Answers to the Exercises

2. Symmetric-Key Cryptography

1. If all keys are equal, then $C_0 = 0 \ldots 0$ or $C_0 = 1 \ldots 1$.
   We consider for example the bits at the positions 2,3,5,7,9,11,13,15,16,18,20,22,24,26,28,1 of $C_0$ and denote this sequence by $b_1, b_2, \ldots, b_{16}$.
   Bit $b_i$ appears as bit number 5 in $k_i$, $i = 1, \ldots, 16$. Thus we have $b_1 = b_2 = \ldots = b_{16}$, because all keys are equal. Additionally we consider the positions 3,4,6,8,10,12,14,16,17,19,21,23,25,27,1,2 of $C_0$. The $i$-th bit in this sequence is the 24-th bit of $k_i$. Thus all bits at these positions are equal. Position 3 appears in both cases. Thus all bits of $C_0$ are equal.
   Similar arguments show that $D_0 = 0 \ldots 0$ or $D_0 = 1 \ldots 1$.
   We obtain the four weak keys by combining the possible values of $C_0$ and $D_0$. If we apply $PC1$ to the four rows

   
   
   \[
   \begin{array}{cccccccc}
   01 & 01 & 01 & 01 & 01 & 01 & 01 & 01 \\
   FE & FE & FE & FE & FE & FE & FE & FE \\
   1F & 1F & 1F & 0E & 0E & 0E & 0E \\
   E0 & E0 & E0 & F1 & F1 & F1 & F1 
   \end{array}
   \]

   

   we see that the four rows are the weak keys of DES. Note that $PC1$ is a permutation on 56 bits. The bits in the positions 8,16,24,32,40,48,56,64 are not used.

2. a. Note that $\overline{k}$ yields $\overline{k}_i$, if $k$ yields $k_i$ and that $E(\overline{x}) = \overline{E(x)}$. Thus

   \[
   f(\overline{x}, \overline{k}) = P(S(E(\overline{x}) \oplus \overline{k})) = P(S(E(x) \oplus k)) = f(x, k)
   \]

   and

   \[
   \phi_i(\overline{x}, \overline{y}) = (\overline{x} \oplus f_i(\overline{y}), \overline{y})
   = (\overline{x} \oplus f(\overline{y}, \overline{k}_i), \overline{y})
   = (\overline{x} \oplus f(y, k_i), \overline{y})
   = (\overline{x} \oplus f_i(y), \overline{y})
   = (x \oplus f_i(y), \overline{y})
   = \phi_i(x, y).
   \]
Hence we get
\[
\begin{align*}
\text{DES}_k(x) &= \text{IP}^{-1}(\phi_{16}(\mu(\phi_{15}(\ldots \mu(\phi_2(\mu(\phi_1(\text{IP}(x))))))))) \\
&= \text{IP}^{-1}(\phi_{16}(\mu(\phi_{15}(\ldots \mu(\phi_2(\mu(\phi_1(\text{IP}(x))))))))) \\
&\quad \vdots \\
&= \text{IP}^{-1}(\phi_{16}(\mu(\phi_{15}(\ldots \mu(\phi_2(\mu(\phi_1(\text{IP}(x))))))))) \\
&= \text{DES}_k(x).
\end{align*}
\]

b. \(\text{DES}(k, x) = y\) implies
\[
\text{DES}(\bar{k}, x) = \text{DES}(\bar{k}, \bar{x}) = \overline{\text{DES}(k, x) = y}.
\]

Assume \(c = \text{DES}_k(m)\) and \(\bar{c} = \text{DES}_k(\bar{m})\) are known.
Choose \(k'\) and compute \(y = \text{DES}(k', \bar{m})\).

i. If \(y = \bar{c}\), then the key is \(k'\).

ii. If \(y = c\), then the key is \(\bar{k}\).

Thus, we can test the two keys \(k'\) and \(\bar{k}\) with one encryption.

3. We apply MixColumns to the four bytes of one column. Let \(M\) be the \(4 \times 4\) matrix, that defines the MixColumns step. Then \((I_4|M'),\) where \(I_4\) is the \(4 \times 4\) unit matrix, is the generator matrix of a linear code \(C\).
Each codeword consists of 8 bytes. \(C\) is a maximum distance separable (MDS) code, this means, that the minimum distance of \(C\) is 5. The 4 parity bytes of the 4 information bytes \((i_1, i_2, i_3, i_4)\) are computed by the multiplication \((i_1, i_2, i_3, i_4)M' = (i_1, i_2, i_3, i_4)'\). The distance between two codewords from \(C\) is at least 5, which means that two codewords differ at least in 5 positions. Thus if \(M\) is applied to two vectors, that differ in \(k\) bytes, \(1 \leq k \leq 4\), the outputs differ in at least \(5 - k\) bytes.
If we change one byte in the plaintext, then after the first round all 4 bytes of one column in the state are affected. Thanks to ShiftRows these 4 bytes are transferred to different columns in the second round. Then, after the application of MixColumns in the second round all bytes of the state are affected.

4. Let \(f : \{0,1\}^n \rightarrow \{0,1\}^n\) be a permutation, \(x_1\) an initial value and \(x_1, x_2, \ldots\) the sequence obtained by applying \(f\). Then there exists an \(i\) with \(f(x_i) \in \{x_1, \ldots, x_i\}\). Let \(j\) be the first \(i\) with this property. Since \(f\) is a permutation \(f(x_j) = x_1\). Otherwise an element would have two preimages. \((x_1, \ldots, x_j)\) is a cycle of \(f\). The average period of the key stream is the average length of a cycle of a randomly selected permutation.
Let \(S\) be a set, \(|S| = k\) and
\[
C_m = \{c \mid c \text{ is an cycle of length } m \text{ of a permutation on } S\}.
\]

The number of different cycles \((x_1, \ldots, x_m)\) is \(\frac{k(k-1) \ldots (k-m+1)}{m}\) (note \((x_1, \ldots, x_m), (x_2, \ldots, x_m, x_1), \ldots, (x_m, x_1, \ldots, x_{m-1})\) define the same cy-
5. Elements \((x, y)\) in the domain of \(f\) are bit-strings of length \(2(|q - 1|)\).

Elements in the range \(G_q \subset \mathbb{Z}_p^*\) are encoded as bit strings of length \(|p|\).

Since \(|q - 1| = |q| = |p| - 1\), we may consider \(f\) as a compression function.

Assume \((x_1, y_1), (x_2, y_2)\) is a collision of \(f\). Then \(g^{x_1} h^{y_1} = g^{x_2} h^{y_2}\). Thus \(g^{x_1 - x_2} = h^{y_2 - y_1}\). If \(y_1 = y_2\), then \(x_1 = x_2\) and \((x_1, y_1) = (x_2, y_2)\). This is a contradiction, since \((x_1, y_1), (x_2, y_2)\) cannot be equal, as a collision of \(f\). Thus \(y_1 \neq y_2\). We get \(\log_g h = \frac{x_1 - x_2}{y_2 - y_1}\).

6. All bits of the columns defined by \(x = 3, z = 0\) and \(x = 1, z = 63\) and the bit at \(x = 0, y = 0, z = 63\) are 1 bits. All other bits are 0 bits.

7. Given a value \(v \in \{0,1\}^n\), we randomly select messages \(m \in \{0,1\}^*\) and check, if \(h(m) = v\). We don’t have to check, if we have selected a message twice. The probability for this event is negligibly small (note that the set of messages is infinite). The probability that \(h(m) = v\) is \(1/2^n\). Lemma B.12 – with \(p = 1/2^n\) – says that we expect to select \(2^n\) messages \(m\) until \(h(m) = v\). The expected number of steps does not depend on \(v\), and we conclude immediately that the expected number of steps in the brute-force attack against the one-way property of \(h\) is \(2^n\). To attack second pre-image resistance, we consider a message \(m' \in \{0,1\}^*\).

Let \(v = h(m')\). We randomly select messages \(m \in \{0,1\}^*, m \neq m'\) and check, if \(h(m) = v\). As before, the probability that \(h(m) = v\) is \(1/2^n\), and the expected number of steps is \(2^n\).

8. a. Let \(M||IV\) be the MAC-code for \(m_1||m_2||\ldots||m_l\). Then \(M||IV'\) is the MAC-code for \(m_1 \oplus IV \oplus IV'||m_2||\ldots||m_l\). Thus, the adversary Eve can replace the first block \(m_1\) by a block of her choice.

b. If \(c_1||c_2||\ldots||c_l\) is the MAC-code for \(m_1||m_2||\ldots||m_l\). Then \(c_1||c_2||\ldots||c_l\) is a MAC-code for \(m_1||m_2||\ldots||m_i, 1 \leq i \leq l\).
c. If $a$ and $b$ are different messages with (identical) MAC-code $M$, then $a\|c$ and $b\|c$ also have identical MAC-codes for every message $c$. 

Introduction to Cryptography by H. Delfs and H. Knebl
3. Public-Key Cryptography

2. By the Chinese remainder theorem we have

\[ \mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^* \]

and \( \mu \) decomposes into

\[ (\mu_1, \mu_2): \mathbb{Z}_p^* \times \mathbb{Z}_q^* \to \mathbb{Z}_p^* \times \mathbb{Z}_q^*, (x_1, x_2) \mapsto (x_1^e, x_2^e) \]

\( \mu \) is an isomorphism if and only if \( \mu_1, \mu_2 \) are isomorphisms. Now

\[ \mu_1: \mathbb{Z}_p^* \to \mathbb{Z}_p^*, x \mapsto x^e \]

and

\[ \mu_2: \mathbb{Z}_q^* \to \mathbb{Z}_q^*, x \mapsto x^e \]

are isomorphisms if and only if \( \gcd(e, p-1) = 1 \) and \( \gcd(e, q-1) = 1 \). This implies the assertion.

3. Let \( g \) be a primitive root in \( \mathbb{Z}_p^* \)

\[ \text{Exp}: \mathbb{Z}_p^* \to \mathbb{Z}_p^*, \alpha \mapsto g^\alpha \]

is an isomorphism of groups. Let \( k \in \mathbb{N}, x \in \mathbb{Z}_p^* \) and \( x^k = 1 \). Then \( x = g^\nu \) and \( x^k = g^{\nu k} \). Hence \( p - 1 \) divides \( \nu k \). This implies

\[ \left| \{x \in \mathbb{Z}_p^* \mid x^k = 1\} \right| = \left| \{\nu \in \mathbb{Z}_{p-1} \mid \nu k \equiv 0 \mod (p-1)\} \right| = \left| \left\{ \frac{p-1}{d} l \mid 1 \leq l \leq d \right\} \right| = d, \]

where \( d = \gcd(k, p-1) \).

Now \( \mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^* \) and \( x^{e-1} = 1 \) if and only if \( (x_1^{e-1}, x_2^{e-2}) = (1, 1) \), where \( x_1 = x \mod p \) and \( x_2 = x \mod q \). This implies

\[ \left| \{x \in \mathbb{Z}_n^* \mid \text{RSA}_{n,e}(x) = x\} \right| = \gcd(e - 1, p - 1) \gcd(e - 1, q - 1). \]

4. Compute \( \lambda = ed - 1, \lambda = 2^tm, m \text{ odd} \). \( \lambda \) is a multiple of \( \varphi(n) \). Thus \( [a^\lambda] = 1 \) for all \( [a] \in \mathbb{Z}_n^* \). Let

\[ W_n := \left\{ [a] \in \mathbb{Z}_n^* \mid a^m \equiv 1 \mod n \right\} \]

or there is an \( i, 0 \leq i \leq t - 1 \), with \( a^{2^im} \equiv -1 \mod n \).

Let \( [a] \notin W_n \). Then there is an \( i, 0 \leq i \leq t - 1 \), with \( a^{2^{i+1}m} \equiv 1 \mod n \) and \( a^{2^i m} \not\equiv \pm 1 \mod n \). Then \( [a^{2^im}] \) and \([1]\) are square roots of \([1]\), and the

\[ H. Delfs and H. Knebl \]

©H. Delfs and H. Knebl
factors of $n$ can be computed by the Euclidean algorithm (Lemma A.69). Let $W_n := Z_n^* \setminus W_n$ be the complement of $W_n$. Then $|W_n| \geq \frac{\varphi(n)}{2}$ (see below). Hence, choosing a random $[a] \in Z_n^*$ we can compute the factors of $n$ in this way with probability $\geq 1/2$, since $[a]$ is not in $W_n$ with a probability $\geq 1/2$. Repeating the random choice $t$-times, if necessary, we can increase the probability of success to $\geq 1 - 2^{-t}$.

It remains to show that $W_n' := \{[a] \in Z_n^* \mid a^{2^m} \equiv -1 \mod n\}$.

$W_n'$ is not empty, since $[-1] \in W_n'$. Let $r = \max\{i \mid W_n^i \neq \emptyset\}$ and $U := \{a \in Z_n^* \mid a^{2^m} \equiv \pm 1 \mod n\}$.

$U$ is a subgroup of $Z_n^*$ and $W_n \subset U$.

