Introduction to public-key cryptography

Part 1: basic constructions

Jean-Sébastien Coron

University of Luxembourg

Outline

- Part 1: basic constructions
  - History
  - Classical cryptography: block-ciphers, hash functions
  - Public-key cryptography: RSA encryption and RSA signatures, DH key exchange
- Part 2: applications of public-key cryptography (next lecture)
  - Security models
  - How to encrypt and sign securely with RSA. OAEP and PSS.
  - Public-key infrastructure. Certificates, SSL protocol.
  - Bitcoin and the cryptographic blockchain

Mono-alphabetic Cipher

- Each letter is replaced with another letter, according to a fixed substitution

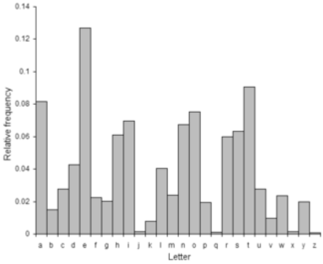
Plaintext :	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Ciphertext :	C G H U Z J T E L Y X I F O P K J W V A B D M S N Q

  Then HELLO WORLD enciphers to EZIIP MPWIU

- Number of possible keys is large
  - $26! = 2^{88.4}$ or 88 bits
  - How much time would it take to recover the key by exhaustive search ?
  - But...

Frequency analysis

- Frequency of letters in English:



- Cryptanalysis of mono-alphabetic cipher
  - The most frequent letter in the ciphertext is likely E,T or A.
  - Substitute and continue with less frequent letters.
  - WEAK

One-time pad (1917)

- Plaintext is xored with the key to produce the ciphertext

$$\begin{array}{rl} \text{Plaintext:} & 0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1 \\ \text{Key:} & \underline{\color{red}{1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0}} \\ \text{Ciphertext:} & 1\,0\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1 \end{array}$$

$\oplus$	0	1
0	0	1
1	1	0

  - $a \oplus b = a + b \bmod 2$

- Proved unbreakable by Shannon (1949) if key is random and as long as the plaintext.
  - Issue: key as long as the plaintext.
  - Used for the hotline between Washington and Moscow during the cold war. The key was delivered via their embassy in the other country.

One-time pad (1917)

- Plaintext is xored with the key to produce the ciphertext

$$
\begin{array}{rl}
\text{Plaintext:} & 0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1 \\
\text{Key:} & \underline{\color{red}{1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0}} \\
\text{Ciphertext:} & 1\,0\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1
\end{array}
$$

$\oplus$	0	1
0	0	1
1	1	0

  - $a \oplus b = a + b \bmod 2$

- Proved unbreakable by Shannon (1949) if key is random and as long as the plaintext.
  - Issue: key as long as the plaintext.
  - Used for the hotline between Washington and Moscow during the cold war. The key was delivered via their embassy in the other country.

One-time pad (1917)

- Plaintext is xored with the key to produce the ciphertext

$$\begin{array}{rl} \text{Plaintext:} & 1\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1 \\ \text{Key:} & \underline{\color{red}{0\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0}} \\ \text{Ciphertext:} & 1\,0\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1 \\ & \uparrow \end{array}$$

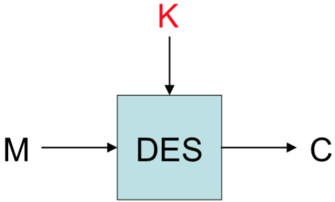
$\oplus$	0	1
0	0	1
1	1	0

  - $a \oplus b = a + b \bmod 2$

- Proved unbreakable by Shannon (1949) if key is random and as long as the plaintext.
  - Issue: key as long as the plaintext.
  - Used for the hotline between Washington and Moscow during the cold war. The key was delivered via their embassy in the other country.

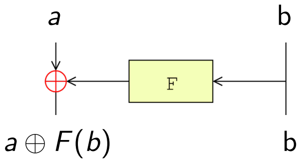
DES block-cipher (1976)

- Data Encryption Standard (DES), published as FIPS PUB 46.
- Developed by NBS (National Bureau of Standards), now NIST (National Institute of Standards and Technology), following an algorithm from IBM.
  - Superseded by the AES, but remains in widespread use.
- Input/output length: 64 bits.
- Key length: 56 bits.

