MTAT.07.003 Cryptology II
Spring 2012 / Exercise session ?? / Example Solution

Exercise (Polynomial Authentication Code + PRF = MAC). *The polynomial message authentication code is secure only if we do not reuse the authentication key. This weakness can be alleviated by using a pseudorandom function, such as the AES block cipher, to stretch the initial seed to many session keys. Let h be a polynomial message authentication code over the field $\mathbb{F}$ and let $\mathcal{F}$ be a (t,ε)-pseudorandom function family over $\mathbb{F}\times\mathbb{F}$. During the key generation sender and receiver agree on a function $f \xleftarrow{u} \mathcal{F}$. To authenticate a message $\boldsymbol{m} = (m_\ell,\ldots,m_1)$, we first generate a nonce $n \xleftarrow{u} \mathbb{F}\times\mathbb{F}$ and then one-time keys $k_1,k_0 \leftarrow f(n)$ and the polynomial authentication code $t = h(\boldsymbol{m},k_1,k_0)$. The resulting message authentication code is (t,n). To verify a tuple $(\boldsymbol{m},t,n)$, receiver repeats the computation to verify that $t = h(\boldsymbol{m},k_1,k_0)$. Prove that this construction is a computationally secure message authentication code.*

Solution. Let RndPolyMac denote the message authentication code defined above. Before we consider its security, it is instructive to consider the security of a stateful message authentication code CtrPolyMac instead. The only difference between RndPolyMac and CtrPolyMac is the way nonce is generated. In the CtrPolyMac, nonces are generated sequentially starting form one. As a consequence, the pseudorandom function is always queried on different values, which simplifies the security analysis.

Security analysis of CtrPolyMac. To show that CtrPolyMac is secure we have estimate how successful can adversary $\mathcal{A}$ be in the game that defines security of message authentication code:

$\mathcal{G}^{\mathcal{A}}$
$f \xleftarrow{u} \mathcal{F}$
$n_0 \leftarrow 0,\ t_0 \leftarrow \perp$
For $i \in \{1,\ldots,q\}$ do
  $\boldsymbol{m}_i \leftarrow \mathcal{A}(n_{i-1},t_{i-1})$
  $n_i \leftarrow n_{i-1}+1$
  $k_1,k_0 \leftarrow f(n_i)$
  $t_i \leftarrow h(\boldsymbol{m}_i,k_1,k_0)$
$\overline{\boldsymbol{m}},\overline{n},\overline{t} \leftarrow \mathcal{A}(n_q,t_q)$
if $(\overline{\boldsymbol{m}},\overline{n},\overline{t}) \in \{(\boldsymbol{m}_1,n_1,t_1),\ldots,(\boldsymbol{m}_q,n_q,t_q)\}$ then **return** 0
$\overline{k}_1,\overline{k}_0 \leftarrow f(\overline{n})$
return $[\overline{t} \stackrel{?}{=} h(\overline{\boldsymbol{m}},\overline{k}_1,\overline{k}_0)]$.

This game models the attack where the adversary $\mathcal{A}$ can query message authentication codes for distinct nonces $1,\ldots,i,\ldots,q$, after which $\mathcal{A}$ can come up with any valid nonce $\overline{n}$. Such a choice is justified, since the sender can control in which order nonces are generated, while the receiver has no control over the order message-tag pairs arrive. Thus, the easiest option for the receiver is to accept all messages with valid authentication tags regardless of nonces. Alternatively, the receiver can keep the database of received nonces and check for duplicates or accept only messages of with sequentially increasing nonces. Neither of these alternatives is practical. The first option requires additional space and time for seeking duplicates, while the second one makes it easy to halt all communications by removing a single message from the channel.

As the first proof step, note that we can replace $\mathcal{F}$ with a function family $\mathcal{F}_{\text{all}}$. The adversaries success can only change by ε if the running time of $\mathcal{A}$ is sufficiently small[1]. Let $\mathcal{G}_1$ be the corresponding game. Next note that there are essentially two possibilities. First, the nonce $\overline{n}$ can be distinct, i.e., $\overline{n} \neq n_i$. Then the random function is evaluated in $q+1$ distinct points and we replace its values by random samples. The