Let $[x] \in W_n'$. By the Chinese Remainder Theorem A.30, there is a $[w] \in Z_n^*$ with $w \equiv x \mod p$ and $w \equiv 1 \mod q$. Then $w^{2^m} \equiv -1 \mod p$ and $w^{2^m} \equiv +1 \mod q$, hence $w^{2^m} \not\equiv \pm 1 \mod n$. Thus, $w \not\in U$, and we see that $U$ is indeed a proper subgroup of $Z_n^*$. Thus $|W_n| \leq \frac{\varphi(n)}{2}$.

6. a. Let $R_{p'} := \{x \in Z_p^* \mid p' \text{ does not divide } \text{ord}(x)\}$.

Note that

i. $p'$ does not divide $\text{ord}(x)$, if and only if $\text{ord}(x)$ divides $a$, and that

ii. $p'$ divides $\nu$, where $\nu$ is defined by a representation $x = g^\nu$ of $x$ with a primitive root $g$.

Thus,

$$|R_{p'}| = |\{x \in Z_p^* \mid \text{ord}(x) \mid a\}| = |\{g^{p'\nu} \mid 1 \leq \nu \leq a\}| = a.$$

b. Let $R_{p'q'} := \{x \in Z_n^* \mid p'q' \text{ does not divide } \text{ord}(x)\}$.

Note $\text{ord}(x)$ is the least common multiple of $\text{ord}(x \mod p)$ and $\text{ord}(x \mod q)$.

$p'q'$ does not divide $\text{ord}(x)$, if and only if $p'$ does not divide $\text{ord}(x \mod p)$ or $q'$ does not divide $\text{ord}(x \mod q)$.

By the Chinese Remainder Theorem, we have

$$Z_p^* \times R_{q'} \cup R_{p'} \times Z_q^* = R_{p'q'},$$

$$Z_p^* \times R_{q'} \cap R_{p'} \times Z_q^* = R_{p'} \times R_{q'}.$$

This implies $|R_{p'q'}| = (p-1)b + a(q-1) - ab$ and

$$\frac{|R_{p'q'}|}{\varphi(n)} = \frac{(p-1)b + a(q-1) - ab}{ap'bq'} = \frac{1}{p'} + \frac{1}{q'} - \frac{1}{p'q'}. $$

7. a. RSA_{n,e}(x) = x^{e^l} = x \text{ if } e^l \equiv 1 \mod \varphi(n). \text{ The last condition is satisfied for } l = \varphi(\varphi(n)).$

Introduction to Cryptography by H. Delfs and H. Knebl
3. Public-Key Cryptography

b. \( x^{e^i} = x \) is equivalent to \( x^{e^i-1} = 1 \). The last equation is equivalent to \( e^i \equiv 1 \mod \text{ord}(x) \).

To prevent the decryption-by-iterated-encryption attack, it is required that \( \text{ord}(e \mod \text{ord}(x)) \) is large for \( x \) and \( e \).

We show that the set of “exceptions”,

\[
\{(x, e) \in \mathbb{Z}_n^* \times \mathbb{Z}_{\varphi(n)}^* \mid \text{ord}(e \mod \text{ord}(x)) < p''q'' \}
\]

is an exponentially small subset of \( \mathbb{Z}_n^* \times \mathbb{Z}_{\varphi(n)}^* \). The frequency of elements \( x \in R_{p'}q'' \) (see Exercise 6) is exponentially small. Let \( x \notin R_{p'}q'' \). Then \( n' = p'q' \) divides \( k \), \( k := \text{ord}(x) \). Then \( \text{ord}(e \mod n') \) divides \( \text{ord}(e \mod k) \).

Thus, if \( p''q'' \) divides \( \text{ord}(e \mod n') \) then \( p''q'' \) divides \( \text{ord}(e \mod k) \) and \( \text{ord}(e \mod k) \) is large.

Let \( R_{p''q''} := \{ e \in \mathbb{Z}_n^* \mid p''q'' \text{ does not divide } \text{ord}(e) \} \).

Let \( f \) be the frequency of elements \( e \in R_{p''q''} \).

By Exercise 6, \( f = \frac{1}{p'} + \frac{1}{q'} - \frac{1}{p''q''} \) is exponentially small. The discussion shows that \( p''q'' \) is a lower bound for the number of iterations of the repeat-until loop for all \( (x, e) \) outside an exponentially small subset of \( \mathbb{Z}_n^* \times \mathbb{Z}_{\varphi(n)}^* \).

9. a. We assume that it is possible to compute discrete logarithms in \( H \) and \( y' \in H \).

b. The verification condition in ElGamal’s signature scheme is \( g^m = y^t r^s \).

\[
y^t r^s = y^t r^s = y^t r^s = g^t z r^s
\]

\[
= g^{t z (p-3)(m-tz)/2} = g^{t z (t(p-1)/2 t^{-1}) m-tz}
\]

\[
= g^{t z (-t^{-1}) m-tz} = g^{t z m-tz} = g^m.
\]

Note that \( gt = p-1 = -1 \mod p \) implies \( t = -g^{-1} \) and \( g = -t^{-1} \) and that

\[
t(p-1)/2 = (-g)^{(p-1)/2} = (-1)^{(p-1)/2} g^{-(p-1)/2} = 1 \cdot (-1) = -1
\]

\( (g^{-(p-1)/2} = -1, \text{ since } g \text{ is a primitive root}) \).

c. The above attack does not work in the DSA signature scheme.

12. The domain parameters \((F_q, a, b, Q, n, h)\) are assumed to be publicly known. Let \((d_A, P_A)\) be Alice’s key pair.

**Signing.** Messages \( m \) to be signed must be elements in \( \mathbb{Z}_n \). In practice, a hash function \( H \) is used to map real messages to elements of \( \mathbb{Z}_n \). The signed message is produced by Alice using the following steps:

a. Alice selects a random integer \( k, 1 \leq k \leq n-1 \).

© H. Delfs and H. Knebl
b. She computes $kQ := (x_{kQ}, y_{kQ})$ and converts $x_{kQ}$ to an integer and sets $\tilde{r} := x_{kQ} \mod n$.

c. She sets $r := (\tilde{r} + m) \mod n$. If $r = 0$, she returns to step 1.

d. She sets $s := (k - rd_A) \mod n$.

e. $(m, r, s)$ is the signed message.

**Verification.** Bob verifies the signed message $(m, r, s)$ as follows:

a. He verifies that $1 \leq r \leq n - 1$ and $0 \leq s \leq n - 1$; if not, then he rejects the signature.

b. He computes $R := sQ + rP_A$ ($P_A$ is the public key of Alice). If $R = O$ the signature is rejected.

c. Let $R = (x_R, y_R)$ and convert $x_R$ as an integer. The signature is accepted if $x_R = (r - m) \mod n$; otherwise it is rejected.

If Alice signed the message $(m, r, s)$, we have $x_R \equiv (r - m) \mod n$:

$$R = sQ + rP$$
$$= (k - rt)Q + (\tilde{r} + m)tQ$$
$$= (k - rt + \tilde{rt} + mt)Q$$
$$= (k + t(\tilde{r} + m - r))Q$$
$$= kQ.$$  

Hence we have $x_R = x_{kQ} \equiv (r - m) \mod n$.

In the Nyberg–Rueppel signature scheme the message $m$ can be recovered from the signature $(r, s)$: $m = r - x_R \mod n$.

The Nyberg–Rueppel signature scheme is existentially forgeable when used without a hash function. An adversary Eve can construct a message $m$ and a valid signature $(r, s)$ on $m$. She selects $r, s, 1 \leq r \leq n - 1$ and $0 \leq s \leq n - 1$, at random. Sets $R = rQ + sP_A$ and $m = r - x_R \mod n$. Thus, $x_R = r - m \mod n$ and $(m, r, s)$ is a valid signed message. If used with a cryptographic hash function $H$, adversary Eve has to find some meaningful message $\tilde{m}$ with $H(\tilde{m}) = m$. Since $H$ has the one-way property, this is practically infeasible.
4. Cryptographic Protocols

1. With this protocol the simple man-in-the-middle attack does not work. A more sophisticated attack is necessary. If the adversary Eve selects $e$ and declares $y^e_A$ as her public key, a man-in-the-middle attack works:
   a. Eve intercepts $c$ and forwards it unchanged to Bob.
   b. Eve intercepts $d$ and forwards $d^e$ to Alice.

Then Alice computes $k = d^{x_A}g^y_A = g^{kex_A}$ with Bob. Whereas Bob believes that he shares $k = c^{x_B}y^e_B = g^{kex_B}$ with Eve. Eve cannot compute the session key $k$. However, she can masquerade as Alice.


OneOfTwoSquareRoots($x_1, x_2$)

Case: Peggy knows a square root $y_1$ of $x_1$ (the other case follows analogously):
   1. Peggy chooses $r_1, r_2 \in \mathbb{Z}^*_n$ and $e_2 \in \{0, 1\}$ at random and sets $a = (a_1, a_2) = (r_1^2, r_2^2x_2^{e_2})$. Peggy sends $a$ to Vic.
   2. Vic chooses $e \in \{0, 1\}$ at random. Vic sends $e$ to Peggy.
   3. Peggy computes

   $e_1 = e \oplus e_2,$
   $b = (b_1, b_2) = (r_1y_1^{e_1}, r_2)$

   and sends $b, e_1, e_2$ to Vic.
   4. Vic accepts, if and only if

   $e = e_1 \oplus e_2,$
   $b_1^2 = a_1x_1^{e_1}, b_2^2 = a_2x_2^{e_2}.$

The completeness, soundness and zero-knowledge properties are analogously proven as in Protocol 4.5.

3. a. Let $x \in \text{QNR}_n^{+1}$. $a = r^2x^\sigma \in \text{QR}_n \iff \sigma = 0$. Thus $\sigma = \tau$ and Vic will accept.
   b. Let $x \in \text{QR}_n$. Then $a = r^2x^\sigma \in \text{QR}_n$, for $\sigma \in \{0, 1\}, r \in \mathbb{Z}^*_n$. Thus $\tau$ is always 1 and prob($\sigma = \tau$) = 1/2. Thus a dishonest Peggy can convince Vic with probability 1/2 if $x \in \text{QR}_n$.
   c. Let $V^*$ be a dishonest verifier defined by the following

Protocol 4.2.

PQR$_n$

1. $V^*$ randomly chooses $r \in \mathbb{Z}^*_n$ with $(\frac{x}{n}) = 1$ and sends $a = r$ to Peggy.
2. Peggy computes $\tau := \begin{cases} 0 & \text{if } a \in \text{QR}_n \\ 1 & \text{if } a \notin \text{QR}_n \end{cases}$ and sends $\tau$ to Vic.
3. $V^*$ outputs $\tau$.

Note $\tau = 0$ if $r \in \text{QR}_n$ and $\tau = 1$ if $r \notin \text{QR}_n$. Thus $V^*$ can decide after interaction with Peggy, whether a randomly chosen $r$ is a quadratic residue. Without Peggy’s help he cannot do this according to the quadratic residuosity assumption (see Section 4.3.1 and Definition 6.11).

d. **Algorithm 4.3.**

```c
int S(int x)
1. select $r \in Z_n^*$ and $\sigma \in \{0, 1\}$ uniformly at random
2. return $(\tilde{a}, \tilde{\tau}) \leftarrow (r^2 x^\sigma, \sigma)$
```

By construction, the random variables $S(x)$ and $(P, V)(x)$ are identically distributed for $x \in \text{QNR}_n$.

e. Vic proofs to Peggy after step 1 that he knows a square root of $a$ or of $a/x$ by using the protocol of Exercise 2. He can only succeed, if he followed the protocol in step 1. Thus he is a honest verifier and d) applies.

4. The idea is as in Exercise 3e). The verifier proves that he follows the protocol in step 1, i.e., that he sends a message which he encrypted with the public key. For this purpose, he shows that he knows the $e$-th root of the message he transmitted.

To show that a prover Peggy knows the $e$-th root $x$ of $y$, the following protocol may be used.

**Protocol 4.4.**

$e$-th root($y$)

1. Peggy chooses $r \in Z_n^*$ at random and sets $a = r^e$. Peggy sends $a$ to Vic.
2. Vic chooses $\sigma \in \{0, 1\}$ at random. Vic sends $\sigma$ to Peggy.
3. Peggy computes $b = rx^\sigma$ and sends $b$ to Vic, i.e., Peggy sends $r$, if $e = 0$, and $rx$, if $\sigma = 1$.
4. Vic accepts, if and only if $b^e = ay^\sigma$.

The completeness, soundness and zero-knowledge properties are analogously proven as in Protocol 4.5.

5. a. Alice commits to 0, if $c \in \text{QR}_n$ and to 1, if $c \notin \text{QR}_n$.

Note: $c \in \text{QR}_n \iff -c \notin \text{QR}_n$.

b. $c_1 c_2 = r_{1,r_2}^2 (-1)^{b_1 + b_2 mod 2} = (r_{1,r_2})^2 (-1)^{b_1 \oplus b_2}$

c. $c_1$ and $c_2$ commit to the same value, if $c_1 c_2 \in \text{QR}_n$. They commit to different values, if $c_1 c_2 \notin \text{QR}_n$. Both cases can be proven by zero-knowledge proofs (see Section 4.2.4 and Exercise 3).

6. The access structure can be realized, if $P_1$ gets three shares, $P_2$ two shares and $P_3, P_4, P_5$ and $P_6$ each get one share in a $(5, n)$-Shamir threshold scheme.

