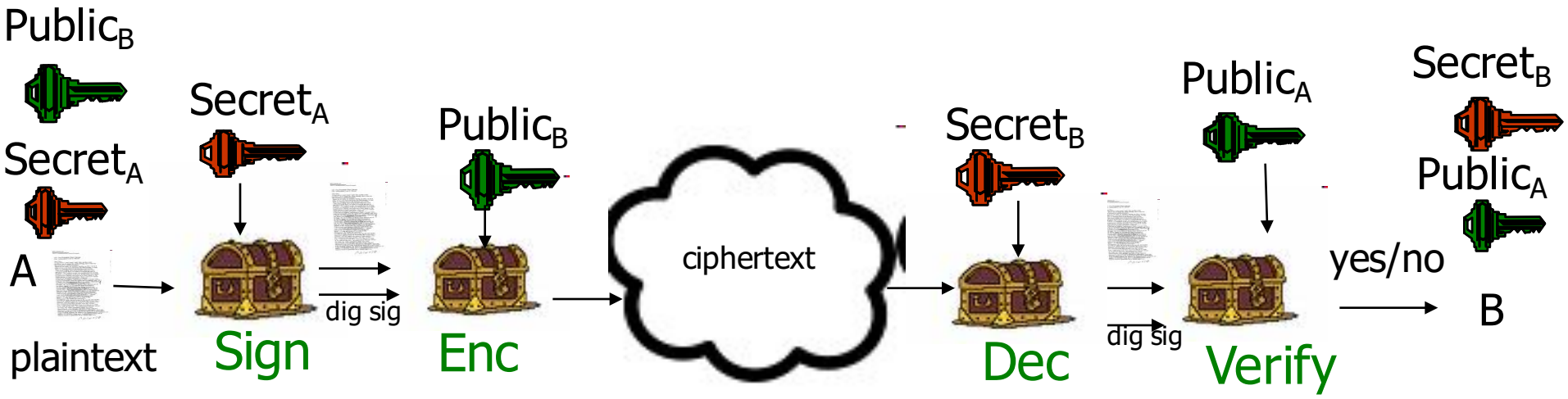


Digital signatures

Use of hash function for efficiency



Confidentiality, authentication and integrity



Timing Attacks

- developed by Paul Kocher in mid-1990's
- exploit timing variations in operations
  - eg. multiplying by small vs large number
  - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Other attacks

RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis*

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

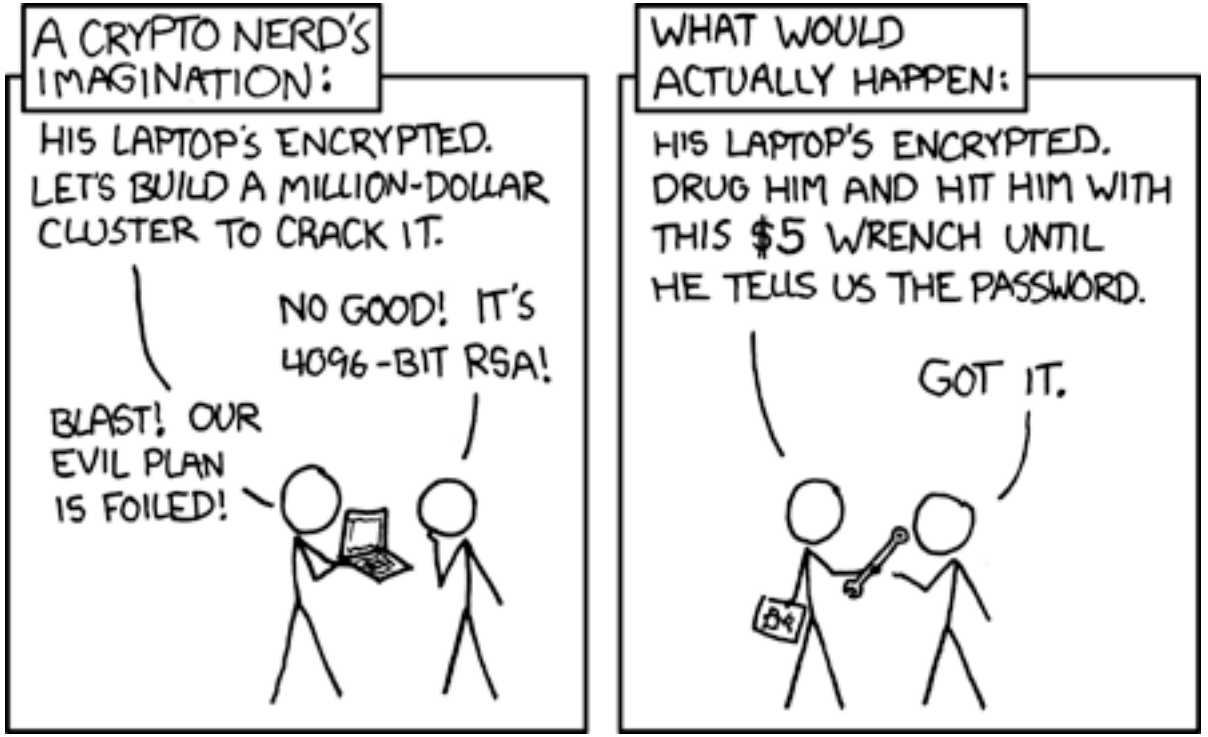
Adi Shamir
Weizmann Institute of Science
adi.shamir@weizmann.ac.il

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

December 18, 2013

In this paper we describe a new acoustic cryptanalysis key extraction attack, applicable to GnuPG's current implementation of RSA. The attack can extract full 4096-bit RSA decryption keys from laptop computers (of various models), within an hour, using the sound generated by the computer during the decryption of some chosen ciphertexts. We experimentally demonstrate that such attacks can be carried out, using either a plain mobile phone placed next to the computer, or a more sensitive microphone placed 4 meters away.

Attacks in practice



Attacks in practice, ctnd

- Most attacks on crypto happen because of bad implementation
- Crypto implementation is **hard**
  - Always use functions that already exist and have been extensively peer-reviewed
  - Many aspects → Random seeds, hardcoded keys, key derivation, ECB,..
  - Even well-established systems have occasional vulnerabilities (SSL's heartbleed, Apple's gotofail)
- If crypto is done well, attacks on the enc message/key are hard (likely too hard)
  - Exploit the human instead