**Prof. Dr. Rudolf Mathar, Jose Calvo, Markus Rothe**

# Tutorial 6
# - Proposed Solution -

Friday, December 4, 2015

## Solution of Problem 1

Recall the RSA cryptosystem: $n = pq$, $p \neq q$ prime and $e \in \mathbb{Z}_{\varphi(n)}$ with $\gcd(e, \varphi(n)) = 1$. The public key is $(n, e)$.

Our pseudo-random generator based on RSA is:

**a)** Select a random seed $x_0 \in \{2, \ldots, n-1\}$.

**b)** Iterate: $x_{i+1} \equiv x_i^e \mod n$, $i = 0, \ldots, t$.

**c)** Let $b_i$ denote the last $h$ bits of $x_i$, where $h = \lfloor \log_2 \lfloor \log_2(n) \rfloor \rfloor$.

**d)** Return the pseudo-random sequence $b_1, \ldots, b_t$ of $h \cdot t$ pseudo-random bits.

## Solution of Problem 2

Given: two hash functions with output length of 64 bits and 128 bits.

**a)** How many messages have to be created, such that the probability of a collision exceeds 0.86?

Birthday paradox: $k$ objects, $n$ bins, $p_{k,n}$, the probability of "no collision", is bounded by

$$
\begin{aligned}
& p_{k,n} \leq \exp\left(-\frac{k(k-1)}{2n}\right) \\
\Rightarrow \quad & 1 - p_{k,n} \geq 1 - \exp\left(-\frac{k(k-1)}{2n}\right) \geq p \\
\Leftrightarrow \quad & \exp\left(-\frac{k(k-1)}{2n}\right) \leq 1 - p \\
\Leftrightarrow \quad & k^2 - k + 2n \log_e(1-p) \\
& = \left(k - \frac{1}{2} + \frac{1}{2}\sqrt{1 - 8n \log_e(1-p)}\right) \cdot \left(k - \frac{1}{2} - \frac{1}{2}\sqrt{1 - 8n \log_e(1-p)}\right) \geq 0
\end{aligned}
$$

With $n = 2^{64} \approx 1.844 \cdot 10^{19}$ and $p = 0.86$, we get $k_{64} \approx 8.517 \cdot 10^9$, and with $n = 2^{128} \approx 3.403 \cdot 10^{38}$, we get $k_{128} \approx 3.658 \cdot 10^{19}$, where $k_{64}$ and $k_{128}$ denote the number of messages needed to get a collision with probability of $p = 0.86$.

**b)** The following solution is an example and other solutions are possible. The main aspect of this exercise is to show the growth in resources for generating collisions the longer the hash function is.

| Hardware resource | 64 bit hash function | 128 bit hash function |
|---|---|---|
| hash function executions | $k_{64} = 8.517 \cdot 10^9$ | $k_{128} = 3.658 \cdot 10^{19}$ |
| memory size | $k_{64} \cdot 64$ bits $\approx 63.5$GiB | $k_{128} \cdot 128$ bits $= 5.45 \cdot 10^{11}$GiB |
| comparisons | $0 + 1 + 2 + \ldots + (k_{64} - 1)$ $= \sum_{i=0}^{k_{64}-1} i = \frac{1}{2} k_{64} (k_{64} - 1)$ $\approx 3.63 \cdot 10^{19}$ | $\frac{1}{2} k_{128} (k_{128} - 1) \approx 6.69 \cdot 10^{38}$ |

## Solution of Problem 3

$$C_i = M_{i+1} \oplus E_K(C_{i-1}), \quad i = 1, \ldots, n-1 \tag{1}$$
$$\mathrm{MAC}_K^{(n)} = E_K(C_{n-1}) \tag{2}$$
$$C_0 = M_1 \tag{3}$$
$$\hat{C}_i = E_K(\hat{C}_{i-1} \oplus M_i), \quad i = 1, \ldots, n-1 \tag{4}$$
$$\widehat{\mathrm{MAC}}_K^{(n)} = E_K(\hat{C}_{n-1} \oplus M_n) \tag{5}$$
$$\hat{C}_0 = 0 \tag{6}$$

We show that the equivalency

$$\mathrm{MAC}_K^{(n)} = \widehat{\mathrm{MAC}}_K^{(n)} \tag{7}$$

holds, by induction over $n$.

