Applied Cryptography

Guest Lectures, Assignment 7, Monday, May 27, 2024

Exercises with answers and grading.

1. **(15 point)** In the first guest lecture given by Benoît Viguier, we took a look at the management of certificates for TLS and certificate authorities (CAs).

   (a) "Let's Encrypt" allows for easy issuing of certificates. Before issuing such certificate, some verifications need to be done to guarantee security. What are those verifications?

   (b) Name the protocol used by "Let's Encrypt" to perform the verifications of (1a).

   (c) Give a brief description in your own words of how the protocol in (1b) works.

   (d) Name one of the advantages of a multi-tier certificate authority.

   (e) Certificate Revocation Lists (CRLs) are part of the TLS1.3 protocol, name the biggest inconvenience with CRLs.

   (f) Name at least two problems solved by the Online Certificate Status Protocol (OCSP).

   Begin Secret Info:...

   (a) First, verify the ownership of the secret key and then verify the ownership of the domain.

   (b) "Let's Encrypt" uses the ACME protocol.

   (c)
   i. User requests a certificate.
   ii. CA requests proof of ownership of private key by asking to sign a nonce, and placing a random value in the DNS/some random path on the server.
   iii. User sign the nonce, provide CSR and set the value in the DNS.
   iv. CA validates DNS and ownership of private key.
   v. CA returns Certificates.

   (d)
   i. Offline root increases the security.
   ii. If sub-CA is compromised, it can be revoked
   iii. A single root CA can have multiple sub CA for different applications.

   (e) The list could be a potentially large file.

   (f)
   i. It increases server load, enough that it has to be considered.
   ii. It presents privacy problems for the user.
   iii. It increases latency considerably.

   End Secret Info...

2. **(35 points)** In the lecture and in particular the guest lecture by Thom Wiggers, we took a close look at the TLS 1.3 protocol. In the future, such a protocol needs to be adapted to be quantum-secure. In this exercise, we go over several difficulties that might arise when adapting TLS to be quantum-proof.

   Stripped down to its cryptographical components, the TLS 1.3 protocol can be given as follows, where data implies application data encrypted with a key derived from some KDF.

Alice's client		Bob's server
P, G		$P, G, \mathsf{pk}_B, \mathsf{sk}_B$
$x \xleftarrow{\$} \mathbb{Z}_q$, $X \leftarrow G^x$		
	$\xrightarrow{X}$	
		$y \xleftarrow{\$} \mathbb{Z}_q$, $Y \leftarrow G^y$ $\mathsf{ss} \leftarrow X^y$ $K \leftarrow \mathsf{KDF}(\mathsf{ss})$ $\sigma \leftarrow \mathsf{AE}_K($ $\mathsf{cert}_{\mathsf{pk}_B}$ $\mathsf{Sign}(\mathsf{sk}_B, \mathsf{transcript})$ $\mathsf{key\_conf}$ $)$
	$\xleftarrow{Y,\sigma}$	
$\mathsf{ss} \leftarrow Y^x$ $K, K', K'' \leftarrow \mathsf{KDF}(\mathsf{ss})$ $\mathsf{Verif}_K(\sigma)$ $\sigma' \leftarrow \mathsf{AE}_{K'}(\mathsf{key\_conf})$		
	$\xrightarrow{\sigma',\ \mathsf{AE}_{K''}(\mathsf{data})}$	
	$\xleftrightarrow{\mathsf{data}}$	

   (a) Explain what Alice needs to do in $\mathsf{Verif}(\sigma)$

   In post-quantum cryptography, Diffie-Hellman key exchange cannot be used (there is currently no post-quantum alternative), and so we resort to using key encapsulation mechanisms (KEMs), as we have seen in lecture 4.

   (b) Explain which steps in this protocol need to be updated to achieve post-quantum security. In particular, describe the specific functions that need to be replaced by KEMs and post-quantum signatures.

   (c) Use Kyber-768[1] as your key encapsulation mechanism and pick a post-quantum signature algorithm selected by NIST that you think fits, based on size and performance[2]. Assume 'pre-quantum' TLS 1.3 using X25519 for key exchange and RSA-2048 for signatures. Compare both size of transmitted data and clock cycles for the asymmetric cryptography data of this 'pre-quantum' TLS 1.3 and your 'post-quantum' TLS 1.3.

   (d) Based on the increases you get, what kind of problems do you expect when naïvely switching to post-quantum TLS?

   Alternatively, researchers have attempted to come up with modified versions of TLS that are less problematic in a post-quantum context. One of these is KEMTLS[3]. We can give the following schematic description.

