ETH Zürich, D-INFK
Spring 2018

Prof. Ueli Maurer
Christian Badertscher
Fabio Banfi

Cryptography Foundations

Exercise 3

3.1 Key Recycling

Goal*: Cryptographic keys should generally not be re-used. We examine a concrete scenario that causes vulnerabilities.*

Consider the following application of the so-called Encrypt-then-MAC paradigm but where the same key is used for the underlying functions:

Suppose that a single message m is encrypted using the one-time pad and the resulting ciphertext is authenticated using a MAC $f\colon \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^\ell$ with the same key as for the one-time pad, i.e., the message is encrypted as $c := m \oplus k$, then the MAC is computed as $z := f(c,k)$, and the pair (c,z) is transmitted.

View this new scheme as a MAC-function $f'(m,k) := (c,z)$ for c and z computed as above. Show that it is easily possible to win the 1-message MAC forgery game for the MAC f' with probability 1, even if we assume that winning the t-message forgery game for the MAC f is hard for an arbitrary $t > 1$.

3.2 Combining Different Schemes

Goal*: Cryptographic schemes should be composed carefully. We examine a concrete scenario that causes vulnerabilities and discuss security requirements for schemes combining encryption and authentication.*

a) Let (e,d) be a CPA-secure encryption scheme and f a secure MAC-function (under a chosen-message attack), again with the usual understanding of security as hardness of the respective problems (CPA bit-guessing problem and MAC forgery game, respectively).

Assume someone combines these and sends the pair (c,t) of the ciphertext $c = e(m,k,r)$ and the tag $t = f(m,k')$, computed with two different keys k, k'. This methodology is called *Encrypt-and-MAC*. Show that it is possible that an adversary learns the full message that is sent. Which additional property must the MAC have to prevent this?

As the plaintext (as opposed to the ciphertext) is authenticated, a modified ciphertext will still be accepted by the receiver if it decrypts to the same plaintext. Suppose the protocol aborts the connection in case a MAC verification fails, and such an abort is visible to an attacker, e.g., by observing a connection reset (in the TCP connection).

b) Assume that the MAC does not exhibit the weakness discussed in **a)**. Show that the error message can still be exploited to attack the security of the protocol, even if the encryption scheme is secure according to the standard definitions!

Hint: Devise a tailor-made encryption scheme, e.g., use the one-time pad with an encoding that expands the plaintext before encrypting.

3.3 Computing Discrete Logarithms in Specific Groups

Goal*: Discover that, while computing discrete logarithms is believed to be hard for certain groups, it is actually easy in other groups.*

Let G be a cyclic group of order $n = |G|$. For a generator g of G and an element y in G we want to compute the discrete logarithm x of y to the base g. This means to find the $x \in \{0, \dots, n-1\}$ such that $y = g^x$.

a) Show that when $G = \mathbb{Z}_n$ is the additive group of integers modulo n, discrete logarithms can easily be computed.

b) Assume that you have an efficient algorithm to compute discrete logarithms to the base g in G. Let $h \in G$ be another generator of G. Show how you can use your algorithm to efficiently compute discrete logarithms to the base h.

c) Assume that G has even order, so $n = 2m$. Show how, in this case, you can efficiently compute whether the discrete logarithm x of y to the base g is even or odd. This means to compute the reduction $x \bmod 2$ of x modulo 2.

d) Suppose now that the order of G is divisible by 2^k, so $n = 2^k m$. Construct an efficient algorithm to compute $x \bmod 2^k$.

e) Now generalize your idea of **c)** to the case of other small prime divisors. So, give an efficient algorithm to compute $x \bmod p^k$ if p^k divides n (and p is small compared to n).

f) Finally show how to efficiently compute x if all prime divisors of n are small.

3.4 Resources and Converters in the Diffie-Hellman Key-Agreement

Goal*: This task is to familiarize yourself with the style we describe our resources and converters.*

Recall the Diffie-Hellman construction presented in the lecture. Specify as pseudo-code the initiator's converter $\mathsf{dh_I}$ and the responder's converter $\mathsf{dh_R}$ that are attached at interfaces A and B of the assumed resource. That is, specify the converter used in the expression $\mathsf{dh_I}^A \mathsf{dh_R}^B [\bullet\!\!\longrightarrow, \longleftarrow\!\!\bullet]$. Recall that the authenticated (single-message) channel $\bullet\!\!\longrightarrow$ is specified in Exercise 2.3, and that $\longleftarrow\!\!\bullet$ is shorthand for an authenticated (single-message) channel that is defined as $\bullet\!\!\longrightarrow$ but with A being the receiver interface and B being the sender interface.

Also, specify the constructed key-resource as pseudo-code and pay attention to the details, in particular when the key is output and what information is leaked at interface E. Try to stick to the basic notation style of Exercise 2.3.

For notational simplicity, you may find it helpful to write init for the first unary input at interface A and getKey for the remaining unary inputs.

Hint: Recall that resources are systems where for each input at an interface, exactly one output at the same interface is returned. Converters are systems where for each input at the outside interface, a finite number of outputs at the inside interface are generated, before the result of this activation is returned at the outside interface.

Discussion of solutions:

12/13.3.2018 (Tasks 3.1 and 3.2)

19/20.3.2018 (Tasks 3.3 and 3.4)

The Monday and Tuesday sessions of each week cover the same material.