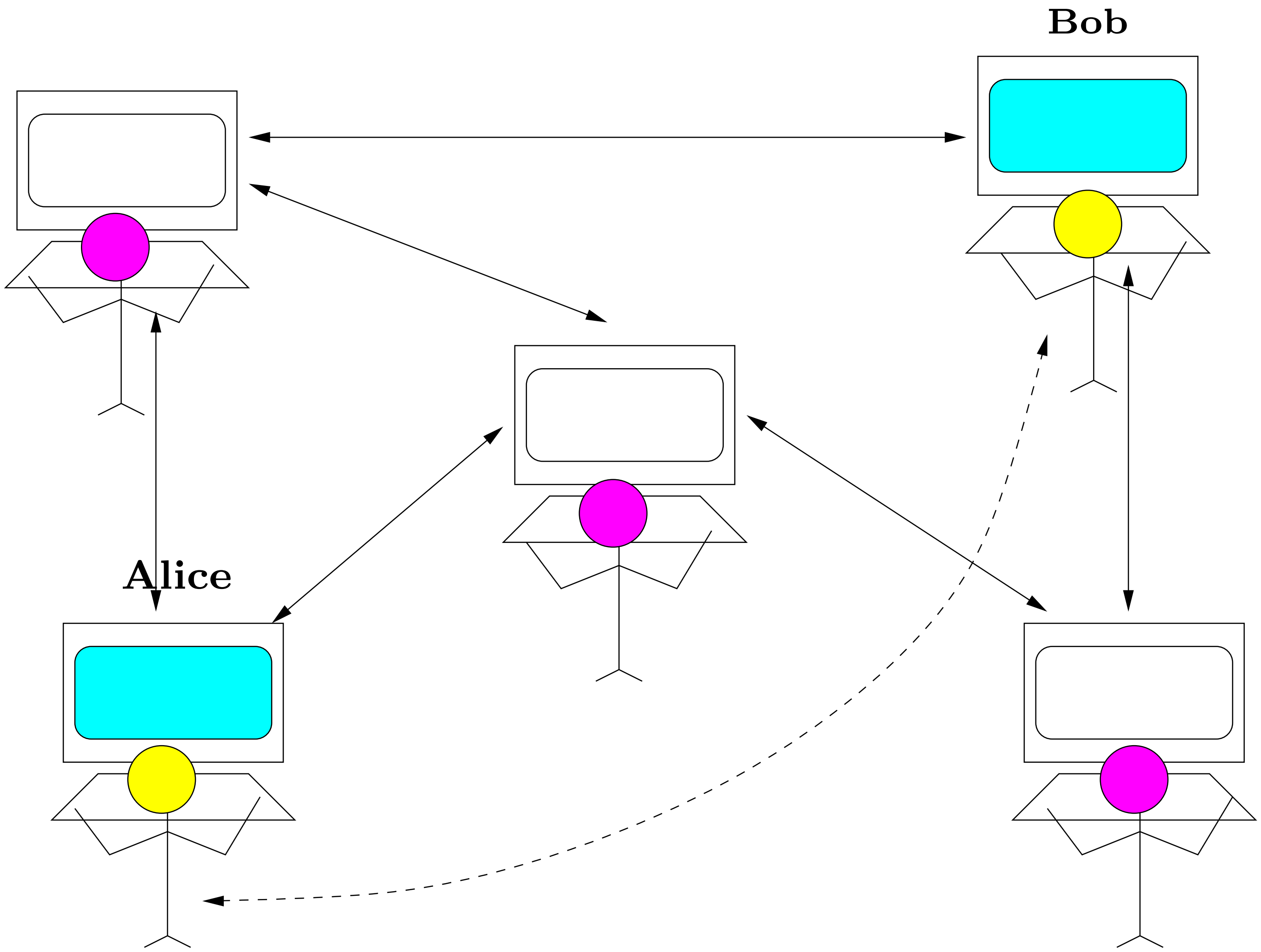
Cryptographic Protocols

Kumar Neeraj Verma

TU München

Winter Semester 2005

Communication over computer networks



Security problems

- Adversary can spy on messages,
- delete messages,
- modify messages,
- impersonate as Alice to Bob,
- deny having sent or received a message
- ...

