Modern Cryptography Jan 15, 2020

Homework 13

Lecturer: Daniel Slamanig, TA: Guillermo Perez, Karen Klein Due: 23.59 CET, Jan 22, 2020

To get credit for this homework it must be submitted no later than Wednesday, January 15th via TUWEL. If you have not registered for the tutorial (192.063 Tutorial on Introduction to Modern Cryptography 2019W) on TUWEL, please do so. If you are unable to register for the course on TUWEL for some reason, submit your homework via email to via email to `guillermo.pascualperez@ist.ac.at`, please use "MC19 Homework 13" as subject.

1. Derandomizing signatures

   - **(3 Points)** Let $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be an $\mathsf{EUF\text{-}CMA}$ secure signature scheme, where the signing algorithm Sign is probabilistic. In particular, algorithm Sign uses randomness r chosen from a space $\mathcal{R}$. We let $\mathsf{Sign}(sk, m; r)$ denote the execution of algorithm Sign with randomness r. Let F be a secure pseudo-random function (PRF) with key space $\mathcal{K}$ and output space being $\mathcal{R}$. Prove that the signature scheme $\Sigma' = (\mathsf{KGen}', \mathsf{Sign}', \mathsf{Verify})$ is also $\mathsf{EUF\text{-}CMA}$ secure, where

     KGen': Run $(sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)$, $k \leftarrow_\$ \mathcal{K}$, set $sk' := (sk, k)$ and return (sk', pk).
     Sign': Compute $r \leftarrow F(k, m)$, $\sigma \leftarrow \mathsf{Sign}(sk, m; r)$ and output σ.

   Solution: Let's first consider an scheme Σ'' which is intermediate to Σ and Σ'. Σ'' is similar to Σ' except that instead of a PRF F it relies on a function f (to which it has oracle access) picked at random from the set of all functions. That is, $\Sigma'' = (\mathsf{KGen}, \mathsf{Sign}'', \mathsf{Verify})$, where

   Sign'': Compute $r \leftarrow f(m)$, $\sigma \leftarrow \mathsf{Sign}(sk, m; r)$ and output σ.

   Since f is a random function, it is not hard to see that the advantages that a forger has against Σ'' and Σ are the same.

   The Reduction. Next, we design an algorithm R which relates the security of the PRF F to the security of Σ'' and Σ'. That is, R plays the PRF indistinguishability game and the $\mathsf{euf\text{-}cma}$ game with a forger A at the same time. R is given access to an oracle $\mathcal{F}(\cdot)$ which is either $F_k(\cdot)$ (real) or $f(\cdot)$ (random) and its goal is to distinguish real from random. First it runs KGen to sample a key-pair (pk, sk) and passes pk on to A, whose goal is to forge a signature on pk. Whenever A queries for the signature on a message m, R first relays it to $\mathcal{F}$ to obtain $r = \mathcal{F}(m)$. Next, it computes the signature $\sigma = \mathsf{Sign}(sk, m; r)$ and returns it to the adversary. At the end of the $\mathsf{euf\text{-}cma}$ game A returns a forgery (m^*, σ^*) and R outputs 1 iff this forgery valid.

   Analysis. Since F is a secure PRF, we know that

   $$\left|\Pr\left[\mathsf{R}^{F_k(\cdot)} = 1\right] - \Pr\left[\mathsf{R}^{f(\cdot)} = 1\right]\right| = \mathsf{negl}(\lambda). \tag{1}$$

Note that when $\mathcal{F}(\cdot) = F_k(\cdot)$, R simulates Σ' to A; on the other hand, when $\mathcal{F}(\cdot) = f(\cdot)$, R simulates Σ'' to A. It follows from eq.(1) that

$$\left|\Pr\left[\mathsf{Sig\text{-}forge}^{\mathsf{euf\text{-}cma}}_{\mathsf{A},\Sigma'} = 1\right] - \Pr\left[\mathsf{Sig\text{-}forge}^{\mathsf{euf\text{-}cma}}_{\mathsf{A},\Sigma''} = 1\right]\right| = \mathsf{negl}(\lambda).$$

Since Σ'' is a secure signature scheme,

$$\Pr\left[\mathsf{Sig\text{-}forge}^{\mathsf{euf\text{-}cma}}_{\mathsf{A},\Sigma'} = 1\right] = \mathsf{negl}(\lambda),$$

and by basic algebraic manipulation we get

$$\Pr\left[\mathsf{Sig\text{-}forge}^{\mathsf{euf\text{-}cma}}_{\mathsf{A},\Sigma''} = 1\right] = \mathsf{negl}(\lambda).$$

□

2. Attack on derandomized signatures

   - **(1.5 Points)** Consider the Schnorr signature scheme (see slide 21 of Lecture 13) using the derandomization strategy in Task 1. Present a detailed attack that recovers the secret signing key if we assume that you can introduce a single bit fault into the signing process (as discussed on slide 23 of Lecture 13).

