Resit Exam — Security

February 28, 2017.   12:30 – 15:30

You can score a maximum of 100 points. Each question indicates how many points it is worth. You are NOT allowed to use books or notes, or a (smart) phone. You may answer in Dutch or in English. Read each exercise carefully (RTFE). Please write clearly, and don't forget to put your name and student number on each page that you hand in. You can keep this exam yourself.

1. (**25 points**) (PKI and protocol insight)
   This exercise considers a situation where a group of people, called voters, can cast a vote electronically using RSA. It is assumed that each participant X has a private key d_X and a corresponding certificate C_X containing the associated public key e_X and the participant's name X, signed by some Certificate Authority.

   We consider the following security properties for the voting process.

   (A) **secrecy**: no-one except voter V should be able to learn what V voted.

   (B) **single-vote**: no-one should be able to vote more than once.

   Votes are sent to a Polling Station PS that collects votes. It is assumed that connections do not reveal side-information (like IP addresses).

   (a) (**10 points**) Consider the following protocol.

$$V \longrightarrow PS \;:\; \{[\text{vote}]_{d_V}, C_V\}_{e_{PS}}$$

   Discuss briefly whether goals (A) and (B) are achieved: how or how not.

   Begin Secret Info: ....

   - *Goal (A) is **not** achieved since PS can see both the vote and the voter's identity V in the certificate C_V*
   - *Goal (B) is achieved when PS keeps a list of voter identities. Double occurrence of an identity indicates double voting.*

Grading (total 10):	
(A) no	*2*
correct explanation	*3*
(B) yes	*2*
correct explanation	*3*

   End Secret Info ....

   Now consider the following more complicated version, where voter V first gets a blind signature on a fresh nonce N.

$$V \longrightarrow PS \;:\; \{(PS|N)\cdot\{r\}_{e_{PS}}, C_V\}_{e_{PS}}$$

   where r is a random blinding factor, and | is concatenation

$$\begin{aligned} PS \longrightarrow V &\;:\; \{[PS|N]_{d_{PS}}\cdot r\}_{e_V} \\ V \longrightarrow PS &\;:\; \{\text{vote}, [PS|N]_{d_{PS}}\}_{e_{PS}} \end{aligned}$$

(b) (**5 points**) Explain what PS does after receiving the first message, in order to generate the signed name-and-nonce $[PS|N]_{d_{PS}}$ in the second message.

Begin Secret Info: ....

The Polling Station PS puts an RSA-signature on $N \cdot \{r\}_{e_{PS}}$ as in:

$$\begin{aligned}[(PS|N)\cdot\{r\}_{e_{PS}}]_{d_{PS}} &= \left((PS|N)\cdot r^{e_{PS}}\right)^{d_{PS}} \\ &= (PS|N)^{d_{PS}} \cdot r^{e_{PS}\cdot d_{PS}} \\ &= [PS|N]_{d_{PS}} \cdot r.\end{aligned}$$

Grading (total 5):	
PS must sign $(PS|N)\cdot\{r\}_{e_{PS}}$	*2*
correct explanation/calculation	*3*

End Secret Info ....

(c) (**10 points**) Briefly discuss whether goals (A) and (B) are achieved with this second protocol: how or how not.

Begin Secret Info: ....

- *Secrecy (A) is achieved because PS does not learn who sends in a vote*
- *Single-voting (B) is also achieved if PS only puts one blind signature per voter V. Hence PS must keep a record of V's in the first message.*

Grading (total 10):	
(A) yes	*2*
correct explanation	*3*
(B) yes	*2*
correct explanation	*3*

End Secret Info ....

2. (**22 points**) security protocols, symmetric key usage, hash functions

A typical organization runs various services a user can log in to, using the same account. A way to provide 'single sign-on' to these services is by using so-called 'tickets'. In such a system, a user requests a ticket from a central *ticket granting server* (TGS), and then uses this ticket to authenticate to a *service* ($S_1, S_2, S_3, \ldots$). In this question, we will use the following symmetric keys:

- K_{S_i} is a secret key shared between the TGS and service S_i (TGS has a list);
- K_A is shared between user Alice and the TGS;
- K_{AS_i} is a (session) key distributed to A and S_i, freshly generated by the TGS.

Protocol 1:

$$\begin{aligned} A &\longrightarrow TGS : A, S_i \\ &\qquad\qquad TGS \text{ generates a fresh } K_{AS_i} \\ TGS &\longrightarrow A \quad : K_A \oplus K_{AS_i}, \quad K_{S_i} \oplus K_{AS_i} (= ticket) \\ A &\longrightarrow S_i \quad : K_{AS_i}\{A\}(= authenticator), \quad K_{S_i} \oplus K_{AS_i} (= ticket) \end{aligned}$$

(a) (**5 points**) Protocol 1 contains a crucial flaw; show how Alice can obtain the secret key of a service.

