Applied Cryptography

Public Key Cryptography, Assignment 4, Wednesday May 11, 2022

Exercises with answers and grading.

1. **(15 points)** In the lecture post quantum cryptography was introduced. Using your own words, and with the help of the slides and the internet answer the following:
   (a) What is necessary for a cryptosystem to be called post-quantum?
   (b) Why are we interested in post-quantum cryptosystems?
   (c) What are the main advantages of post-quantum cryptography as opposed to quantum cryptography?
   (d) Assume a quantum adversary that is in possession of a large universal quantum computer. How much time, in processing time, does he need to break a password of length 10, uniformly chosen from the set of all passwords containing any letter A-z and any special character, but no numerical characters 0-9?
   (e) Answer the previous question in the case the adversary is not in possession of a quantum computer.
   (f) What is the required length of the keys of a symmetric cryptosystem against quantum adversaries for 128 bits of security?
   (g) What is the required length of RSA keys against quantum adversaries for 128 bits of security?
   (h) Define one of the hard problems: LWE, MQ, Syndrome decoding and explain how it can be used to construct a public key cryptosystem. Use at most 250-300 words

Begin Secret Info:.....

   (a) The complexity of breaking a post-quantum cryptosystem needs to be modelled assuming both a classical and a quantum adversary. This means that quantum speed-ups coming from quantum algorithms must be taken into account when targeting a certain number of bits of security.
   (b) We know that most cryptosystems currently in use today rely on the hardness of integer factorization and discrete logarithms in finite groups. Both are efficiently solved by Shor's algorithm and therefore *not* post-quantum. Assuming a (near) future with quantum computers, we will need to use post-quantum cryptosystems if we want secure public key cryptography.
   (c) Whereas quantum cryptography requires devices to run over quantum channels and therefore use quantum hardware, post-quantum cryptography relies only on classical channels and classical hardware. This means we do not have to upgrade every possible device to (presumably expensive) quantum hardware, but only have to upgrade the software on these devices to post-quantum cryptosystems. Furthermore, for hardware acceleration purposes, we can already work in advance to provide these for systems that run on classical hardware. Also, there are a number of open problems in quantum cryptography that are not yet solved, and the current solution still requires classical channels. So, just switching to quantum cryptography still requires using classical channels for security.
   (d) Assuming 83 characters in total, give or take, there are 10^{83} possible passwords. If one quantum query takes x seconds, then in total we need $\left(10^{83}\right)^{1/2} \cdot x$ seconds because of the quantum speed-up by Grover's algorithm. Any reasonable explanation of why a query takes x seconds is okay.

(e) Assuming 83 characters in total, give or take, there are 10^{83} possible passwords. If one query takes y seconds, then in total we need $10^{83} \cdot y$ seconds to check all posibilities. Any reasonable explanation of why a query takes y seconds is okay.

(f) By Grover's speed-up, the keys need to be twice as long to provide 128 bits of quantum security. For example, AES-256 has a security of 2^{256} classicaly, but with Grover this is reduced to 2^{128}. Hence, AES keys would need a length of 256 bits.

(g) Any reasonable explanation using Shor's complexity of $\approx \mathcal{O}(n^3)$ is correct. So, assuming factorization of an n-bit number takes roughly n^3 operations, we want $n^3 = 2^{128}$ and so $n = 6981463658332$ bits. This is a bit too much for usability.

(h) Any reasonable solution is accepted.

End Secret Info .....

