Hybrid Cryptography

with examples in Ruby and Go

Romek Szczesniak
security consultant
Hardcore Happy Cat Ltd

Eleanor McHugh
system architect
Games With Brains

January 2015

romek

- an applied cryptographer since 1995
- secures systems from Biometrics to Firewalls
- specialises in PKI, Smartcards, Biometrics

ellie

- commercial developer since 1995
- mission-critical & performance sensitive systems
- specialises in Ruby and Go

design credits

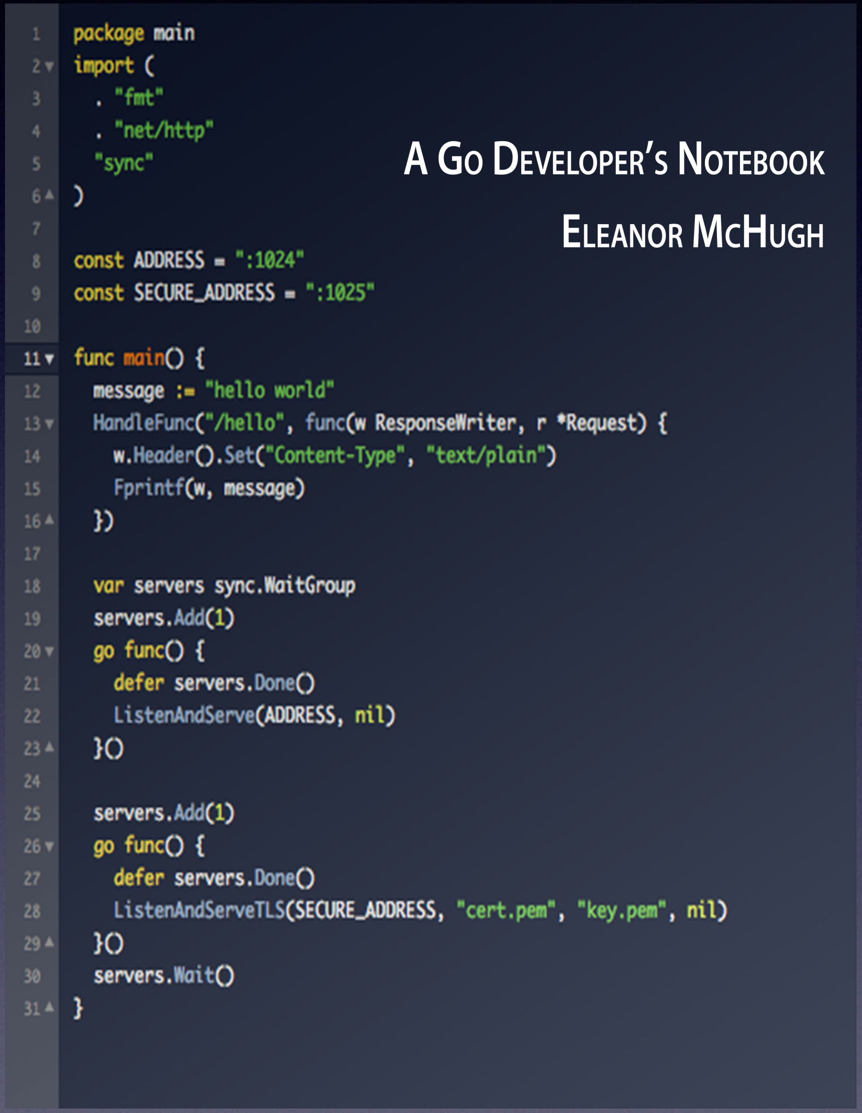
JUST WHEN YOU THOUGHT YOU HAD SEEN IT ALL ON THE TUBE
YOU CAN NOW TOUCH IN TO PAY WITH YOUR PHONE.
EE
Travel the whole of London with just your phone. Search Cash on Tap.

Sunday, February 9, 2014, 00:26 by Keith Micallef

New ID cards to be rolled out this week

IDMLT11025507<59999998M<<<<<<<
7001017F2305248MLT130524<<<<<<9
<<JOANNA<<<<<<<<<<<<<<<<<<<<

The new cards may have a lifespan of 10 years. Photo: Matthew Mirabelli


A Go Developer's Notebook

Eleanor McHugh

```go
package main
import (
  . "fmt"
  . "net/http"
  "sync"
)

const ADDRESS = ":1024"
const SECURE_ADDRESS = ":1025"

func main() {
  message := "hello world"
  HandleFunc("/hello", func(w ResponseWriter, r *Request) {
    w.Header().Set("Content-Type", "text/plain")
    Fprintf(w, message)
  })

  var servers sync.WaitGroup
  servers.Add(1)
  go func() {
    defer servers.Done()
    ListenAndServe(ADDRESS, nil)
  }()

  servers.Add(1)
  go func() {
    defer servers.Done()
    ListenAndServeTLS(SECURE_ADDRESS, "cert.pem", "key.pem", nil)
  }()
  servers.Wait()
}
```

HSBC Global Connections

hybrid cryptography?

