Applied Cryptography

Symmetric Cryptography, Assignment 5, Wednesday, May 25, 2022

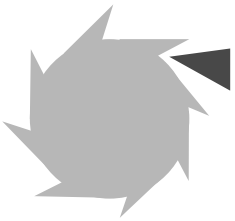
Exercises with answers and grading.

1. **(20 points)** In the lecture of May 18th, Sofia talked (a.o.) about secure communication, and the role of forward and backward security in secure communication. This question is about how the Signal protocol achieves these properties. We start with a warm up.

   (a) What is forward secrecy, and what is backward secrecy?

   (b) Under which condition does a Diffie-Hellman key exchange provide forward secrecy?

   The Signal protocol (`https://signal.org/docs/`) uses the so-called Double Ratchet algorithm (`https://signal.org/docs/specifications/doubleratchet/`), whose main goal is to provide forward secrecy. The name comes from a mechanical device "ratchet" that performs circular movement only in one direction. Similarly to this device, the Double Ratchet algorithm does not allow going back in a chain of derived keys.

   We start first with a single ratchet protocol. Assume Alice and Bob already share a secret key K, and use this to derive chain keys CK and message keys MK, leading to the following naive protocol to send encrypted messages. Here, KDF is a key-derivation function, Enc an encryption algorithm and Dec the associated decryption algorithm.

Alice, knows: K		**Bob**, knows: K
Alice sending message		
$CK_1^s, MK_1 \leftarrow \text{KDF}(K)$		
$N \leftarrow 1$		
$c_1 \leftarrow \text{Enc}(MK_1, m_1)$		
	$\xrightarrow{c_1, N}$	
		$CK_1^r, MK_1 \leftarrow \text{KDF}(K)$
		$m_1 \leftarrow \text{Dec}(MK_1, c_1)$
Bob sending 2 messages back		
		$CK_2^r, MK_2 \leftarrow \text{KDF}(CK_1^r)$
		$N \leftarrow 2$
		$c_2 \leftarrow \text{Enc}(MK_2, m_2)$
	$\xleftarrow{c_2, N}$	
		$CK_3^r, MK_3 \leftarrow \text{KDF}(CK_2^r)$
		$N \leftarrow 3$
	$\xleftarrow{c_3, N}$	$c_3 \leftarrow \text{Enc}(MK_3, m_3)$

   (c) Does this provide forward secrecy when an MK_i is leaked? What about backward secrecy?

(d) Assume leakage of CK_i at a certain moment in time. What can an attacker do with CK_i?

(e) Assume Alice and Bob have a lively conversation using the above protocol. What might go wrong with the counter N?

Considering the vulnerabilities of the naive protocol, let's explore how the double ratchet improves security. The following protocol describes a simplified view of the Double Ratchet algorithm as used in the Signal protocol. Capital letters A_i and B_i indicate public keys for Diffie-Hellman, with corresponding private keys a_i and b_i.

Alice, knows: K, B_0		**Bob**, knows: K, b_0
Alice sending message		
$a_0 \xleftarrow{\$} \mathbb{Z}_q, A_0 \leftarrow g^{a_0} \pmod p$ $SS_0 = B_0^{a_0} \pmod p$ $RK_0, CK_0^s \leftarrow \text{KDF}(K, SS_0)$ $CK_1^s, MK_1 \leftarrow \text{KDF}(CK_0^s)$ $N \leftarrow 1$ $c_1 \leftarrow \text{Enc}(MK_1, m_1)$	$\xrightarrow{c_1, N, A_0}$	$SS_0' \leftarrow A_0^{b_0} \pmod p$ $RK_0, CK_0^r \leftarrow \text{KDF}(K, SS_0')$ $CK_1^r, MK_1 \leftarrow \text{KDF}(CK_0^r)$ $m_1 \leftarrow \text{Dec}(MK_1, c_1)$
Bob sending 2 messages back		
		$b_1 \xleftarrow{\$} \mathbb{Z}_q, B_1 \leftarrow g^{b_1} \pmod p$ $SS_1 \leftarrow A_0^{b_1} \pmod p$ $RK_1, CK_0^r \leftarrow \text{KDF}(RK_0, SS_1)$ $CK_1^r, MK_1 \leftarrow \text{KDF}(CK_0^r)$ $N \leftarrow 1$ $c_2 \leftarrow \text{Enc}(MK_1, m_2)$
	$\xleftarrow{c_2, N, B_1}$	
		$CK_2^r, MK_2 \leftarrow \text{KDF}(CK_1^r)$ $N \leftarrow 2$ $c_3 \leftarrow \text{Enc}(MK_2, m_3)$
	$\xleftarrow{c_3, N, B_1}$	

(f) Explain the steps Alice performs to decrypt the messages. What is the role of RK_i, CK_i and MK_i in these messages?