2. **(20 points)** Consider the following generalization of one time signatures:

- **Key generation:**
  - Generate a pair of secret and public key $(\mathsf{sk}_1, \mathsf{pk}_1)$ using Lamport's OTS for 256-bit messages and which uses SHA-256
  - Set a state to $S = ()$ and $\sigma_0 = ()$
- **Signing:** To sign a message M_i, $i = 1, 2, \ldots$ the signer
  - Generates a new key pair $(\mathsf{sk}_{i+1}, \mathsf{pk}_{i+1})$
  - Computes a signature $\sigma_i = (M_i, \mathsf{pk}_{i+1}, \mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i, \mathsf{pk}_{i+1})), \sigma_{i-1})$ where $\mathsf{Sign}_{\mathsf{sk}_i}$ is the signing algorithm of Lamport OTS using the secret key sk_i.
  - Add $(M_i, \mathsf{pk}_{i+1}, \mathsf{sk}_{i+1}, \mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i, \mathsf{pk}_{i+1})))$ to the state S
- **Verification:** To verify the signature $\sigma_i = (M_i, \mathsf{pk}_{i+1}, \sigma_{i,OTS}, \sigma_{i-1})$
  - Check $\mathsf{Vf}_{\mathsf{pk}_j}(M_j, \mathsf{pk}_{j+1}, \sigma_{j,OTS}) = 1$ for all $j \in \{1, 2, \ldots, i\}$

(a) What should be the length of the output of H_0?

(b) How long is the signature of the 12-th message? Assume the length of the i-th message is L_{M_i}.

(c) What is the advantage of this scheme compared to MSS introduced in the lectures? What is the disadvantage?

(d) Show that a forgery is possible if instead of $\mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i, \mathsf{pk}_{i+1}))$ the signer includes $\mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i))$

(e) Show that a forgery is possible, if in the signature generation of σ_i, we omit σ_{i-1}, and in the verification process we set $j \in \{i\}$, and provide the OTS public key pk_i together with the signature.

(f) Show that a forgery is possible if the adversary is able to find second preimages for H_0

(g) Can you think of a way to improve the efficiency of the scheme using Merkle trees? If yes, please describe the solution in detail, with a justification of the improved efficiency. Different solutions are possible, and will be accepted, provided there isn't an obvious security flaw.

Begin Secret Info: .....

(a) The Lamport OTS keys $(\mathsf{sk}_i, \mathsf{pk}_i)$ sign 256-bit messages, so the output of H_0 should be 256 bits.

(b) We sum each individual component for the length of the $i-$th message: the length of M_i is L_{M_i}, the length of pk_{i+1} is $2 * 256 * 256$ and the length of $\sigma_{i,OTS}$ is $256 * 256$ again. Then, write L_{i-1} for the length of the previous message. Then, $L_i = L(M_i) + L(\mathsf{pk}_{i+1}) + L(\sigma_{i,OTS}) + L_{i-1} = L_{M_i} + L_{i-1} + 3 * 256 * 256 + 256 * 256$. With $L_0 = 0$, we get $L(M_{12}) = \sum_{i=0}^{12} L_{M_i} + 3 * 2^{16}$

(c) Advantage: You can sign as many messages as you want. Disadvantage: The size of the signature grows with each message

(d) Just change the public key. There is no link that connects the new public key to the signature. The forgery is: $\sigma_i = (M_i, \mathsf{pk}'_{i+1}, \mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i)), \sigma_{i-1})$

(e) There is no link to the public key of the signature scheme pk, so the adversary can create his own $(\mathsf{pk}'_i, \mathsf{sk}'_i)$, and use them to create a signature $\sigma_i = (M_i, \mathsf{pk}_{i+1}, \mathsf{Sign}_{\mathsf{sk}'_i}(H_0(M_i, \mathsf{pk}_{i+1})))$

(f) Then the adversary can use $\sigma_i = (M'_i, \mathsf{pk}'_{i+1}, \mathsf{Sign}_{\mathsf{sk}_i}(H_0(M_i, \mathsf{pk}_{i+1})), \sigma_{i-1})$ for which $H_0(M_i, \mathsf{pk}_{i+1}) = H_0(M'_i, \mathsf{pk}'_{i+1})$

(g) A solution is accepted when it uses Merkle trees and explains in detail how the efficiency is improved.

End Secret Info .....

3. **(15 points)**

(a) Provide concrete plausible practical examples for the two identity misbinding attacks (Attack 1 and 2) from the lectures.

