Computer Security
Coursework Exercise CW1

Cryptography and Protocols

School of Informatics
University of Edinburgh
`http://www.inf.ed.ac.uk/teaching/courses/cs`

This is an individual assessed coursework exercise. It will be awarded a mark out of 25. It is one of two assessed exercises in the Computer Security course. Each exercise is worth 12.5% of the final result for the course. The deadline for completing this coursework is 16:00, 16th March 2015. The final page summarises submission instructions. This exercise explores some topics in cryptography and protocols. This answers should be submitted on paper, by hand at the ITO. To answer the questions you will need to consult the lecture slides and additional resources such as the referenced papers that are cited in the lecture slides.

Question 1 The following questions are intended to help you further understand some of the concepts learned in the lectures regarding the use of pseudo-random number generators, and key entropy.

(a) A bit source S produces a statistically biased random sequence of bits $b_1b_2b_3\ldots$ and it is known that for each bit b_i, $prob(b_i = 0) = p$ and $prob(b_i = 1) = (1-p)$, for some $0 < p < 1$ where $p \neq 1/2$. Devise a simple algorithm to extract from S an unbiased random bit sequence $a_1a_2a_3\ldots$ (i.e., produce a bit sequence such that $prob(a_i = 0) = prob(a_i = 1) = 1/2$). Justify that your algorithm works as required.

ANSWER: Consider the following algorithm:

```
i := 1
while(true){
  if $b_i \neq b_{i+1}$
    output $b_i$
  i := i + 2
}
```

At each iteration of the while loop, the algorithm outputs:

- **0 with probability** $p(1-p) - prob(b_i = 0 \wedge b_{i+1} = 1) = p(1-p)$
- **1 with probability** $(1-p)p - prob(b_i = 1 \wedge b_{i+1} = 0) = (1-p)p$
- **nothing with probability** $p^2 + (1-p)^2 - prob(b_i = b_{i+1}) = p^2 + (1-p)^2$

So, in the resulting sequence we have for all $j \in \mathbb{N}$ $prob(a_j = 0) = prob(a_j = 1) = p(p-1)$.

(b) A colleague is preparing a presentation in which he wants to demonstrate the superiority of the public key cryptosystem RSA over the conventional cryptosystem AES. One of his arguments is that because RSA uses a key size of 2048 bits, while AES uses a key size of 128 bits, RSA offers a more secure alternative (because 2048 is a much larger number than 128). Do you agree with this argument? Why or why not?

ANSWER: The best known attack for breaking AES-128 is brute-forcing the key space. That is the complexity of the attack in in $O(2^{128})$. However, there are better attacks than brute-force against RSA. Indeed, the security of the RSA algorithm relies on the difficulty of large numbers factorization. The current best known algorithms for factoring large numbers is the Number Field Sieve that takes sub exponential time. Thus, RSA keys must be longer for equivalent resistance to attacks, than AES keys. RSA Security claims that 1024-bit RSA keys are equivalent in strength to 80-bit symmetric keys, 2048-bit RSA keys to 112-bit symmetric keys and 3072-bit RSA keys to 128-bit symmetric keys.

(c) A vendor is attempting to sell cryptographic software to your company and suggests that their software uses a key length of 160 bits, and hence requires $O(2^{160})$ operations to attack the cipher. The vendor explains that these 160 bits are derived as follows: the software repeatedly selects 20 random characters from the set $\{a, \dots, z\}$ and that since 8 bits are used to encode each character, the key length is $8 \times 20 = 160$ bits long. Do you believe his claim that this cipher offers 160 bits of security through its key? Why or why not?

ANSWER: The size of the key space is actually 26^{20}. So a brute-force attack takes $\mathcal{O}(26^{20}) \sim \mathcal{O}(2^{94}) << \mathcal{O}(2^{160})$

Question 2 Attack on variants of raw RSA.