Security problems

- Adversary can spy on messages,
- delete messages,
- modify messages,
- impersonate as Alice to Bob,
- deny having sent or received a message
- ...

How to secure communication over an insecure network ?

Cryptography and cryptographic protocols to the rescue ...

Encrypting and decrypting messages

. . . the naive way:

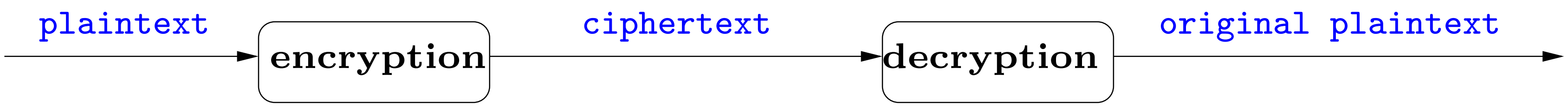
Instead of Alice $\longrightarrow$ Bob:

`This is Alice.  My credit card number is 1234567890123456`

We have Alice $\longrightarrow$ Bob:

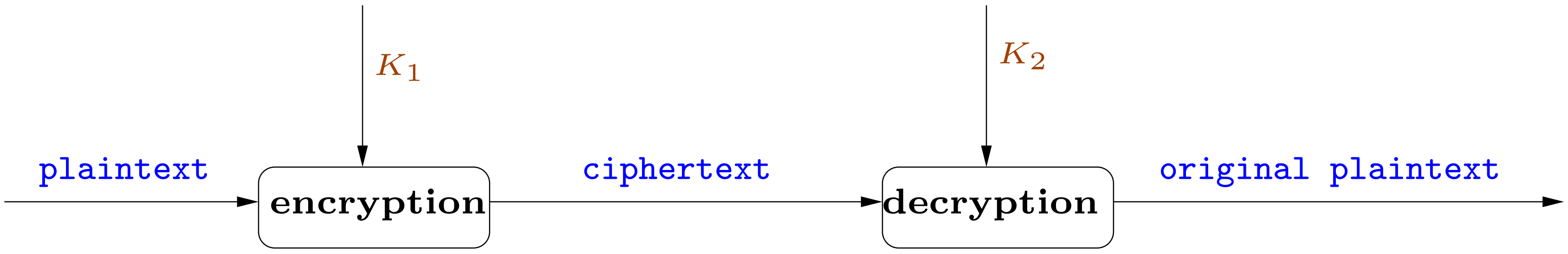
`6543210987654321 si rebmun drac tiderc yM .ecilA si sihT`

Alice and Bob agree on the method of encryption and decryption.



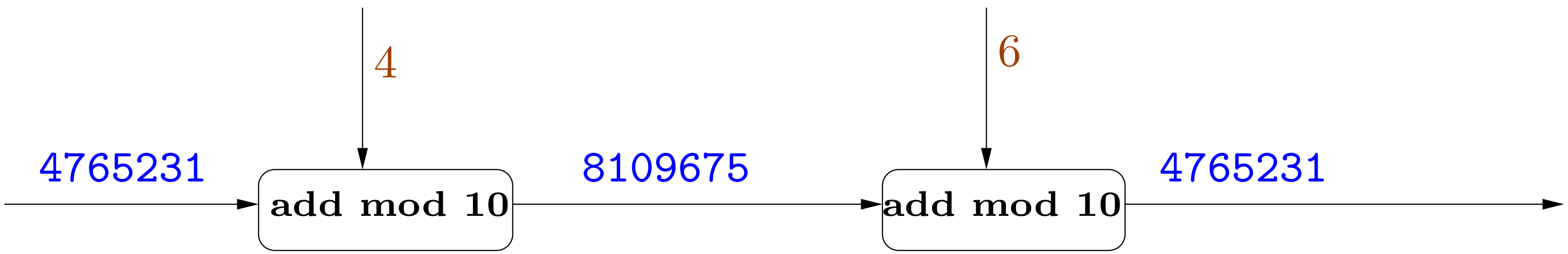
Cryptography with keys

Today we instead have the following picture:



The encryption and decryption algorithms are assumed to be publicly known.