Introduction to Cryptography by H. Delfs and H. Knebl
7. Assume $P_i$ has $p_i$ shares of a $(t,n)$-Shamir threshold scheme. Then $p_1 + p_2 \geq t$ and $p_3 + p_4 \geq t$. Thus $p_1 + p_2 + p_3 + p_4 \geq 2t$. $p_1 + p_2 < t$ implies $p_3 + p_4 \geq t$. Thus $\{P_1, P_3\}$ or $\{P_2, P_4\}$ are also able to reconstruct the secret.

8. We use the notations of Section 4.5. The encryption scheme allows to encrypt every message $m = g^v, 0 \leq v \leq q - 1$. Thus, a voter could encrypt up to $(q - 1)/2$ “yes-” or “no-votes”. If an authority posts $w_jg$ or $w_jg^{-1}$, the tally is decreased or increased by $\lambda_{i,j}$.

9. The voter $V_j$ can duplicate the vote $c_i = (c_{i,1}, c_{i,2})$ of the voter $V_i$. For this purpose, he selects $\alpha$ and sets $c_j = (c_{i,1}g^\alpha, c_{i,2}h^\alpha)$. He has to prove that his vote is a correctly formed one, by the protocol OneOfTwoPairs (Protocol 4.20). We first discuss the case, where the interactive version of the proof is applied.

a. The voter $V_j$ can derive from the voter $V_i$’s proof $(a, d, b) = OneOfTwoPairs(g, h, (y_1, z_1), (y_2, z_2))$, where

$$
y_1 = c_{i,1}, \ z_1 = c_{i,2}g, \ y_2 = c_{i,1}, \ z_2 = c_{i,2}g^{-1},$$

$$a = (a_1, a_2, a_3, a_4),$$

$$d = (d_1, d_2), \ b = (b_1, b_2),$$

the proof

$$(\tilde{a}, \tilde{d}, \tilde{b}) = OneOfTwoPairs(g, h, (\tilde{y}_1, \tilde{z}_1), (\tilde{y}_2, \tilde{z}_1)),$$

where

$$\tilde{y}_1 = y_1g^\alpha, \ \tilde{z}_1 = z_1h^\alpha$$

$$\tilde{y}_2 = y_2g^\alpha, \ \tilde{z}_2 = z_2h^\alpha$$

$$\tilde{a} = a, \ \tilde{d} = d,$$

$$\tilde{b} = (b_1 - d_1\alpha, b_2 - d_2\alpha).$$

b. With the non-interactive proof, the attack does not work. Replacing the argument $(y_i, z_i, i = 1, 2)$ of the hash function will cause a different output. Note, the hash function is assumed to be collision resistant. To duplicate a vote, an identical copy of the ballot must be used. However, it will be detected, if a ballot is posted twice.

10. Given a public key $pk = (n, g)$ and a ciphertext $c$, Peggy interactively proves to Vic that she knows a plaintext $m \in \mathbb{Z}_n$ and a randomness value $\rho \in \mathbb{Z}_n^*$ such that $c = E_{pk}(m, \rho) = g^m\rho^\alpha$. $(m, \rho)$ is Peggy’s witness.

Let $r \in \mathbb{N}, r < \varphi(n)$ be a large number, e.g., $r := \sqrt{n}$. 

©H. Delfs and H. Knebl
Protocol 4.5.
KnowledgeOfThePlaintext(c)

a. Peggy randomly chooses a plaintext \( m' \in \mathbb{Z}_n \) and a randomness \( \rho' \in \mathbb{Z}_n^* \), computes \( c' = E_{pk}(m', \rho') = g^{m' \rho'} \) and sends \( c' \) to Vic.

b. Vic randomly selects a challenge \( e \in \{0, 1, \ldots, r\} \) and sends \( e \) to Peggy.

c. Peggy computes \( \tilde{m} := em + m' \mod n \) and \( \tilde{\rho} := \rho^e \rho' \mod n \) and sends \( \tilde{m}, \tilde{\rho} \) to Vic.

d. Vic accepts if \( c^e c' = E_{pk}(\tilde{m}, \tilde{\rho}) = g^{\tilde{m} \tilde{\rho}} \).

The protocol is a special honest-verifier zero-knowledge proof of knowledge, as the following arguments show.

Obviously, the interactive proof is complete. To prove soundness, assume that a prover \( P^* \), honest or not, is able to convince Vic with probability \( > \frac{1}{r} \). Then, \( P^* \) is able to send an element \( c' \) in the first move, such that she is able to compute correct answers \( (\tilde{m}_1, \tilde{\rho}_1) \) and \( (\tilde{m}_2, \tilde{\rho}_2) \) for at least two distinct challenges \( e_1, e_2 \in \{1, \ldots, r\}, e_1 > e_2 \). Since Paillier is homomorphic, \( \tilde{m}_1 - \tilde{m}_2 \) is the plaintext \( (e_1 - e_2) m \) in \( c^{e_1 - e_2} \). Therefore, \( P^* \) immediately derives

\[
    m = (\tilde{m}_1 - \tilde{m}_2) \cdot (e_1 - e_2)^{-1},
\]

which means that \( P^* \) is an honest prover, who knows the plaintext. If \( (e_1 - e_2) \) were not a unit mod \( n \), then \( P^* \) would be able to crack the secret key by computing \( \gcd(e_1 - e_2, n) \).

Given a challenge \( e \), the following simulator \( S \) generates correctly distributed accepting transcripts \( (c', e, \tilde{m}, \tilde{\rho}) \). \( S \) randomly generates elements \( \tilde{m} \in \mathbb{Z}_n, \tilde{\rho} \in \mathbb{Z}_n^* \) and sets \( c' := E_{pk}(\tilde{m}, \tilde{\rho}) = g^{\tilde{m} \tilde{\rho}} \). Thus, the protocol is a special honest-verifier zero-knowledge protocol.

BlindRSASig(m)

1. Vic randomly chooses \( r \in \mathbb{Z}_n^* \), computes \( m = r^m \) and sends it to Peggy.

2. Peggy computes \( \sigma = m^d \) and sends it to Vic.

3. Vic computes \( \sigma r^{-1} \) and gets the signature of \( m \).

12. a. \( ry^r g^{-s} = mg^k g^{ex} g^{-(xr+k)} = m \).

b. Choose any \( r, s \) with \( 1 \leq r \leq p - 1 \) and \( 1 \leq s < q - 1 \) and let \( m := ry^r g^{-s} \). Then \( (m, r, s) \) is a signed message. This kind of attack is always possible, if the message can be recovered from the signature, as in the basic Nyberg–Rueppel scheme.

c. Use a collision-resistant hash function \( h \) and hash before encrypting, or, if you want to preserve the message recovery property, apply a suitable bijective redundancy function \( R \) to the message to be signed (see [MenOorVan96]).
Let \((m, r, s)\) be a valid signature. Without the first check, an attacker may sign messages \(\tilde{m}\) of his choice. He computes 
\[
g^{k} = r m^{-1}
\]
by the extended Euclidean algorithm. Then, he uses the Chinese remainder theorem to determine a \(\tilde{r} \in \mathbb{Z}\) with 
\[
\tilde{r} \equiv \tilde{m} g^k \mod p \quad \text{and} \quad \tilde{r} \equiv r \mod q.
\]
Then \((\tilde{m}, \tilde{r}, s)\) passes the verification, if 
\[
1 \leq \tilde{r} \leq p - 1
\]
is not checked.

**ProofRep\((g_1, g_2, y)\)**

1. Peggy randomly chooses \(r_1, r_2 \in \mathbb{Z}_q\), computes 
\[
a = g^{r_1} g^{r_2}
\]
and sends it to Vic.
2. Vic randomly chooses \(c \in \mathbb{Z}_q\) and sends it to Peggy.
3. Peggy computes 
\[
b_i = r_i - c x, \quad i = 1, 2,
\]
and sends \((b_1, b_2)\) to Vic.
4. Vic accepts the proof, if 
\[
a = g_1^{b_1} g_2^{b_2} y^c,
\]
otherwise, he rejects it.
1. Peggy randomly chooses $r_1, r_2 \in \mathbb{Z}_q$, computes $\pi = g_1^{r_1} g_2^{r_2}$ and sends it to Vic.

2. Vic randomly chooses $u \in \mathbb{Z}_q^*$, $v_1, v_2, w \in \mathbb{Z}_q$ and computes
   
   \begin{align*}
   a &= \pi^u g_1^{v_1} g_2^{v_2} y^w, \\
   c &= h(m||a), \tau = (c - w)u^{-1}.
   \end{align*}

   Vic sends $\tau$ to Peggy.

3. Peggy computes $b = (b_1, b_2) = (r_1 - cx_1, r_2 - cx_2)$ and sends it to Vic.

4. Vic verifies whether
   
   \[ \pi = g_1^{b_1} g_2^{b_2} y^\tau, \]

   computes $b = (b_1, b_2) = (u\bar{b}_1 + v_1, u\bar{b}_2 + v_2)$ and gets the signature $\sigma(m) = (c, b)$ of $m$.

The verification condition for a signature $(c, b)$ is $c = h(m||g_1^{b_1} g_2^{b_2} y^\tau)$. 

Introduction to Cryptography by H. Delfs and H. Knebl
5. Probabilistic Algorithms

1. The desired Las Vegas algorithm works as follows:
   Repeat
   1. Compute \( y = A(x) \).
   2. Check by \( D(x, y) \), whether \( y \) is a correct solution for input \( x \).
   3. If the check yields 'yes', then return \( y \) and stop. Otherwise, go back to 1.

   The expected number of iterations is \( 1/\text{prob}(A(x) \text{ correct}) \) (by Lemma B.12) and hence \( \leq P(|x|) \). The binary length of an output \( y \) is bounded by \( R(|x|) \). Thus, the running time of \( D(x, y) \) is bounded by \( S(|x| + R(|x|)) \).

2. We define the algorithm \( \tilde{A} \) on input \( x \) as follows:
   a. Let \( t(x) := \tilde{P}(|x|)^2Q(|x|) \).
   b. Compute \( A(x) t(x) \)-times, and obtain the results \( b_1, \ldots, b_{t(x)} \in \{0, 1\} \).
   c. Let
   \[
   \tilde{A}(x) := \begin{cases} 
   +1 & \text{if } \frac{1}{t(x)} \sum_{i=1}^{t(x)} b_i \geq a \\
   0 & \text{if } \frac{1}{t(x)} \sum_{i=1}^{t(x)} b_i < a.
   \end{cases}
   \]

   From Corollary B.25 applied to the \( t(x) \) independent computations of \( A(x) \), we get for \( x \in \mathcal{L} \)
   \[
   \text{prob} \left( \frac{1}{t(x)} \sum_{i=1}^{t(x)} b_i < a \right) < \frac{P(|x|)^2}{4t(x)} < \frac{1}{Q(|x|)},
   \]
   and for \( x \notin \mathcal{L} \)
   \[
   \text{prob} \left( \frac{1}{t(x)} \sum_{i=1}^{t(x)} b_i \geq a \right) < \frac{P(|x|)^2}{4t(x)} < \frac{1}{Q(|x|)}.
   \]

3. a. The probability that \( A(x) \) returns at least one 1 during \( t \) executions of \( A(x) \) is 0 if \( x \notin \mathcal{L} \), and \( > 1 - (1 - 1/Q(|x|))^t \) if \( x \in \mathcal{L} \). For \( t \geq \ln(2)Q(|x|) \), we have \( (1 - 1/Q(|x|))^t \leq 1/2 \) (see proof of Proposition 5.7).

   b. Consider an \( \mathcal{NP} \)-problem \( \mathcal{L} \) and a deterministic polynomial algorithm \( M(x, y) \) that answers the membership problem for \( \mathcal{L} \) with certificates of length \( \leq L(|x|) \). Selecting \( y \in \{0, 1\}^{L(|x|)} \) by coin tosses and calling \( M(x, y) \), we get a probabilistic polynomial algorithm \( A(x) \) with deterministic extension \( M(x, y) \). Conversely, a probabilistic polynomial algorithm \( A \) that decides the membership in \( \mathcal{L} \) yields a deterministic \( M \). These considerations show that a problem \( \mathcal{L} \) is in \( \mathcal{NP} \) if and only if there is a probabilistic polynomial algorithm \( A \) with values in \( \{0, 1\} \).
such that \( \text{prob}(A(x) = 1) > 0 \) if \( x \in \mathcal{L} \), and \( \text{prob}(A(x) = 1) = 0 \) if \( x \notin \mathcal{L} \).

Now, the inclusion \( \mathcal{RP} \subseteq \mathcal{NP} \) is obvious (to obtain this inclusion, only the 'conversely'-direction of our considerations is necessary).

4. Let \( A(x) \) be a Las Vegas algorithm for the membership in a \( \mathcal{ZPP} \)-problem \( \mathcal{L} \), and let \( P(|x|) \) be a polynomial bound for the expected running time of \( A \). We define a Monte Carlo algorithm \( \tilde{A}(x) \) as follows. We call \( A(x) \). If \( A(x) \) returns after less than \( P(|x|) \) steps, we set \( \tilde{A}(x) = A(x) \). Otherwise, let \( \tilde{A}(x) = 0 \). Then \( \tilde{A} \) is an algorithm for the membership in \( \mathcal{L} \), as it is required for \( \mathcal{RP} \)-problems.