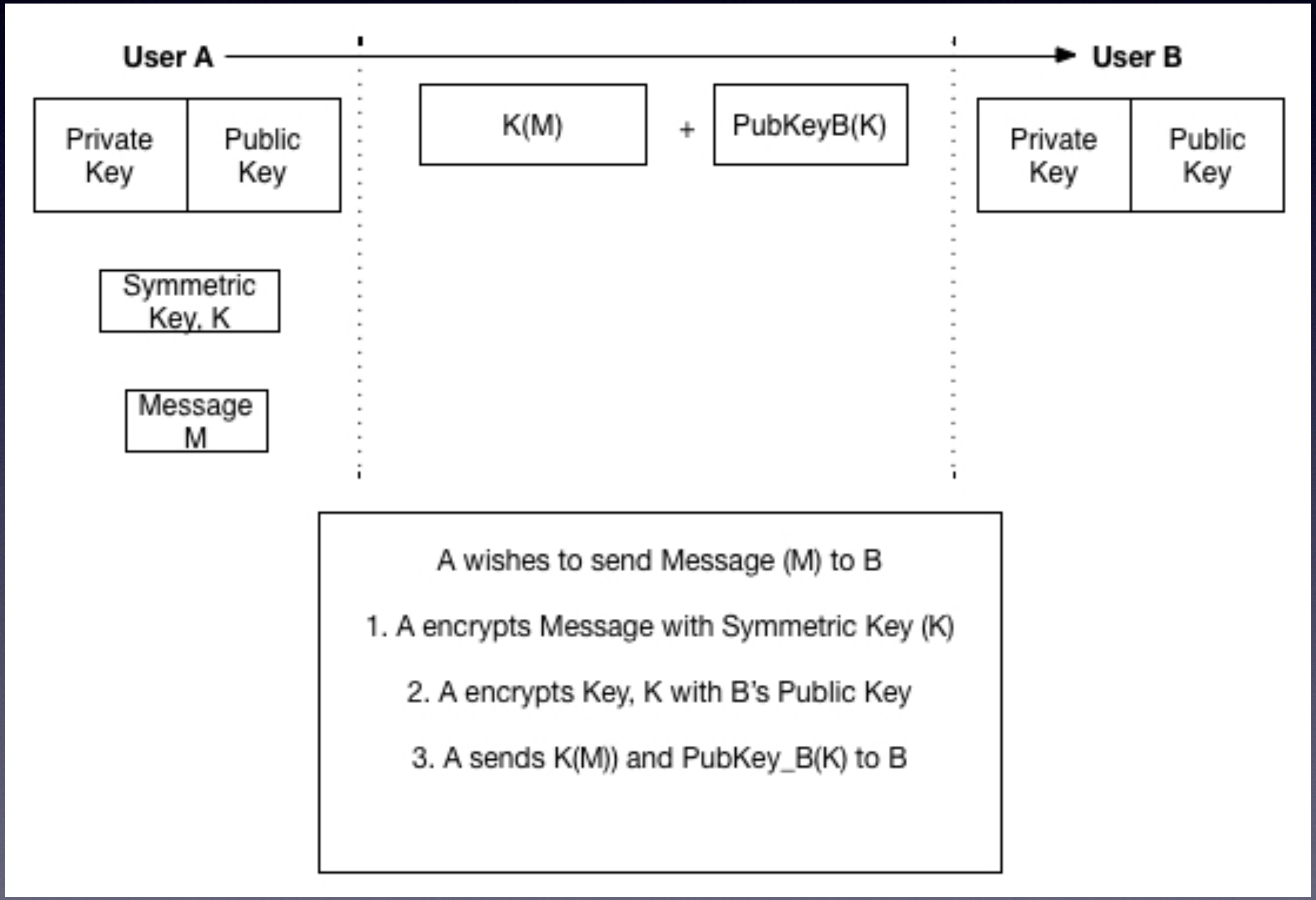
- a mode of encryption that merges two or more encryption systems
- incorporates a combination of asymmetric and symmetric encryption to benefit from the strengths of each form of encryption
- these strengths are respectively defined as speed and security

hybrid encryption is considered a highly secure type of encryption hybrid encryption is considered a highly secure type of encryption ***as long as*** the public and private keys are fully secure

history

- rarely mentioned in the literature
- Cramer & Shoup (2004)
- Dent (2005, 2009)
- Telnic DNS (2006)
- commonly discussed post-Snowden (2012)
- used in PGP and PKCS#7

encryption



A wishes to send Message (M) to B

1. A encrypts Message with Symmetric Key (K)

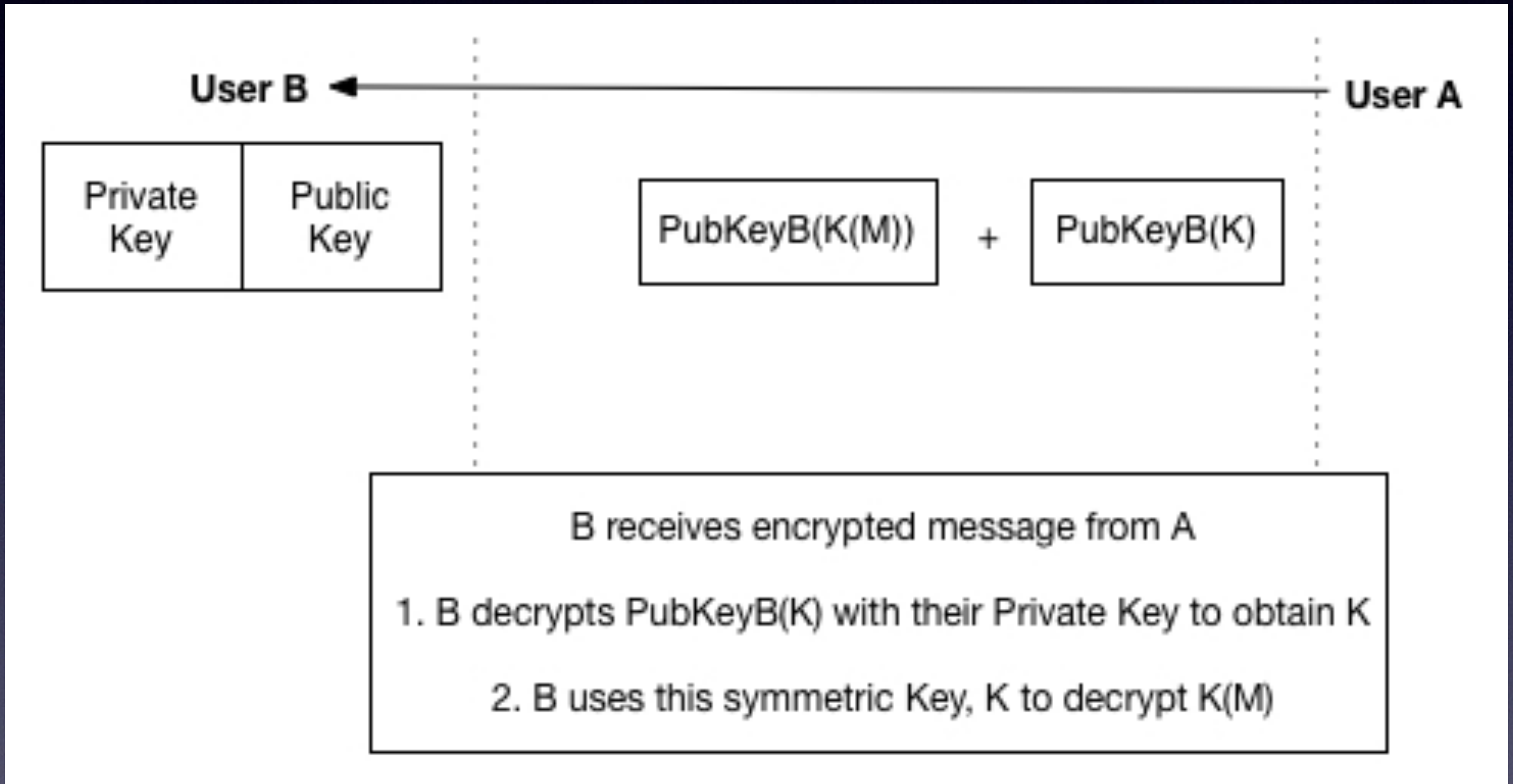
2. A encrypts Key, K with B's Public Key