[1]To determine the true bound we have to do the reduction. Intuition says that $t_{\mathcal{A}} \leq t - \mathrm{O}(q)$

following game

$\mathcal{G}_2^{\mathcal{A}}$

$n_0 \leftarrow 0,\ t_0 \leftarrow \perp$
For $i \in \{1, \ldots, q\}$ do
  $\boldsymbol{m}_i \leftarrow \mathcal{A}(n_{i-1}, t_{i-1})$
  $n_i \leftarrow n_{i-1} + 1$
  $k_1, k_0 \leftarrow \mathbb{F}$
  $t_i \leftarrow h(\boldsymbol{m}_i, k_1, k_0)$
$\overline{\boldsymbol{m}}, \overline{n}, \overline{t} \leftarrow \mathcal{A}(n_q, t_q)$
if $\overline{n} \in \{n_1, \ldots, n_q\}$ then **return** 0
$\overline{k}_1, \overline{k}_0 \leftarrow \mathbb{F}$
return $[\overline{t} \stackrel{?}{=} h(\overline{\boldsymbol{m}}, \overline{k}_1, \overline{k}_0)]$

measures the success of $\mathcal{A}$ in this setting. Note that the condition $(\overline{\boldsymbol{m}}, \overline{n}, \overline{t}) \notin \{(\boldsymbol{m}_1, n_1, t_1), \ldots, (\boldsymbol{m}_q, n_q, t_q)\}$ by the construction, as $\overline{n}$ is distinct form other nonces. Hence, we could remove the irrelevant if sentence from $\mathcal{G}_2$. Next, note that the keys of the final challenge are created after $\mathcal{A}$ outputs its forgery $\overline{\boldsymbol{m}}, \overline{n}, \overline{t}$. As a result, it is easy to see that $\mathcal{A}$ conducts impersonation attack against h. By the security of polynomial message authentication code

$$\Pr\left[\mathcal{G}_2^{\mathcal{A}} = 1\right] \leq \frac{1}{|\mathbb{F}|} \ .$$

The second case when the nonce $\overline{n}$ coincides with n_j for some $j \in \{1, \ldots, n_q\}$ is a bit more difficult to analyse. By the construction the keys $\overline{k}_1, \overline{k}_0$ generated to check the message tag $\overline{t}$ coincide with the keys $k_0^{(j)}, k_1^{(j)} = f(n_j)$. The corresponding situation is modelled by the following game:

$\mathcal{G}_{2+j}^{\mathcal{A}}$

$n_0 \leftarrow 0,\ t_0 \leftarrow \perp$
For $i \in \{1, \ldots, q\}$ do
  $\boldsymbol{m}_i \leftarrow \mathcal{A}(n_{i-1}, t_{i-1})$
  $n_i \leftarrow n_{i-1} + 1$
  $k_1^{(i)}, k_0^{(i)} \leftarrow \mathbb{F}$
  $t_i \leftarrow h(\boldsymbol{m}_i, k_1, k_0)$
$\overline{\boldsymbol{m}}, \overline{n}, \overline{t} \leftarrow \mathcal{A}(n_q, t_q)$
if $\overline{n} \neq n_j$ then **return** 0
if $\overline{\boldsymbol{m}} = \boldsymbol{m}_j$ then **return** 0
return $[\overline{t} \stackrel{?}{=} h(\overline{\boldsymbol{m}}, k_1^{(j)}, k_0^{(j)})]$

where the check $\overline{\boldsymbol{m}} = \boldsymbol{m}_j$ assures that the adversary does not submit a message and a tag that she has seen before. Now it is easy to see that $\mathcal{A}$ succeeds in the game $\mathcal{G}_{2+j}$ only if she manages to conduct a successful substitution attack against the polynomial message authentication code with keys $k_1^{(j)}, k_0^{(j)}$. Consequently

$$\Pr\left[\mathcal{G}_{2+j}^{\mathcal{A}} = 1\right] \leq \frac{\ell}{|\mathbb{F}|} \ .$$

Since one of the events $\overline{n} \notin \{n_1, \ldots, n_q\}$ or $\overline{n} = n_j$ must occur during a successful attack we get

$$\Pr\left[\mathcal{G}_1^{\mathcal{A}} = 1\right] = \Pr\left[\mathcal{G}_2^{\mathcal{A}} = 1\right] + \Pr\left[\mathcal{G}_3^{\mathcal{A}} = 1\right] + \cdots + \Pr\left[\mathcal{G}_{1+q}^{\mathcal{A}} = 1\right] \leq \frac{q\ell + 1}{|\mathbb{F}|} \ .$$

Consequently, we have proven that for all $(t - \mathrm{O}(q))$-time adversaries $\mathcal{A}$:

$$\Pr\left[\mathcal{G}^{\mathcal{A}} = 1\right] \leq \frac{q\ell + 1}{|\mathbb{F}|} + \varepsilon$$

and thus the CtrPolyMac is indeed computationally secure.