5. Proposition 5.6 can be improved such that the probability of success of the algorithm \( \tilde{A} \) is exponentially close to 1. More precisely: By repeating the computation \( A(x) \) and by returning the most frequent result, we get a probabilistic polynomial algorithm \( \tilde{A} \) such that

\[
\text{prob}(\tilde{A}(x) = f(x)) > 1 - 2^{-Q(|x|)} \text{ for all } x \in X.
\]

The proof is completely analogous to the proof of Proposition 5.6. The Chernoff bound is used instead of Proposition 5.6 (which is a consequence of the Weak Law of Large Numbers B.24). The Chernoff bound implies that

\[
\text{prob} \left( \sum_{j=1}^{t} S_j > \frac{t}{2} \right) \geq 1 - 2e^{-\frac{t^2}{2P(|x|)^2}}.
\]

For \( t > \ln(2)P(|x|)^2(Q(|x|) + 1) \), we get the desired result.
6. One-Way Functions and the Basic Assumptions

1. Let $I_k := \{ n \in \mathbb{N} \mid n = pq, p, q \text{ distinct primes}, |p| = |q| = k \}$. The set of keys of security parameter $k$ is $I_k = \{(n, e) \mid n \in I_k, e \in \mathbb{Z}_{\phi(n)}^* \}$.

Let $p_k$ be the uniform distribution on $I_k$ and let $q_k$ be the distribution $i \leftarrow U(1^k)$, given by $S$. Then

$$p_k(n, e) = \frac{1}{|I_k|} \cdot \frac{1}{\text{aver}(\mathbb{Z}_{\phi(n)}^*)} \quad \text{and} \quad q_k(n, e) = \frac{1}{|I_k|} \cdot \frac{1}{|\mathbb{Z}_{\phi(n)}^*|},$$

where $\text{aver}(\mathbb{Z}_{\phi(n)}^*)$ is the average value taken over $n \in I_k$. As we observed in the proof of Proposition 6.6 (referring to Appendix A.2), $\varphi(x) > \frac{x}{6\log(x)}$. Hence,

$$\varphi(n) > |\mathbb{Z}_{\phi(n)}^*| = \varphi(\varphi(n)) > \frac{\varphi(n)}{c\log(k)}$$

($c$ a constant). This implies $q_k(n, e) \leq c\cdot\log(k)\cdot p_k(n, e)$. In particular, $q_k$ is polynomially bounded by $p_k$.

b. Analogous to a).

2. The number of primes of length $k$ is of order $2^k/k$ (by the Prime Number Theorem A.82). Thus, we expect to get a prime after $O(k)$ iterations if we randomly choose $k$-bit strings and apply a probabilistic primality test (see Lemma B.12). A probabilistic primality test takes $O(k^3)$ steps (step = binary operation) and therefore, the expected running time to generate a random prime of length $k$ is $O(k^4)$. To choose a random $e \in \mathbb{Z}_{\phi(n)}^*$ and to check, whether it is a unit (by Euclid’s algorithm A.5), takes $O(k^2)$ steps. The probability of getting a unit is $\varphi(\varphi(n))/\varphi(n)$, with $\varphi(n) = (p - 1)(q - 1)$. Let $d := \{\text{average}_{n \in I_k} \varphi(n)/\varphi(\varphi(n))\}$. In the uniform sampling algorithm of Proposition 6.8, we expect to get a key after generating $d$ moduli $n = pq$ and $d$ exponents. Applying the admissible key generator from Exercise 1, we expect to get a key after generating one modulus and $d$ exponents. Thus, the expected running time of the uniform key generator of Proposition 6.8 is about $d \times$-times the expected running time of the admissible key generator from Exercise 1. We have $d \leq 6\log(2k)$ (Appendix A.2).

3. $f$ is certainly not a strong one-way function: Half of the elements of $X_j$ are even. For every $(x, y) \in D_n$, with $x$ or $y$ is even and $xy < 2^n - 1$, a pre-image $(2, x/2, y/2)$ of $f_n(x, y)$ is immediately computed.

Let $\hat{D}_n := \{(x, y) \in D_n \mid x, y \text{ are primes with } |x| = |y| = |n/2|\}$. We have (by the Prime Number Theorem A.82)

$$|\hat{D}_n| \approx \left(\frac{2^{|n/2|} - 1}{|n/2|} - 1\right)^2 \geq \frac{2^n - 3}{(n - 1)/2^2} \geq \frac{2^n - 3}{n^2} = \frac{2^n}{2n^2}.$$
On the other hand, \(|D_n| = \sum_{j=2}^{n-2} 2^{j} 2^{n-j} < n2^n\) and hence
\[
\frac{|\tilde{D}_n|}{|D_n|} \geq \frac{1}{2n^3}.
\]

By the factoring assumption, the pre-image of \(xy\) cannot be efficiently computed with a non-negligible probability for \((x, y) \in \tilde{D}_n\). Thus, the probability of success of an adversary algorithm is \(\leq 1 - 1/2n^3\).

4. Let \(A_1\) be the algorithm that calls \(A\) and then returns the difference \((a_1 - a'_1, \ldots, a_r - a'_r)\) of \(A\)'s outputs. As we already observed in the proof of Proposition 4.43, \(A_1\) computes a non-trivial representation \(1 = \prod_{j=1}^r g_j^{a_j}\) of 1 if and only if \(A\) computes two distinct representations \(\prod_{j=1}^r g_j^{a_j} = \prod_{j=1}^r g_j^{a'_j}\) of the same element in \(G_q\).

To compute the discrete logarithm of an element \(y \in G_q\) with respect to \(g\), we use the algorithm \(B\) (see the algorithm given in the proof of Proposition 4.43):

**Algorithm 6.1.**

```plaintext
int B(int p, q, g, y)
1   if y = 1
2      then return 0
3   else select i \(\in\) \(\{1, \ldots, r\}\) and
4      u_j \(\in\) \(\{1, \ldots, q - 1\}\), 1 \(\leq\) j \(\leq\) r, uniformly at random
5      g_i \(\leftarrow\) y^{u_i}
6      g_j \(\leftarrow\) g^{u_j}, 1 \(\leq\) j \(\neq\) i \(\leq\) r, is chosen at random
7      \((a_1, \ldots, a_r) \leftarrow A(g_1, \ldots, g_r)
8      if a_i \(\neq\) 0 mod q
9         then return x \(\leftarrow\) \(-(a_i u_i)^{-1} \left(\sum_{j \neq i} a_j u_j\right) \mod q
10        else return 0
```

If \(A_1\) returns a non-trivial representation and if \(a_i \neq 0\) (modulo q), then
\[
y^{-u_i a_i} = \prod_{j \neq i} g_j^{a_j u_j},
\]
and \(B\) correctly returns \(\log_g(y)\) of \(y\) with respect to the base \(g\).

If \(y \neq 1\), then \(y\) is a generator of \(G_q\) and \(y^{u_i}\) is an element which is randomly and uniformly chosen from \(G_q \setminus \{1\}\), and this random choice is independent of the choice of \(i\). If \(A_1\) returns a non-trivial representation of 1, then at least one \(a_i \neq 0\) mod q and therefore, the probability that we get a position \(i\) with \(a_i \neq 0\) mod q by the random choice of \(i\), is \(\geq \frac{1}{r} \geq \frac{1}{T(|p|)}\). Thus,
\[
\text{prob}(B(p, q, g, y) = \log_g(y)) \geq \text{prob}(A(p, q, g_1, \ldots, g_r) = (a_1, \ldots, a_r) \neq 0, 1 = \prod_{j=1}^r g_j^{a_j}):
\]

Introduction to Cryptography by H. Delfs and H. Knebl
Let $I_k \equiv \{ (n, e) \mid n = pq, p, q \text{ distinct primes } \mid |p| = |q| = k, e \in \mathbb{Z}_{\varphi(n)}^* \}$ be the set of public RSA keys with security parameter $k$. By Exercise 1, the RSA assumption remains valid if we replace $(n, e) \equiv I_k$ by $n \equiv J_k, e \equiv \mathbb{Z}_{\varphi(n)}$. In the following sequence of distributions, each distribution polynomially bounds its successor.

a. $n \equiv J_k, e \equiv \mathbb{Z}_{\varphi(n)}$

b. $n \equiv J_k, e \equiv \{ f < 2^{2k} \mid f \text{ prime to } \varphi(n) \}$

c. $n \equiv J_k, \tilde{p} \equiv \{ f \in \text{Primes}_{\leq 2k} \mid f \text{ does not divide } \varphi(n) \}$

The example after Definition B.33, shows that a) bounds b), since the number of primes of binary length $\leq 2k$ is about $\frac{2^k}{k}$ (Theorem A.82). By Proposition B.34, we conclude that the RSA assumption remains valid if we replace $(n \equiv J_k, e \equiv \mathbb{Z}_{\varphi(n)})$ by $(n \equiv J_k, \tilde{p} \equiv \{ f \in \text{Primes}_{\leq 2k} \mid f \text{ does not divide } \varphi(n) \})$. By Lemma B.32, this distribution - we call it $q$ - can be replaced by $(n \equiv J_k, \tilde{p} \equiv \text{Primes}_{\leq 2k})$, since both distributions are polynomially close. Namely, we have for large $k$ (up to some constant)

$$|\{ f \in \text{Primes}_{\leq 2k} \mid f \text{ does not divide } \varphi(n) \}| \geq |\text{Primes}_{\leq 2k}| - \log_2(2k),$$

hence by Theorem A.82

$$\left\lfloor q(\tilde{p}, n) = \frac{1}{|J_k|} \cdot \frac{1}{|\text{Primes}_{\leq 2k}|} \right\rfloor$$

$$\leq \frac{1}{|J_k|} \cdot \frac{1}{|\text{Primes}_{\leq 2k}|} \cdot \left( \frac{|\text{Primes}_{\leq 2k}|}{|\text{Primes}_{\leq 2k}| - \log_2(2k)} - 1 \right)$$

$$\approx \frac{1}{|J_k|} \cdot \frac{1}{|\text{Primes}_{\leq 2k}|} \left( \frac{2^{2k}}{2k \log_2(2k)} - 1 \right)$$

$$\approx \frac{1}{|J_k|} \cdot \frac{1}{|\text{Primes}_{\leq 2k}|} \left( \frac{2k \log_2(2k)}{2^{2k}} \right) \approx \frac{k}{2^{2k}} \cdot \frac{2k \log_2(2k)}{2^{2k}} \leq \frac{k^5}{2^{6k}}.$$
Since the number of tuples \((\tilde{p}, n)\) is of order \(O\left(\frac{2^k}{k^2}\right)\), the polynomial closeness follows.

Finally, by Theorem A.84, we have for a prime \(\tilde{p}\) (up to some constant)
\[
|\{f \in \text{Primes}_k \mid \tilde{p} \text{ divides } f - 1\}| \approx \frac{1}{\tilde{p} - 1} \frac{2^k}{k} \leq \frac{2^k}{2k^2},
\]
hence
\[
|J_{k,\tilde{p}}| \geq \frac{9\cdot2^k}{4k^2} \text{ and then } 4 \cdot |J_{k,\tilde{p}}| \geq |J_k| \approx \frac{2^k}{2k^2}.
\]
We see that \((\tilde{p} \not\in J_k, \tilde{p} \not\in \text{Primes}_{\leq 2k})\) polynomially bounds \((\tilde{p} \not\in J_k, \tilde{p} \not\in \text{Primes}_{\leq 2k}, n \not\in J_{k,\tilde{p}})\). This finishes the proof.

6. Let \(b \in \{0, 1\}\). Assume that there is a positive polynomial \(P\), such that
\[
\Pr (B_i(x) = b : i \leftarrow K(1^k), x \leftarrow D_i) = 1 - \frac{1}{P(k)},
\]
for infinitely many \(k\). Then the constant algorithm \(A(i, y)\) that always returns \(b\) successfully computes the hard-core bit
\[
\Pr (A(i, f_i(x)) = B_i(x) : i \leftarrow K(1^k), x \leftarrow D_i) = \Pr (B_i(x) = b : i \leftarrow K(1^k), x \leftarrow D_i) \geq \frac{1}{2} + \frac{1}{P(k)},
\]
a contradiction.

7. Assume there is an algorithm \(A\) with
\[
\Pr (A(i, f_i(x), B_i(x)) = 1 : i \leftarrow K(1^k), x \leftarrow D_i) = \Pr (A(i, f_i(x), z) = 1 : i \leftarrow K(1^k), x \leftarrow D_i, z \leftarrow \{0, 1\}) > \frac{1}{P(k)}
\]
for some positive polynomial \(P\) and for \(k\) in an infinite subset \(\mathcal{K}\) of \(\mathbb{N}\) (Replacing \(A\) by \(1 - A\), if necessary, we may omit the absolute value).

Let \(\tilde{A}\) be the following algorithm with inputs \(i \in I, y \in R_i\):

a. Randomly choose a bit \(b \leftarrow \{0, 1\}\).

b. If \(\tilde{A}(i, y, b) = 1\), then return \(b\), else return \(1 - b\).