(a) Assume that Alice wants to keep her RSA modulus N secret to everybody except to Bob. Alice uses $e = 3$ as public exponent. To encrypt a message m, Bob computes $c = m^3 \bmod N$ and sends c to Alice. Assume that Eve gets $c_1 = m_1^3 \bmod N$ and $c_2 = m_2^3 \bmod N$ and already knows m_1 and m_2; explain how Eve can recover N.

ANSWER: $c_1 = m_1^3 \pmod N$ and $c_2 = m_2^3 \pmod N$. But by definition this means that $N | m_1^3 - c_1$ and $N | m_2^3 - c_2$. So N is a common divisor of $m_1^3 - c_1$ and $m_2^3 - c_2$. So N is a multiple of gcd$(m_1^3 - c_1, m_2^3 - c_2)$. Furthermore, because $e = 3$ is small it must be that N is a "small" multiple of gcd$(m_1^3 - c_1, m_2^3 - c_2)$. So with high probability N is one of the first few multiples of gcd$(m_1^3 - c_1, m_2^3 - c_2)$. Could try and check the first few multiples. In fact there are even possibilistic attacks if m_1 and m_2 are related by a known relation.

For more on how to use RSA see

[1] Dan Boneh. Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society (AMS), 46(2):203213, 1999.

(b) Assume that Alice and Bob want to share the same modulus N but use different public exponent. Alice uses $e_A = 3$ and Bob uses $e_B = 5$. Let d_A and d_B be the corresponding private exponents. Explain how Alice can recover d_B from d_A.

ANSWER: If you assumed that Alice knows $\phi(N)$ for computing d_A, she can compute d_B using the Extended Eucledian algorithm. However Alice doesn't

need to be given $\phi(N)$, she can efficientlty recover it from N, e_A and d_A. See [1] above for more details.

(c) Assume that Alice and Bob want to share the same modulus N but use different public exponent. Alice uses $e_A = 3$ and Bob uses $e_B = 5$. Now Charlie wants to encrypt a message m for Alice and Bob. He sends:

$$c_A = m^3 \bmod N$$

to Alice and

$$c_B = m^5 \bmod N$$

to Bob. Explain how Eve can recover m from N, c_A and c_B.

ANSWER: Note that a and b are coprime, so we can compute a and b such that $3a + 5b = \gcd(3,5) = 1$. a and b can be computed using the extended Eucledian algorithm. $a = 2$ and $b = -1$ are possible solutions. Then we need to compute $c_A^2 \pmod N$ and $c_B^{-1} \pmod N$. By assumption on the message space, $\gcd(c_B, N) = 1$, and thus $c_B^{-1} \pmod N$ can be computed using the extended Eucledian algorithm. Finally compute $c_A^2 \pmod N) * c_B^{-1} \pmod N) = c_A * c_B^{-1} \pmod N) = m^{3*2} * m^{-5} \pmod N) = m \pmod N) = m$.

Question 3 Alice and Bob believe they share an k-bit secret key, and want to confirm that they do agree on the the same key. In order to achieve this over an insecure channel, while preventing an attacker from learning the secret key, they execute the following protocol. Let k_A be the key held by Alice, and k_B the key held by Bob.

1. Alice generates a random k-bit value r.
2. Alice computes $x = k_A \oplus r$, and sends x to Bob.
3. Bob computes $y = k_B \oplus x$ and sends y to Alice.
4. Alice compares r and y. $If r = y$, then she knows $k_A = k_B$ that is, she and Bob do agree on the same secret key.

(a) Show how an attacker can learn the shared secret key.

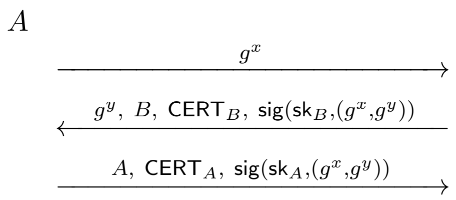
ANSWER: The attacker randomly selects r' and sends it to Bob, who will think it comes from Alice. Bob will send back $r' \oplus k_B$. Now the attacker can just xor his randomly selected value r' with the message sent by Bob and retrieve k_B. Indeed, $k_B = (r' \oplus k_B) \oplus r'$.