The security lies in the (secret) keys.



Cryptography of the pre-computer age

Substitution ciphers: each character is mapped to the another character. The famous Caesar cipher: A $\rightarrow$ D, B $\rightarrow$ E, $\ldots$, Z $\rightarrow$ C.

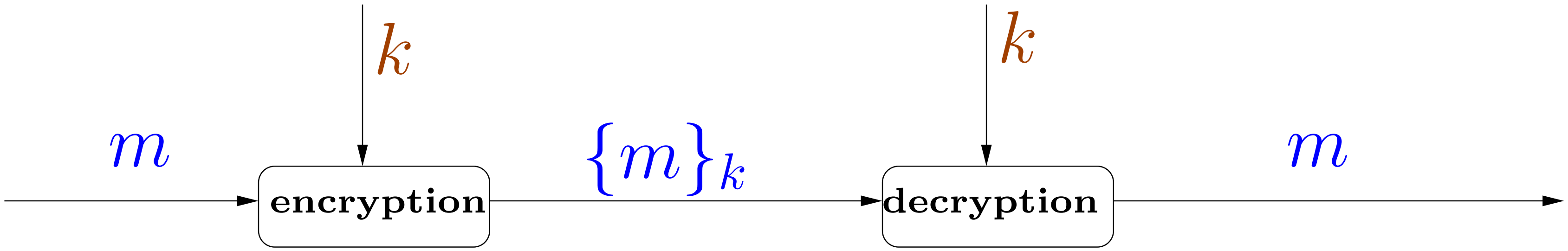
transposition cipher: shuffling around of characters.

Plaintext: `this is alice my credit card number is 1234567890123456`

```
thisisalic
emycreditc
ardnumberi
s123456789
0123456
```

Ciphertext: `teas0 hmr11 iyd22 scn33 iru44 sem55 adb66 lie7i tr8cc i9`

Private key cryptography



- The same key k is used for encryption and decryption
- Given message m and key k, we can compute the encrypted message $\{m\}_k$
- Given the encrypted message $\{m\}_k$ and the key k, we can compute the original message m

Private key cryptography

Suppose K_{ab} is a private key shared between A and B.

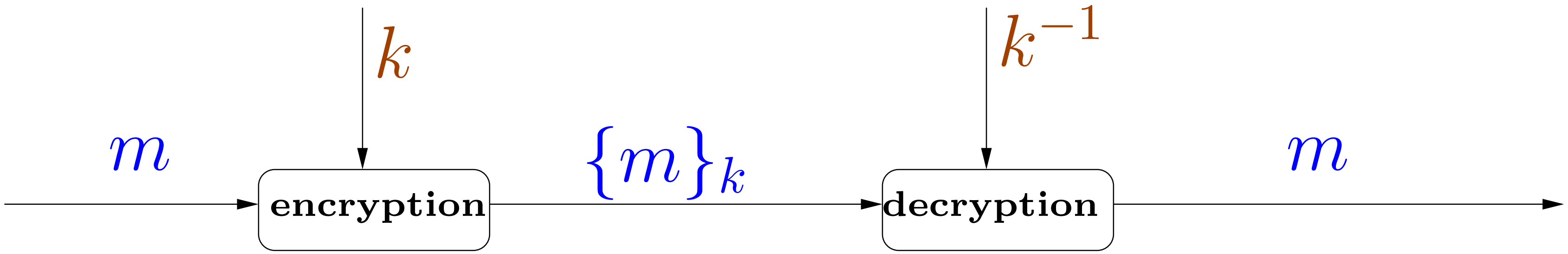
A can send a message m to B using private key cryptography:

$$A \longrightarrow B : \{m\}_{K_{ab}}$$

Only B can get back the message m.