3. A sends K(M)) and PubKey_B(K) to B

encryption

- User A encrypts the Message with the symmetric key
- User A encrypts the symmetric key with the receiver’s public key
- User A sends the encrypted message and the encrypted key to User B

decryption



B receives encrypted message from A

1. B decrypts PubKeyB(K) with their Private Key to obtain K

2. B uses this symmetric Key, K to decrypt K(M)

decryption

- User B knows how the Message is encrypted
- User B decrypts the symmetric key with his private key
- User B decrypts the Message using the symmetric key

an example workflow

1. create public key pair for user B (RSA-4096)
2. create symmetric key K (AES-256-CBC)
3. encrypt $K(M_B)$ and $Pub_B(K)$ for message M_B
4. send $Pub_B(K)$ and $K(M_B)$ to user B
5. decrypt K with $Priv_B$
6. decrypt M_B with K
7. send $K(M_A)$ to user A
8. change keys and repeat as required
9. all keys are stored in Base 64 encoding

key features

- a point-to-point cryptosystem
- fast, easy-to-use, user-specific system
- independent of underlying cryptosystems
- may change algorithms at any point
- may change keys at any point

weasel words

- danger! experimental code presented here!
- all such code is provided for entertainment purposes only and should be used with extreme caution, under adult supervision, *et al.*
- any resemblance to actual code and concepts, living or dead, is purely coincidental

a simple example

- hybrid encryption with text strings
- ruby 1.8 and later
- uses OpenSSL as its crypto library

```ruby
#!/usr/bin/env ruby -w
require 'rubygems'
require 'openssl'
require 'base64'

class Hybrid
    def initialize
        @privkey=0
        @pubkey=0
        @sessionkey=0
        @iv=0
        @f=0
        @g=0
    end
end

h = Hybrid.new
```

```
class Hybrid
    def keygen
        @privkey=OpenSSL::PKey::RSA.new(4096,65537)
        @pubkey=@privkey.public_key
        puts "4096-bit Key generated"
        @sessionkey=OpenSSL::Random.random_bytes(256/8){ putc "." }
    end
end
```

