CS 463 Take-Home Midterm (Dr. Lawlor)

0.) YOUR NAME: ____Ann Tupek_____

Your finished electronic version of this exam document is due back by midnight 2013-03-06 as a PDF file. Please write clearly and professionally, using correct spelling and complete sentences, but keep things to the point. You are encouraged to use hyperlinks where appropriate. Everything you write should be your own work, in your own words.

This is an exam, but it is open web, open books, and open notes. In fact, you may use any non-living reference material—thus the spirit world is OK; but classmates, friends, or faculty are not, even if reached via email or SMS.

—

1.) After the fiasco last year where Phil sold our ATM software update private key on eBay, and the Baltislavonian mafia were able to push that malware card skimmer update out to most of our customers' ATMs, our new CTO has been pretty anxious about preventing that sort of thing from happening again. She's from the air force, where they enforced a "no lone zone" around their nuclear missile launch controls, and is convinced something similar should be able to work to protect our RSA private key, where no single computer has access to the key.

She thinks it could work where instead of keeping the whole private key exponent *d* on one single machine, after generating the key normally you split it up by choosing a random number *k*, keep *k* on one machine, and keep *d-k* on the other machine. Then to sign an update's hash *m* we somehow run it through both machines to end up with the usual RSA signature m^d just like normal.

1.a.) Can we make the math work out? How?

Yes, we can make the math work out: $m^k * m^{d-k} = m^{k+d-k} = m^d$.

1.b.) What data needs to go between the two machines, and what do the machines need to compute?

The update's hash m needs to go to both machines. On machine A, we raise m to the k, and on machine B, we raise m to the d-k. Then, either machine A needs to send its part of the signature to machine B, or machine B needs to send its part of the signature to machine A. Whichever machine receives the other's signature needs to multiply it by the exponential computation that it did, and the result will be the correct RSA signature.

1.c.) If an attacker, or a rogue sysadmin, has total control of one of the machines but not the other, could they sign updates? We don't want them to be able to sign updates, so we're hoping the answer is "NO".

If the attacker/rogue sysadmin has control of the machine that receives the part of the signature from the other machine, then they could figure out the exponent that the hash was raised to on that other machine, because they also received the hash m. Call the part of the signature they received x, $x=m^k$. Because they also have the hash m, they could take $\log_m$ of x to get k, the exponent that m was raised to to get x, $k = \log_m x$. They would therefore have the ability to generate d directly and sign updates. So, we can either hope that the attacker/rogue sysadmin has control of the first machine that sends its data on, and does not control the second machine that computes both its own half and then the full signature (probably not a risk that we want to take). Or, we can have both machines send on their data to a third machine that does not receive the hash m itself, and only receives m^k and m^{d-k}, and multiplies them together for the RSA signature m^d.

—

2.) Diffie-Hellman key exchange is normally implemented using modular exponentiation or elliptic curve point multiplication, both of which sound hard. So it's often described in terms of color mixing, where for example we might blend various quantities of CMY pigments (Cyan, Magenta, and Yellow, the primary pigments) to achieve a color mixture, and we can set it up so we can blend the pigments in any order and we'll arrive at the same color.

Mathematically, we can use vector notation to describe the amount of C, M, and Y pigments. So we might start with a public agreed start color like magenta, C=0, M=1, and Y=0:

startCMY=(0,1,0) magenta

Each side would blend this color half and half with their secret colors, say

aSecretCMY=(1,0,0) cyan

bSecretCMY=(0,1,1) brick red

giving these public colors, which are exchanged publicly as the mixture:

publicCMY = 1/2 * startCMY + 1/2 * secretCMY

aPublicCMY=1/2 * (0,1,0) + 1/2 * (1,0,0) = (0.5,0.5,0) purple

bPublicCMY=1/2 * (0,1,0) + 1/2 * (0,1,1) = (0,1,0.5) red

Each side would then blend the other's public color with 1/3 of their own secret color to give an even three-way blend of the start color and both secret colors, in this case resulting in:

