Applied Cryptography

Public Key Cryptography, Assignment 6, Monday, May 13, 2024

Exercises with answers and grading.

1. **(25 points)** In the last 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, $\mathcal{D}$, that is in possession of a large universal quantum computer. How much processing time does $\mathcal{D}$ need to break a password of length 10 character characters, 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 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 if we want to achieve 128 bits of security?

   (g) What is the required length of RSA keys against quantum adversaries if we want to achieve 128 bits of security?

   (h) Choose of the of the following problems: LWE, MQ or Syndrome decoding. State the definition of the problem you chose 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 possibilities. 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} classically, 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. **(25 points)** In the lecture, we have seen SIGMA-protocols and how these prevent identity misbinding attacks.

   (a) Provide two 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. describe 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 not. Does removing the identities sent in the clear change the situation?

Begin Secret Info:..

(a) *There are many examples, a couple of them are already in the lecture notes (Alice buys Bob's house thinking it is Eve's house).*

(b) *In SIGMA-1 the verification of either identity is verified by using a MAC and encrypted with a key that Eve cannot get. Eve can try to replace the identity, but will never be able to make the tag and the ciphertext match.*

(c) *Eve can also include the identity of Bob (for attack 1, and Alice for attack 2) in it's own signature.*

(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 . . . .