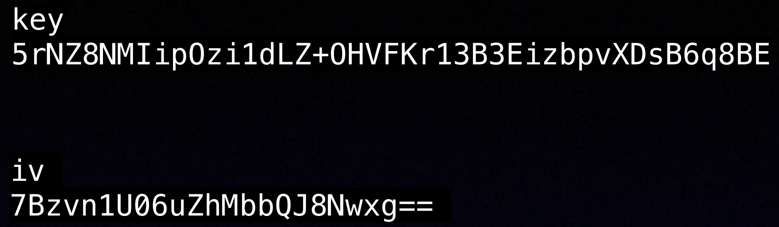
```
-----BEGIN RSA PUBLIC KEY-----
MIICCgKCAgEA5DL16QdI+0uaBpprF9nxmKO5mkgnWvcmoCMRBxFaEpwjSOCiiYjq
DdwXjChywMQQgx34nzqerXXKWjSIpLyy6sZV0akudiQ00JxnIv0y+STKZStzeNqF
FlTTfSksVRIMGJ6JkRvtZQ3I+uYkuqyfSDpr4/rEivYk2oz9Ru3Zj6WMEUeqsYJA
sz7mc5iFR+1Sr7RvRSAYXqxe6wM0PicSZ0vRGkSCbCvHXKNi4HteTGTXFXVr+s4l
3XfyF8i46e7tEq/9skJf9oaGxBhU26ALVQEH/xFc/TzFwCG5NDdVvdOcb8euE/sN
DG6SvCNJ5+ClSevJ74n4eSo8ScQU9t6lnITQXlTaDYCibbjjknPBCE9e/puoD3KF
YlvERwPTXtarLE/huZrx1llEubNaJjxrMoeJSIrs57DP7U6v4uQoTDbQM6yauwJC
pj7eOdd/S+HHpDLdad+mDEKJGwqFbafalb2WrkxYgkDq4Loeipmge/zIxZxBQAsB
dkCY+rSn6lskPcagfTfoAmx+0A+0A3cJP92oKzs0X2/flhuQAlrh5WmS6SSMVndt
988ayJ9z3QghxkNB59OgNleQjkKGxsoPTF/8Yvg0UBC4tVeTVpvROmFKX81tbPos
yxfnJ9xqUPaX0azMqZrOWPUMty2spyhZ4IMru/xviRoZ2NMjOY5O9dECAwEAAQ==
-----END RSA PUBLIC KEY——
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIJKAIBAAKCAgEA5DL16QdI+0uaBpprF9nxmKO5mkgnWvcmoCMRBxFaEpwjSOCi
iYjqDdwXjChywMQQgx34nzqerXXKWjSIpLyy6sZV0akudiQ00JxnIv0y+STKZStz
eNqFFlTTfSksVRIMGJ6JkRvtZQ3I+uYkuqyfSDpr4/rEivYk2oz9Ru3Zj6WMEUeq
sYJAsz7mc5iFR+1Sr7RvRSAYXqxe6wM0PicSZ0vRGkSCbCvHXKNi4HteTGTXFXVr
+s4l3XfyF8i46e7tEq/9skJf9oaGxBhU26ALVQEH/xFc/TzFwCG5NDdVvdOcb8eu
E/sNDG6SvCNJ5+ClSevJ74n4eSo8ScQU9t6lnITQXlTaDYCibbjjknPBCE9e/puo
D3KFYlvERwPTXtarLE/huZrx1llEubNaJjxrMoeJSIrs57DP7U6v4uQoTDbQM6ya
uwJCpj7eOdd/S+HHpDLdad+mDEKJGwqFbafalb2WrkxYgkDq4Loeipmge/zIxZxB
QAsBdkCY+rSn6lskPcagfTfoAmx+0A+0A3cJP92oKzs0X2/flhuQAlrh5WmS6SSM
Vndt988ayJ9z3QghxkNB59OgNleQjkKGxsoPTF/8Yvg0UBC4tVeTVpvROmFKX81t
bPosyxfnJ9xqUPaX0azMqZrOWPUMty2spyhZ4IMru/xviRoZ2NMjOY5O9dECAwEA
AQKCAgEA0kmn1RLyjSiRCq64K6Wafme5/NOq+Keyv3UxFstFrsqVtW3UOluiHB2K
0YzgmoTTFpDC8LDLUtuuGkw48140nichJHD8MMCSrv7CCDs+AtuFa4+L/H2akQag
UcFkagyUewd1i/QpYqs+Xv9AL4otyhiUHeWTwt6q/X9ZU0iR6U7L8YySXrvCNaus
IDAX+j1XqjTjKNc3vd6oJXexZ+kHi4sRaVxit53sPJEP5/+n2Uw/7DVlyRy5Rgpn
XMWKqYCUlVj6t4908S/s9r3ZTP6CEtY9cS6l+3NKZBBvpA+uAp0Dlvpyj1UVJDSt
IZR/YZ/hkWooj4YcJEPohK2eCBUKlKEwbhEhV9HDwjLHxsUoT1N9AUO4nd78RZdY
/YI0Y5QAaGCieWjgqrajW6jFTn4IzCGU487rMS1XoUubdzJjysyrr43Si9bL6mJz
xQi8j23Xv4aTHjHuuNKyCMOu5iuNLPjC8sSaLoAN8NZETvOPpL7KydN721S1rojO
igZEsruGsMrC3ZUIwWy1Q9uwoHnhlxF5rEIJONdlSdJPz3BIMleO79dSiFZKT4dt
5n0tjo/4u3Jdr/hpNSws8llX1c0PO9srQ3bhofe73EEo1hyDclttHLsLm28stbN8
Zzfvnw7FAYKx2VV86uL79wQaiPNxREUBr3wHpeTL3/s9Gg+/JkECggEBAPfa7/xm
w+pLCvnqwprrU2pVMOVbI+uCU1aytNqMZyi5zeXR30452bAxSLtGJhKTUdhCvv6b
td/+pQoQvn4tZxsHD7AJsMT8/gVIq/5w/k1m5X7p4cO5erCGtYZP2chTQsBkycpf
QGJppRBoVdgxe2qkfY2Cv0u524e0MawKToPaBhzG4e9GGyzpbCRZ3q8I2hx/hX0F
2n4f62N51HM9TDzutoSin0qLhvNf8E9I7Kym4XWIrT6b3ms6Yd94Rp3ljhDCFrTE
WOxLqD4H3jf7JlJ8j+58r3SYiQ0mN01y5DnD9tGi/VYxqFr1777VD/laGm7d1D2Z
fTcCbN4TFFGE0gkCggEBAOuyqtC4DiyjPJ5310pyDEjkiC2myEDAZS+BjPWTZVRo
qItGO8E5fuDTX44bENZ8/8isyVPTSCcGXhsaRTiFqb/NwGVK3c0t+2qDYCiqBuGp
K1Ph4KwX/0nPClFSnEkUfXej4PDB8G9C7h2XCVh2kT85/qfWcezzF6UUrEGuUYlL
641VB3tr6RT0SMGz7m+2ovWyYy9VOr+K54LItLobIeFH8MsdP9JCIFiBHZ+D+ma7
q79g59jHgAEu2vgYfKaCpXAp2j6tOgWqPWF7mliqFD/GJKXyaaObXfz+ZpkEgqE/
vgQREOO/c5VVZFB/bOemJLUYM1WytQAbldMN01Rt14kCggEAOWywTXpByfa5BE4v