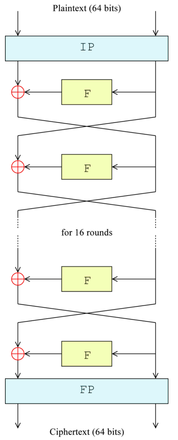


DES

- Feistel cipher
  - Transforms a function $F : \{0,1\}^n \to \{0,1\}^n$ into a permutation $P : \{0,1\}^{2n} \to \{0,1\}^{2n}$
  - For DES, $n = 32$.
- Single round

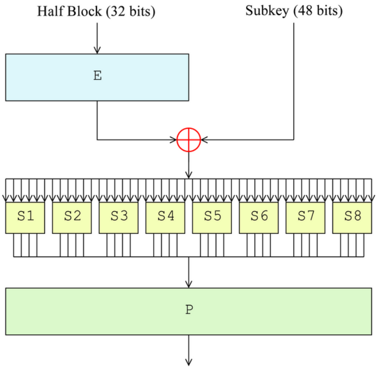


  - Why is it a permutation ?



DES

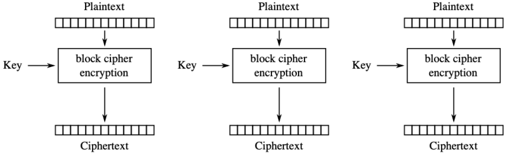
- F function



- SBox: $S_i : \{0,1\}^6 \rightarrow \{0,1\}^4$
  - How much memory is needed to store the 8 SBoxes ?

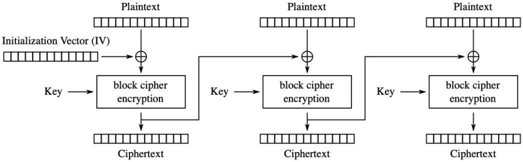
DES modes of operation

- Encrypting longer messages (> 64 bits)
- FIPS-81: DES modes of operation
- ECB: WEAK



Electronic Codebook (ECB) mode encryption

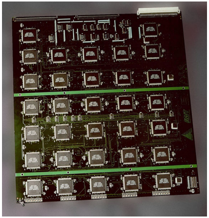
- CBC: OK



Cipher Block Chaining (CBC) mode encryption

Security of DES

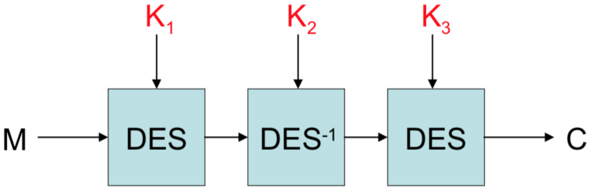
- Problem: key is too short (56 bits). Exhaustive search has become feasible
  - How much time would take exhaustive search on a modern computer ?
- DES cracker from Electronic Frontier Foundation (EFF). Breaks DES in 2 days (1998).



- Other attacks
  - Differential cryptanalysis (Biham and Shamir, 1990). 2^{47} chosen plaintexts. Linear cryptanalysis (Matsui, 1993). 2^{43} known plaintexts.

Triple DES

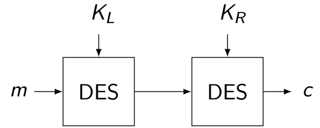
- Block cipher
  - 64-bit input and output, 168-bit key



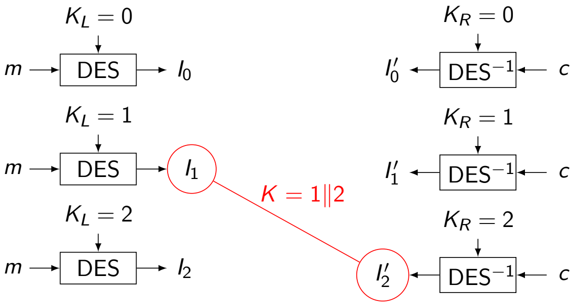
  - Why DES^{-1} instead of DES in the middle ?
- Slowly disappearing, replaced by AES
  (6 times faster in software).

Meet in the middle attack

- Why Triple DES instead of Double DES ?