Applying Lemma B.13 we get
\[
\Pr (\tilde{A}(i, f_i(x)) = B_i(x) : i \leftarrow K(1^k), x \leftarrow D_i) = \frac{1}{2} + \Pr (\tilde{A}(i, f_i(x)) = b : i \leftarrow K(1^k), x \leftarrow D_i | B_i(x) = b)
\]
\[
= \frac{1}{2} + \Pr (A(i, f_i(x), B_i(x)) = 1 : i \leftarrow K(1^k), x \leftarrow D_i) - \Pr (A(i, f_i(x), b) = 1 : i \leftarrow K(1^k), x \leftarrow D_i, b \leftarrow \{0, 1\}) > \frac{1}{2} + \frac{1}{P(k)}.
\]

Introduction to Cryptography by H. Delfs and H. Knebl
6. One-Way Functions and the Basic Assumptions

for the infinitely many \( k \in \mathcal{K} \). Hence, \( B \) is not a hard-core predicate.

Conversely, if \( \tilde{A}(i, y) \) is a probabilistic polynomial algorithm with

\[
\operatorname{prob}(\tilde{A}(i, f_i(x)) = B_i(x) : i \leftarrow K(1^k), x \leftarrow D_i) > \frac{1}{2} + \frac{1}{P(k)}
\]

for infinitely many \( k \), then the algorithm \( A \) with

\[
A(i, y, z) := \begin{cases} 1 & \text{if } z = \tilde{A}(i, y), \\ 0 & \text{else.} \end{cases}
\]

successfully distinguishes between the distributions.

8. The analogous proposition is:

The following statements are equivalent.

a. For every probabilistic polynomial algorithm \( A \) with inputs \( i \in \mathcal{I}, x \in X_i \) and output in \( \{0, 1\} \) and every positive polynomial \( P \), there is a \( k_0 \in \mathbb{N} \) such that for all \( k \geq k_0 \)

\[
| \operatorname{prob}(A(i, x) = 1 : i \leftarrow I_k, x \leftarrow X_i) \\
- \operatorname{prob}(A(i, x) = 1 : i \leftarrow I_k, x \leftarrow X_i) | \leq \frac{1}{P(k)}
\]

b. For every probabilistic polynomial algorithm \( A \) with inputs \( i \in \mathcal{I}, x \in X_i \) and output in \( \{0, 1\} \) and all positive polynomials \( Q, R \) there is a \( k_0 \in \mathbb{N} \) such that for all \( k \geq k_0 \)

\[
\operatorname{prob}\{ i \in I_k \mid | \operatorname{prob}(A(i, x) = 1 : x \leftarrow X_i) \\
- \operatorname{prob}(A(i, x) = 1 : x \leftarrow X_i) | > \frac{1}{Q(k)}\} \leq \frac{1}{R(k)}
\]

The proof now runs in the same way as the proof of Proposition 6.17. The main difference is that we need an algorithm \( \text{Sign}(i) \) which computes the sign of

\[
\operatorname{prob}(A(i, x) = 1 : x \leftarrow X_i) - \operatorname{prob}(A(i, x) = 1 : x \leftarrow X_i)
\]

with high probability if the absolute value of this difference is \( \geq \frac{1}{\tilde{T}(k)} \) (with \( \tilde{T} \) a polynomial). This algorithm is constructed analogously. We use the fact that the probabilities can be approximately computed with high probability by a probabilistic polynomial algorithm (Proposition 6.18).

9. see [GolMic84].
7. Bit Security of One-Way Functions

1. 

\[ 17 \in \text{QR}_p, \]
\[ \text{PSqrt}(17) = 13 \notin \text{QR}_p, 13 \cdot 2^{-1} = 16 \]
\[ \text{PSqrt}(16) = 4 \in \text{QR}_p \]
\[ \text{PSqrt}(4) = 2 \notin \text{QR}_p, 2 \cdot 2^{-1} = 1 \]
\[ \text{PSqrt}(1) = 1 \in \text{QR}_p \]

Thus we have \( \text{Log}_{p,g}(17) = 01010 \) (in binary encoding).

2. Algorithm 7.1.

```cpp
int BinSearchLog(int p, g, y)
1 int : l, r, i
2 l ← 0; r ← p − 1
3 while l ≤ r do
4 i ← div (l + r)
5 if A1(p, g, y) = 1
6 then l ← i
7 else r ← i + 1
8 y ← y^2
9 return l
```

3. a. We compute the \( t \) least-significant bits as in the proof of Proposition 7.5. Let \( k = |p|, y = g^{x_{k-1} \ldots x_0}, x_i \in \{0, 1\}, i = 0, \ldots, k − 1 \). The bit \( x_0 \) is 0, if and only if \( y \in \text{QR}_p \) (Proposition A.55). This condition can be tested with the criterion of Euler for quadratic residuosity (Proposition A.58).

We replace \( y \) by \( yg^{-1} \), if \( x_0 = 1 \). Thus, we can assume \( x_0 = 0 \). We get the square roots \( y_1 = g^{x_{k-1} \ldots x_1} \) and \( y_2 = g^{x_{k-1} \ldots x_1+(p-1)/2} \) of \( y \).

Since \( p − 1 = 2^t q, q \) odd, the \( t \) least-significant bits of \( p − 1 \) are 0.

\[ \log y_1 \text{ and } \log y_2 \text{ coincide in the } t \text{ least-significant bits (} t \geq 1 \). \]

Algorithm 7.2.

```cpp
int A(int p, g, x)
1 d ← ε
2 for c ← 0 to \( t \) − 1 do
3 if x ∈ QR_p
4 then d ← d[0]
5 else d ← d[1]
6 x ← xg^{-1}
7 x ← \text{Sqrt}(p, g, x)
8 return d
```

Introduction to Cryptography by H. Delfs and H. Knebl
b. Let \( \{u, v\} = \text{Sqrt}(y) \). Then \( \text{Lsb}_{t-1}(\log_{p,g}(u)) \neq \text{Lsb}_{t-1}(\log_{p,g}(v)) \) (the logarithms differ by \( \frac{p-1}{2} \)). Observe that you can compute these bits by a).

**Algorithm 7.3.**

```c
int A(int p, g, y)
1    \{u, v\} ← Sqrt(y)
2    if A1(p, g, y) = Lsb_{t-1}(\log_{p,g}(u))
3        then return u
4    else return v
```

A computes the principal square root of \( y \). The assertion now follows by Proposition 7.5.

c. Let \( P \) be a positive polynomial and \( A_1 \) be a probabilistic polynomial algorithm such that

\[
\text{prob}(A_1(p, g, g^x) = \text{Lsb}_t(x) : x ∈ \mathbb{Z}_{p-1}) ≥ \frac{1}{2} + \frac{1}{P(k)},
\]

where \( p \) is an odd prime, \( p = 2^t q, q \) odd, and \( g \) is a primitive root \( \text{mod} p \). As in b) we get a probabilistic polynomial algorithm \( A \) such that

\[
\text{prob}(A(p, g, y) = \text{PSqrt}_{p,g}(y) : y \leftarrow \text{QR}_n) ≥ \frac{1}{2} + \frac{1}{P(k)}.
\]

This contradicts the discrete logarithm assumption (see Theorem 7.7).

4. a. Let \( p - 1 = 2^t q, q \) odd, \( y = g^x \). Compute the \( t \) least-significant bits of \( x \) by the Algorithm of Exercise 7.2. Guess the next \( j - t \) bits from \( x \) (note there are only polynomially many, namely \( O(k) \), alternatives). Thus, we can assume that the \( j \) least-significant bits of \( x \) are known.
Algorithm 7.4.
\[
\begin{align*}
\text{int } & A(\text{int } p, g, y) \\
1 & \ L_j \leftarrow j \text{ least-significant bits of } x \\
2 & \ d \leftarrow L_j \\
3 & \ \text{for } c \leftarrow j \text{ to } k - 1 \text{ do} \\
4 & \ b \leftarrow A_1(p, g, y, L_j) \\
5 & \ \text{if } \text{Lsb}(L_j) = 1 \\
6 & \ y \leftarrow yg^{-1} \\
7 & \ \{u, v\} \leftarrow \text{Sqrt}(p, g, y) \\
8 & \ \text{if } \text{Lsb}(\log_{p,g}(u)) = \text{Lsb}(L_j) \\
9 & \ y \leftarrow u \\
10 & \ \text{else } y \leftarrow v \\
11 & \ L_j \leftarrow b||\text{Lsb}_{j - 1}(L_j) \ldots \text{Lsb}_{1}(L_j) \\
12 & \ d \leftarrow b||d \\
13 & \ \text{return } d
\end{align*}
\]

b. The probability of success of $A_1$ can be increased as in Lemma 7.8. Observe that you can compute \text{Lsb}_j(x) from \text{Lsb}_j(x + r)$, where $r$ is randomly chosen, if $x + r \leq p - 1$. Use this to compute \text{Lsb}_j(x) with probability almost 1 for small values of $x$. Then continue as in the proof of Theorem 7.7 to prove statement b).

5. Assume there is a positive polynomial $P \in \mathbb{Z}[X]$ and an algorithm $A_1$ such that
\[
\text{prob}(A_1(p, g, \text{Lsb}_t(x), \ldots, \text{Lsb}_{t + j - 1}(x), g^x)) = \text{Lsb}_{t + j}(x) : (p, g) \leftarrow I_k, x \leftarrow \mathbb{Z}_{p - 1}) > \frac{1}{2} + \frac{1}{P(k)}.
\]
for infinitely many $k$. By Proposition 6.17, there are polynomials $Q, R$, such that
\[
\text{prob}\left\{(p, g) \in I_k \mid \text{prob}(A_1(p, g, \text{Lsb}_t(x), \ldots, \text{Lsb}_{t + j - 1}(x), g^x)) = \text{Lsb}_{t + j}(x) : x \leftarrow \mathbb{Z}_{p - 1}) > \frac{1}{2} + \frac{1}{Q(k)}\right\} > \frac{1}{R(k)},
\]
for infinitely many $k$. From the preceding Exercise 4, we conclude that there is an algorithm $A_2$ and a positive polynomial $S \in \mathbb{Z}[X]$ such that
\[
\text{prob}\left\{(p, g) \in I_k \mid \text{prob}(A_2(p, g, g^x) = x : x \leftarrow \mathbb{Z}_{p - 1}) \geq 1 - \frac{1}{S(k)}\right\} > \frac{1}{R(k)},
\]
for infinitely many $k$. By Proposition 6.3, there is a positive polynomial $T \in \mathbb{Z}[X]$ such that

Introduction to Cryptography by H. Delfs and H. Knebl
prob(A_2(p, g, g^x) = x : (p, g) \in I_k, x \in \mathbb{Z}_{p-1}) > \frac{1}{T(k)},

for infinitely many k, a contradiction to the discrete logarithm assumption.

6.

<table>
<thead>
<tr>
<th>t</th>
<th>a_t</th>
<th>u_t</th>
<th>a_t x</th>
<th>Lsb(a_t x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0.5</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>0.75</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0.875</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.9375</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0.46875</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>0.234375</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus we have a = 5, u = \frac{15}{64}.

7.

<table>
<thead>
<tr>
<th>t</th>
<th>a_t</th>
<th>u_t</th>
<th>returned bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>196</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>98</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>49</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>220</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>0.125</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>0.5625</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>223</td>
<td>0.78125</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>307</td>
<td>0.890625</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>349</td>
<td>0.9453125</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>370</td>
<td>0.47265625</td>
<td>1</td>
</tr>
</tbody>
</table>

We get a = 370, ax = \lfloor \frac{121}{256} \cdot 391 + 1 \rfloor and x = a^{-1}ax = 196.

8. Observe that

Msb(x) = Lsb(2x) and
Lsb(x) = Msb(2^{-1}x).

Thus, an algorithm A(n, e, y) computing Lsb(x) can be used to compute Msb(x) (Msb(x) = A(n, e, 2^ey)) and vice versa.

10. Let \( y = x^c \). Observe that \( 2^c y = (2x)^c \).

11. Algorithm 7.5.

\[
\text{int } RSA^{-1}(\text{int } y) \\
1 \quad \text{for } i \leftarrow 1 \text{ to } k - 1 \text{ do} \\
2 \quad \text{if } \text{LsbRSA}^{-1}(y) = 0 \\
3 \quad \quad \text{then } \text{LSB}[i] \leftarrow 0 \\
4 \quad \quad y \leftarrow y2^{-c} \text{ mod } n \\
5 \quad \text{else } \text{LSB}[i] \leftarrow 1 \\
6 \quad y \leftarrow (n - y)2^{-c} \text{ mod } n \\
7 \quad t[k] \leftarrow \text{LsbRSA}^{-1}(y); t[1] = \ldots = t[k - 1] = 0 \\
8 \quad \text{for } i \leftarrow k - 1 \text{ downto } 1 \text{ do} \\
9 \quad \quad t \leftarrow \text{Shift}(t) \\
10 \quad \text{if } \text{LSB}[i] = 1 \\
11 \quad \quad \text{then } t \leftarrow \text{Delta}(n, t, k - i + 1) \\
12 \quad \text{return } t
\]

\text{Shift}(t) \text{ returns the bits of } t, \text{ shifted one position to the left, filling the emptied bit with 0. Delta}(s, t, i) \text{ returns for } t \leq s \text{ the } i \text{ least-significant bits of } s - t. \text{ The remaining bits are 0.}

12. a. We define \( L_j : \mathbb{Z}_n^* \rightarrow \{0, 1\}^j, \ x \mapsto x \text{ mod } 2^j \).