6FS9btVuDrWfDOGVDXE6FaiR/g2OdsC5TBZ7KSdCAqGuEH+xZrmQJs1Mxijpc/uN
Jw695LUuHUsheYJkGDVOJBVp1eURJuZpOD+w/VU4mXXGr3Ma9Bhl6E1JTYPMipCh
0wUj4wFZVYAFcjYNdtN47rM0nbfV0rUBg75qbW1ncMSho0wZvKCO/PhuNuqOTu3b
GxgIodVs1C4ZWdwZ2ClSNAxhSV8gvWp9ORRD4/QS2QO02MBmuds+B4O2Vojw4e5Q
vgeiSVoyvr6EqC7vEezYw0jrN7b/aHKq312B9BEnCr+yg8MsfKNImT0GlcgqEQm6
m2h6gQKCAQBhe/pObXHfYHyYBnUTI2yVUYBJcWvt7CVtqqWEhLwqV0cuo5PfbUpe
7s3c1rD2JakddOmoNADpsyaFCy6KHC6DWDQ1MOvgCx6rhT7mUryZ5QA4p3nnc91w
x6M603I0f7cNHsjQi0ZInmQh9PA2mIOmpPQAsx9Xo4uqCYzddZ3frXj1ca+wiodS
1V6qTyNVLTLlcCy5zQSJaIgsfZrSRpqStNCREb3t1s/OC0kXStzsVL7KXuhFru3w
j1KdvnL/45VNeOH9fmQ7J5hPk3HZLi9F2UwbHtI2ivIqy4Xf0A+/Zb/PqsdTi0Hh
B/p/mNSQUxVnmWTSEyHts3saWeOITg4RAoIBABx5l+sBHe+LQifjXZiFexCVQCeq
4Uimg/DCZAUcqFIc+a08gocgMgTwYp9lvwNO+VuBiCuvHa0iWGrMEERuMMb4D4PJ
4jjEvXXgZ4ncNS0U1a07ITNlw+fc08dzzysy9fSw9KnP9rdJh4uItneKxA0tLQnv
Ry0SbPrkzc5mb3OUvhYluCcT4w+p5ikWbgdRwhzkRSQlko96PeusXBT3BDEWAEbo
lYOnvEaEfVgmNQiE3JhC4NeX3FRecinORah0Qrf4EE1uWjkqRoGlzzO7UnzT1Gdn
04A9b1bn7oge3u1MU84EH1T12vNVGqcmE4HZV9zZakipzRklFwiWIL6eA4s=
-----END RSA PRIVATE KEY-----
```

```
class Hybrid
    def encrypt
        puts "256-bit Key generated"
        string = "The cat sat on the mat"
        puts "String: #{string}\n"

        c=OpenSSL::Cipher::Cipher.new("aes-256-cbc")
        c.encrypt
        c.key = @sessionkey
        c.iv=@iv=@iv=c.random_iv
        e=c.update(string)
        e << c.final

        @f = Base64.encode64(e)
        @g = Base64::encode64(@pubkey.public_encrypt(@sessionkey))
    end
end
```



```
key
5rNZ8NMIipOzi1dLZ+OHVFKr13B3EizbpvXDsB6q8BE


iv
7Bzvn1U06uZhMbbQJ8Nwxg==
class Hybrid
    def decrypt
        dec=0
        @sessionkey=0 # Reset session key
        @sessionkey=@privkey.private_decrypt(Base64.decode64(@g))
        dec=OpenSSL::Cipher::Cipher.new("aes-256-cbc")
        dec.decrypt
        dec.key = @sessionkey
        dec.iv=@iv
        d=dec.update(Base64.decode64(@f))
        d << dec.final
        puts "Decrypted #{d}\n"
    end
end
```

```
class Hybrid
    def display
        puts
        puts "Ciphertext: #{@f}\n"
        puts "Encrypted Symmetric Key:\n#{@g}\n"
    end
end

h.keygen
h.encrypt
h.display
h.decrypt
```

```
4096-bit Key generated
256-bit Key generated
String: The cat sat on the mat

Ciphertext: Z8VZggOHDWXswdl+igZDH9CoqMp6ZlCEmW7xc41ZfzE=

Encrypted Symmetric Key:
RE5kOLxkeSmYeJyws0g/pmegwC4PF1NPUY3E7gylGgGaBS9M84T8VqbNNT9Q
z7lWKysOAH5zNMfcrUmfj1mdp4cv9OUvzsfAiSUQVu/2iIYh/jwygJ/w8yCF
JAjTYvkvd4Td/4Vs+Gm8WgAnM2M8oxzYrAfp5u7dqcy9pgsg6o6T9mBPzfB/
pWjsPDtLkbV2xRL4fgJXBtsjRMI1ewO3hNimEXEyqTC9bShHGKDnsZrDwG/r
B6ZVZ6JKNoOTlCSaPCsgdKgd+nqfDNsvfduzVxg4Ev2Mh52LjHXLlRDOPel2
uL0tN8FXPY4wNaq/39tuLXxu24Nsl/BCsKPhe2nGJ4F0GZ/HTkdjPtxGS6/Q
57siMnxxWTkO9tM9JvqGyD75707EgdlQZR5Az5Ulq7u2LMJZ6HuZiEBMzgD9
Cxb4ST9TJxiFxu6MtVicVRuus1BkYFv6FJ2wdf+1+2mqPvQwSrUqu269VuGJ
g12xpgYY2UiwL9mtE8xW6BvfFZEesJSFXXiQQ8+I/28JWbxzuy8gLpmKHz36
WocbrMvTlb4nwWDbilUQBIpp4bUJHk0090mcfiJAUn3nLuqycwevVDeibhRK
UkpBzPGGVi8TthOYsKSfcQBuj2542t/k/CrpVGSnEf3QrotKQLNZPB2SpKx2
HmTRBbuMZe6UDYZyZfYHdbo=

Decrypted The cat sat on the mat
```