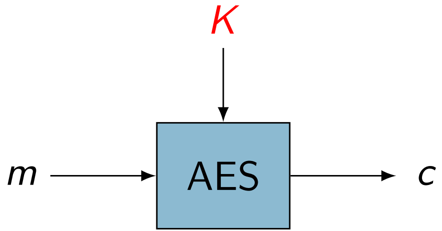
- Meet in the middle attack with known m and c



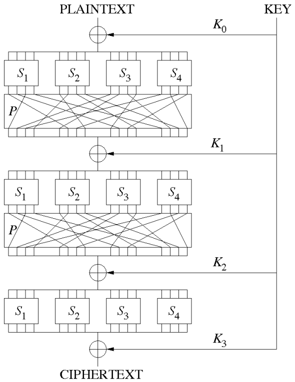
- Complexity: 2^{56}, so Double DES is not more secure than DES.

AES block cipher

- Most widely used block-cipher today
- NIST standard since 2001 (DES replacement)
- Input/output length: 128 bits.
- Key length: 128/192/256 bits.

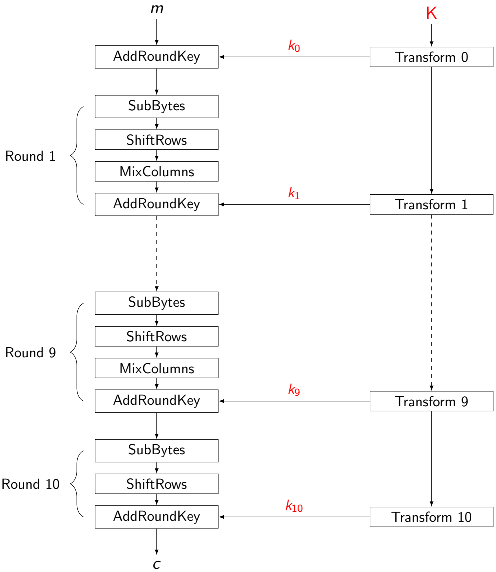


- Substitution-permutation network (SPN)
  - Several rounds of substition boxes (SBoxes) and linear layer.
  - SBox should be a permutation
  - Unlike DES, all input bits are modified in each round.



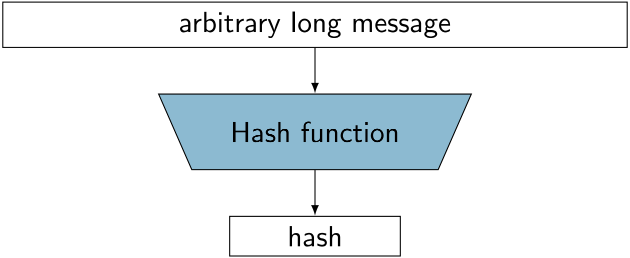
Example of SPN with 3 rounds

Overview of AES-128



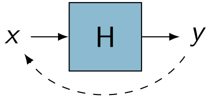
Hash function

- Hash function
  - Takes as input a message of arbitrary length and outputs a string of fixed length.
- Examples of hash functions:
  - SHA-1 (1995): 160 bits
  - SHA-2 (2001): 224, 256, 384 and 512 bits
  - SHA-3 (2015): 224, 256, 384 and 512 bits

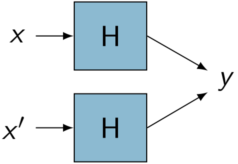


Properties of hash functions

- Preimage resistance
  - Given y, it is infeasible to find x such that $y = H(x)$



- Collision resistance
  - It is infeasible to find $x \neq x'$ such that $H(x) = H(x')$



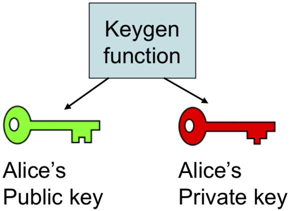
- Birthday paradox
  - For a n-bit hash function, it is possible to find a collision in $2^{n/2}$ operations.
  - Therefore to provide λ bits of security, must have output size at least 2λ bits.

Applications of hash functions

- Integrity of messages or files
  - Given $h = H(m)$, one can check that m has not been modified by recomputing $H(m)$ and checking that $h = H(m)$.
  - To protect the integrity of m, we don't have to store a copy of the long message m, we only have to store the short h.
- Commitment scheme
  - To commit on m, Alice sends $h = H(r\|m)$ to Bob, without revealing m.
  - She can later reveal m (and r) to Bob who checks $h = H(r\|m)$
- Proof of work (Bitcoin)
  - Find m such that $H(m)$ starts with k zero bits. This requires 2^k hash computations on average.
  - One can verify m by computing $H(m)$.