(b) Show in detail that SIGMA-I indeed prevents the two identity misbinding attacks (Attack 1 and 2) from the lectures.

(c) Recall the ISO 9796 protocol from the lectures, in which the identity of the receiver was included in the signatures σ_A, σ_B to prevent an identity misbinding attack.

Alice's client		Bob's server
$P, G, \mathsf{pk}_B, a, \mathsf{sk}_A$		$P, G, \mathsf{pk}_A, b, \mathsf{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$	
$\mathsf{Vf}_{\mathsf{pk}_B}(\sigma_B)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$B \leftarrow G^b$, $\sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B, Alice)$
$K_{A,B} \leftarrow B^a$		$K_{B,A} \leftarrow A^b$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A, Bob)$	$\xrightarrow{\sigma_A}$	$\mathsf{Vf}_{\mathsf{pk}_A}(\sigma_A)$

Show in detail (i.e. describing an attack) why including the identity of the sender in the signatures does not prevent identity misbinding attacks.

(d) Assume the protocol uses RSA signatures. Because the idea from (a) does not work, the designers decided to include the shared key $K_{A,B} = K_{B,A}$ in the signatures σ_A, σ_B instead of the identities. Does this idea prevent identity misbinding attacks? Explain why. (Hint: No, it doesn't. Please show an attack.) Does removing the identities sent in clear change the situation?

Begin Secret Info: .....

(a) The examples should fit the attacks. Anything similar to the examples given in the lecture (but not the same) is enough. Of course, it is best to have examples that are practically relevant.

(b) In attack 1, Eve would not be able to properly form the second message that should be B, C_E, where C_E needs to be formed using unknown keys for Eve, but should contain Eve's identity.

In attack 2, the same holds but for the third message.

(c) The attacks are the following:

Attack 1:

Alice's client		Eve		Bob's server
$P, G, \mathsf{pk}_B, \mathsf{pk}_E,$ a, sk_A		$P, G, \mathsf{pk}_A, \mathsf{pk}_B,$ e, sk_E		$P, G, \mathsf{pk}_A, \mathsf{pk}_E,$ b, sk_B
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$		$\xrightarrow{\text{Alice; } A}$	
				$B \leftarrow G^b$
$\mathsf{Vf}_{\mathsf{pk}_E}(\sigma_E)$	$\xleftarrow{\text{Eve; } B;\ \sigma_E}$	$\sigma_E = \mathsf{Sign}_{\mathsf{sk}_E}(A, B, Eve)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$\sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B, Bob)$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A, Alice)$	$\xrightarrow{\sigma_A}$		$\xrightarrow{\sigma_A}$	$\mathsf{Vf}_{\mathsf{pk}_A}(\sigma_A)$
$K_{A,B} \leftarrow B^a$				$K_{B,A} \leftarrow A^b$

Attack 2:

Alice's client		Eve		Bob's server
$P, G, \mathsf{pk}_B, \mathsf{pk}_E,$ a, sk_A		$P, G, \mathsf{pk}_A, \mathsf{pk}_B,$ e, sk_E		$P, G, \mathsf{pk}_A, \mathsf{pk}_E,$ b, sk_B
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$		$\xrightarrow{\text{Eve; } A}$	
				$B \leftarrow G^b$
$\mathsf{Vf}_{\mathsf{pk}_B}(\sigma_B)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$		$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$\sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B, Bob)$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A, Alice)$	$\xrightarrow{\sigma_A}$	$\sigma_E = \mathsf{Sign}_{\mathsf{sk}_E}(B, A, Eve)$	$\xrightarrow{\sigma_E}$	$\mathsf{Vf}_{\mathsf{pk}_E}(\sigma_E)$
$K_{A,B} \leftarrow B^a$				$K_{B,A} \leftarrow A^b$

(d) Actually any hash-then-sign signature will produce the same effect. In principle, Eve needs to sign the unknown shared key, and this looks impossible. But using the public key she can compute $H(A, B, K_{A,B})$, and this is enough to the produce a signature!

End Secret Info .....