a complex example

- ruby hybrid encryption with web pages
- acquire a web page
- roundtrip encrypt the web page

```ruby
#!/usr/bin/env ruby -w
require 'rubygems'
require 'openssl'
require 'base64'
require 'nokogiri'
require ‘open-uri'

class HybridHTML < Hybrid
end

h = HybridHTML.new
```

```
Allegra:FOSDEM eleanor$ ruby hybrid-html.rb
4096-bit Key generated
4096-bit Key generated

Enter Web Page

http://minimalsites.com/

Ciphertext: YCqp6e
+Vngs84RtKdVTcsmhX5C9xzb6mDxOwJjSME8rAYTKqIi/pX1u1DH3a
2t3OrhJCbiX7mcYxFaqXfJWqHh6mhdVSGqFvgt7Qvg4zIr0Yo+nn9b4ZYGM6
6shKQ+OL6luFVY3K7QBQwQZJIyiGC8Y+6agOC7yMdOCTeYbFeaG6cuFuvLvb
IMGtdWWVo0mCls1BwBZutVn7+xNODcKBhvoHjpXnpsJZYLoSP6XUFnGHwoCG
hbpkTdxFW3wJ1y4cJyr8baX99OxjqkLSeYjd9PL7efJWXykJGJ4f53S9vzkI
4h79CX6yX3KR22rqWQtUzku3soILATIn38MRCClCwOfBXpC3nP6cDLOUZNYV
muNdFJ3xY7ZSNqA8UiUQIeUFaIKhDfclhRQ1gvseu8TVdww/vYrFXUEXEgvi
1nFMeLRnb/TroPSbCYvO/gUC3+wT5X8ScvzHiD1a36w+PS0o1DHeS2ren66S
RKs6DAyAnY4+9f7hF97xAWGGNUEGiVSbam+5S/naiuLya22dVxZaEVP8SVkL
4TLJEbS9EWm/MYDSjQEpzVmFA8esuPaIiKJ/r/ae7KO0x+PfJx9Wt+HiI+H1
uBEYLDWcC2LWktqcZzzLBgA0xP6kyHk7BYB47IZ0mfzwBsPR+sDcvgIUtLfT
4QVsQc7sgCjDkPWWaVGU+AduIzHIpNpZNSJHn/KGLsx+UIbCJnSFlSSQM86P
SnleRHbIAt9Xb7vmsFQDmeRQDJlNUpY6CfkU6Lj6ATydvuWIFsQMwu2HXNri
9SXLgcQ+zV1MwK9OJlwgF5HAAKPO2AKHu/jl8zuh8S8xHQk7FZ1tk5T/cbdH
fP2DDlwkZdwX0J3nztXVmghjjAweTlRoZYdeYzLFheBf8773bCS6NoXWJVLP
LO9YkPCy/8h2ktoitcMsRpE8fk5Cq4AaHpOYovhKh8yUTDdYcCzh4XqQnTav
Q3GiuhbnklB+MQmFMiPKL0OTdy++DUmvvFE4G+GjzMZJWGvi+RDH2gzRdZ+F
gVBEhKbniju9dRhg172OJ+in+rdjR1V7d0gmLWwoeWqQrABzj2y2lHAlm25G
5wVAFcRqhEJxstASfBlMXgX9fqzRt3JSSgsrJquOfTAjSk0rWI5F2R2ebsvR
BU1QW55hP9/cTHp/tGnTaSBd+ZoG/dSdV/UOKfgcflB1qsunGmIkNu3tx9vB
c7sjLKOyWdvtRWpfVGlOV0tepIA3JXucUnYCslQbX7uK8BDSe/dLYvT8d+6H
lCEuxBp1ugAlKch3rDKOy0Tnmts0bgTyMbTIllrP69M6l0wr5KBr3jUE36N3
7PI=

Encrypted Symmetric Key:
YeNraRPZFzeQN+IixCKuSSdWkTf7MxIUfWXqxRbJ62olwmylc7JJF8xbU4t/
suDMrLGjVIr07oAvCbaoXmxoUQmj8dQA129Pqtxz712JOyd3RvMSeRFMaBNC
kjehFvTs53mu1HKO9bEHlZLxNm9c8+H31plc3hWGIpdy3YV+B6K+TNSRCFDY
BV5arzDgBC1v3kGe/nifkh3Ph1A9AsWXLyUzLhebLkn1pa6Ojgue1IarhSCi
XThojRUAuuCI6eMXbdE0K3fTe+pR7hklbcjraHwj/tIg9c15A6zFhTwUXEsE
DEaPz9x3LSMR4KKnL0i8fdJsmuYpyESkc4ueCJBoEY2ww/n+8zhsJNVMD8yu
r2o8RBnP4HRtxOvGRZ+1S+ddg5Q5KzHkp3+Qpfc1N8cpJ0q5HZeALDHKhVC+
IRirrjd92cDxm3ujCsyDWIrkeellhpPhFNX7PhUowyIyhKA9Rr1q7TKsGVON
CDDLxV6VELc2im9r/+ghHM3relizr7K4yfo7ErhxvP4kmFMoStG/lB94/lgO
9Th4YaLWZReqFIjQGe8kwmWwpq3CH1ZMbb3JE/3Vgcc1ogn0m2ajGC3+79N0
f4lQSTwvZPWuQNxPGtfzFwui0gdZ+mq793g2PHB4gBxMkniz96pscwjXLp2M
sOnrc/ZSb5EH/g9TGqALgHE=

Decrypted

Minimal Sites


@import url(http://css-reset-sheet.googlecode.com/svn/
reset.css);
@import url(http://fonts.googleapis.com/css?family=Karla);

html, *{
    font-family: 'Karla', sans-serif;
    font-weight: 400;
    background-color: #fff;
color: #999;
       font-size: 1.5em;
       line-height: 1.3em;
       text-rendering: optimizeLegibility;
}
body{
padding: 0 3em 3em;
}
.hello{
margin: 2.4em 0;
}
.hello > h1{
color: #000;
}

.links{
    list-style: none;
    padding-right: 6em;
    padding-top: .4em;
    border-top: 1px solid #000;
display: inline-block;
         font-size: .7em;
}
.links > li{
display: inline-block;
         margin-right: 0.4em;
}
.links > li > a{
    text-decoration: none;
color: #000;
}
@media only screen and (max-width : 568px) {
    body{
padding: 0 1em 1em;
         font-size: .9em;
    }
    .hello{
margin: 1em 0;
    }
    .links{
display: block;
         padding-right: 0;
    }
}




Hello, Minimal Sites is taking a nap.
See you in 2015.
```

```
class HybridHTML < Hybrid
    def encrypt
        puts "4096-bit Key generated"
        puts "\nEnter Web Page\n\n"

        file = /\n/.match(gets()).pre_match()
        string=Nokogiri::HTML open(file)

        c=OpenSSL::Cipher::Cipher.new("aes-256-cbc")
        c.encrypt
        c.key = @sessionkey
        c.iv=@iv=@iv=c.random_iv
        e=c.update(string)
        e << c.final
        @f = Base64.encode64(e)
        @g = Base64::encode64(@pubkey.public_encrypt(@sessionkey))
    end
end
```

beyond http

- nothing to stop us encrypting HTTPS pages too
- difficult to show in terminal
- DNS NAPTRs (RFCs 3401-3405)
- needs further explanation…

THE NAPTR RESOURCE

NAMING AUTHORITY POINTER

- this table shows the two modes of NAPTR as they appear in a DNS zone record

TTL			order	preference	flag	service type	regex + replacement	terminator
600	IN	NAPTR	100	50	“u”	“E2U+sip”	“!^.*$!sip:joe@fish.com!”	.
600	IN	NAPTR	100	51	“”	“”	“”	test.com

TTL	time in seconds before record must be resolved from an authoritative server
order	order in which records should be evaluated
preference	preference within a given order index
flag	“u” for a standard terminal record resource record as specified in the RFCs
service type	ENUM to URI + service type for an ENUM-specific service type
regex	regular expression to use for matching
replacement	string to replace the matched URI with
terminator	either “.” or the target domain name for a non-terminal record

NAPTR CRYPTO RECIPE

- take a standard NAPTR record

IN	NAPTR	100	11	“u”	“E2U+email:mailto”	“!^.*$!mailto:romeks@gmail.com!”	.

- encrypt with session key & initialisation vector

initialisation vector	+FgTpo7SPyd7cZx+cGVAtg==
session key	/wVEQmHS4vhwO/AJTDqpGoXwMYYHiSUmZShY7GSCcrl=

- store results in an encrypted replacement field

IN	NAPTR	100	11	“u”	“E2U+email:mailto”	“!^.*$! HVnGeCBG4ISOvghq8jwylpFQmvotfaSjdgQ88ExkaIU=!”	.

to recap

- this system is highly flexible
- protocol independent
- fast and algorithm independent
- easy to setup and use
- lightweight
- great for user -> user communication

a complex example

- a web storage service in go
- stores and retrieves arbitrary text messages
- client and server interacting over http
- RSA encryption for symmetric key transfer
- stream-based AES encryption for messages

the server

```
var server = NewFileServer("localhost:1024")

func main() {
	server.GET("/", ServerStatus)

	server.GET("/key", PublicKey)
	server.POST("/key/:id", StoreKey)

	server.POST("/user", RegisterUser)
	server.GET("/user/:id", UserStatus)

	server.GET("/file/:id", ListFiles)
	server.GET("/file/:id/:filename", RetrieveFile)
	server.POST("/file/:id/:filename", StoreFile)

	server.ListenAndServe()
}
```