   Solution: Let's first write the resulting signature scheme $\Sigma'_S = (\mathsf{KGen}'_S, \mathsf{Sign}'_S, \mathsf{Verify}'_S)$:

   - KGen'_S: $G, q, g) \leftarrow \mathcal{G}(1^n)$, $x \xleftarrow{\$} \mathbb{Z}_q$, $k \xleftarrow{\$} \mathcal{K}$. Returns $(sk, pk) = ((x, k), (G, q, g, y := g^x))$ and a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
   - Sign'_S: Define $\ell := F(k, m)$ and compute $I := g^\ell$, $r := H(I, m)$, $s := rx + \ell \mod q$. Output the signature $\sigma = (r, s)$.
   - Verify'_S: compute $I := g^s \cdot y^{-r}$ and output 1 if $H(I, m) = r$ (verification does not change).

   To perform the attack, we will query for the signature on any given message twice getting signatures (r_1, s_1) and (r_2, s_2) respectively, introducing a single bit fault into the second execution. In particular, we will introduce the fault during the computation of $r := H(I, m)$, so that the resulting signature (r_2, s_2) satisfies $r_2 \neq r_1$ (and therefore also $s_2 \neq s_1$, since ℓ is the same in both executions, as it was already computed at the time of the fault). The difference between r_2 and r_1 can be a single bit, all we care about is that $r_2 - r_1 \neq 0$. Overall, we obtain two signatures, each satisfying the equation

   $$s_i = r_i x + \ell \mod q\,.$$

   Combining both, we get:

   $$\begin{aligned} & s_1 - r_1 x = s_2 - r_2 x \mod q \\ \Rightarrow \quad & x = (s_2 - s_1)(r_2 - r_1)^{-1} \mod q\,. \end{aligned}$$

   Since $r_2 - r_1 \neq 0$, the inverse exists and we have recovered x. □

3. One-time signatures

- Let us consider the signature scheme Σ in Fig. 1 with message space $\mathcal{M} = \{0,1\}^*$ and hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ (which we assume to be sampled randomly from the hash function family $\{H_k\}_{k\in\mathcal{K}}$).
  - **(0.5 Points)** Show that the scheme is correct

$\mathsf{KGen}(1^\lambda)$:	$\mathsf{Sign}(sk, m)$:	$\mathsf{Verify}(pk, m, \sigma)$:
- $\mathcal{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{GGen}(1^\lambda)$; - $H \leftarrow\!\!\$\ \{H_k\}_{k\in\mathcal{K}}$; - $\alpha, \beta \leftarrow\!\!\$\ \mathbb{Z}_q$; - $u := g^\alpha; v := g^\beta$; - $(sk, pk) := ((\alpha, \beta), (u, v, \mathcal{G}, H))$ - **return** (sk, pk).	- $\sigma := H(m)\alpha + \beta \mod q$; - **return** σ.	- **if** $g^\sigma = v \cdot u^{H(m)}$ **return** 1 **else return** 0.

Figure 1: Signature scheme Σ.

Solution: Let $m \in \mathcal{M}$ be arbitrary, let $((\alpha, \beta), (u, v, \mathcal{G}, H)) = (sk, pk)$ be a key pair generated with KGen, and let $\sigma = \mathsf{Sign}(sk, m)$. Then we have

$$g^\sigma = g^{H(m)\alpha+\beta} = (g^\alpha)^{H(m)} \cdot g^\beta = u^{H(m)} \cdot v.$$

Hence $\mathsf{Verify}(pk, m, \sigma) = 1$. □

- **(3.5 Points)** Prove the following theorem:

  Theorem 1 *If the discrete-logarithm problem is hard relative to $\mathcal{G}$ and H is modeled as a random oracle, then the signature scheme is* EUF-1-CMA *secure.*

  Solution: Assume that A is a PPT adversary that breaks the EUF-1-CMA security game with non-negligible probability. We construct a PPT adversary B using A as a black box that solves the discrete logarithm problem with respect to GGen with non-negligible probability. B on input of $(\mathcal{G}, x = g^\tau)$ is going to use the discrete logarithm challenge x as u and set up v in a way that allows it to compute without knowledge of the τ a valid signature on a message m as long as its hash value equals a particular $h \in \mathbb{Z}_p$. Signatures on messages without this property, on the other hand, will allow B to compute the discrete logarithm of x. Since H is modeled as a random oracle B can program it such that it will evaluate to h on the single signing query m.