(g) Describe, informally, what steps Alice needs to perform to send a message back to Bob.

(h) Explain what an attacker learns from the leakage of: i) RK_i, ii) CK_i, iii) MK_i.

(i) Identify the first ratchet from the naive protocol. What is the second ratchet?

(j) If, instead of using $RK_i, CK_0 \leftarrow \text{KDF}(RK_{i-1}, SS_j)$ we used $RK_i, MK_1 \leftarrow \text{KDF}(RK_{i-1}, SS_j)$, and we skip the step $CK_1, MK_1 \leftarrow \text{KDF}(CK_0)$. To encrypt a second message, we would simply compute $MK_2 \leftarrow \text{KDF}(MK_1)$. Is this less secure? What do we lose?

Begin Secret Info:

(a) ***Forward secrecy:*** *If a long-term secret leaks, messages sent before the leakage are still secure.* ***Backward secrecy:*** *The opposite, even though a long-term secret leaks, future sessions are still secure.*

(b) *When ephemeral keys are used per session: even when Alice's or Bob's long term secrets are leaked, the session before this leakage can still be considered secure. (After leakage, you can no longer be certain that your session really is with Alice or Bob.)*

(c) *Leakage of MK_i only allows an attacker to decrypt message m_i, and does not allow him to derive earlier or later chainkeys CK_i. So, both forward and backward secrecy are ok.*

(d) *Leakage of CK_i breaks backward secrecy, forward secrecy is still ok.*

(e) *It is hard to keep the counter in sync between Alice and Bob, and this causes mix-ups of the protocol. However, CK_N can always be derived on either side and N is send along so decryption of the message is still possible, but multiple messages may be encrypted with the same key MK_N.*

(f) *First, update Bob's public key to the new version, then compute the shared secret. Use this to update the root key RK_i and derive the chain key CK_i. The chain key CK_i is used to derive the message key MK_i for the i-th message in the chain as well as to update the chain key to CK_{i+1}. The message key is then used the decrypt the message.*

(g) *Generate a new pair (a_i, A_i), and derive the secret key using the latest public key Bob has sent. Use this to update the root key, and derive a chain key and message key. Then encrypt the message using the message key.*

(h) *i) Leakage of RK_i breaks backwards secrecy until either sides generates a new ephemeral Diffie-Helmann key. This is also called the "self-healing" property of the Double-Ratchet algorithm. ii) Leakage of CK_i allows an attacker to decreypt all the messages m_j in that session with $j \geq i$, as the attacker can derive CK_j and MK_j. iii) Leakage of MK_i only allows an attacker to decrypt the message c_i.*

(i) *The first ratchet are the updates to the Diffie-Hellman keys, such that they stay ephemeral, but are linked to provide forward secrecy and backwards secrecy. The second ratchet are the chain keys derived using the KDF, that provide forward secrecy during a session.*

(j) *We essentially lose the second (symmetric-key) ratchet: all messages in one chain now rely on the single secret DH-key SS_i and so we have no forward secrecy within the chain of messages, until we update the DH-key.*

End Secret Info .....

2. **(15 points)** In the lecture of May 25th, Peter talked (a.o.) about the Noise framework and the WireGuard protocol. The first part of this question is about certain symmetric cryptographic aspects within Noise. Refer to `http://www.noiseprotocol.org` for background study.

   (a) One authenticated encryption algorithm supported by Noise is AES-GCM. What is the official nonce size (in bits) of AES-GCM?

   (b) What is the effective nonce size (in bits) of AES-GCM as used in Noise?

   (c) Does the difference in official nonce size and actual nonce size sacrifice the *security* of the scheme? Explain your answer.

   The second part of this question is about certain symmetric cryptographic aspects within WireGuard. Refer to `https://www.wireguard.com/protocol` for background study.

   (d) What is the cryptographic hash function used in WireGuard?

(e) WireGuard uses the HKDF mode, which, as you learned in the symmetric crypto part of the course, is heavily based on the HMAC mode. As a matter of fact, WireGuard uses standalone HMAC as well. What is the exact output size (in bits) and what is the maximal key size (in bits) for the specific HMAC construction in WireGuard?