github.com/feyeleanor/webcryptodemo

```
import "crypto/rsa"
import “net/http"
import "github.com/julienschmidt/httprouter"

type FileServer struct {
	PEM string
	*rsa.PrivateKey
	Started time.Time
	Address string
	*httprouter.Router
	UserDirectory
	Requests int
}

func (s *FileServer) ListenAndServe() {
	s.Started = time.Now()
	http.ListenAndServe(s.Address, s.Router)
}
```

github.com/feyeleanor/webcryptodemo

```
import “encoding/base32“
import “crypto/rand"

type FileStore map[string]string

type user struct {
	Key           []byte
	ID             string
	Registered time.Time
	FileStore
}

type UserDirectory map[string]*user

func (u *UserDirectory) NewUserToken() string {
	b := make([]byte, 30)
	if _, e := rand.Read(b); e != nil {
		panic(fmt.Sprintf(“rand.Read failed: %v", e))
	}
	return base32.StdEncoding.EncodeToString(b)
}
```

github.com/feyeleanor/webcryptodemo

```
import "html/template"
import "os"

var templates = template.Must(
        template.ParseFiles("server_status.txt", “server_status.html",
                "user_status.txt", "user_status.html", "list_files.txt", "list_files.html"))

func renderTemplate(w io.Writer, t string, v interface{}) {
        if e := templates.ExecuteTemplate(os.Stderr, t+".txt", v); e != nil {
                fmt.Println(e)
        }
        if e := templates.ExecuteTemplate(w, t+".html", v); e != nil {
                fmt.Println(e)
        }
}
```

github.com/feyeleanor/webcryptodemo

server_status.html

```
<html>
	<head>
		<title>Server Status</title>
	</head>
	<body>
		<table>
			<tr>
				<td>launched</td>
				<td>{{.Started}}</td>
			</tr>
			<tr>
				<td>current time</td>
				<td>{{.Now}}</td>
			</tr>
			<tr>
				<td>users</td>
				<td>{{.Users}}</td>
			</tr>
			<tr>
				<td>files</td>
				<td>{{.Files}}</td>
			</tr>
			<tr>
				<td>requests</td>
				<td>{{.Requests}}</td>
			</tr>
		</table>
	</body>
</html>
```

server_status.txt

```
= = = = = = = = =  Server Status  = = = = = = = = =
launched     {{.Started}}
current time {{.Now}}
users        {{.Users}}
files        {{.Files}}
requests     {{.Requests}}
= = = = = = = = = = = = = = = = = = = = = = = = = =
```

github.com/feyeleanor/webcryptodemo

```
func ServerStatus(w http.ResponseWriter, r *http.Request, _ httprouter.Params) {
	renderTemplate(w, "server_status", server)
}

func PublicKey(w http.ResponseWriter, r *http.Request, _ httprouter.Params) {
	server.Requests++
	w.Header().Set("Content-Type", "text/plain; charset=utf-8")
	Fprint(w, server.PEM)
}
```

github.com/feyeleanor/webcryptodemo

```
import "crypto/rand"
import "crypto/rsa"
import "crypto/sha1"

func EncryptRSA(key *rsa.PublicKey, m, l []byte) ([]byte, error) {
	return rsa.EncryptOAEP(sha1.New(), rand.Reader, key, m, l)
}

func DecryptRSA(key *rsa.PrivateKey, m, l []byte) ([]byte, error) {
	return rsa.DecryptOAEP(sha1.New(), rand.Reader, key, m, l)
}
```

github.com/feyeleanor/webcryptodemo

```
import "crypto/rsa"
import "crypto/x509"
import "encoding/pem"

func LoadPrivateKey(b []byte) (r *rsa.PrivateKey, e error) {
	if block, _ := pem.Decode(b); block != nil {
		if block.Type == "RSA PRIVATE KEY" {
			r, e = x509.ParsePKCS1PrivateKey(block.Bytes)
		}
	}
	return
}

func LoadPublicKey(k string) (r interface{}, e error) {
	b, _ := pem.Decode([]byte(k))
	return x509.ParsePKIXPublicKey(b.Bytes)
}
```

github.com/feyeleanor/webcryptodemo

```
import "crypto/rsa"
import "crypto/x509"
import "encoding/pem"

func PublicKeyAsPem(k *rsa.PrivateKey) (r string) {
	if pubkey, e := x509.MarshalPKIXPublicKey(&k.PublicKey); e == nil {
		r = string(pem.EncodeToMemory(&pem.Block{
			Type:  "RSA PUBLIC KEY",
			Bytes: pubkey,
		}))
	} else {
		panic(e)
	}
	return
}
```

github.com/feyeleanor/webcryptodemo

the client

```
var PublicKey *rsa.PublicKey

func main() {
    PublicKey = GetServerKey()
    u, k := RegisterUser()
    UserStatus(k, u)

    f := "this is a test file"
    StoreFile(k, u, "test", f)
    UserStatus(k, u)
    RetrieveFile(k, u, "test")
    if rf, e := RetrieveFile(k, u, "test"); e == nil {
        switch b, e := ioutil.ReadAll(rf); {
        case e != nil:
            Println(e)
        case string(b) != f:
            Println("Test file corrupted:", string(b))
        default:
            Println("file returned correctly")
        }
    }
}
```