We get the RSA-inversion by rational approximation by using the equations

\[
\begin{align*}
    a_0 &= 1, & u_0 &= 0, \\
    a_t &= 2^{-1}a_{t-1}, & u_t &= \frac{1}{2} (u_{t-1} + \text{Lsb}(a_{t-1}x)).
\end{align*}
\]

We have

\[
L_{j-1}(\overline{a_t}x) = \frac{1}{2} L_j(\overline{a_{t-1}}x + \text{Lsb}(\overline{a_{t-1}}x)n),
\]

and we compute \( L_j(\overline{a_t}x) \) for \( t \geq 0 \) by

\[
\text{Guess } L_j(\overline{a_t}x), \\
L_j(\overline{a_t}x) = \text{Lsb}_j(\overline{a_t}x)2^{j-1} + \frac{1}{2} L_j(\overline{a_{t-1}}x + \text{Lsb}(\overline{a_{t-1}}x)n).
\]

and get

Introduction to Cryptography by H. Delfs and H. Knebl
Algorithm 7.6.

```latex
\text{int } A_2(\text{int } n, e, y) \\
1 \quad a \leftarrow 1, u \leftarrow 0 \\
2 \quad \text{guess } Lst_j \leftarrow L_j(a_0x) \\
3 \quad \text{for } t \leftarrow 1 \text{ to } k \text{ do} \\
4 \quad \quad u \leftarrow \frac{1}{2}(u + \text{Lsb}(Lst_j)) \\
5 \quad \quad a \leftarrow 2^{-j}a \mod n \\
6 \quad Lst_j \leftarrow A_1(n, e, a^ey \mod n)2^{j-1} + \frac{1}{2}(Lst_j + \text{Lsb}(a_{t-1}x)n) \\
7 \quad \text{return } a^{-1}[un + 1] \mod n
```

b. With the notations from the proof of Theorem 7.14 we have

\[ A_{t,i} = a_t + ia_{t-1} + b = (1 + 2t)a_t + b, \]
\[ W_{t,i} = [u_t + iu_{t-1} + v]. \]

and if \( W_{t,i} = q \) (see proof of proof of Theorem 7.14) we have

\[ \frac{A_{t,i}x}{W_{t,i}} = \frac{a_t x + ia_{t-1} x + bx}{W_{t,i}}. \]

Thus, we get

\[ L_j(\frac{A_{t,i}x}{W_{t,i}}) = L_j(\frac{a_t x}{W_{t,i}}) + L_j(\frac{ia_{t-1} x}{W_{t,i}}) + L_j(\frac{bx}{W_{t,i}}) - L_j(W_{t,i}n) \mod 2^j \]

and

\[ \text{Lsb}_j(\frac{A_{t,i}x}{W_{t,i}})2^{j-1} + L_j(\frac{A_{t,i}x}{W_{t,i}}) = \]
\[ \text{Lsb}_j(\frac{a_t x}{W_{t,i}})2^{j-1} + L_j(\frac{a_t x}{W_{t,i}}) + L_j(\frac{ia_{t-1} x}{W_{t,i}}) + L_j(\frac{bx}{W_{t,i}}) - \]
\[ L_j(W_{t,i}n) \mod 2^j, \quad \text{hence} \]
\[ \text{Lsb}_j(\frac{a_t x}{W_{t,i}})2^{j-1} = \]
\[ \text{Lsb}_j(\frac{A_{t,i}x}{W_{t,i}})2^{j-1} + L_j(\frac{A_{t,i}x}{W_{t,i}}) - L_j(\frac{a_t x}{W_{t,i}}) + L_j(\frac{ia_{t-1} x}{W_{t,i}}) - L_j(\frac{bx}{W_{t,i}}) + L_j(W_{t,i}n) \mod 2^j. \]

We use the last equation to get \( \text{Lsb}_j(\frac{a_t x}{W_{t,i}}) \) by a majority decision computing \( \text{Lsb}_j(\frac{A_{t,i}x}{W_{t,i}}) \) by algorithm \( A_1 \). Observe that the other terms of the right side of the equation are known. \( L_{j-1}(\frac{a_t x}{W_{t,i}}) \) and \( L_{j-1}(\frac{A_{t,i}x}{W_{t,i}}) \) can be recursively computed from \( L_j(\frac{a_{t-1} x}{W_{t,i}}) \) and \( L_j(\frac{a_t x}{W_{t,i}}) \):

\[ L_{j-1}(\frac{a_t x}{W_{t,i}}) = \frac{1}{2}(L_j(\frac{a_{t-1} x}{W_{t,i}} + \text{Lsb}(\frac{a_{t-1} x}{W_{t,i}})n), \]
\[ L_{j-1}(\frac{A_{t,i}x}{W_{t,i}}) = (1 + 2t)L_j(\frac{a_t x}{W_{t,i}}) + L_j(\frac{bx}{W_{t,i}}) \mod 2^{j-1}. \]

Initially we have to guess \( L_j(\frac{a_t x}{W_{t,i}}) \) and \( L_j(\frac{bx}{W_{t,i}}) \). This is polynomial in \( k \), because \( j \leq \lfloor \log_2(2k) \rfloor \).

We can modify the Algorithm from Lemma 7.15 to get an algorithm which computes \( L_j(a_t x) \) with probability almost 1. From \( L_j(a_t x) \) we can easily derive \( \text{Lsb}(a_t x) \), and we can use \( \text{Lsb}(a_t x) \) in Algorithm 7.17 and continue as in Section 7.2.

13. The proof is analogous to the proof of Exercise 5.

© H. Delfs and H. Knebl
8. One-Way Functions and Pseudorandomness

1. If $\hat{A}(i, z)$ is a probabilistic polynomial algorithm that distinguishes between the sequences generated by $\pi \circ G$ and true random sequences (see Definition 8.2), then $A(i, z) : (i, z) \mapsto \hat{A}(i, \Pi(i, z))$ distinguishes the sequences generated by $G$ from true random sequences.

2. Examples can be constructed by one-way permutations $f = (f_i : D_i \rightarrow D_i)_{i \in I}$ with hard-core predicate $B$, like the RSA family. Consider the pseudorandom generator $G$ with $G_i(x) := (f_i(x), B_i(x))$, which generates from a randomly chosen seed $x \leftarrow D_i$ a pseudorandom sequence of length $|x| + 1$. $G$ is computationally perfect, by Exercise 7 in Chapter 6. Let $\pi$ be the permutation $\pi_i(y, b) := (f_i^{-1}(y), b)$ ($y \in D_i, b \in \{0, 1\})$. Then $\pi_i(G_i(x)) = (x, B(x))$, and we see that $\pi \circ G$ is not computationally perfect (since $B(x)$ is computable from $x$).

3. The proof is an immediate consequence of Exercise 1 (consider the permutation $(x_1, \ldots, x_{i(k)}) \mapsto (x_{i(k)}, \ldots, x_1)$ and Yao’s Theorem 8.7.

4. Assume there is a probabilistic polynomial statistical test $A(i, z)$ and a positive polynomial $R$ such that

$$\text{prob}(A(i, G_i^l(x)) = 1 : i \leftarrow K(1^k), x \leftarrow \{0, 1\}^{Q(k)})$$
$$- \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0, 1\}^{Q(k)+l}) > \frac{1}{R(k)},$$

for $k$ in an infinite subset $K$ of $\mathbb{N}$ (replacing $A$ by $1 - A$ if necessary we may drop the absolute value).

For $k \in K$ and $i \in I_k$ we consider the following sequence of distributions

$$d_{i,0}, d_{i,1}, \ldots, d_{i,l}$$

on $\{0, 1\}^m$, where $m = m(k) := Q(k) + l$.

$$d_{i,0} = \{(b_1, \ldots, b_l, x) : (b_1, \ldots, b_l) \leftarrow \{0, 1\}^l, x \leftarrow \{0, 1\}^{Q(k)}\}$$
$$d_{i,1} = \{(b_1, \ldots, b_{l-1}, G_i^l(x)) : (b_1, \ldots, b_{l-1}) \leftarrow \{0, 1\}^{l-1}, x \leftarrow \{0, 1\}^{Q(k)}\}$$
$$d_{i,2} = \{(b_1, \ldots, b_{l-2}, G_i^l(x)) : (b_1, \ldots, b_{l-2}) \leftarrow \{0, 1\}^{l-2}, x \leftarrow \{0, 1\}^{Q(k)}\}$$
$$\vdots$$
$$d_{i,r} = \{(b_1, \ldots, b_{l-r}, G_i^r(x)) : (b_1, \ldots, b_{l-r}) \leftarrow \{0, 1\}^{l-r}, x \leftarrow \{0, 1\}^{Q(k)}\}$$
$$\vdots$$
$$d_{i,l} = \{G_i^l(x) : x \leftarrow \{0, 1\}^{Q(k)}\}.$$

---

5 We use the following notation: $\{S(x) : x \leftarrow X\}$ denotes the image of the distribution on $X$ under $S : X \rightarrow Z$, i.e., the probability of $z \in Z$ is given by the probability for $z$ appearing as $S(x)$ if $x$ is randomly selected from $X$.
$d_{i,0}$ is the uniform distribution, $d_{i,l}$ is the distribution induced by $G^d_i$. For $k \in \mathcal{K}$, we have

$$
\frac{1}{R(k)} < \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)
- \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)
= \sum_{r=0}^{l-1} (\text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)
- \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)).
$$

Define the algorithm $\tilde{A}$ as follows:

a. Randomly choose $r$, with $0 \leq r < l$.

b. Choose random bits $b_1, b_2, \ldots, b_{l-r-1}$.

c. For $z = (z_1, \ldots, z_{Q(k)+1}) \in \{0,1\}^{Q(k)+1}$ let

$$
\tilde{A}(i, z) := A(i, b_1, \ldots, b_{l-r-1}, z_1, G^d_i((z_2, \ldots, z_{Q(k)+1}))).
$$

We have

$$
\text{prob}(\tilde{A}(i, G_i(x)) = 1 : i \leftarrow K(1^k), x \leftarrow \{0,1\}^{Q(k)})
- \text{prob}(\tilde{A}(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^{Q(k)+1})
= \sum_{r=0}^{l-1} \text{prob}(r) \cdot (\text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)
- \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)))
= \frac{1}{l} \sum_{r=0}^{l-1} (\text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)
- \text{prob}(A(i, z) = 1 : i \leftarrow K(1^k), z \leftarrow \{0,1\}^m)))
> \frac{1}{IR(k)},
$$

for the infinitely many $k \in \mathcal{K}$. This contradicts the assumption that $G$ is computationally perfect.

5. The proof runs in the same way as the proof of Yao’s Theorem 8.7. An additional input $y \in Y_i$ has to be added to the algorithms $A$ and $\tilde{A}$ and the probabilities

$$
\text{prob}(\tilde{A}(i, f_i(x), z) = \ldots : i \leftarrow K(1^k), x \leftarrow X_i, z \leftarrow \ldots)
$$

must also be taken over $x \leftarrow X_i$. The distributions $p_{i,r}$ are modified to

$$
p_{i,r} = \{(f_i(x), G_{i,1}(x), G_{i,2}(x), \ldots, G_{i,r}(x), b_{r+1}, \ldots, b_{Q(k)} : (b_{r+1}, \ldots, b_{Q(k)}) \leftarrow \{0,1\}^{Q(k)-r}, x \leftarrow X_i\}.
$$

©H. Delfs and H. Knebl
6. Assume there is a probabilistic polynomial algorithm $A(i, y)$, such that

$$\text{prob}(A(i, f_i(x)) = C_i(B_{i,1}(x), \ldots, B_{i,l}(k(x))) : i \leftarrow K(1^k), x \xleftarrow{\$} D_i)$$

$$> \frac{1}{2} + \frac{1}{P(k)}$$

for $k$ in an infinite subset $K$ of $\mathbb{N}$.

Define the algorithm $\tilde{A}(i, y, z_1, \ldots, z_l)$ as follows:

$$\tilde{A}(i, y, z_1, \ldots, z_l) := \begin{cases} 1 & \text{if } A(i, y) = C_i(z_1, \ldots, z_l), \\ 0 & \text{else} \end{cases}$$

We have

$$\text{prob}(A(i, f_i(x)) = C_i(z_1, \ldots, z_l) : i \leftarrow K(1^k), x \xleftarrow{\$} D_i, (z_1, \ldots, z_l) \xleftarrow{\$} \{0,1\}^l)$$

$$= \text{prob}(A(i, f_i(x)) = 0 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i) \cdot \text{prob}(C_i(z_1, \ldots, z_l) = 0 : (z_1, \ldots, z_l) \xleftarrow{\$} \{0,1\}^l)$$

$$+ \text{prob}(A(i, f_i(x)) = 1 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i) \cdot \text{prob}(C_i(z_1, \ldots, z_l) = 1 : (z_1, \ldots, z_l) \xleftarrow{\$} \{0,1\}^l)$$

$$= \text{prob}(A(i, f_i(x)) = 0 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i) \cdot \frac{1}{2}$$

$$+ \text{prob}(A(i, f_i(x)) = 1 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i) \cdot \frac{1}{2}$$

$$= \frac{1}{2}$$

Hence

$$| \text{prob}(\tilde{A}(i, f_i(x), z_1, \ldots, z_l) = 1 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i, (z_1, \ldots, z_l) \xleftarrow{\$} \{0,1\}^l)$$

$$- \text{prob}(\tilde{A}(i, f_i(x), B_{i,1}(x), \ldots, B_{i,l}(x)) = 1 : i \leftarrow K(1^k), x \xleftarrow{\$} D_i) |$$

$$= | \frac{1}{2} - \text{prob}(A(i, f_i(x)) = C_i(B_{i,1}(x), \ldots, B_{i,l}(x))) : i \leftarrow K(1^k), x \xleftarrow{\$} D_i |$$

$$> \frac{1}{2} + \frac{1}{P(k)} - \frac{1}{2}$$

$$> \frac{1}{P(k)}$$

for infinitely many $k$. This is a contradiction.