*Proof.* $n = 1$:

$$\mathrm{MAC}_K^{(1)} \overset{(2)}{=} E_K(C_0) \overset{(3)}{=} E_K(M_1) \overset{(6)}{=} E_K(\hat{C}_0 \oplus M_1) \overset{(5)}{=} \widehat{\mathrm{MAC}}_K^{(1)}$$

$n \to n+1$:

$$\begin{aligned}
\mathrm{MAC}_K^{(n+1)} &\overset{(2)}{=} E_K(C_n) \overset{(1)}{=} E_K(M_{n+1} \oplus E_K(C_{n-1})) \\
&\overset{(2)}{=} E_K\left(M_{n+1} \oplus \mathrm{MAC}_K^{(n)}\right) \\
&\overset{(7)}{=} E_K\left(M_{n+1} \oplus \widehat{\mathrm{MAC}}_K^{(n)}\right) \\
&\overset{(5)}{=} E_K\left(M_{n+1} \oplus E_K\left(\hat{C}_{n-1} \oplus M_n\right)\right) \\
&\overset{(4)}{=} E_K\left(M_{n+1} \oplus \hat{C}_n\right) \overset{(4)}{=} \hat{C}_{n+1} = \widehat{\mathrm{MAC}}_K^{(n+1)}
\end{aligned}$$

□

## Solution of Problem 4

**a)** Alice sends to Bob:

$$c = e(m \parallel h(k_2 \parallel m), k_1)\,.$$

Bob validates as follows:

- $d(c, k_1) = m' \parallel h(k_2 \parallel m)$
- compute $h(k_2 \parallel m')$ with the known key $k_2$
- verify $h(k_2 \parallel m) = h(k_2 \parallel m')$

Background:

- two keys are use to separate encryption and validation
- e.g., two keys can have different security levels
- encryption can be omitted if the message is not secret, but integrity is still ensured
- if a part of the key is lost, then the system is not entirely broken

**b)** *Method (i)*: Let both, $(K_1, L_1)$ and $(K_2, L_2)$ belong to Bob.

First, Alice sends the following to Bob:

$$c = e(m \parallel h(s \parallel m) \parallel e(s, K_2), K_1)\,.$$

Then, Bob validates:

- $d(c, L_1) = m \parallel h(s \parallel m) \parallel e(s, K_2)$
- $d(e(s, K_2), L_2) = s$
- compute $h(s \parallel m')$ with session key $s$
- verify $h(s \parallel m) = h(s \parallel m')$.

*Method (ii)*: Let $(K_A, L_A)$ belong to Alice and $(K_B, L_B)$ belong to Bob, then a possible protocol for message validation is.

- Bob sends to Alice: $c_1 = e(s, K_A)$
- Alice calculates: $d(c_1, L_A) = s$
- Alice then sends to Bob: $c_2 = e(m \parallel h(s \parallel m), K_B)$
- Bob calculates $d(c_2, L_B) = m \parallel h(s \parallel m)$, computes $h(s \parallel m')$ with session key $s$, and verifies that $h(s \parallel m) = h(s \parallel m')$.

There are many more protocols possible.

**c)** There is no authentication for Alice and Bob. Eve can easily impersonate Alice:

Eve sends the following $c$ to Bob using Bob's public key:

$$c = e(\tilde{m} \parallel h(s \parallel \tilde{m}) \parallel e(s, K_2), K_1)$$

with her own forged message $\tilde{m}$ impersonating Alice. Bob does not know that this message is actually from Eve.

To counteract this attack, the message must be securely linked to Bob's identity. This demand can be accomplished by a signature. For instance:

$$c = e(m \parallel sig_A(m) \parallel h(s \parallel sig_A(m) \parallel e(s, K_2), K_1).$$