# Principles of Computer Security
# Exercises 4

Stuart Golodetz

February 26, 2006

1. (a) The keys are $56$ bits long, so there are $2^{56}$ possible keys to try. In the worst case, the attack thus takes $2^{56}\ \mu s = 72057594037927936\ \mu s \approx 2280$ years (3SF). In the expected case, the attack takes half that, i.e. $1140$ years or so. (The best case, of course, sees the attack taking $1\ \mu s$: in practice, the probability of this happening is so small as to be negligible!)

   (b) The only ways I can think of to improve on the attack are:

   - Use lots of computers running in parallel. In practice, you'd need a LOT of computers, but with 832200 running in parallel you could cut the time down to a day, which is feasible if you're the US government, say: there are a lot more computers than that in existence, remember! It's also theoretically feasible if you're a hacker and have managed to gain access to a lot of people's machines with a trojan or something.
   - TODO

   (c) To protect against this sort of attack, we can try and ensure that the attacker has to decrypt the whole message to learn its meaning. In other words, even if he succeeds in decrypting any single block, the attacker won't discover part of the message. The easiest way to do this is just to rearrange the message before encrypting it with DES.

   The block size of DES is $64$ bits. If the message length $n$ isn't a multiple of $64$ then we can pad the message with $\times$ characters until it is. We can thus work with the assumption that there are $m = \frac{n}{64}$ blocks used for a message of length $n$. We rearrange the original message $c_0 ... c_{n-1}$ so that block $i$ s.t. $0 \leq i < m$ is the encryption of $c_i c_{i+m} c_{i+2m} ... c_{i+(m-1)m}$.

   Let's give a simple concrete example (using (for simplicity) a block size of $4$ bits, meaning there are $\frac{16}{4} = 4$ blocks):

   ```
   This is a test××
   0123012301230123
             ↓
   ```
   $\underbrace{T\ \ as}_{Block\ 0}\quad \underbrace{hi\ \ t}_{Block\ 1}\quad \underbrace{ist\times}_{Block\ 2}\quad \underbrace{s\ \ e\times}_{Block\ 3}$

   The actual example is unimportant: the point is that decrypting any of the blocks just gives you a load of gibberish. To decrypt the original message, you have to decrypt all the blocks, something we expect to take a lot longer than the $1\ \mu s$ the attacker was spending decrypting just a few blocks.

2. (a) We want $e$ to be a small odd integer coprime to $\phi(n) = (p-1)(q-1)$. It's clear that $79$ is both small (roughly speaking!), odd and integral, so all that remains to check is that it's coprime to $(p-1)(q-1) = 42 \times 72 = 3024$. In other words, we want to show that $gcd(3024, 79) = 1$. Well, using Euclid's Algorithm:

$$gcd(3024, 79) = gcd(79, 22) = gcd(22, 13) = gcd(13, 9) = gcd(9, 4)$$
$$= gcd(4, 1) = gcd(1, 0) = 1$$

   (b) We know that $d = e^{-1} \bmod \phi(n) = 79^{-1} \bmod 3024$, so we can use the Extended Euclidean Algorithm to calculate it. We proceed as follows:

$$\begin{aligned}
1 = 1 - 0 &= 1 - (4 - 4 \cdot 1) = -1 \cdot 4 + 5 \cdot 1 \\
&= -1 \cdot 4 + 5(9 - 2 \cdot 4) = 5 \cdot 9 - 11 \cdot 4 \\
&= 5 \cdot 9 - 11(13 - 9) = -11 \cdot 13 + 16 \cdot 9 \\
&= -11 \cdot 13 + 16(22 - 13) = 16 \cdot 22 - 27 \cdot 13 \\
&= 16 \cdot 22 - 27(79 - 3 \cdot 22) = -27 \cdot 79 + 97 \cdot 22 \\
&= -27 \cdot 79 + 97(3024 - 38 \cdot 79) \\
&= 97 \cdot 3024 - 3713 \cdot 79
\end{aligned}$$

So the multiplicative inverse of $79$ modulo $3024$ is derived by getting $-3713$ into the right range (i.e. $(0, 3024)$) and turns out to be $-3713 + 2 \cdot 3024 = 2335$. Checking this, we calculate that $79 \cdot 2335 \bmod 3024 = 184465 \bmod 3024 = 1$.