7. Assume that the bits $B_{i,1}, \ldots, B_{i,l}$ are not simultaneously secure. From the stronger version of Yao’s Theorem, Exercise 5, we conclude that there

Introduction to Cryptography by H. Delfs and H. Knebl
is a probabilistic polynomial algorithm \( A \), a positive polynomial \( P \) and a \( j_k, 1 \leq j_k \leq l(k) \) such that

\[
\begin{align*}
\text{prob}(A(i, f_i(x), B_{i,1}(x) \ldots B_{i,j_k-1}(x)) &= B_{i,j_k}(x) : i \leftarrow K(1^k), x \overset{\mu}{\leftarrow} X_i) \\
&> \frac{1}{2} + \frac{1}{P(k)},
\end{align*}
\]

for infinitely many \( k \). This is a contradiction.

8. The statement, which is analogous to Theorem 8.4, is almost identical to the statement of Theorem 8.4:

For every probabilistic polynomial algorithm \( A \) with inputs \( i \in I_k, z \in \{0,1\}^{l(k)Q(k)}, y \in D_i \) and output in \( \{0,1\} \) and every positive polynomial \( P \in \mathbb{Z}[X] \), there is a \( k_0 \in \mathbb{N} \) such that for all \( k \geq k_0 \)

\[
|\text{prob}(A(i, z, y) = 1 : i \leftarrow K(1^k), z \overset{\mu}{\leftarrow} \{0,1\}^{l(k)Q(k)}, y \overset{\mu}{\leftarrow} D_i) \\
- \text{prob}(A(i, G_i(x), f_{i,Q(k)}^l(x)) = 1 : i \leftarrow K(1^k), x \overset{\mu}{\leftarrow} D_i)| \leq \frac{1}{P(k)}.
\]

The proof runs as the proof of Theorem 8.4. There are only the following differences:

In the distributions \( p_{i,r} \), the elements \( b_i \) have to be chosen from \( \{0,1\}^{l(k)} \);

\( b_j \overset{\mu}{\leftarrow} \{0,1\}^{l(k)} \), and \( X_i \) has to be set as \( X_i := \{0,1\}^{l(k)Q(k)} \times D_i \).

We define the algorithm \( \bar{A} \) as follows:

On inputs \( i \in I_k, y \in D_i, w \in \{0,1\}^{l(k)} \)

a. Randomly choose \( r \), with \( 0 \leq r < Q(k) \).

b. Randomly choose \( b_1, b_2, \ldots, b_{Q(k)-r-1} \) in \( \{0,1\}^{l(k)} \).

c. For \( y = f_i(x) \) let \( \bar{A}(i, y, w) := \)

\[
A(i, b_1, \ldots, b_{Q(k)-r-1}, w, B_i(f_i(x)), B_i(f_{i}^2(x)), \ldots, B_i(f_{i}^l(x)), f_{i}^{l+1}(x)).
\]

Then

\[
\begin{align*}
|\text{prob}(\bar{A}(i, f_i(x), B_i(x)) = 1 : i \leftarrow K(1^k), x \overset{\mu}{\leftarrow} D_i) \\
- \bar{A}(i, y, w) = 1 : i \leftarrow K(1^k), y \overset{\mu}{\leftarrow} D_i, w \overset{\mu}{\leftarrow} \{0,1\}^{l(k)}| \\
= \sum_{r=0}^{Q(k)-1} \text{prob}(r) \cdot (\text{prob}(A(i, z, y) = 1 : i \leftarrow K(1^k), (z, y) \overset{p_{i,r+1}}{\leftarrow} X_i) \\
- \text{prob}(A(i, z, y) = 1 : i \leftarrow K(1^k), (z, y) \overset{p_{i,r}}{\leftarrow} X_i)) \\
= \frac{1}{l(k)} \sum_{r=0}^{Q(k)-1} (\text{prob}(A(i, z, y) = 1 : i \leftarrow K(1^k), (z, y) \overset{p_{i,r+1}}{\leftarrow} X_i) \\
- \text{prob}(A(i, z, y) = 1 : i \leftarrow K(1^k), (z, y) \overset{p_{i,r}}{\leftarrow} X_i)) \\
> \frac{1}{l(k)P(k)},
\end{align*}
\]

\( \copyright \) H. Delfs and H. Knebl
for infinitely many $k$. This contradicts the fact that $B$ is an $l$-bit hard-core predicate.
9. Provable Secure Encryption

1. The affine cipher is perfectly secret. Namely, let \( m \in \mathbb{Z}_n \) and \( c \in \mathbb{Z}_n \). We look for the number of keys \( (a, b) \) such that \( m \) is encrypted as \( c \). \( a \) is a unit modulo \( n \), so we have \( \varphi(n) \) choices for \( a \). Since \( b = c - a \cdot m \mod n \), the choice of \( a \) determines \( b \). We conclude that there are \( \varphi(n) \) keys \( (a, b) \) which transform \( m \) to \( c \). If the keys are selected uniformly at random, as assumed, this means that \( \mathrm{prob}(c|m) = \frac{\varphi(n)}{\varphi(n)n} = \frac{1}{n} \). The probability is independent of \( m \), which implies that the affine cipher is perfectly secret (Proposition 9.4).

2. Knowing the key and the ciphertext, the plaintext \( m \) can be derived. Hence, \( H(M|KC) = 0 \). Therefore, we have

\[
0 \leq I(M; K|C) = H(K|C) - H(K|MC)
= H(M|C) - H(M|KC) = H(M|C).
\]
Hence, \( H(K|C) \geq H(M|C) \), because \( H(K|MC) \geq 0 \).

3. It is not ciphertext-indistinguishable, as the following considerations show. Let \( (p, g, y := g^x) \) be an ElGamal public key. We have \( E_{p, g, y}(m) = (g^k, y^km) \) for plaintexts \( m \in \mathbb{Z}_p^* \). Applying \( \log_{p, g} \) to both components of \( E_{p, g, y}(m) \), we get \( (k, kx + \log_{p, g}(m)) \in \mathbb{Z}_2^{2−1} \). If \( p − 1 = 2^t \cdot a \) odd, then the \( t \) least-significant bits of \( \log_{p, g}(z) \) can be easily computed for \( z \in \mathbb{Z}_p^* \) (see Section 7.1, Exercise 3 in Chapter 7). In particular, we can compute the \( t \) least-significant bits of \( k \), \( x \), \( \log_{p, g}(y^km) \). Since \( (kx \mod p − 1) \mod 2^t = kx \mod 2^t \), we can compute the \( t \) least-significant bits of \( kx \mod (p − 1) \) and hence also of \( \log_{p, g}(m) \). Thus, we can distinguish between plaintexts, whose \( \log_{p, g} \) differ in their \( t \) least-significant bits.

If we consider only the least-significant bit, this means that we can distinguish between quadratic residues and non-residues.

4. Note that \( S(i) \) returns two distinct messages \( m_0 \neq m_1 \). We have for every pair \( m_0 \neq m_1 \)

\[
\begin{align*}
\mathrm{prob}(A(i, m_0, m_1, c) = m : i \leftarrow K(1^k), m \leftarrow \{m_0, m_1\}, c \leftarrow E(m)) &= \frac{1}{2} \cdot \mathrm{prob}(A(n, e, m_0, m_1, c) = m_0 : (n, e) \leftarrow I_k, c \leftarrow E(m_0)) \\
&\quad + \frac{1}{2} \cdot \mathrm{prob}(A(n, e, m_0, m_1, c) = m_1 : (n, e) \leftarrow I_k, c \leftarrow E(m_1)) \\
&= \frac{1}{2} + \frac{1}{2} \cdot [\mathrm{prob}(A(n, e, m_0, m_1, c) = m_0 : (n, e) \leftarrow I_k, c \leftarrow E(m_0)) \\
&\quad - \mathrm{prob}(A(n, e, m_0, m_1, c) = m_0 : (n, e) \leftarrow I_k, c \leftarrow E(m_1))].
\end{align*}
\]
5. For $m \in \{0, 1\}^r$, we denote by $\overline{m}$ the padded $m$.
Assume that the encryption scheme is not ciphertext-indistinguishable.
Then, by Exercise 4, there is a probabilistic polynomial algorithm $A$ and
a positive polynomial $P$ such that for infinitely many $k$: For all $(n, e) \in I_k$
there are $m_{0,n,e}, m_{1,n,e} \in \{0, 1\}^r$, $m_{0,n,e} \neq m_{1,n,e}$ such that

$$\text{prob}(A(n, e, m_{0,n,e}, m_{1,n,e}, c) = m_{0,n,e} : (n, e) \leftarrow I_k, c \leftarrow \text{RSA}_{n,e}(\overline{m_{0,n,e}}))$$
$$- \text{prob}(A(n, e, m_{0,n,e}, m_{1,n,e}, c) = m_{0,n,e} : (n, e) \leftarrow I_k,$$
$$c \leftarrow \text{RSA}_{n,e}(\overline{m_{1,n,e}})) > \frac{1}{P(k)}.$$

Here, observe that there are only polynomially many, namely $<4k^2$, message pairs $\{m_0, m_1\}$, so we can omit the sampling algorithm $S$ (all message pairs can be considered in polynomial time).

Let $Q$ be a positive polynomial with $\deg Q > \deg P + 1$. Replacing $A$ by a modification, if necessary, we may assume that the probability of those $(n, e) \leftarrow I_k, m_0, m_1 \leftarrow \{0, 1\}^r$, such that either

$$\text{prob}(A(n, e, m_0, m_1, c) = m_0 : c \leftarrow \text{RSA}_{n,e}(\overline{m_0}))$$
$$- \text{prob}(A(n, e, m_0, m_1, c) = m_0 : c \leftarrow \text{RSA}_{n,e}(\overline{m_1})) \geq 0,$$

or the absolute value of the difference is $\leq \frac{1}{Q(k)}$, is $\geq 1 - \frac{1}{Q(k)}$. (The sign of the difference may be computed by a probabilistic polynomial algorithm with high probability, see Proposition 6.18 and, e.g., the proof of Proposition 6.17. Replace the output by its complement, if the sign is negative).

Then

$$\text{prob}(A(n, e, m_0, m_1, c) = m_0 : (n, e) \leftarrow I_k, m_0 \leftarrow \{0, 1\}^r,$$
$$m_1 \leftarrow \{0, 1\}^r \setminus \{m_0\}, c \leftarrow \text{RSA}_{n,e}(\overline{m_0}))$$
$$- \text{prob}(A(n, e, m_0, m_1, c) = m_0 : (n, e) \leftarrow I_k, m_0 \leftarrow \{0, 1\}^r,$$
$$m_1 \leftarrow \{0, 1\}^r \setminus \{m_0\}, c \leftarrow \text{RSA}_{n,e}(\overline{m_1})) > \frac{1}{2^{2r}} \frac{1}{2P(k)} \geq \frac{1}{8k^2P(k)}.$$

Let $\tilde{A}$ be the following algorithm with inputs $(n, e) \in I, y \in \mathbb{Z}_n, z \in \{0, 1\}^r$:

a. Randomly select $m_1 \leftarrow \{0, 1\}^r, z \neq m_1$.

b. $\tilde{A}(n, e, y, z) := \begin{cases} 1 & \text{if } A(n, e, z, m_1, y) = z, \\
0 & \text{else} \end{cases}$.
6. In order to decrypt, the recipient of the encrypted message \(c_1 \ldots c_n\) uses his secret trapdoor information to compute the elements \(x_j = f_i^{-1}(c_j)\). Then, he obtains \(m\) as \(B_i(x_1) \ldots B_i(x_n)\).

To prove security, assume that the scheme is not ciphertext-indistinguishable. Let \(S\) be a sampling algorithm and \(A\) be a distinguishing algorithm such that

\[
\text{prob}(A(i, m_0, m_1, c) = m_0 : i \leftarrow K(1^k), \{m_0, m_1\} \leftarrow S(i), c \leftarrow E(m_0)) \\
\quad - \text{prob}(A(i, m_0, m_1, c) = m_1 : i \leftarrow K(1^k), \{m_0, m_1\} \leftarrow S(i), c \leftarrow E(m_0)) > \frac{1}{P(k)},
\]

for some positive polynomial \(P\) and infinitely many \(k\) (see Exercise 4).