Begin Secret Info: ....

Alice receives $K_A \oplus K_{AS_i}$ and $K_{S_i} \oplus K_{AS_i}$.
These can be used to obtain $(K_A \oplus K_{AS_i}) \oplus (K_{S_i} \oplus K_{AS_i}) = K_A \oplus K_{S_i}$.
Since Alice has K_A, she can compute $(K_A \oplus K_{S_i}) \oplus K_A = K_{S_i}$.

Grading (total 5):	
Cancel K_{AS_i} using XOR	*2*
Cancel K_A using XOR	*3*
Alternatively:	
Claim Alice has K_{AS_i}	*1*
Use this to obtain K_{S_i} from ticket	*3*

End Secret Info ....

Now consider Protocol 2, where we instead use some secure block cipher to encrypt the session key.

Protocol 2:

$$\begin{aligned} A &\longrightarrow TGS : A, S_i \\ &\qquad\qquad TGS \text{ generates a fresh } K_{AS_i} \\ TGS &\longrightarrow A \quad : K_A\{K_{AS_i}\}, \quad K_{S_i}\{K_{AS_i}\}(= ticket) \\ A &\longrightarrow S_i \quad : K_{AS_i}\{A\}(= authenticator), \quad K_{S_i}\{K_{AS_i}\}(= ticket) \end{aligned}$$

After completing the protocol, the service uses the session key to decrypt the *authenticator*. If that works, authentication is considered successful. To prevent replay attacks, assume each service stores every ticket it has seen, and will never accept the same ticket twice.

(b) (**7 points**) Assume Alice (A) starts using Protocol 2 to authenticate to service S_1. Show how an attacker Eve (E) can make Alice authenticate to S_2 instead.

Begin Secret Info: ....

$$\begin{aligned} A &\longrightarrow E(TGS) : A, S_1 \\ E(A) &\longrightarrow TGS \quad : A, S_2 \\ A &\longleftarrow TGS \quad : K_A\{K_{AS_2}\}, \quad K_{S_2}\{K_{AS_2}\}(= ticket) \\ A &\longrightarrow E(S_1) \quad : K_{AS_2}\{A\}(= authenticator), \quad K_{S_2}\{K_{AS_2}\}(= ticket) \\ E &\longrightarrow S_2 \quad : K_{AS_2}\{A\}(= authenticator), \quad K_{S_2}\{K_{AS_2}\}(= ticket) \end{aligned}$$

The main trick here is to get Alice to create a valid authenticator for S_2.

Grading (total 7):	
Get TGS to create $ticket$ using K_{S_2}	*3*
Show that Alice creates $authenticator$	*3*
Send $ticket$ and $authenticator$ to S_2	*1*

End Secret Info . . . .

(c) (**4 points**) Prevent the problem that is abused in (b).

Begin Secret Info: . . . .
This is prevented by limiting the authenticator to the intended service, i.e. include S_i: $authenticator = K_{AS_i}\{A, S_i\}$.

Grading (total 4):	
Show understanding that authenticator re-use is the problem	*2*
Correctly limit validity of the authenticator	*2*
Alternatively:	
Fix by encrypting S_i in first message	*3*

End Secret Info . . . .

We started by assuming that the *TGS* and each user share a secret key (e.g. K_A, for Alice). However, typically they only share a *password*. This is then used to derive a shared secret key. Assume this is done by computing $K_A = h(password_A)$, where h is some fast cryptographically secure hash function.

(d) (**6 points**) Assume Alice has not chosen a very strong password (e.g. only five characters) and an eavesdropper listens in on a run of Protocol 2. Show how an attacker can retrieve the password using an *offline attack*. Clearly state how the correct password can be recognized.

Begin Secret Info: . . . .
An attacker obtains $K_A\{K_{AS_i}\}$ and $K_{AS_i}\{A\}$ from the protocol run. They can now iterate over passwords to compute $K'_A = h(password'_A)$ for every attempt $password'_A$. For each attempt, they can try to decrypt $K_A\{K_{AS_i}\}$ to obtain K'_{AS_i}, and then use this to decrypt $K_{AS_i}\{A\}$. If this decryption succeeds and A is obtained, this indicates that the guess is correct, i.e. $password'_A = password_A$.

Grading (total 6):	
Iterate over passwords to compute candidates for K_A	*1*
Decrypt $K_A\{K_{AS_i}\}$ to obtain candidate for K_{AS_i}	*2*
Decrypt $K_{AS_i}\{A\}$ to obtain candidate for A	*2*
Indicate correctness when A decrypted correctly	*1*

End Secret Info . . . .