(c) The encryption of $42$ is the result of the calculation:

$$\begin{aligned} & 42^e \bmod n \\ = \; & 42^{79} \bmod 3139 \\ = \; & [(42^2 \bmod 3139) \cdot ((42^{11} \bmod 3139)^7 \bmod 3139)] \bmod 3139 \\ = \; & [1764 \cdot (2235^7 \bmod 3139)] \bmod 3139 \\ = \; & [1764 \cdot 2106] \bmod 3139 \\ = \; & 1547 \end{aligned}$$

(d) The result should clearly be $42$, since it would be pretty pointless if we encrypted a message and it decrypted to a different message! To decrypt, we calculate as follows:

$$\begin{aligned} & 1547^d \bmod n \\ = \; & 1547^{2335} \bmod 3139 \\ = \; & (1547^5 \bmod 3139)^{467} \bmod 3139 \\ = \; & 2880^{467} \bmod 3139 \\ = \; & [(2880^5 \bmod 3139) \cdot (2880^{462} \bmod 3139)] \bmod 3139 \\ = \; & [2794 \cdot ((2880^6 \bmod 3139)^{77} \bmod 3139)] \bmod 3139 \\ = \; & [2794 \cdot (1463^{77} \bmod 3139)] \bmod 3139 \\ = \; & [2794 \cdot ((1463^7 \bmod 3139)^{11} \bmod 3139)] \bmod 3139 \\ = \; & [2794 \cdot (216^{11} \bmod 3139)] \bmod 3139 \\ = \; & [2794 \cdot 1119] \bmod 3139 \\ = \; & 42 \end{aligned}$$

3. First of all we prove the hint:

$$\begin{aligned} & M^{(p-1)(q-1)} \bmod p \\ = \; & (M^{(p-1)} \bmod p)^{(q-1)} \bmod p && \{\text{by (a)}\} \\ = \; & 1^{(q-1)} \bmod p && \{\text{by Fermat's Little Theorem}\} \\ = \; & 1 \bmod p \\ = \; & 1 \end{aligned}$$

It's also analogously true that $M^{(p-1)(q-1)} \bmod q = 1$. Now, if $M^{(p-1)(q-1)} \bmod p = 1$ then $\exists x \cdot M^{(p-1)(q-1)} = px+1$ and analogously therefore $\exists y \cdot M^{(p-1)(q-1)} = qy+1$. Whence $px = qy$ and by the fundamental theorem of arithmetic (i.e. numbers have a unique prime factorisation), it must be the case that $x = qz$ for some $z$ (since $p$ and $q$ are relatively prime and $px$ must contain $q$ as a factor). Then $M^{(p-1)(q-1)} = (pq)z + 1$, whence $M^{(p-1)(q-1)} \bmod (p \cdot q) = 1$, as required. (Note that the same result would have followed from the Chinese Remainder Theorem we're learning about in ADSA, but this shows it explicitly.)

We now need to derive our main result, i.e. that $M^{ed} \bmod n = M$. We observe that $ed = 1 \bmod (p-1)(q-1)$, so $ed = 1 + k(p-1)(q-1)$ for some integer $k$. Consider:

$$\begin{aligned} & M^{ed} \bmod p \\ = \; & M M^{k(p-1)(q-1)} \bmod p \\ = \; & M(M^{(p-1)})^{k(q-1)} \bmod p \\ = \; & M(M^{(p-1)} \bmod p)^{k(q-1)} \bmod p \end{aligned}$$

Now, provided $M \not\equiv 0 \bmod p$, this is equal (by Fermat's Little Theorem) to $M \cdot 1^{k(q-1)} \bmod p = M \bmod p = M$. (By contrast, if $M \equiv 0 \bmod p$, then $M^{ed} \bmod p = 0^{ed} \bmod p = 0 \bmod p = M \bmod p$ anyway.)

An entirely analogous argument will show that $M^{ed} \bmod q = M$ as well. Whence as before, we can deduce that $M^{ed} \bmod n = M$, and we're done.