A and B need to agree beforehand on a key K_{ab} which should not be disclosed to any one else

Public key cryptography



- A chooses pair (K_a, K_a^{-1}) of keys such that
  - messages encrypted with K_a can be decrypted with K_a^{-1}
  - K_a^{-1} cannot be calculated from K_a
- A makes K_a public: this is the public key of A
- A keeps K_a^{-1} secret: this is the private key of A

Public key cryptography

Then any B can send a message to A which only A can read:

$$B \longrightarrow A : \{m\}_{K_a}$$

Sometimes we have the additional property: messages encrypted with K_a^{-1} can be decrypted with K_a

Then A can send a message m to B

$$A \longrightarrow B : \{m\}_{K_a^{-1}}$$

and B is sure that the message m was encrypted by A. Hence we have authentication

One way hash functions

Properties of a one way hash function H:

– Given M, it is easy to compute $H(M)$ (called message digest).
– Given $H(M)$ is is difficult to find M' such that $H(M) = H(M')$.

A sends to B the message M together with the encrypted hash value $\{H(M)\}_{K_{ab}}$.

Efficient means of demonstrating authenticity, since $H(M)$ is of a fixed size.

Public key cryptography in practice

[user1@host1] `ssh user2@host2`

```
The authenticity of host 'host2 (xyz.xyz.xy.xy)' can't be
established.
RSA key fingerprint is
**:**:**:**:**:**:**:**:**:**:**:**:**:**:**:**.
Are you sure you want to continue connecting (yesno)? yes
Warning: Permanently added 'host2,xyz.xyz.xy.xy' (RSA)
to the list of known hosts.
Password:********

Welcome to host2
```

[user2@host2]

Cryptography is not enough!

Intruder is more clever. He can attack even if the cryptographic algorithms are perfect.

Alice tells Bank to transfer $\pounds 5000$ to Charlie's (intruder) account:

$$A \longrightarrow B : \{A, B, \text{ transfer 5000 euros } \ldots\}_{K_{ab}}$$

- B believes that message comes from A
- Charlie has no way to decrypt the message

Cryptography is not enough!

Intruder is more clever. He can attack even if the cryptographic algorithms are perfect.

Alice tells Bank to transfer $\pounds 5000$ to Charlie's (intruder) account:

$$A \longrightarrow B : \{A, B, \text{ transfer 5000 euros } \ldots\}_{K_{ab}}$$

- B believes that message comes from A
- Charlie has no way to decrypt the message
- But: Charlie can send the same message again to the bank

Intruder can replay known messages (freshness attack)

Solution: use session key

Generate fresh random value (nonce) for each new session and use it as a key for that session.

Solution: use session key

Generate fresh random value (nonce) for each new session and use it as a key for that session.

How to agree on a fresh key for each session?

Solution: use session key

Generate fresh random value (nonce) for each new session and use it as a key for that session.

How to agree on a fresh key for each session?

A sends to B the new key K_{ab} at the beginning of the session:

$$A \longrightarrow B : K_{ab}$$

And then uses it during that session.

Solution: use session key

Generate fresh random value (nonce) for each new session and use it as a key for that session.

How to agree on a fresh key for each session?

A sends to B the new key K_{ab} at the beginning of the session:

$$A \longrightarrow B : K_{ab}$$

And then uses it during that session.

Doesn't work. What about

$$A \longrightarrow B : \{K_{ab}\}_{K_{long}}$$

Using a long term key to agree on a session key.

A more complex solution

A and B both choose a nonce each.

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

A more complex solution

A and B both choose a nonce each.

$$
\begin{aligned}
&1. \quad A \longrightarrow B : \{A, N_a\}_{K_b} \\
&2. \quad B \longrightarrow A : \{N_a, N_b\}_{K_a} \\
&3. \quad A \longrightarrow B : \{N_b\}_{K_b}
\end{aligned}
$$

The second message is to assure A that B is active and N_b is fresh.