  Before we describe B in detail we argue that without loss of generality we may assume that A satisfies the following properties:

  · A never queries the random oracle RO on the same input twice, since otherwise we can replace it by an adversary with the same success probability that forwards "fresh" queries to the random oracle, stores the corresponding answer in a list, and answers all other queries using its list.

  · When A queries the singing oracle on a message m then $\mathsf{RO}(m)$ is already defined, since otherwise we could replace A with an adversary that makes one additional RO query and achieves the same success probability.

· Let m be the message the adversary has seen a signature on and m^* the message output as part of the forgery attempt. Then $m \neq m^*$, since otherwise we could replace A with an adversary that simply aborts in this case.

We now describe B. Let Q_{RO} be an upper bound on the number of random-oracle queries of A. Note that since A runs in polynomial time Q_{RO} is polynomially bounded. On input of the DL-challenge $(\mathcal{G}, x)$ with $x = g^\tau$, B samples $\sigma, h \leftarrow\!\!\$\, \mathbb{Z}_q$ and sets $u \leftarrow x$, $v \leftarrow x^{-h} \cdot g^\sigma$. Then it samples an index $j \leftarrow\!\!\$\, \{1, \ldots, Q_{\mathsf{RO}}\}$ and runs adversary A on input $pk = (u, v, \mathcal{G})$.

Since H is modeled as a random oracle B has to provide A with a random-oracle procedure RO. Adversary B answers the jth query m_j to RO with h. All other queries are answered with values from $\mathbb{Z}_q$ sampled uniformly at random.

At some point A asks for a signature on a message m. If $m = m_j$ adversary B answers the query with σ. Else it aborts.

At the end of the game A outputs (m^*, σ^*). Let $h^* = \mathsf{RO}(m^*)$. If $h^* = h$ adversary B aborts. Otherwise it returns $(\sigma^* - \sigma)/(h^* - h) \bmod q$ as the solution of its discrete logarithm challenge.

It is easy to verify that B runs in polynomial time. We will show that:

(a) If B does not abort and A succeeds in forging a signature then B computes the discrete logarithm of x.

(b) The probability that B does not abort is bounded from below by $(1 - 1/q)/Q_{\mathsf{RO}}$.

(c) B provides A with a perfect simulation of the $\mathsf{EUF\text{-}1\text{-}CMA}$ game unless it aborts.

Items (a),(b),(c) imply that

$$\begin{aligned}
\Pr[\mathsf{DLog}_{\mathsf{B}}] &= \Pr[\mathsf{DLog}_{\mathsf{B}} \mid \text{no abort}] \Pr[\text{no abort}] + \Pr[\mathsf{DLog}_{\mathsf{B}} \mid \text{abort}] \Pr[\text{abort}] \\
&\geq \Pr[\mathsf{DLog}_{\mathsf{B}} \mid \text{no abort}] \Pr[\text{no abort}] \\
&\geq \frac{1 - 1/q}{Q_{\mathsf{RO}}} \Pr[\mathsf{DLog}_{\mathsf{B}} \mid \text{no abort}] \\
&\geq \frac{1 - 1/q}{Q_{\mathsf{RO}}} \Pr[\mathsf{EUF\text{-}1\text{-}CMA}_{\mathsf{A}}],
\end{aligned}$$

which is non-negligible since Q_{RO} is polynomially bounded and $\Pr[\mathsf{EUF\text{-}1\text{-}CMA}_{\mathsf{A}}]$ was assumed to be non-negligible. Thus if the discrete logarithm problem is hard with respect to GGen then the scheme is EUF-1-CMA secure.

It remains to prove (a),(b), and (c). Regarding (a), consider the forgery (m^*, σ^*). If σ^* is a valid signature on m^* then by our choice of u and v we have

$$g^{\sigma^*} = u^{h^*} \cdot v = g^{\tau h^*} \cdot g^{-\tau h + \sigma},$$

which is equivalent to

$$\sigma^* - \sigma = \tau(h^* - h) \bmod q.$$

If B does not abort we have that $h^* \neq h$ and B correctly computes $\tau = (\sigma^* - \sigma)/(h^* - h) \bmod q$.

