Information security. The protection of information and information systems against unauthorized access or modification of information, whether in storage, processing, or transit, and against denial of service to authorized users.

Complete mediation

For every requested action, check authenticity, integrity, and authorization.
Current

... ...

Bad

... ...

Principles of Computer System Design © Saltzer & Kaashoek 2009

BY-NC-SA
Open design principle

Let anyone comment on the design. You need all the help you can get.
Minimize secrets

Because they probably won’t remain secret for long.
Economy of mechanism

*The less there is, the more likely you will get it right.*
Minimize common mechanism

*Shared mechanisms provide unwanted communication paths.*
Fail-safe defaults

Most users won’t change them, so make sure that defaults do something safe.
Least privilege principle

Don’t store lunch in the safe with the jewels.
Computer system

Principal

Request

Perform action

Authentication module

Authorization module

Authentic?

Authorized?

Guard

OK

Perform action

Object

Audit trail

Log

Yes/no

Yes/no

OK

Principles of Computer System Design © Saltzer & Kaashoek 2009
Request
To: service
From: Alice
Buy Generic Moneymaking, Inc.

Service

guard

Alice’s trading account
Alice 
secure area

M
SIGN

Tag

M

VERIFY

ACCEPT or REJECT

Bob 
secure area
Alice uses her private key $K_1$ to sign the message $M$, creating a tag $Tag$. This tag is then transmitted to Bob, who uses his private key $K_2$ to verify the signature. Bob then decides whether to accept or reject the message. The secure areas ensure that the keys are not exposed.
ENCRYPT

K1

M

secure area

ENCRYPT (M, K1)

insecure area

DECRYPT

K2

DECRYPT (ENCRYPT (M, K1), K2)

secure area
Comparison of access control systems

<table>
<thead>
<tr>
<th>System</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ticket</td>
<td>Quick access check</td>
<td>Revocation is difficult</td>
</tr>
<tr>
<td></td>
<td>Tickets can be passed around</td>
<td>Tickets can be passed around</td>
</tr>
<tr>
<td>List</td>
<td>Revocation is easy</td>
<td>Access check requires searching a list</td>
</tr>
<tr>
<td></td>
<td>Audit possible</td>
<td></td>
</tr>
<tr>
<td>Agency</td>
<td>List available</td>
<td>Revocation might be hard</td>
</tr>
</tbody>
</table>
Alice

From: Alice

Send me the quiz
Rule 1: Delegating authority:

If A says (B speaks for A) then B speaks for A

Rule 2: Use of delegated authority.

If A speaks for B and A says (B says X) then B says X

Rule 3: Chaining of delegation.

If A speaks for B and B speaks for C then A speaks for C
procedure RC4_GENERATE ()
   \(i \leftarrow (i + 1) \mod 256\)
   \(j \leftarrow (j + S[i]) \mod 256\)
   \(\text{SWAP}(S[i], S[j])\)
   \(t \leftarrow (S[i] + S[j]) \mod 256\)
   \(k \leftarrow S[t]\)
   return \(k\)

procedure RC4_INIT (seed)
   for \(i\) from 0 to 255 do
      \(S[i] \leftarrow i\)
      \(K[i] \leftarrow \text{seed}[i]\)
      \(j \leftarrow 0\)
   for \(i\) from 0 to 255 do
      \(j \leftarrow (j + S[i] + K[i]) \mod 256\)
      \(\text{SWAP}(S[i], S[j])\)
      \(i \leftarrow j \leftarrow 0\)
<table>
<thead>
<tr>
<th>input</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_0$</td>
<td>$s_{0,0}$</td>
<td>$o_0$</td>
</tr>
<tr>
<td>$i_1$</td>
<td>$s_{1,0}$</td>
<td>$o_1$</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$s_{2,0}$</td>
<td>$o_2$</td>
</tr>
<tr>
<td>$i_3$</td>
<td>$s_{3,0}$</td>
<td>$o_3$</td>
</tr>
<tr>
<td>$i_4$</td>
<td>$s_{0,1}$</td>
<td>$o_4$</td>
</tr>
<tr>
<td>$i_5$</td>
<td>$s_{1,1}$</td>
<td>$o_5$</td>
</tr>
<tr>
<td>$i_6$</td>
<td>$s_{2,1}$</td>
<td>$o_6$</td>
</tr>
<tr>
<td>$i_7$</td>
<td>$s_{3,1}$</td>
<td>$o_7$</td>
</tr>
<tr>
<td>$i_8$</td>
<td>$s_{0,2}$</td>
<td>$o_8$</td>
</tr>
<tr>
<td>$i_9$</td>
<td>$s_{1,2}$</td>
<td>$o_9$</td>
</tr>
<tr>
<td>$i_{10}$</td>
<td>$s_{2,2}$</td>
<td>$o_{10}$</td>
</tr>
<tr>
<td>$i_{11}$</td>
<td>$s_{3,2}$</td>
<td>$o_{11}$</td>
</tr>
<tr>
<td>$i_{12}$</td>
<td>$s_{0,3}$</td>
<td>$o_{12}$</td>
</tr>
<tr>
<td>$i_{13}$</td>
<td>$s_{1,3}$</td>
<td>$o_{13}$</td>
</tr>
<tr>
<td>$i_{14}$</td>
<td>$s_{2,3}$</td>
<td>$o_{14}$</td>
</tr>
<tr>
<td>$i_{15}$</td>
<td>$s_{3,3}$</td>
<td>$o_{15}$</td>
</tr>
</tbody>
</table>
procedure AES (in, out, key)

state ← in // copy in into state
ADDROUNDKEY (state, key) // mix key into state
for r from 1 to 9 do
    SUBBYTES (state) // substitute some bytes in state
    SHIFTROWS (state) // shift rows of state cyclically
    MIXCOLUMNS (state) // mix the columns up
    ADDROUNDKEY (state, key[r×4, (r+1)×4 – 1]) // expand key, mix in
SUBBYTES (state)
SHIFTROWS (state)
ADDROUNDKEY (state, key[10×4, 11×4 – 1])
out ← state // copy state into out
(a) Encipher

(b) Decipher
1. \{ClientHello, client_version, randomclient, session_id, cipher_suites, compression_f\}

2. \{ServerHello, server_version, randomserver, session_id, cipher_suite, compression_f\}

3. \{ServerCertificate, certificate_list\}

4. \{ServerHelloDone\}

5. \{ClientKeyExchange, ENCRYPT (pre_master_secret, ServerPubKey)\}

6. \{ChangeCipherSpec, cipher_suite\}

7. \{Finished, MAC (master_secret, messages 1, 2, 3, 4, 5)\}

8. \{ChangeCipherSpec, cipher_suite\}

9. \{Finished, mac (master_secret, messages 1, 2, 3, 4, 5, 7)\}

10. \{Data, plaintext\}
structure X_509_v3_certificate
  version
  serial_number
  signature_cipher_identifier
  issuer_signature
  issuer_name
  subject_name
  subject_public_key_cipher_identifier
  subject_public_key
  validity_period
procedure DELETE_FILE (file_name)
    auth ← CHECK_DELETE_PERMISSION (file_name, this_user_id)
    if auth = PERMITTED
        then DESTROY (file_name)
    else signal ("You do not have permission to delete file_name")
Enigma Rotor with eight contacts

Side view, showing contacts.  

Edge view, showing some connections.