The third message is to assure B that A is active and N_a is fresh.

A more complex solution

A and B both choose a nonce each.

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, N_a\}_{K_b} \\
2. & B \longrightarrow A : \{N_a, N_b\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

The second message is to assure A that B is active and N_b is fresh.
The third message is to assure B that A is active and N_a is fresh.

Expected security property: N_a and N_b are known only to A and B.
Expected authentication property: A and B are assured that they are talking to each other.

$$A \longrightarrow B : \{A, B, N_a, N_b \text{ transfer 5000 euros } \ldots\}_{K_b}$$

A more complex solution

A and B both choose a nonce each.

$$1. \quad A \longrightarrow B : \{A, N_a\}_{K_b}$$

$$2. \quad B \longrightarrow A : \{N_a, N_b\}_{K_a}$$

$$3. \quad A \longrightarrow B : \{N_b\}_{K_b}$$

The second message is to assure A that B is active and N_b is fresh.
The third message is to assure B that A is active and N_a is fresh.

Expected security property: N_a and N_b are known only to A and B.
Expected authentication property: A and B are assured that they are talking to each other.

$$A \longrightarrow B : \{A, B, N_a, N_b \text{ transfer 5000 euros } \ldots\}_{K_b}$$

How secure is this ? How to guarantee security ?

Cryptography and cryptographic protocols

- Cryptography deals with algorithms for encryption, decryption, random number generation, etc. Cryptographic protocols use cryptography for exchanging messages.

- Attacks against cryptographic primitives involves breaking the algorithm for encryption, etc. Attacks against cryptographic protocols may be of completely logical nature.

- Cryptographic protocols may be insecure even if the underlying cryptographic primitives are completely secure.

- Hence we often separate the study of cryptographic protocols from that of cryptographic primitives.

Difficulty in ensuring correctness of cryptographic protocols

- Infinitely many sessions
- Infinitely many participants
- Infinitely many nonces
- Sessions are interleaved
- Adversary can replace messages by any arbitrary message: infinitely branching system

Back to our example

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

Back to our example

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

This is the well-known Needham-Schroeder public-key protocol.

Published in 1978. Attack found after 17 years in 1995 by Lowe.

Man in the middle attack

$$A \xrightarrow{\{A, N_a\}_{K_c}} C\ (A) \xrightarrow{\{A, N_a\}_{K_b}} B$$

$$A \xleftarrow{\{N_a, N_b\}_{K_a}} C\ (A) \xleftarrow{\{N_a, N_b\}_{K_a}} B$$

$$A \xrightarrow{\{N_b\}_{K_c}} C\ (A) \xrightarrow{\{N_b\}_{K_b}} B$$

Man in the middle attack

$$\text{A} \xrightarrow{\{A, N_a\}_{K_c}} \text{C (A)} \xrightarrow{\{A, N_a\}_{K_b}} \text{B}$$

$$\text{A} \xleftarrow{\{N_a, N_b\}_{K_a}} \text{C (A)} \xleftarrow{\{N_a, N_b\}_{K_a}} \text{B}$$

$$\text{A} \xrightarrow{\{N_b\}_{K_c}} \text{C (A)} \xrightarrow{\{N_b\}_{K_b}} \text{B}$$

Even very simple protocols may have subtle flaws

Consequences

Suppose B is the server of a bank.

C, who can now pretend to be A:

$$C \longrightarrow B : \{N_a, N_b, \text{ transfer £5000 from account of } A \text{ to account of } C\}_{K_b}$$

A fix: the Needham-Schroeder-Lowe protocol [Lowe,1985]

B includes his identity in the message he sends:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, Na\}_{K_b} \\
2. & B \longrightarrow A : \{B, N_a, N_b\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

A fix: the Needham-Schroeder-Lowe protocol [Lowe,1985]

B includes his identity in the message he sends:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, Na\}_{K_b} \\
2. & B \longrightarrow A : \{B, N_a, N_b\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

Is it secure?

