Modern Cryptography	January 25, 2019

Solutions to Homework 13

Lecturer: Daniel Slamanig, TA: Karen Klein

1. Hash-and-Sign

- **(3 Points)** Provide a formal proof of security of the hash-and-sign paradigm, i.e. prove the following theorem:

Theorem 1 *If Σ is a secure signature scheme for messages of length k and Γ is collision resistant, then Σ' is a secure signature scheme (for arbitrary-length messages).*

Solution: Let $\mathcal{A}'$ be an arbitrary PPT adversary against EUF-CMA security of $\Sigma' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$. Define $\mathsf{Collision}$ as the event that throughout a run of the security game $\mathsf{SigForge}^{\mathsf{euf-cma}}_{\mathcal{A}',\Sigma'}(n)$ the attacker $\mathcal{A}'$ queries a signature for a message m_i such that $H^s(m_i) = H^s(m^*)$. Then we can bound the success probability of A' as

$$\mathsf{Pr}[\mathcal{A}' \text{ wins}] = \mathsf{Pr}[\mathcal{A}' \text{ wins } \wedge \mathsf{Collision}] + \mathsf{Pr}[\mathcal{A}' \text{ wins } \wedge \neg\mathsf{Collision}]$$

We first bound the probability that $\mathcal{A}'$ wins and $\mathsf{Collision}$ happens. To this aim, we construct a PPT algorithm $\mathcal{A}^H$ against the collision-resistance (see definition 5.1 and 5.2 from the first part of the course) of the hash function family $\Gamma = (\mathsf{Gen}_H, H)$ that runs $\mathcal{A}'$ as a subroutine. First, $\mathcal{A}^H$ gets a challenge $s \leftarrow \mathsf{Gen}_H(1^n)$ and runs $\mathsf{Gen}(1^n)$ to receive a key pair $(\mathsf{pk}, \mathsf{sk})$. It sets $\mathsf{pk}' = (\mathsf{pk}, s)$ and $\mathsf{sk}' = (\mathsf{sk}, s)$. Then it sends pk' to $\mathcal{A}'$. Since $\mathcal{A}^H$ knows sk', it can answer all signature queries m_i of $\mathcal{A}'$ just by running the algorithm $\mathsf{Sign}'_{\mathsf{sk}'}(m_i)$. Furthermore, $\mathcal{A}^H$ stores all queries m_i together with the corresponding hash values $H^s(m_i)$ in a list $\mathcal{Q}$. When $\mathcal{A}'$ outputs a forgery (m^*, σ^*), then $\mathcal{A}^H$ computes $H^s(m^*)$ and searches for a pair $(m_i, H^s(m_i)$ in the list $\mathcal{Q}$ such that $H^s(m_i) = H^s(m^*)$. If it finds such a pair, $\mathcal{A}^H$ outputs (m^*, m_i), otherwise it outputs an arbitrary pair (m_1, m_2). Clearly, if $\mathsf{Collision}$ happens then $\mathcal{A}^H$ finds m_i such that $H^s(m_i) = H^s(m^*)$ and, furthermore, if $\mathcal{A}'$ is successful it also holds $m^* \notin \mathcal{Q}$, thus, $m^* \neq m_i$ and $\mathcal{A}^H$ wins. By collision resistance of Γ it follows

$$\mathsf{negl}_1(n) \geq \mathsf{Pr}[\mathsf{Hash-coll}_{\mathcal{A}^\mathsf{H},\Gamma}(\mathsf{n}) = 1] \geq \mathsf{Pr}[\mathcal{A}' \text{ wins } \wedge \mathsf{Collision}].$$