Refined analysis. The bound obtained above is very loose, since we do not take into account the probability that outputs the nonce in the right format. By taking this into account, we get much sharper bounds

$$\Pr\left[\mathcal{G}_2^{\mathcal{A}} = 1\right] = \Pr\left[\overline{n} \notin \{n_1, \ldots, n_q\}\right] \cdot \frac{1}{|\mathbb{F}|}$$
$$\Pr\left[\mathcal{G}_{2+j}^{\mathcal{A}} = 1\right] = \Pr\left[\overline{n} = j\right] \cdot \frac{\ell}{|\mathbb{F}|}$$

We emphasise that this is possible only because the security of polynomial message authentication code is unconditional. If we would have had computational security guarantees, then this type of success estimation would have been impossible—to get a bound, we have to construct a reduction but this would require efficient sampling over the runs which satisfy the condition $\overline{n} \notin \{n_1, \ldots, n_q\}$ or $\overline{n} = j$ and the latter is non-trivial task in if the computing power is limited. Now improved upper bounds allow us to obtain better bound

$$\Pr\left[\mathcal{G}_1^{\mathcal{A}} = 1\right] \leq \Pr\left[\overline{n} \notin \{n_1, \ldots, n_q\}\right] \cdot \frac{1}{|\mathbb{F}|} + \cdots + \Pr\left[\overline{n} = q\right] \cdot \frac{\ell}{|\mathbb{F}|} \leq \frac{\ell}{|\mathbb{F}|}$$

where the last inequality follows from the fact that maximum is larger than average. As a result we have obtained that if the running time of $\mathcal{A}$ is sufficiently small then

$$\Pr\left[\mathcal{G}^{\mathcal{A}} = 1\right] \leq \frac{\ell}{|\mathbb{F}|} + \varepsilon$$

and thus CtrPolyMac is indeed secure. □

Security analysis of RndPolyMac. The security game for RndPolyMac differs only by the way the nonce is computed:

$\mathcal{G}^{\mathcal{A}}$
$f \xleftarrow{u} \mathcal{F}$
$n_0 \leftarrow 0,\ t_0 \leftarrow \perp$
For $i \in \{1, \ldots, q\}$ do
  $\boldsymbol{m}_i \leftarrow \mathcal{A}(n_{i-1}, t_{i-1})$
  $n_i \leftarrow \mathbb{F} \times \mathbb{F}$
  $k_1, k_0 \leftarrow f(n_i)$
  $t_i \leftarrow h(\boldsymbol{m}_i, k_1, k_0)$
$\overline{\boldsymbol{m}}, \overline{n}, \overline{t} \leftarrow \mathcal{A}(n_q, t_q)$
if $(\overline{\boldsymbol{m}}, \overline{n}, \overline{t}) \in \{(\boldsymbol{m}_1, n_1, t_1), \ldots, (\boldsymbol{m}_q, n_q, t_q)\}$ then **return** 0
$\overline{k}_1, \overline{k}_0 \leftarrow f(\overline{n})$
return $[\overline{t} \stackrel{?}{=} h(\overline{\boldsymbol{m}}, \overline{k}_1, \overline{k}_0)]$.

It is easy to see that the analysis of CtrPolyMac would be sufficient if all nonces would be distinct. Since nonces are drawn randomly form a large set, the probability of nonce collisions is negligible:

$$\Pr\left[\exists i, j : n_i = n_j\right] = \binom{q}{2} \cdot \frac{1}{|\mathbb{F}|^2} = \frac{q(q-1)}{2\,|\mathbb{F}|^2} \ .$$

Secondly, it is straightforward to see that if we always query message pairs $(0,\ldots,0,0)$ and $(0,\ldots,0,1)$, then the colliding nonces allow us to restore both sub keys k_0 and k_1 and thus authenticate any message corresponding to this nonce. Consequently, the RndPolyMac scheme is insecure in case of nonce collisions. Thus we can estimate the success probability as follows:

$$\Pr\left[\mathcal{G}_1^{\mathcal{A}}=1\right]=\Pr\left[\mathcal{G}_1^{\mathcal{A}}=1\wedge\exists i,j:n_i=n_j\right]+\Pr\left[\mathcal{G}_1^{\mathcal{A}}=1\wedge\forall i,j:n_i\neq n_j\right]\ .$$

As the first probability can be bounded by the collision probability and the second by the the bounds from CtrPolyMac analysis, we obtain:

$$\Pr\left[\mathcal{G}^{\mathcal{A}}=1\right]\leq\frac{q(q-1)}{2\left|\mathbb{F}\right|^2}+\frac{\ell}{|\mathbb{F}|}+\varepsilon\ .$$

The latter concludes our security proof.