A variant of the Needham-Schroeder-Lowe protocol

Suppose now we change the place of B in the second message:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, Na\}_{K_b} \\
2. & B \longrightarrow A : \{N_a, N_b, B\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

A variant of the Needham-Schroeder-Lowe protocol

Suppose now we change the place of B in the second message:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, Na\}_{K_b} \\
2. & B \longrightarrow A : \{N_a, N_b, B\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

Does this affect security?

Type flaw

An attack on the variant of the Needham-Schroeder-Lowe protocol [Millen]:

$$\mathsf{C} \xrightarrow{\{A, C\}_{K_b}} \mathsf{B}$$

$$\mathsf{B} \xrightarrow{\{C, \underbrace{N_b, B}_{N_c}\}_{K_a}} \mathsf{A}$$

$$\mathsf{C} \xleftarrow{\{N_b, B, N_a, A\}_{K_c}} \mathsf{A}$$

Security properties

Secrecy:

- some data M is unknown to the intruder (reachability property).
- global secrecy: a message is secret all the time.
- local secrecy: a message is secret till the corresponding session has not ended.

Security properties

Authentification:

- If A accepts a message M as coming from B then B actually sent M.
- If A received a message of form M_1 then B sent a message of form M_2.
- If A got a message of form M then B was active

Security properties

Anonymity: an external person should not be able to know about the sender of a message. E.g. for electronic voting, mobile telephony.

Non-repudiation: an agent should not be able to deny that he sent some message. E.g. for electronic contract signing

Fairness: E.g. in electronic contract signing no party should have an advantage over the other. The following electronic contract signing protocol is not fair for A:

$$A \rightarrow B : Sig_A(m)$$

$$B \rightarrow A : Sig_B(m)$$

An overview of cryptography

RSA

Proposed by Ron Rivest, Adi Shamir and Leonard Adleman in 1978.

One of the most well-known public-key algorithms.

Its security is believed to derive from the difficulty of the integer factorization problem: decomposing an integer into its prime factors.

Based on modular arithmetic:

$$
\begin{aligned}
0 = 15 = 30 = 45 &\pmod{15} \\
4 = 19 = 34 = 49 &\pmod{15} \\
10 + 70 = 80 = 5 &\pmod{15} \\
6 \times 8 = 48 = 3 &\pmod{15} \\
11^2 = 121 = 1 &\pmod{15}
\end{aligned}
$$

Numbers x and y are relatively prime if $gcd(x, y) = 1$.

Euler phi function $\phi(n)$ is the number of positive integers smaller than n and relatively prime to n.

$$\text{If } gcd(a, n) = 1 \text{ then } a^{\phi(n)} = 1 \pmod{n}$$

Now suppose p and q are two distinct prime numbers and $n = pq$.

The set of positive integers smaller than n and relatively prime to n are $\{1, \ldots, pq - 1\} \setminus \{p, 2p, \ldots, (q - 1)p, q, 2q, \ldots, (p - 1)q\}$.

Hence $\phi(n) = pq - 1 - p - q + 2 = (p - 1)(q - 1)$.

Randomly choose two large distinct prime numbers p and q. $n = pq$.

Choose e such that e and $\phi(n)$ are relatively prime.

Compute d such that $ed = 1 \pmod{\phi(n)}$

I.e. $d = e^{-1} \pmod{\phi(n)}$ (use Euclid's algorithm)

Public key $= (n, e)$, encryption: $C = M^e \pmod{n}$

Private key $= d$, decryption: $M = C^d \pmod{n}$

We have $M^{ed} = M^{k\phi(n)+1} = (M^{\phi(n)})^k M = M \pmod{n}$.

The whole message is first divided into smaller portions $< n$.

Also works if M is not relatively prime to n.

$M = ap$ where $0 < a < q$.

$M^{ed} = (M^{\phi(n)})^k M = (a^{q-1}p^{q-1})^{p-1}M = 1^{p-1}M = M \pmod{q}$

$M^{ed} = 0 = M \pmod{p}$

Hence $M^{ed} = M \pmod{n}$

We use the fact that if $a = b \pmod{p}$ and $a = b \pmod{q}$ where p, q are primes then $a = b \pmod{pq}$.