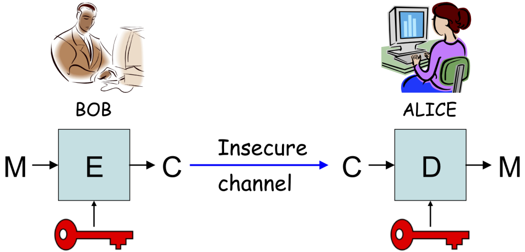
Public-key cryptography

- Invented by Diffie and Hellman in 1976. Revolutionized the field.
- Each user now has two keys
  - A public key
  - A private key
  - Should be hard to compute the private key from the public key.
- Enables:
  - Asymmetric encryption
  - Digital signatures
  - Key exchange, identification, and many other protocols.



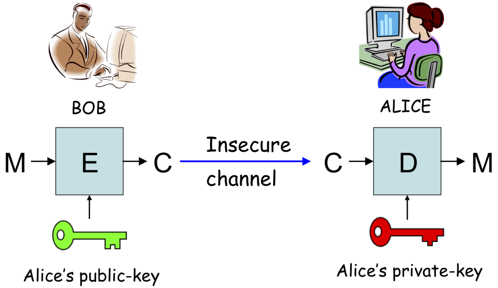
Key distribution issue

- Symmetric cryptography
  - Problem: how to initially distribute the key to establish a secure channel ?



Public-key encryption

- Public-key encryption (or asymmetric encryption)
  - Solves the key distribution issue

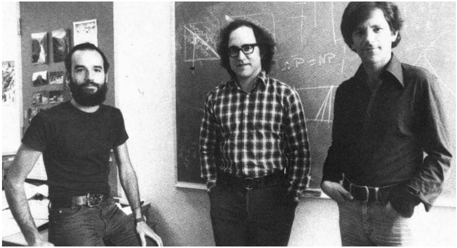


Analogy: the mailbox

- Bob wants to send a letter to Alice
  - Bob obtains Alice's adress
  - Bob puts his letter in Alice's mailbox
  - Alice opens her mailbox and read Bob's letter.
- Properties of the mailbox
  - Anybody can put a letter in the mailbox
  - Only Alice can open her mailbox

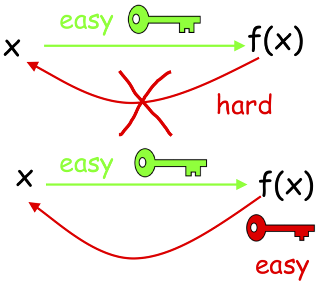
The RSA algorithm

- The RSA algorithm is the most widely-used public-key encryption algorithm
  - Invented in 1977 by Rivest, Shamir and Adleman.
  - Implements a trapdoor one-way permutation
  - Used for encryption and signature.
  - Widely used in electronic commerce protocols (SSL), secure email, and many other applications.



Trapdoor one-way permutation

- Trapdoor one-way permutation
  - Computing $f(x)$ from x is easy
  - Computing x from $f(x)$ is hard without the trapdoor



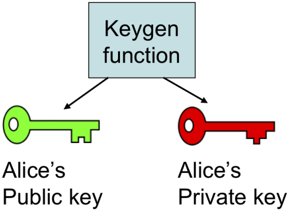
- Public-key encryption
  - Anybody can compute the encryption $c = f(m)$ of the message m.
  - One can recover m from the ciphertext c only with the trapdoor.

RSA

- Key generation:
  - Generate two large distinct primes p and q of same bit-size $k/2$, where k is a parameter.
  - Compute $n = p \cdot q$ and $\phi = (p-1)(q-1)$.
  - Select a random integer e such that $\gcd(e, \phi) = 1$
  - Compute the unique integer d such that

    $$e \cdot d \equiv 1 \pmod{\phi}$$

    using the extended Euclidean algorithm.
  - The public key is (n, e).
  - The private key is d.