In the case that $\mathsf{Collision}$ does not happen we reduce the security of Σ' to the security of the fixed-length signature scheme Σ. We construct an adversary $\mathcal{A}$ against the security of Σ as follows: Given a public key pk, $\mathcal{A}$ computes $s \leftarrow \mathsf{Gen}_H(1^n)$ and sends $\mathsf{pk}' = (\mathsf{pk}, s)$ to $\mathcal{A}'$. Whenever $\mathcal{A}'$ asks for a signature on a message m_i, $\mathcal{A}$ computes $H^s(m_i)$, queries $\sigma_i \leftarrow \mathsf{Sign}_{sk}(H^s(m_i))$ and forwards σ_i to $\mathcal{A}'$. Clearly, this perfectly simulates the $\mathsf{Sign}'_{\mathsf{sk}'}$ oracle. As soon as $\mathcal{A}'$ outputs a pair (m^*, σ^*), $\mathcal{A}$ outputs $(H^s(m^*), \sigma^*)$. If $\mathsf{Collision}$ does not happen then $H^s(m^*) \neq H^s(m_i)$ for all previously queried messages m_i. If furthermore $\mathcal{A}'$ succeeds, i.e., (m^*, σ^*) is a valid forgery with respect to Σ', then $(H^s(m^*), \sigma^*)$ is a valid forgery with respect to Σ since by definition

$$1 = \mathsf{Vrfy}'_{\mathsf{pk}'}(m^*, \sigma^*) = \mathsf{Vrfy}_{\mathsf{pk}}(H^s(m^*), \sigma^*).$$

Security of Σ now implies

$$\mathsf{negl}_2(n) \geq \Pr[\mathcal{A} \text{ wins}] \geq \Pr[\mathcal{A}' \text{ wins } \wedge \neg\mathsf{Collision}].$$

Combining the two results, we get

$$\Pr[\mathcal{A}' \text{ wins}] \leq \mathsf{negl}_1(n) + \mathsf{negl}_2(n) =: \mathsf{negl}(n)$$

which proves security of Σ'. □

2. RSA signatures

- **[12.3 in book, 2nd edition] (2 Points)** In the lecture we have seen an attack on the textbook RSA signature scheme in which an attacker forges a signature on an arbitrary message using two signing queries. Show how an attacker can forge a signature on an arbitrary message using a single signing query.

  Solution: Suppose an attacker $\mathcal{A}$ wants to forge a signature on $m \in \mathbb{Z}_N^*$. To this aim, $\mathcal{A}$ sets $m' := [m^{-1} \bmod N]$ and queries a signature on m'. Receiving $\sigma' \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m') = [(m')^d \bmod N]$, the attacker computes $\sigma := [(\sigma')^{-1} \bmod N]$ and outputs (m, σ). It holds

  $$\sigma^e = (\sigma')^{-e} = (m')^{-ed} = (m)^{ed} = m \bmod N.$$

  Thus, (m, σ) is a valid forgery. □

3. DSA Signatures

- **[12.7 in book, 2nd edition] (2 Points)** Consider a variant of DSA in which the message space is $\mathbb{Z}_q$ and H is omitted. (So the second component of the signature is now $s := k^{-1} \cdot (m + xr) \bmod q$.) Show that this variant is not secure.

  Solution: A possible no-message attack arises from setting $r = F(y)$, thus, implicitly setting $k = x$. Since the secret key x is of course unknown to the attacker but it has to output $s = k^{-1}(m + rx) = x^{-1}(m + rx) \bmod q$, the idea is now to set m in such a way that x cancels out in the expression and s only depends on known parameters. Setting $m = 0$ achieves this goal and it is easy to check that $(m, (r, s)) = (0, (F(y), F(y)))$ is a valid forgery:

  $$F(g^{ms^{-1}} y^{rs^{-1}}) = F(g^0 y^1) = F(y) = r.$$

  More general, one can set $r := F(g^a y^b)$, i.e., $k := a + bx \bmod q$, and $m := rab^{-1} \bmod q$ to get

  $$s := k^{-1}(m+rx) = (a+bx)^{-1}(m+rx) = (a+bx)^{-1}(rab^{-1}+rx) = (a+bx)^{-1}rb^{-1}(a+bx) = rb^{-1}.$$

  By construction, $(m, (r, s)) = (rab^{-1}, (F(g^a y^b), F(g^a y^b) \cdot b^{-1}))$ is a valid forgery. □

4. One-time signatures

- **(1 Point)** Write down the experiment for existential unfogeability under a one-time non-adaptive chosen message attack ($\mathsf{EUF\text{-}1\text{-}naCMA}$ security).