4. The protocol description is as follows:

$$\begin{array}{l} 1.\ a \rightarrow b : a, \{\{a, n_a\}_{PK(b)}\}_{SK(a)} \\ 2.\ b \rightarrow a : \{n_a, n_b, k_{ab}\}_{PK(a)} \\ 3.\ a \rightarrow b : \{n_b\}_{PK(b)} \end{array}$$

The protocol achieves the following:

(a) Both $a$ and $b$ know $k_{ab}$ and no-one else does. The justification for this is that $b$ knows $k_{ab}$ because he created it and the only time it's sent in a message is when it's sent to $a$ and encrypted with $a$'s public key: in that instance, only $a$ can read it.

(b) $a$ knows that $b$ wants to talk to him and that the key $k_{ab}$ is current and from $b$. The justification for this is that he sent his nonce $n_a$ to $b$ encrypted using $PK(b)$ and received it back in a message: only $b$ could possibly have decrypted the message to determine $n_a$ (in the timespan involved, at least), so $b$ must have sent the message containing it and the session key $k_{ab}$.

(c) $b$ knows that $a$ wants to talk to him, since he sent his nonce $n_b$ to $a$ encrypted with $a$'s public key and received it back in a message: $a$ is the only other person who could know $n_b$, so $a$ must have sent it back and want to talk to him.

5. One (rather complicated) scheme would be as follows:

$1.\ c \rightarrow b : c, \{\{c, n_c, m, amount\}_{PK(b)}\}_{SK(c)}$

Providing $amount \leq balance$ :

$2.\ b \rightarrow m : b, \{\{b, n_b\}_{PK(m)}\}_{SK(b)}$

$3.\ m \rightarrow b : \{n_b, n_m\}_{PK(b)}$

$4.\ b \rightarrow c : \{\{\{b, c, n_m, amount\}_{PK(m)}, n_c\}_{SK(b)}\}_{PK(c)}$

$5.\ c \rightarrow m : c, \{\{\{b, c, n_m, amount\}_{PK(m)}, description(goods), n_c\}_{PK(m)}\}_{SK(c)}$

Providing $price(goods) = amount$ :

$6.\ m \rightarrow b : \{n_b, n_c, c, amount\}_{PK(b)}$

{Bank transfers amount from c to m}

$7.\ b \rightarrow m : \{n_m\}_{PK(m)}$

$8.\ m \rightarrow c : goods$

{If goods don't turn up, complain to the bank!}

The way this works is as follows:

(a) First of all, $c$ tells the bank 'I am $c$, I want to buy something from this merchant for this amount and here's a nonce you can use to convince me what I'm getting from you is fresh.'

(b) The bank checks the amount against $c$'s balance and, provided there's enough in the account, informs the merchant that the customer wants to make a transaction, giving the merchant its nonce and requesting one in return.

(c) The bank gives the customer a token to use for the transaction - on receiving this, the merchant will know this came from the bank because only the bank knows his nonce $n_m$. The bank assures the customer this is recent by using the customer's nonce $n_c$.

(d) The customer sends the token to the merchant, along with a description of the goods and his nonce $n_c$, which the merchant can use to prove to the bank that the customer really wants this transaction to go ahead (this prevents merchants arbitrarily getting banks to transfer money around).

(e) The merchant tells the bank to transfer the money from $c$ to $m$.

(f) The bank tells the merchant he's done this (by sending $n_m$, which only he and $m$ know).

(g) The merchant sends the customer the goods (we hope!) Failing which, legal proceedings might ensue.

What does this protocol achieve? Well:

- The merchant can be sure the customer will pay without needing a credit card number (the transaction is handled by the bank).
- The bank can be sure that transactions only occur when both the merchant and the customer want them to (the various nonces are used to ensure this).
- The customer can be sure that, at worst, the merchant can transfer some of his money this one time and renege on the deal: with the traditional credit card scheme, the merchant has the customer's card number and could pass this off to unscrupulous third parties, etc. He can also be sure that only transactions that he explicitly authorises will go ahead. Finally, he can be sure that the details of the transaction (other than the amount involved) are unknown to the bank.

6. TODO: I wasn't really sure how to go about this one, I'll have to see how to do it in class.