RSA encryption

- Encryption with public-key (n, e)
  - Given a message $m \in [0, n-1]$ and the recipent's public-key (n, e), compute the ciphertext:

$$c = m^e \bmod n$$

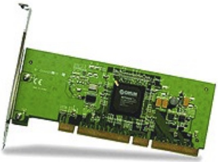
- Decryption with private-key d
  - Given a ciphertext c, to recover m, compute:

$$m = c^d \bmod n$$

- Message encoding
  - The message m is viewed as an integer between 0 and $n-1$
  - One can always interpret a bit-string of length less than $\lfloor \log_2 n \rfloor$ as such a number.

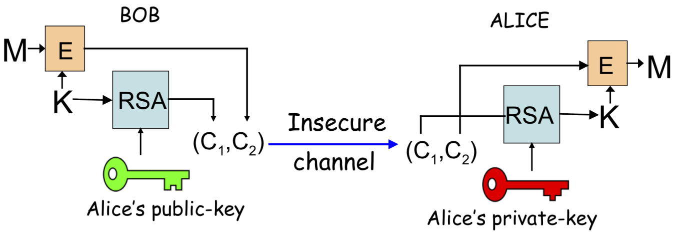
Implementation of RSA

- Required: computing with large integers
  - more than 1024 bits.
- In software
  - big integer library: GMP, NTL
- In hardware
  - Cryptoprocessor for smart-card
  - Hardware accelerator for PC.



Speed of RSA

- RSA much slower than AES and other secret key algorithms.
- To encrypt long messages
  - encrypt a symmetric key K with RSA
  - and encrypt the long message with K

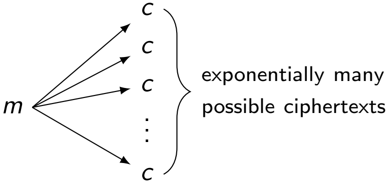


Security of RSA

- The security of RSA is based on the hardness of factoring.
  - Given $n = p \cdot q$, it should be difficult to recover p and q.
  - No efficient algorithm is known to do that. Best algorithms have sub-exponential complexity.
  - Factoring record (2020): a 829-bit RSA modulus n.
  - In practice, one uses at least 1024-bit RSA moduli.
- However, there are many other lines of attacks.
  - Attacks against textbook RSA encryption
  - Low private / public exponent attacks
  - Implementation attacks: timing attacks, power attacks and fault attacks

Elementary attacks

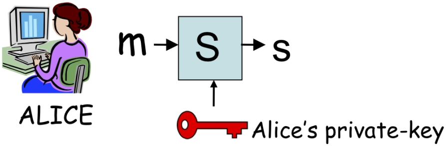
- Textbook RSA encryption: dictionary attack
  - If only two possible messages m_0 and m_1, then only $c_0 = (m_0)^e \bmod N$ and $c_1 = (m_1)^e \bmod N$.
  - $\Rightarrow$ encryption must be probabilistic.



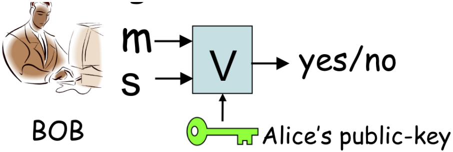
- Example: PKCS#1 v1.5 (1993)
  - $\mu(m) = 0002 \| r \| 00 \| m$
  - $c = \mu(m)^e \bmod N$
  - Still insufficient
    (Bleichenbacher's attack, 1998)

Digital signatures

- A digital signature σ is a bit string that depends on the message m and the user's public-key pk
  - Only Alice can sign a message m using her private-key sk



  - Anybody can verify Alice's signature of the message m given her public-key pk



Digital signature

- A digital signature provides:
  - Authenticity: only Alice can produce a signature of a message valid under her public-key.
  - Integrity: the signed message cannot be modified.
  - Non-repudiation: Alice cannot later claim that she did not sign the message

The RSA signature scheme

- Key generation :
  - Public modulus: $N = p \cdot q$ where p and q are large primes.
  - Public exponent : e
  - Private exponent: d, such that $d \cdot e = 1 \mod \phi(N)$
- To sign a message m, the signer computes :
  - $s = m^d \mod N$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \mod N$
  - Anybody can verify the signature