Solution: We define the experiment $\mathsf{SigForge}^{\mathsf{EUF-1-naCMA}}_{\mathcal{A},\Pi}$ between an adversary $\mathcal{A}$ and a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows:

- On input only the security parameter 1^n (and possibly some further public parameters, e.g., specifying the message space), $\mathcal{A}$ outputs a single message m he wants to get a signature for.
- Keys $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{Gen}(1^n)$ are chosen and a signature $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ is computed. $\mathcal{A}$ receives pk and σ.
- $\mathcal{A}$ outputs a forgery (m^*, σ^*).
- The output of the experiment is 1 if and only if $m \neq m^*$ and $\mathsf{Vrfy}_{\mathsf{pk}}(m^*, \sigma^*) = 1$.

□

- **(2 Points)** For the one-time signatures under the discrete logarithm problem from the lecture (slide 24) show the following theorem:

Theorem 2 *If the discrete-logarithm problem is hard relative to $\mathcal{G}$, then the signature scheme is* EUF-1-naCMA *secure.*

Solution: Let $\mathcal{A}$ be an arbitrary PPT adversary against the signature scheme. We construct an algorithm $\mathcal{A}'$ for the discrete logarithm as follows: $\mathcal{A}'$ gets as input a tuple (G, q, g, g^z), where $(G, q, g) \leftarrow \mathcal{G}(1^n)$ and $z \leftarrow \mathbb{Z}_q$ uniformly random. Running $\mathcal{A}$ on public parameters (G, q, g), $\mathcal{A}'$ receives a message m for which it needs to output a signature. $\mathcal{A}'$ chooses $\sigma \leftarrow \mathbb{Z}_q$ uniformly at random and sets $h := g^z$, i.e., implicitly $z = x$, and $c := g^m h^\sigma$. It send $\mathsf{pk} := (h, c)$ and σ to $\mathcal{A}$. By construction, σ is a valid signature for m with respect to the public key $\mathsf{pk} := (h, c)$. G, q, g and h are chosen exactly as in the real game. If $x \neq 0$, then for any m the distribution of $c := g^{m+x\sigma}$ with $\sigma \leftarrow \mathbb{Z}_q$ is identical to the distribution of $c_{\text{real}} := g^y$ with $y \leftarrow \mathbb{Z}_q$ in the real game. In both cases, σ is uniquely determined given c, h and m. Since $x = 0$ happens with negligible probability $1/q$, the view of $\mathcal{A}$ when given $\mathsf{pk} = (c, h)$ and σ is computational indistinguishable from its view in a real execution of $\mathsf{SigForge}^{\mathsf{EUF-1-naCMA}}_{\mathcal{A},\Pi}$. When $\mathcal{A}$ outputs a forgery (m^*, σ^*), $\mathcal{A}'$ first checks $m \neq m^*$ and $\sigma \neq \sigma^*$. If $\mathcal{A}$ is successful, this is always true and thus, in this case, $\mathcal{A}'$ outputs $z' := (m - m^*)(\sigma - \sigma^*)^{-1} \bmod q$. Otherwise $\mathcal{A}'$ outputs $z' \leftarrow \mathbb{Z}_q$ uniformly at random. If $\mathcal{A}$ succeeds and outputs a valid forgery, then

$$g^{m^*+z\sigma^*} = g^{m^*} h^{\sigma^*} = c = g^m h^\sigma = g^{m+z\sigma}.$$

This implies $m^* + z\sigma^* = m + z\sigma$, hence $z = (m - m^*)(\sigma^* - \sigma)^{-1} \bmod q$. Thus, if $x \neq 0$ and $\mathcal{A}$ succeeds, then $\mathcal{A}'$ solves the discrete logarithm problem. Assuming the hardness of the discrete logarithm problem relative to $\mathcal{G}$ it follows

$$\begin{aligned}\mathsf{Pr}[\mathcal{A} \text{ wins}] &= \mathsf{Pr}[\mathcal{A} \text{ wins } \wedge\ x \neq 0] + \mathsf{Pr}[\mathcal{A} \text{ wins } \wedge\ x = 0] \\ &\leq \mathsf{Pr}[\mathcal{A}' \text{ solves DLog}] + 1/q \leq \mathsf{negl}(n).\end{aligned}$$

This proves EUF-1-naCMA-security of the scheme. □