Another possible solution is for the attacker to just intercept x and y, and then retrieve $k_B = x \oplus y$.

(b) Show how an attacker can make Alice and Bob believe they do not share the same key.

ANSWER: The attacker needs to trick Alice into believing that she doesn't share the same key with Bob. For that the attacker intercepts Bob's response y, and sends any other message $r' \neq y$ to Alice. Note that if Alice and Bob do share the same key, Bob will be sending r (the one selected by Alice in the first step of the protocol). Alice will then compare the value she received (r') and the one she selected in the first step (r). As these are different she will think that Bob and herself do not share the same key.

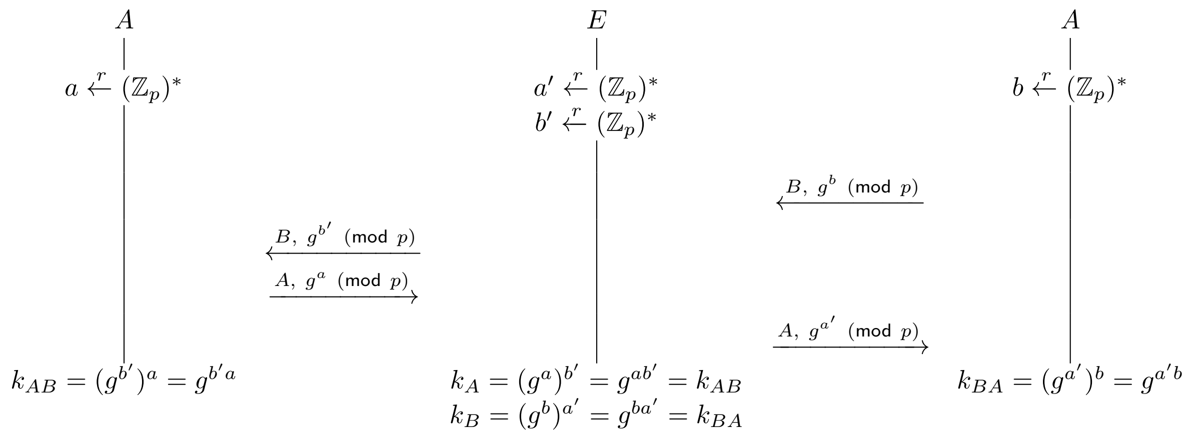
Question 4 In class, we saw the Diffie-Hellman protocol, which is a two-party key establishment protocol secure against passive attackers. However, as we saw, the Diffie-Hellman protocol is insecure against active attackers. Indeed, a malicious agent can mount a man-in-the-middle attack to learn a key not intended for him. This attack is possible because their is no mechanism to authenticate the two parties to one another. We consider the following extension of the Diffie-Hellman protocol to thwart this attack. We assume that the parties A and B have a private signing key sk_A and sk_B respectively, and a certificate on the corresponding public key CERT_A and CERT_B respectively.


A
B
g^x
g^y, B, CERT_B, $\mathsf{sig}(\mathsf{sk}_B,(g^x,g^y))$
A, CERT_A, $\mathsf{sig}(\mathsf{sk}_A,(g^x,g^y))$

The result is a shared secret $K_{AB} = g^{xy}$ from which the parties derive a session-key.

(a) Briefly explain the purpose of the signatures in the protocol above. How does it defend against the attack discussed in class?