3. (**16 points**) (Diffie-Hellman computations and insight)
Alice and Bob will execute the Diffie-Hellman protocol. The domain parameters are modulus $p = 101$ and generator $g = 19$. Alice uses 39 as her secret number and Bob uses 48 as his secret number.

   (a) (**3 points**) What is the purpose of the Diffie-Hellman protocol, i.e., what do Alice and Bob want to achieve?

   Begin Secret Info: . . . .
   Share a secret key, nothing more and nothing less
   End Secret Info . . . .

   (b) (**3 points**) Describe the protocol with messages exchanged and computations executed on each side.

   Begin Secret Info: . . . .
   Alice sends her public number to Bob, Bob sends his public number to Alice, then both exponentiate the received public number with their own private number. The result at both sides is the shared secret.

   End Secret Info . . . .

   (c) (**4 points**) The basic Diffie-Hellman protocol is vulnerable to a (wo-)man-in-the-middle attack because Bob cannot verify that the message he receives actually comes from Alice and vice versa. Explain how to extend the protocol such that Bob can be sure that what he receives comes from Alice (we do not require the other way round in this question). Also clearly explain what must be set up for that.

   Begin Secret Info: . . . .
   Alice gets a long-term public key pair and signs with the private key the public number she sends to Bob and sends the signature along. She needs to get the public key to Bob in an authenticated way. They can agree on the phone, use mutual friends to sign it or even a CA, assuming trustworthy CA's exist. TOFU is not a valid answer.

Grading (total 4):	
signature with long-term public key pair	*2*
public key must be authenticated	*2*

   End Secret Info . . . .

   (d) (**3 points**) Compute Alice's public number that plays a role in the protocol

   Begin Secret Info: . . . .
   $19^{39} = 71$

Square and multiply:

$$\begin{aligned}19^2 &= 58\\ 19^4 &= 31\\ 19^8 &= 52\\ 19^9 &= 52 \times 19 = 79\\ 19^{18} &= 79^2 = 80\\ 19^{19} &= 80 \times 19 = 5\\ 19^{38} &= 5^2 = 25\\ 19^{39} &= 25 \times 19 = 71\end{aligned}$$

1 point for square-and-multiply and 2 points for correct result.
End Secret Info . . . .

(e) (**3 points**) Do the computation Bob will do at the end of the protocol and say what it is that he has computed

Begin Secret Info: . . . .
He has computed the shared secret and that is $71^{48} = 56$
Square and multiply:

$$\begin{aligned}71^2 &= 92\\ 71^3 &= 92 \times 71 = 68\\ 71^6 &= 68^2 = 79\\ 71^{12} &= 79^2 = 80\\ 71^{24} &= 80^2 = 37\\ 71^{48} &= 37^2 = 56\end{aligned}$$

1 point for shared secret, 1 point for square-and-multiply and 1 points for correct result.
End Secret Info . . . .

4. (**15 points**) RSA system, modular arithmetic
Take $p = 11$ and $q = 19$ as prime numbers for RSA.

(a) (**4 points**) Compute the modulus n and the result $\varphi(n)$ of applying the Euler function φ to n.

Begin Secret Info: . . . .

$n = p \cdot q = 11 \cdot 19 = 209$, $\varphi(n) = (p-1) \cdot (q-1) = 10 \cdot 18 = 180.$

Grading (total 4):

aspect:	*points*
Correct formula for n and $\varphi(n)$	*2*
Correct computation	*2*

End Secret Info . . . .

(b) (**6 points**) Assume we choose as public exponent $e = 17$. Find the corresponding private exponent d. Describe your computations.

Begin Secret Info: ....

We need to invert $e \mod \varphi(n)$, i.e. *find* d *such that* $d \cdot 17 \equiv 1 \mod 180$.

We do this with the extended Euclidean algorithm.

$180 - 11 \cdot 17 = -7$

$17 - 2 \cdot 7 = 3$

$7 - 2 \cdot 3 = 1$

So $1 = 7 - 2 \cdot (17 - 2 \cdot 7) = 5 \cdot 7 - 2 \cdot 17$, *further*

$1 = 5 \cdot (11.17 - 180) - 2 \cdot 17 = 53 \cdot 17 - 5 \cdot 180$

So the public exponent is 53.

Grading (total 6):	
aspect:	*points*
Using extended Euclidean algorithm	*3*
correct computation and value of d	*3*

End Secret Info ....

Assume we have found an USB stick with someone's RSA private key on it that was stored as p, q and d. We don't know whose key it is and to find out we wish to find the corresponding modulus and public exponent. After doing that we will be able to compare it with public keys we can find on the web.

(c) (**5 points**) The private key data on the USB stick is $p = 23$, $q = 19$ and private exponent 317. Explain how to compute the modulus and especially the public exponent and then do the actual computations.