[1] `https://pq-crystals.org/kyber/index.shtml`

[2] See an overview by Bas Westerbaan at `https://blog.cloudflare.com/nist-post-quantum-surprise/`

[3] Introduced in `https://eprint.iacr.org/2020/534.pdf`

Alice's client		Bob's server
P, G		$P, G, \mathsf{pk}_B, \mathsf{sk}_B$
$(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KEM.Keygen}()$		
	$\xrightarrow{\mathsf{pk}_e}$	
		$(\mathsf{ss}_e, \mathsf{ct}_e) \leftarrow \mathsf{KEM.Encaps}(\mathsf{pk}_e)$ $K_1, K_1' \leftarrow \mathsf{KDF}(\mathsf{ss}_e)$ $\sigma \leftarrow \mathsf{AE}_{K_1}(\mathsf{cert}_{\mathsf{pk}_B})$
	$\xleftarrow{\mathsf{ct}_e, \sigma}$	
$\mathsf{ss}_e \leftarrow \mathsf{KEM.Decaps}(\mathsf{ct}_e, \mathsf{sk}_e)$ $K_1, K_1' \leftarrow \mathsf{KDF}(\mathsf{ss}_e)$ $(\mathsf{ss}_B, \mathsf{ct}_B) \leftarrow \mathsf{KEM.Encaps}(\mathsf{pk}_B)$ $\sigma' \leftarrow \mathsf{AE}_{K_1'}(\mathsf{ct}_B)$		
	$\xrightarrow{\sigma'}$	
		$\mathsf{ss}_B \leftarrow \mathsf{KEM.Decaps}(\mathsf{ct_B}, \mathsf{sk_B})$
$K_2, K_2', K_2'' \leftarrow \mathsf{KDF}(\mathsf{ss}_e \mid\mid \mathsf{ss}_B)$		$K_2, K_2', K_2'' \leftarrow \mathsf{KDF}(\mathsf{ss}_e \mid\mid \mathsf{ss}_B)$
	$\xrightarrow{\mathsf{AE}_{K_2}(\mathsf{key\_conf}), \mathsf{data}}$	
	$\xleftarrow{\mathsf{AE}_{K_2'}(\mathsf{key\_conf}), \mathsf{data}}$	
	$\xleftrightarrow{\mathsf{data}}$	

(e) On what client-to-server message can Alice send encrypted application data in KEMTLS?

(f) Using the same schemes you picked before, calculate the size of transmitted data using KEMTLS.

(g) What are advantages of using KEMTLS compared to TLS, and what are the disadvantages?

(h) TLS has a long history of vulnerability to padding oracle attacks. Describe on a high level the idea behind these attacks.

(i) Is KEMTLS vulnerable to a padding oracle attack? Justify your answer.

(j) KEMTLS at first sight appears to completely avoid the usage of digital signatures. This is however not the case. Where and for what purpose are digital signatures used in KEMTLS? How many signatures in use can you count and which are they?

Begin Secret Info: ....

(a) Alice verifies σ using the K that she derives, and then needs to check that the certificate is correct to ensure pk_B *really belongs to Bob. Knowing this, she can verify the signature of the transcript. Last, she checks if her keys agree with the key confirmation.*

(b) The first steps need to be replaced by a KEM, but also we need post-quantum signatures for both the certificate and the signature of the transcript!

(c) This depends on the signature that is chosen, but it should be quite obvious that this grows the size a lot.

(d) Bas gave some initial hints in his lecture: the problem is that these objects become too big for packet sizes and this can cause a lot of errors in relays and hops. Some boxes in the middle of a TLS exchange do not expect or cannot handle this sudden increase in size and your connections can break. Also, some specific devices, such as smartcards or embedded devices, might not be calculated for larger TLS, and they wouldn't be able to perform handshakes anymore.

(e) On her second message already, as she gets anything from Bob that she needs to send authenticated encrypted application data: she has K_2, K_2', K_2'' given just ct_e *and σ.*

(f) *This should be consistent with the previous answer, but with fewer signatures.*

(g) *An advantage is that we rely less on signatures: they appear only in the certificates and no longer in the protocol itself, they have been replaced by KEMs. Post-quantum KEMs perform much better than post-quantum signatures. A downside is that we lose the non-interactive nature we had thanks to Diffie-Hellman: with DH, either party can compute the shared secret given just a public key, whereas in a KEM we are restricted to an encapsulation decapsulation order.*

(h) *Given a wrong padding, TLS gave out a warning that the padding was incorrect, which gave information on the secret. In this way, by checking if the padding was wrong or right, it was possible to learn the secret bit-by-bit.*

(i) *The design of KEMTLS prevents padding oracle attacks.*

(j) *KEMTLS still uses signatures in the PKI, so the signatures in here are in the certificate that is sent by Bob. Such a certificate usually holds between 1 and 3 signatures, as it also contains the certificate for CAs (for root certificates it is likely to be on the device already.)*

End Secret Info . . . .