We now prove (b). B can either abort when A poses its signing query m or after it outputs its forgery attempt. The former happens if $m \neq m_j$. Since we may assume

that $m \in \{m_1, \ldots, m_{Q_{\mathsf{RO}}}\}$, and j was sampled uniformly in $\{1, \ldots, Q_{\mathsf{RO}}\}$ and is independent of all the adversaries queries we obtain that $\Pr[m = m_j] = 1/Q_{\mathsf{RO}}$. Now consider the forgery attempt (m^*, σ^*). Since H is modeled as a random oracle and hence its outputs are uniformly distributed we have that $\Pr[\mathsf{RO}(m) = \mathsf{RO}(m^*)] = 1/q$, where we used that we may assume $m \neq m^*$. Thus in the latter case with probability at least $1 - 1/q$ adversary B does not abort. Overall, we can bound the probability of B not aborting by $(1/1 - q)/Q_{\mathsf{RO}}$.
Finally we show that (c) holds. Assume that B does not abort. Consider σ returned as response to the signing query on m. Since B did not abort we have $m = m_j$ and hence $\mathsf{RO}(m) = h$. Thus $v \cdot u^{\mathsf{RO}(m)} = x^{-h} g^{\sigma} \cdot x^h = g^{\sigma}$ and σ is indeed a valid signature on m. We conclude by showing that all values provided to A have the correct distribution. For $i \in \{1, \ldots, Q_{\mathsf{RO}}\}$ denote by h_i the value $\mathsf{RO}(m_i)$ and consider the joint distribution of $(u, v, h_1, \ldots, h_{Q_{\mathsf{RO}}})$. In the scheme as defined in Figure 1 this is the uniform distribution over $\mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q^{Q_{\mathsf{RO}}}$. In the game simulated by B for all i$\neq j$ the values h_i are uniformly random in $\mathbb{Z}_q$ and independent of all other values. The remaining values are $u = g^{\tau}$, $v = g^{-\tau h + \sigma}$, and $h_j = h$. To see that their distribution is indeed uniform over $\mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q$, note that, for all $u' = g^{\tau'}, v' = g^{\nu'} \in \mathbb{G}$, and $h' \in \mathbb{Z}_p$ we have $\Pr[u = u', v = v', h = h'] = \Pr[\tau = \tau', h = h', \sigma = \nu' + \tau' h'] = q^{-3}$. Hence, unless B aborts, the simulation of the $\mathsf{EUF\text{-}1\text{-}CMA}$ game is perfect. □

– **(1.5 Point)** Show that Σ is not two-time secure: given signatures on two distinct messages m_0 and m_1 in $\mathbb{Z}_q$, the adversary can forge the signature on every message $m \in \mathbb{Z}_q$ of its choice.

Solution: The central observation for this exercise is, that, unless the hash values of the messages collide, two valid signatures are enough to compute the secret key of the scheme.
More precisely, we construct an adversary A against EUF-q-CMA for $q \geq 2$ as follows. A picks two arbitrary messages m_1, m_2 with $m_1 \neq m_2$. Let $h_1 = H(m_1)$ and $h_2 = H(m_2)$. If $h_1 = h_2$ then A asks for a signature σ_1 on m_1. It then returns the forgery (σ_1, m_2). If, on the other hand, $h_1 \neq h_2$ then it asks for signatures σ_1, σ_2 on m_1 and m_2. It then computes the unique solution (α, β) of the following system of linear equations over $\mathbb{Z}_q$

$$\begin{pmatrix} h_1 & 1 \\ h_2 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}.$$

Then for arbitrary $m_3 \notin \{m_1, m_2\}$ it returns the forgery (σ_3, m_3) with $\sigma_3 = H(m_3)\alpha + \beta$. We argue that in both cases A computed a valid signature on the respective message.
Indeed, if $h_1 = h_2$ then

$$u^{H(m_2)} \cdot v = u^{h_2} \cdot v = u^{h_1} \cdot v = g^{\sigma_1},$$

where the last equation holds since σ_1 is a valid signature on m_1. It follows that $\mathsf{Verify}(pk, m_2, \sigma_1) = 1$. If, on the other hand, $h_1 \neq h_2$ then since σ_1 and σ_2 are valid signatures we have that

$$g^{\sigma_1} = u^{h_1} \cdot v \quad \text{and} \quad g^{\sigma_2} = u^{h_2} \cdot v,$$

which is equivalent to

$$\sigma_1 = h_1\alpha + \beta \quad \text{and} \quad \sigma_2 = h_2\alpha + \beta,$$

where addition and multiplication are in $\mathbb{Z}_q$. The equations can be rewritten as the system of linear equations from above. Note that the matrix has a nonzero determinant since $h_1 \neq h_2$. Thus the systems unique solution is the private key (α, β) which in turn allows to compute signatures on arbitrary messages. □