sharedCMY = 2/3 * yourPublicCMY + 1/3 * mySecretCMY

aSharedCMY = 2/3 * (0,1,0.5) + 1/3 * (1,0,0) = (0.33, 0.66, 0.33) muddy brown

bSharedCMY = 2/3 * (0.5,0.5,0) + 1/3 * (0,1,1) = (0.33, 0.66, 0.33) muddy brown

2.a.) If the public colors exchanged are:

aPublicCMY = (0.5,0.5,0.5) gray

bPublicCMY = (0.5,1,0) blue-purple

What are each side's secret colors, and what is the shared secret color?

startCMY/2+secretCMY/2=publicCMY

aSecret = 2 (publicCMY – startCMY/2) = 2((0.5, 0.5, 0.5)-(0.0, 1.0, 0.0)/2)
= 2((0.5, 0.5, 0.5)-(0.0, 0.5, 0.0))
=2(0.5, 0.0, 0.5)
= (1.0, 0.0, 1.0) = **GREEN**

bSecret = 2(publicCMY-startCMY/2) = 2((0.5, 1.0, 0.0)-(0.0, 1.0, 0.0)/2)
= 2((0.5, 1.0, 0.0)-(0.0, 0.5, 0.0))
=2(0.5, 0.5, 0.0)
= (1.0, 1.0, 0.0) = **BLUE**

Shared Secret = 2/3(aPublic)+1/3(bSecret)=2/3(0.5, 0.5, 0.5)+1/3(1.0, 1.0, 0.0)
=(0.33, 0.33, 0.33)+(0.33, 0.33, 0.0)
=(0.66, 0.66, 0.33) = **GRAY-BLUE**

2.b.) Is it a good idea to use linear functions for cryptographic operations?

No, it is not a good idea to use linear functions for cryptographic operations because they are easily reversible.

—

3.) A company's sales flier says, in part: “Our proprietary encryption algorithm is so advanced, even **we** don't understand how it works! The core operation is called binary bitwised [*sic*] rotation, invented by Ronald Rivest the father of modern cryptography, and so complex it's beyond the standard C++ operators. And our algorithm repeats this provably mathematical secure operation a **dozen** times per int!”

3.a.) Manager: What are the advantages and drawbacks of doing business with this company?

The drawback to doing business with this company is that their encryption is easy to decrypt because they are not mixing in a key with the data, so whatever data you encrypt is easy for an adversary to decrypt. However, if you want to steal this company’s secrets, and if they encode their own secret documents with this algorithm, it will be easy to decipher them, and then use your knowledge to take control of the company and institute a better encryption algorithm.

3.b.) Mathematician: Briefly explain to them how “binary bitwised [*sic*]” rotation actually works, using appropriately simple words.

A binary bitwise rotation shifts the binary digits over one place for each rotation. For a left rotation, the high bit at the end is rotated around to be the new low bit, and for a right rotation, the low bit at the end is likewise rotated around to be the new high bit.

For example, rotating 1001 0110 left one rotation results in 0010 1101.

3.c.) Theoretical Cryptanalyst: Explain an operation equivalent to a series of a dozen fixed-bitcount rotations.

An operation that is equivalent to a series of a dozen fixed-bitcount rotations is four fixed-bitcount rotations in either direction.

3.d.) Applied Cryptanalyst: The company has “encrypted” a four-letter ASCII message as the little-endian integer 0x2DEAE42C. What is the message?

I don’t know. The left bitwise rotation I applied twelve times gives me: 0xae42c2de which does not translate into ASCII letters. A right bitwise rotation results in 0x42c2deae, and again, no ASCII letters. I did find some ASCII on left rotation 3&11: W!ao and on right rotation 5: aoW!. Since I wasn’t able to decrypt this, it makes me second-guess my answers to the above questions. However, I’m sure that this is easy to decrypt and I’m just missing something stupid, so I’m going to stick with my above answers.

—

4.) Create an RSA private key and signing request with a subject field that includes your real name in the CN field.