(f) Describe a way in which you think you can improve the way WireGuard uses the hash function you mentioned in (d). Explain your answer. [Hint: you may have to do a quick literature study to answer this question.]

(g) What cryptographic security property on cryptographic hash functions can you use to justify your suggestion of (f). Explain your answer informally.

Begin Secret Info: .....

(a) 96 *bits.*

(b) 64 *bits, see* `http://www.noiseprotocol.org/noise.html#the-aesgcm-cipher-functions`

(c) No, because the nonce is a counter, see `http://www.noiseprotocol.org/noise.html#overview-of-handshake-state-machine`*. It does influence the durability of the scheme, as you can only use it* 2^{64} *times, and you must make sure not implement it with random nonces.*

(d) BLAKE2s.

(e) In Wireguard, the HMAC-call to BLAKE2s is of the form `HMAC-Blake2s(key, input, 32)`*. The RFC*[1] *states that the key is of size at most* 32 *bytes, the input can be of arbitrary length, and the output is exactly* 32 *bytes (because that is the parameter fed to* `Blake2s`*.*

(f) According to `https://datatracker.ietf.org/doc/html/rfc7693`*: "BLAKE2 does not require a special "HMAC" (Hashed Message Authentication Code) construction for keyed message authentication as it has a built-in keying mechanism." So, according to the RFC, one can basically do BLAKE2s on input of (roughly)* $K\|M$*. Alternatively, according to* `https://eprint.iacr.org/2016/827.pdf`*, the BLAKE2 mode is indifferentiable from a random oracle, which (due to lecture 5, slide 31) means that BLAKE2s on input of (roughly)* $K\|M$ *works.*

(g) BLAKE2 is indifferentiable from a random oracle (see `https://eprint.iacr.org/2016/827.pdf`*).*

End Secret Info .....

3. **(15 points)** In the lecture of June 1st, Thom will talk (a.o.) about TLS. This question is about an attack related to this topic, namely Bleichenbacher's attack. For this question, read the following paper: `http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf`.

   (a) Describe the RSA PKCS#1 v1.5 padding.

   (b) What is the high level idea of the Bleichenbacher's attack?

   (c) Explain in detail the blinding step of the attack?

   (d) What can the attacker learn after obtaining one positive oracle answer?

   (e) What is the probability of obtaining one positive oracle answer?

   (f) Describe how to make a Bleichenbacher oracle for TLS. Write down a diagram.

   (g) Do a literature survey and find 3 subsequent attacks based on the Bleichenbacher's attack. Write 2–3 sentences describing each of them. Don't forget to provide references.

[1] `https://datatracker.ietf.org/doc/html/rfc7693`

Begin Secret Info:

It is hard to explain this better than Bleichenbacher himself already did. Therefore, for the solutions we point to the precise paragraph in the paper with small explanations where necessary.

(a) *See the start of section 2 and Figure 1.*

(b) *See the start of secion 3.1. Given c, the PKCS conformity of $c \cdot s^e$ (i.e. having confirmation that the plaintext of a given ciphertext conforms to the PKCS padding) leaks information on m. By gathering enough samples $c \cdot s^e$ that are PKCS conforming, we can recover m. All the attacker needs is a* padding oracle *that returns whether a plaintext is conforming or not.*

(c) *See the middle paragraph of page 4: given c, we look for a $c_0 = c \cdot s_0^e$ that is PKCS conforming, which is the starting point for the attack. We do this by just randomly sampling values s_0 until we get a c_0 that is PKCS conforming. Then, we apply steps 2-4 to c_0 so that we "find" the message m_0. Then, knowing s_0 we can recover m too.*

(d) *Precisely the start of page 4; the first two bytes of ms are* `00` *and* `02`*, so $2B \leqslant ms < 3B$.*

(e) *Beginning of 3.2, especially the start of page 6.*

(f) *See section 5 fully. TLS can be made into a Bleichenbachter oracle by taking c to be the encrypted* `pre_master_secret`*. Querying a server with a modfied $c' = c \cdot s^e$ then results in a failure if the associated message m' of c' is not PKCS conforming; the server sends an altert message if the message is not PKCS conforming. Thus, we can query some c' and TLS acts as an oracle to confirm if c' is or is not PKCS conforming.*

(g) *Any reasonable list of attacks is considered correct.*

End Secret Info