ANSWER: The original Diffie-Hellman has no authentication mechanism to ensure the two parties that they are indeed talking to each other. In class, we saw that the DH protocol is subject to the following man in the middle attack


A
E
A
$a \xleftarrow{r} (\mathbb{Z}_p)^*$
$a' \xleftarrow{r} (\mathbb{Z}_p)^*$
$b' \xleftarrow{r} (\mathbb{Z}_p)^*$
$b \xleftarrow{r} (\mathbb{Z}_p)^*$
$B,\ g^b \pmod p$
$B,\ g^{b'} \pmod p$
$A,\ g^a \pmod p$
$A,\ g^{a'} \pmod p$
$k_{AB} = (g^{b'})^a = g^{b'a}$
$k_A = (g^a)^{b'} = g^{ab'} = k_{AB}$
$k_B = (g^b)^{a'} = g^{ba'} = k_{BA}$
$k_{BA} = (g^{a'})^b = g^{a'b}$

where Eve has caused

- **A to think that she is communicating securely with B and that they have both agreed to the key k_{AB};**
- **B to think that she is communicating securely with A and that they have both agreed to the key k_{BA};**
- **Eve has learned the keys k_{AB} and k_{BA} which were intended to remain secret from her.**

In the variant proposed in the statement of Problem 2, A and B sign their view on k_{AB} and k_{BA}. Now, because Eve cannot forge A or B's signature she cannot mount the attack on the original DH protocol on this variant of the protocol. In particular, she cannot sign with the secret signing key of A the message $(g^{a'}, g^b)$. In other words she cannot build message $\mathsf{sign}(\mathsf{sk}_A, (g^{a'}, g^b))$. Similarly, she cannot sign with the secret signing key of B the message $(g^a, g^{b'})$. In other words she cannot build message $\mathsf{sign}(\mathsf{sk}_A, (g^a, g^{b'}))$.

(b) Show that an active man-in-the-middle, Eve, can cause:

- A to think that she is communicating securely with B (as required),
- but B to think he is communicating securely with Eve.

In other words, B is fooled into thinking that the subsequent encrypted messages he is receiving (from A) are coming from Eve. Note that Eve cannot eavesdrop on the resulting encrypted channel.
Hint: Eve replaces the third message. You may assume that Eve also has a certificate, CERT_E, on her public signature verification key sk_E.

ANSWER: If Eve intercepts the third message in an honest execution of the protocol, and replaces it with the following message:

$$E,\ \mathsf{CERT}_E,\ \mathsf{sig}(\mathsf{sk}_E, (g^x, g^y))$$

which she can because she can obtain g^x and g^y from the first to messages of the session, then

- **A will think that she is communicating securely with B (as required),**
- **but B will think he is communicating securely with Eve.**

This is possible because in the first two messages g^x and g^y are not linked to A and B in a secure way.

(c) Describe how Eve can use this attack to steal money from A. For example, suppose A gives expert advice in a private chat room run by B, and that she gets paid for that.

ANSWER: Eve could also register as an expert on Bob's private chat to sell her advice. Then she could just relay to A the messages sent from B to her. A will accept these messages as coming from B for her and will reply with her advice. Now Eve, will intercept A's responses and relay them to B as if coming from herself and will get paid for the advice in place of A.

(d) Propose a way to fix the protocol to defend against this attack. Explain why your fix prevents the attack from Question 4(b).

ANSWER: To fix this problem, A and B need to link g^x and g^y to the two parties of this protocol. This could be achieved as follows

$$A \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad B$$

$$\xrightarrow{\qquad\qquad\qquad\qquad g^x \qquad\qquad\qquad\qquad}$$

$$\xleftarrow{g^y,\ B,\ \mathsf{CERT}_B,\ \mathsf{sig}(\mathsf{sk}_B,(g^x,g^y)),\ \mathsf{aenc}(g^x\cdot g^y,(A,B,g^x,g^y))}$$

$$\xrightarrow{A,\ \mathsf{CERT}_A,\ \mathsf{sig}(\mathsf{sk}_A,(g^x,g^y)),\ \mathsf{aenc}(g^x\cdot g^y,(A,B,g^x,g^y))}$$