github.com/feyeleanor/webcryptodemo

```go
func GetServerKey() (v *rsa.PublicKey) {
	if b, e := Do("GET", KEY); e == nil {
		if k, e := LoadPublicKey(string(b)); e == nil {
			v = k.(*rsa.PublicKey)
		} else {
			panic(e)
		}
	}
	return
}
```

```
Allegra:Hybrid eleanor$ ./server
= = = = = = = = = = User Status = = = = = = = = = =
ID       66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
Key      6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====
Files    0
= = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = User Status = = = = = = = = = =
ID       66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
Key      6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====
Files    1
= = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = User Status = = = = = = = = = =
ID       66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
Key      6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====
Files    1
= = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = User Status = = = = = = = = = =
ID       66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
Key      MD45J5O2JUNTR2OBALT6BWWWBBLU3XS7HSRJWRX5LV5RS2UBQ6FA====
Files    1
= = = = = = = = = = = = = = = = = = = = = = = = = =
= = = = = = = = = = User Status = = = = = = = = = =
ID       66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
Key      MD45J5O2JUNTR2OBALT6BWWWBBLU3XS7HSRJWRX5LV5RS2UBQ6FA====
Files    1
= = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
Allegra:Hybrid eleanor$ ./client
GET http://localhost:1024/key --> 200 OK
POST http://localhost:1024/user --> 200 OK
66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
GET http://localhost:1024/user/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
--> 200 OK
<html>
            <head>
                        <title>User Status</title>
            </head>
            <body>
                        <table>
                                    <tr>
                                                <td>ID</td>

<td>66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R</td>
                                    </tr>
                                    <tr>
                                                <td>Key</td>

<td>6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====</td>
                                    </tr>
                                    <tr>
                                                <td>Files</td>
                                                <td>0</td>
                                    </tr>
                        </table>
            </body>
</html>
POST http://localhost:1024/file/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R/
test --> 200 OK
<html>
            <head>
                        <title>User Status</title>
            </head>
            <body>
                        <table>
                                    <tr>
                                                <td>ID</td>

<td>66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R</td>
                                    </tr>
                                    <tr>
                                                <td>Key</td>

<td>6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====</td>
                                    </tr>
                                    <tr>
                                                <td>Files</td>
                                                <td>1</td>
                                    </tr>
                        </table>
            </body>
</html>
GET http://localhost:1024/user/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
--> 200 OK
<html>
            <head>
                        <title>User Status</title>
            </head>
            <body>
                        <table>
                                    <tr>
                                                <td>ID</td>

<td>66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R</td>
                                    </tr>
                                    <tr>
                                                <td>Key</td>

<td>6M5V5LC3BXVCQVRNKVX25I5XJSIG56JS6JK4K2GWDY4M3WS5G77A====</td>
                                    </tr>
                                    <tr>
                                                <td>Files</td>
                                                <td>1</td>
                                    </tr>
                        </table>
            </body>
</html>
GET http://localhost:1024/file/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R/
test --> 200 OK
this is a test file
POST http://localhost:1024/key/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
--> 200 OK
<html>
            <head>
                        <title>User Status</title>
            </head>
            <body>
                        <table>
                                    <tr>
                                                <td>ID</td>

<td>66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R</td>
                                    </tr>
                                    <tr>
                                                <td>Key</td>

<td>MD45J5O2JUNTR2OBALT6BWWWBBLU3XS7HSRJWRX5LV5RS2UBQ6FA====</td>
                                    </tr>
                                    <tr>
                                                <td>Files</td>
                                                <td>1</td>
                                    </tr>
                        </table>
            </body>
</html>
GET http://localhost:1024/user/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R
--> 200 OK
<html>
            <head>
                        <title>User Status</title>
            </head>
            <body>
                        <table>
                                    <tr>
                                                <td>ID</td>

<td>66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R</td>
                                    </tr>
                                    <tr>
                                                <td>Key</td>

<td>MD45J5O2JUNTR2OBALT6BWWWBBLU3XS7HSRJWRX5LV5RS2UBQ6FA====</td>
                                    </tr>
                                    <tr>
                                                <td>Files</td>
                                                <td>1</td>
                                    </tr>
                        </table>
            </body>
</html>
GET http://localhost:1024/file/66I2PXXEYJ2UU6AY5VTIE4I5KACRVXVN74ADRUFSWPMQED4R/
test --> 200 OK
this is a test file
file returned correctly
```

github.com/feyeleanor/webcryptodemo

```
func RegisterUser() (u string, k []byte) {
    k = GenerateAESKey(256)
    if key, e := EncryptRSA(PublicKey, []byte(k), []byte("REGISTER")); e == nil {
        if v, e := Do("POST", USER, string(key)); e == nil {
            u = printResponse(v, e, k)
        }
    }
    return
}

func RetrieveFile(key []byte, id, tag string) (f io.Reader, e error) {
    r, e := Do("GET", FILE, id, tag)
    f = bytes.NewBufferString(printResponse(r, e, key))
    return
}
```

github.com/feyeleanor/webcryptodemo

```
func Do(m, r string, p ...string) (b []byte, e error) {
        do(NewRequest(m, r, p...), func(res *http.Response) {
                b, e = ioutil.ReadAll(res.Body) })
        return
}

func DoEncrypted(k []byte, m, r string, p ...string) (b []byte, e error) {
        do(NewEncryptedRequest(k, m, r, p...), func(res *http.Response) {
                DecryptAES(res.Body, k, func(s *cipher.StreamReader) {
                        b, e = ioutil.ReadAll(s) }) })
        return
}

func do(req *http.Request, f func(*http.Response)) {
        if res, e := http.DefaultClient.Do(req); e == nil {
                Printf("%v %v --> %v\n", req.Method, req.URL, res.Status)
                f(res)
        } else {
                Println(e)
        }
        return
}
```

github.com/feyeleanor/webcryptodemo

```
import "crypto/aes"
import "crypto/rand"

func GenerateAESKey(n int) (b []byte) {
    switch n {
    case 128: b = make([]byte, 16)
    case 192: b = make([]byte, 24)
    case 256: b = make([]byte, 32)
    }
    rand.Read(b)
    return
}

func GenerateIV() (b []byte, e error) {
    b = make([]byte, aes.BlockSize)
    if _, e = rand.Read(b); e != nil {
        panic(e)
    }
    return
}
```

github.com/feyeleanor/webcryptodemo

```
import “crypto/cipher"
import "io"

func SendIV(w io.Writer, k []byte, f func([]byte)) {
    if iv, e := GenerateIV(); e == nil {
        if _, e = w.Write(iv); e == nil {
            f(iv)
        } else {
            fmt.Println(e)
        }
    }
}

func EncryptAES(w io.Writer, k []byte, f func(*cipher.StreamWriter)) (e error) {
    var b cipher.Block
    if b, e = aes.NewCipher(k); e == nil {
        SendIV(w, k, func(iv []byte) {
            f(&cipher.StreamWriter{S: cipher.NewCFBEncrypter(b, iv), W: w})
        })
    }
    return
}
```

github.com/feyeleanor/webcryptodemo

```
import "io"

func ReadIV(r io.Reader, f func([]byte)) {
	iv := make([]byte, aes.BlockSize)
	if _, e := r.Read(iv); e == nil {
		f(iv)
	} else {
		fmt.Println(e)
	}
}

func DecryptAES(r io.Reader, k []byte, f func(*cipher.StreamReader)) (e error) {
	ReadIV(r, func(iv []byte) {
		var b cipher.Block
		if b, e = aes.NewCipher([]byte(k)); e == nil {
			f(&cipher.StreamReader{S: cipher.NewCFBDecrypter(b, iv), R: r})
		} else {
			fmt.Println(e)
		}
	})
	return
}
```

github.com/feyeleanor/webcryptodemo

Questions…?

Romek Szczesniak
romeks@gmail.com