4.a.) Record the process you used here, from installation to creating the key:

I already had OpenSSL installed on my machine. I'm running version 0.9.8za and it was installed on 6/5/14. To generate the RSA private key and signing request, on the command line, I typed in:

```
openssl req -new -newkey rsa:2048 -nodes -keyout mycert.key -out mycert.csr -
subj  '/O=UAF/OU=CS680/CN=AnnTupek/C=US/ST=Alaska/L=Fairbanks'
```

I then opened up mycert.csr with a text editor and copied and pasted it into crApto.

4.b.) Upload the key signing request to crApto to get it signed by my CA "lawCA", and paste the resulting signed PEM file here:

```
-----BEGIN CERTIFICATE-----
MIIDWDCCAsGgAwIBAgICBngwDQYJKoZIhvcNAQELBQAwgZ8xCzAJBgNVBAYTAlVT
MQ8wDQYDVQQIDAZBbGFza2ExEjAQBgNVBAcMCUZhaXJiYW5rczEOMAwGA1UECgwF
TGF3Q0ExGTAXBgNVBAsMEE1hbHdhcmUgRGl2aXNpb24xHjAcBgNVBAMMFUxhd0NB
IE1hbHdhcmUtT25seSBDQTEgMB4GCSqGSIb3DQEJARYRbGF3bG9yQGFsYXNrYS5l
ZHUwHhcNMTUwMzA1MDMyMjAyWhcNMTcxMTI5MDMyMjAyWjBjMQswCQYDVQQGEwJV
UzEPMA0GA1UECBMGQWxhc2thMRIwEAYDVQQHEwlGYWlyYmFua3MxDDAKBgNVBAoT
A1VBRjEOMAwGA1UECxMFQ1M2ODAxETAPBgNVBAMTCEFublR1cGVrMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuS3DB0AeM9pr5lRurmIf03+4PmEAWKbY
IVtN/VyoSfX49AxaKD0Pba2JaZRcAyAcIQPPVAFdqhnNpaF8Kb23UcOqa8dY5Psx
Jj6GIdQMsOj/TkyNdwBI5e0iu3MpDuZy1qNc4JvcxIpKxE09ohh2oa63HSj/VWy8
5UUWP2huFuaZ5tFJ0RYAWBMvcBYvOIUG06H8Rs8PFhHeidkxt1tC/hnvV0+/s/kx
fTWPTMnJt0MlFhITSGrOhE6Np0FKA2+3PaIh0Fx3MOm0Z1y9PyPjGEf3X7/h/z8f
anTiqtK6/i7XFfZ/IcB+Gc17mM9N3F80ffXbQ7BHUQ6IECOmyEMXOQIDAQABo1ow
WDAdBgNVHQ4EFgQUiSuqpg0VCaPv/gUgTNEqyfoq1d0wHwYDVR0jBBgwFoAUcHnd
0VMnqiaaMlRtEGXPflr0g1IwCQYDVR0TBAIwADALBgNVHQ8EBAMCBaAwDQYJKoZI
hvcNAQELBQADgYEAEPRe9l49AtKhQpHcYNYmzLRTFbTcde4zPHWi1zvWZlCHWdZn
qQth3VCj50C/nEbN1ECpjrP6j/bd29ZAe69e/a0QgYccLhfhDCt/qw01UaMdYI+2
fwyCwifx0IBqV3ZvnhZbc6TDOMIlajJJfra4EFrYPfL2chnLglRKMthpU3g=
-----END CERTIFICATE-----
```

—

5.) Currently, your company's active directory passwords are stored using “reversible encryption”, which allows them to be synchronized to your PHP bulletin board system, which stores them in a MySQL database.

5.a.) Is this a bad idea? If so, how would you change it? If not, why not?

Yes, this is a bad idea because storing passwords using reversible encryption is equivalent to storing them as plaintext should the encryption key be compromised. Instead, I would store a hash of the password, salted with the username and the domain of the bulletin board.