Block algorithms

Given encryption and decryption algorithms that work on blocks of fixed sizes (e.g. 64 bits), how to deal with messages of arbitrary sizes.

Electronic Codebook Mode (ECB): encrypt each block independently.

$\{P_1 \ldots P_n\}_k = \{P_1\}_k \ldots \{P_n\}_k$

This is similar to looking up in a dictionary with 2^{64} entries.

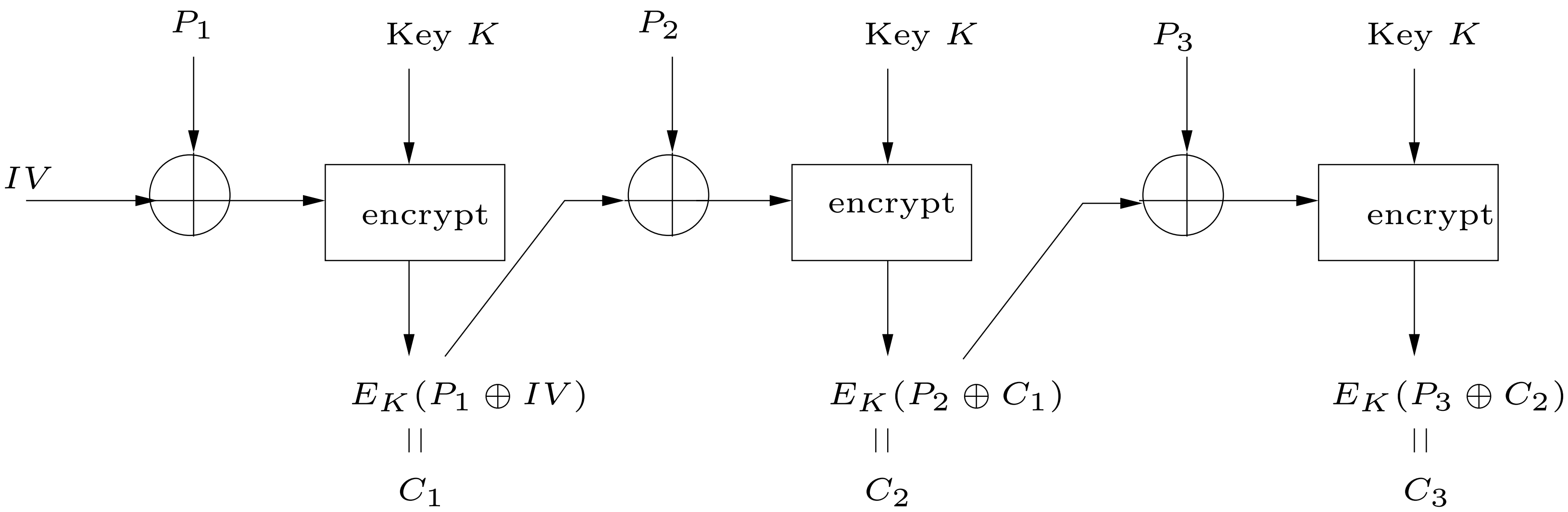
Subject to block replay attacks.

Example of block replay attack.

Interbank money transfers:

Date/Timestamp	1 block
Sending bank name	1 block
Receiving bank name	1 block
Depositor's Name	6 blocks
Depositor's Account	2 blocks
Amount of deposit	1 block

Cipher Block Chaining Mode (CBC)



Encryption	Decryption
$C_1 = E_K(IV \oplus P_1)$	$P_1 = D_K(C_1) \oplus IV$
$C_2 = E_K(C_1 \oplus P_2)$	$P_2 = D_K(C_2) \oplus C_1$
$C_3 = E_K(C_2 \oplus P_3)$	$P_3 = D_K(C_3) \oplus C_2$

Choose a random initialization vector (IV) for each message.