Hash-and-sign paradigm

- There are many attacks on basic RSA signatures:
  - Existential forgery: $r^e = m \pmod{N}$
  - Chosen-message attack: $(m_1 \cdot m_2)^d = m_1^d \cdot m_2^d \pmod{N}$
- To prevent from these attacks, one usually uses a hash function. The message is first hashed, then padded.

$$m \longrightarrow H(m) \longrightarrow 1001\ldots0101\|H(m)$$

$$\downarrow$$

$$\sigma = \left(1001\ldots0101\|H(m)\right)^d \bmod N$$

- Example: PKCS#1 v1.5 (1993)

$$\mu(m) = 0001\ \ \mathtt{FF....FF00}\|c_{\mathsf{SHA}}\|\mathsf{SHA}(m)$$

  - The signature is then $\sigma = \mu(m)^d \bmod N$

Other signature schemes

- Digital Signature Algorithm (DSA) (1991)
  - Digital Signature Standard (DSS) proposed by NIST, specified in FIPS 186.
  - Variant of Schnorr and ElGamal signature schemes
  - Security based on the hardness of discrete logarithm problem.
  - Public-key: $y = g^x \bmod p$
  - Signature: (r, s), where $r = (g^k \bmod p) \bmod q$ and $s = k^{-1}(H(m) + x \cdot r) \bmod p$, where $k \xleftarrow{\$} \mathbb{Z}_q$
- ECDSA: a variant of DSA for elliptic-curves
  - Shorter public-key than DSA (160 bits instead of 1024 bits)
  - Used in Bitcoin to ensure that funds can only be spent by their rightful owners.

Diffie-Hellman key-exchange protocol

- Public parameters: g and p

Bob

$$B = g^b \ [p]$$

Alice

$$A = g^a \ [p]$$

$$\text{Bob} \xrightarrow{B} \text{Alice}$$

$$\text{Bob} \xleftarrow{A} \text{Alice}$$

$$K_B = A^b = (g^a)^b = g^{ab} \ [p]$$

$$K_A = B^a = (g^b)^a = g^{ba} \ [p]$$

$$K_B = K_A$$

Security of Diffie-Hellman

- Based on the hardness of the discrete-log problem:
  - Given $A = g^a \pmod{p}$, find a
  - No efficient algorithm for large prime p.
- No authentication
  - Vulnerable to the man in the middle attack

Diffie-Hellman: meet in the middle attack

Bob | Alice

$B = g^{b}\ [p]$ $\xrightarrow{\quad B \quad}$ $A = g^{a}\ [p]$

$\xleftarrow{\quad A \quad}$

$K_B = A^{b} = g^{ab}\ [p]$ | $K_A = B^{a} = g^{ba}\ [p]$

$$K_B = K_A$$

Diffie-Hellman: meet in the middle attack

Eve

Bob

$B = g^b \ [p]$

$\xrightarrow{B}$

Alice

$A = g^a \ [p]$

$\xleftarrow{A}$

$K_B = A^b = g^{ab} \ [p]$

$K_A = B^a = g^{ba} \ [p]$

$K_B = K_A$

Diffie-Hellman: meet in the middle attack

Bob

Eve

Alice

$$B = g^b \ [p] \xrightarrow{B} \quad \xrightarrow{B' = g^{b'} \ [p]} \quad A = g^a \ [p]$$

$$\xleftarrow{A' = g^{a'} \ [p]} \quad \xleftarrow{A}$$

$$K'_B = (A')^b = g^{a'b} \ [p] \qquad K'_A = (B')^a = g^{b'a} \ [p]$$

$$K'_B = B^{a'} \ [p]$$
$$K'_A = A^{b'} \ [p]$$

Security of Diffie-Hellman

- Based on the hardness of the discrete-log problem:
  - Given $A = g^a \pmod{p}$, find a
  - No efficient algorithm for large prime p.
- No authentication
  - Vulnerable to the man in the middle attack
- Authenticated key exchange
  - Using a PKI. Alice and Bob can sign A and B
  - Password-authenticated key-exchange IEEE P1363.2

Lessons from the past

- Cryptography is a permanent race between construction and attacks
  - but somehow this has changed with modern cryptography and security proofs.
- Security should rely on the secrecy of the key and not of the algorithm
  - Open algorithms enables open scrutiny.