Begin Secret Info: ....

Modulus is just taking the product $23 \cdot 19 = 437$

We need to invert $d \mod \varphi(n)$, i.e. *find* e *such that* $e \cdot 317 \equiv 1 \mod \varphi(n)$. *We do this with the extended Euclidean algorithm.*

We have $\varphi(n) = (23 - 1) \cdot (19 - 1) = 396$

$396 - 317 = 79$

$317 - 4 \cdot 79 = 1$

So $1 = 317 - 4 \cdot (396 - 317) = 5 \cdot 317 - 4 \cdot 396$,

So the public exponent is 5.

Grading (total 5):

aspect:	*points*
Knowing that computing public from private exponent is same as vice versa	*3*
correct computation and values of n and e	*2*

End Secret Info . . . .

5. (**22 points**) (hash properties, hash usage, block cipher modes)
Consider a company A-Corp, which has a database with user records. These records include password data. Furthermore, users can set a password hint themselves. This hint can be shown to users to remind them of their password. A-Corp was hacked and the database was made public, containing the columns `User`, `Password Data` and `Hint`, as shown in the table on the next page. This question is about what could be in the `Password Data` column.

   (a) (**4 points**) After the hack, A-Corp sent an email to its customers. In this email it stated the following:

   > *"The attackers may have obtained access to your username and encrypted password."*

   If this statement is true, what is A-Corp doing wrong in the way they store the password? Explain why that is a problem.

   Begin Secret Info: . . . .
   Passwords should be hashed. When only encryption is used, and the encryption key gets compromised, all passwords will get exposed.

   Grading (total 4):

Hash passwords	*1*
correct explanation	*3*

   End Secret Info . . . .

Many users use insecure passwords. Since the database also includes password hints, for some users we can easily guess the password. On the next page, we display an extract of the leaked database. The table contains password data and password hints for a number of users with easily guessable passwords. Next to it is a column that was added by us. It contains the passwords we can guess, based on the password hints.

`User`	`Password Data`	`Hint`	*Password guess*
Alice	`0581c5efa4097c14`	numbers 1-6	123456
Bobby	`0d136609370da3aa 20f990b0e3f0980b`	2*name+1996	bobbybobby1996
Claire96	`20f990b0e3f0980b`	birth year	1996
Dave	`42eaa074b069b227 2b713b4cca277edc`	9 down	987654321
ErinB	`2fca9b003de39778 2b713b4cca277edc`	password + 1	password1
Frank	`2fca9b003de39778 2fca9b003de39778`	password*2	passwordpassword
Gerald	`e5b172997af51dd1 13f43e1a74dd0e98 1433f6b89ca2a82d 08e1b5be844b0abb`	XKCD 936	correcthorse-batterystaple

Longer passwords give longer encryptions, but the length of the encrypted password is always a multiple of a fixed number (8 bytes in this case).

(b) (**5 points**) What mode of encryption is used? Explain your answer.

Begin Secret Info: ....
ECB, since same blocks of input give the same output.

Grading (total 5):	
ECB	*2*
Correct explanation	*3*

End Secret Info ....

(c) (**3 points**) What is the problem with this mode?

Begin Secret Info: ....
Patterns can remain visible when ECB is used. In this specific case, we can detect for different passwords that parts of them are the same.

Grading (total 3):	
Correct explanation	*3*

End Secret Info ....

(d) (**10 points**) A different company, called Bee Inc., also has a database with user records. Bee Inc. hashes their passwords. Again, we can guess passwords using the password hints:

User	Password Data	Hint	*Password guess*
Arnold	`110edf2294fb8bf4`	numbers 123456	
Bart	`110edf2294fb8bf4`	==123456	123456
Charlie	`110edf2294fb8bf4`	c'est “123456”	
Denise	`e5d8efed9088db0b`	double password	passwordpassword
Erica	`e5d8efed9088db0b`	password twice	
Francine	`2fca9b003de39778`	the password is password	password
Gabe	`2fca9b003de39778`	rhymes with assword	
Harry	`ecba98cca55eabc2`	sixxone	
Iris	`ecba98cca55eabc2`	1*6	111111
Jessica	`ecba98cca55eabc2`	sixones	

We see that an important technique for securely storing passwords is not used.

i. What is the name of this technique?
ii. How does it work?
iii. Why is it important to use this technique?

Begin Secret Info: . . . .

i. Salting should be used.
ii. For different users a different value (the salt) should be hashed together with the password.
iii. Without salting it is possible to hash password guesses and check which records match. With salting guesses must be hashed for every record, increasing the amount of work an attacker needs to do to find passwords.

Grading (total 10):	
Use salting	*3*
Hash different salt with each password	*4*
Reason for salting	*3*

End Secret Info . . . .