For \(m_0, m_1 \in \{0,1\}^n\) and \(0 \leq r \leq n\), we denote by \(s_r(m_0, m_1)\) the concatenation of the first \(n - r\) bits of \(m_0\) with the last \(r\) bits of \(m_1\). Thus, \(s_0(m_0, m_1) = m_0\) and \(s_n(m_0, m_1) = m_1\). We denote by \(m_{j,l}\) the \(l\)-th bit of \(m_j\). Then \(s_r(m_0, m_1) = m_{0,1} m_{0,2} \ldots m_{0,n-r} m_{1,n-r+1} \ldots m_{1,n}\).

For \(0 \leq r \leq n\), let

\[
p_r := \text{prob}(A(i, m_0, m_1, c) = m_1 : i \leftarrow K(1^k), \{m_0, m_1\} \leftarrow S(i), c \leftarrow E(i, s_r(m_0, m_1))).
\]
be the conditional probability assuming that $m_{0,l} = m_{1,l}$. Analogously for the condition $m_{0,l} \neq m_{1,l}$.

With this notation, we have $p_n - p_0 > \frac{1}{nP(k)}$. Since $p_n - p_0 = \sum_{r=0}^{n}(p_{r+1} - p_r)$, there is some $r$, $0 < r \leq n$, with $p_{r+1} - p_r > \frac{1}{nP(k)}$ (Recall $n = Q(k)$).

Hence $s_r(m_0, m_1) = s_{r+1}(m_0, m_1)$ if $m_{0,l} = m_{1,l}$, and thus $p_{r,m_{0,l} = m_{1,l}} = p_{r+1,m_{0,l} = m_{1,l}}$. Therefore, the inequality $p_{r+1} - p_r > \frac{1}{nP(k)}$ also implies

$$\text{prob}(m_{0,l} \neq m_{1,l}) \cdot (p_{r+1,m_{0,l} \neq m_{1,l}} - p_{r,m_{0,l} \neq m_{1,l}}) > \frac{1}{nP(k)}$$

We can approximately compute the probabilities $p_r$ by a probabilistic polynomial algorithm, with high probability (Proposition 6.18). We conclude that for a given positive polynomial $T$, there is a probabilistic polynomial algorithm that on input $1^k$ computes an $r$ with $p_{r+1} - p_r > \frac{1}{nP(k)}$, with probability $\geq 1 - 1/T(k)$.

Now, we give an algorithm $\hat{A}(i, y)$ which successfully computes the predicate $B$. In a preprocessing phase, $\hat{A}$ computes an $r$ with $p_{r+1} - p_r > \frac{1}{nP(k)}$ (with probability $\geq 1 - 1/T(k)$). $\hat{A}$ then uses this $r$ for all inputs $(i, y)$ with $i \in I_k$. Let $l := n - r - 1$. Note that $s_r(m_0, m_1)$ and $s_{r+1}(m_0, m_1)$ differ only in the $l$-th bit. On input $(i, y)$, $\hat{A}$ works as follows:

a. Compute $\{m_0, m_1\} \leftarrow S(i)$.

b. If $m_{0,l} = m_{1,l}$, then return a random $b \leftarrow \{0, 1\}$ and stop.

c. Else, i.e., if $m_{0,l} \neq m_{1,l}$, randomly (and uniformly) choose $x_1, \ldots, x_n$ such that $B_i(x_j)$ equals the $j$-th bit of $s_r(m_0, m_1)$. Let $y_j := f_i(x_j)$.

(Note that $y_1 \parallel y_2 \parallel \cdots \parallel y_n$ is an encryption of $s_r(m_0, m_1)$.)

d. Let $c := y_1 \parallel \cdots \parallel y_{l-1} \parallel y_{l+1} \parallel \cdots \parallel y_n$.

e. If $A(i, m_0, m_1, c) = m_0$, then return $\hat{A}(i, y) = B_i(x_1) = m_{0,l}$. Else, return $\hat{A}(i, y) = 1 - B_i(x_1) = 1 - m_{0,l} = m_{1,l}$.

We want to prove that for some positive polynomial $R$ and infinitely many $k$,

$$\frac{1}{2} + \frac{1}{R(k)} < \text{prob}(\hat{A}(i, f_i(x)) = B_i(x) : i \overset{\$}{\leftarrow} k(1^k), x \overset{\$}{\leftarrow} D_i)$$

$$= \text{prob}(m_{0,l} = m_{1,l}) \cdot \text{prob}(\hat{A}(i, f_i(x)) = B_i(x) | m_{0,l} = m_{1,l})$$

$$+ \text{prob}(m_{0,l} \neq m_{1,l}) \cdot \text{prob}(\hat{A}(i, f_i(x)) = B_i(x) | m_{0,l} \neq m_{1,l})$$

Introduction to Cryptography by H. Delfs and H. Knebl
= \text{prob}(m_{0,l} = m_{1,l}) \cdot \frac{1}{2} \\
+ \text{prob}(m_{0,l} \neq m_{1,l}) \cdot \text{prob}(\tilde{A}(i, f_i(x)) = B_i(x) | m_{0,l} \neq m_{1,l}) \\
= (1),

where the probabilities are taken over \(i \overset{\$}{\leftarrow} k(1^k), x \overset{\$}{\leftarrow} D_i\) and the coin tosses. This will be the desired contradiction to the fact that \(B\) is a hard-core predicate. The following probabilities are computed under the assumption that \(m_{0,l} \neq m_{1,l}\) (we omit the assumption in our notation).

\[
\text{prob}(\tilde{A}(i, f_i(x)) = B_i(x)) \\
= \text{prob}(B_i(x) = m_{0,l}) \cdot \text{prob}(\tilde{A}(i, f_i(x)) = m_{0,l} | B_i(x) = m_{0,l}) \\
+ \text{prob}(B_i(x) = m_{1,l}) \cdot \text{prob}(\tilde{A}(i, f_i(x)) = m_{1,l} | B_i(x) = m_{1,l}) \\
= \frac{1}{2} q_1 + \frac{1}{2} q_2 + \varepsilon \\
= (2)
\]

with

\[
q_1 := \text{prob}(A(i, m_0, m_1, c) = m_0 : i \overset{\$}{\leftarrow} k(1^k), \\
\{m_0, m_1\} \leftarrow S(i), c \leftarrow E(i, s_r(m_0, m_1)) \\
= 1 - \text{prob}(A(i, m_0, m_1, c) = m_1 : i \overset{\$}{\leftarrow} k(1^k), \\
\{m_0, m_1\} \leftarrow S(i), c \leftarrow E(i, s_r(m_0, m_1)) \\
= 1 - p_{r,m_0 \neq m_1,l},
\]

\[
q_2 := \text{prob}(A(i, m_0, m_1, c) = m_1 : i \overset{\$}{\leftarrow} k(1^k), \\
\{m_0, m_1\} \leftarrow S(i), c \leftarrow E(i, s_{r+1}(m_0, m_1)) \\
= p_{r+1,m_0 \neq m_1,l},
\]

and a negligibly small \(\varepsilon\), i.e., given a positive polynomial \(U, \varepsilon \leq \frac{1}{U(k)}\) for sufficiently large \(k\) (see Exercise 7 in Chapter 6).

Thus

\[
(2) = \frac{1}{2} + p_{r+1,m_0 \neq m_1,l} - p_{r,m_0 \neq m_1,l} + \varepsilon.
\]

We insert (2) in (1) and get

\[
(1) = \frac{1}{2} + \text{prob}(m_{0,l} \neq m_{1,l}) \cdot (p_{r+1,m_0 \neq m_1,l} - p_{r,m_0 \neq m_1,l} + \varepsilon) \\
> \frac{1}{2} + (1 - \frac{1}{T(k)}) \cdot \frac{1}{nP(k)} + \varepsilon \\
> \frac{1}{2} + \frac{1}{2nP(k)} = \frac{1}{2} + \frac{1}{2Q(k)P(k)},
\]

for infinitely many \(k\).

The proof is finished.

© H. Delfs and H. Knebl
7. To decrypt an encrypted message $c_1 \ldots c_n$, Bob checks (by using the factorization of $n$), whether $c_i$ is a quadratic residue or not. The security proof is almost identical to the proof of Exercise 6. It leads to a contradiction to statement 2 in Exercise 9 in Chapter 6, which is equivalent to the quadratic residuosity assumption (see Exercise 9).

8. Assume that there is a probabilistic polynomial distinguishing algorithm $\tilde{A}(n,m_0,m_1,c)$, with inputs $n \in I$, $m_0,m_1 \in \mathbb{Z}_n$, $c \in \mathbb{Z}_n^*$, a probabilistic polynomial sampling algorithm $S(n)$ and a positive polynomial $P$ such that

\[
\text{prob}(\tilde{A}(n,m_0,m_1,c) = m ) > \frac{1}{2} + \frac{1}{P(k)},
\]

for infinitely many $k$, where $E(n,m) = g^m r^n$, $r \in \mathbb{Z}_n^*$, $g = [1 + n]$.

We define a probabilistic polynomial algorithm $A = A(n,z)$ to distinguish $n$-th residues and randomly chosen elements in $\mathbb{Z}_n^*$, with inputs $n \in I$, $z \in \mathbb{Z}_n^*$ and output in $\{0,1\}$:

a. Apply $S(n)$ and get $\{m_0,m_1\} := S(n)$.

b. Randomly choose $b$ in $\{0,1\}$: $b \leftarrow \{0,1\}$, set $c = g^{m_b} z$.

c. Let $A(n,z) := \begin{cases} 1 & \text{if } \tilde{A}(n,m_0,m_1,c) = b, \\ 0 & \text{otherwise}. \end{cases}$

If $z$ is a $n$-th residue then $c$ is an encryption of $m_b$. $\tilde{A}$ returns $b$ with probability $> \frac{1}{2} + 1/P(k)$. Then $A$ also returns 1 with probability $> \frac{1}{2} + 1/P(k)$. If $z$ is not a $n$-th residue then $z = g^{m'_b} r^n$, $m_b \neq 0$, $g^{m_b} z = g^{m_b + m'_b} r^n$ is from $\tilde{A}$’s view a random element from $\mathbb{Z}_n^*$, independently chosen from $m_b$. Thus $\tilde{A}$ returns $b$ with probability $1/2$ and $A$ also returns 1 with probability $1/2$. Overall we get

\[
|\text{prob}(A(n,x^n) = 1 : n \leftarrow I_k, x \leftarrow \mathbb{Z}_n^*) - \text{prob}(\tilde{A}(n,x) = 1 : n \leftarrow I_k, x \leftarrow \mathbb{Z}_n^*)| > \frac{1}{2} + \frac{1}{P(k)} - \frac{1}{2} = \frac{1}{P(k)},
\]

for infinitely many $k$. This contradicts the decisional composite residuosity assumption.
10. Unconditional Security of Cryptosystems

1. a) Let \( x_0, x_1 \in \mathbb{F}_2 \), \( x_0 \neq x_1 \). Multiplying in \( \mathbb{F}_2 \) by some element \( x \in \mathbb{F}_2 \) is a linear map over \( \mathbb{F}_2 \). Thus, \( a \mapsto a \cdot (x_1 - x_0) \) can be computed by an \( l \times l \)-matrix \( M \) over \( \mathbb{F}_2 \). \( M \) is invertible, because \( x_1 \neq x_0 \). Let \( M' \) be the first \( f \) rows of \( M \). Then \( M' \) has rank \( f \). Therefore

\[
|\{a \in \mathbb{F}_2^l \mid M' \cdot a = 0\}| = 2^{l-f} \quad \text{and hence}
\]

\[
\text{prob}(\text{msb}(a \cdot x_0)) = \text{msb}(a \cdot x_1) : a \in \mathbb{F}_2^l = 2^{l-f} \cdot 2^{-f} = 2^{-f}.
\]

b) Let \( x_0, x_1, z_0, z_1 \in \mathbb{F}_2^l \), \( x_0 \neq x_1 \) and \( y_0, y_1 \in \mathbb{F}_2^l \). The equation

\[
\begin{pmatrix}
  x_0 & 1 \\
  x_1 & 1
\end{pmatrix}
\begin{pmatrix}
  a_0 \\
  a_1
\end{pmatrix}
= \begin{pmatrix}
  z_0 \\
  z_1
\end{pmatrix}
\]

has exactly one solution, since \( x_0 \neq x_1 \) and hence, the matrix is invertible. Thus

\[
|\{(a_0, a_1) \mid h_{a_0,a_1}(x_0) = y_0, h_{a_0,a_1}(x_1) = y_1\}| = 2^{l-f}2^{l-f} \quad \text{and}
\]

\[
\text{prob}(h_{a_0,a_1}(x_0) = y_0, h_{a_0,a_1}(x_1) = y_1 : a_0 \leftarrow \mathbb{F}_2^l, a_1 \leftarrow \mathbb{F}_2^l) = 2^{-f} \cdot 2^{-f}
\]

\[
= \frac{1}{|\mathbb{F}_2^l|^2}.
\]

2. The straightforward computation by using the definition of Rényi entropy gives

\[
H_2(X) = 3.61, H_2(X \mid Y) = 2.96, H_2(Y) = 0.50.
\]

Thus \( H_2(Y) + H_2(X \mid Y) < H_2(X) \).

3.

\[
H_2(X) = -\log_2 \left( (2^{-n/4})^2 + (2^n - 1) \frac{1 - 2^{-n/4})^2}{(2^n - 1)^2} \right)
\]

\[
= -\log_2 \left( 2^{-n/2} + \frac{(1 - 2^{-n/4})^2}{2^n - 1} \right) \approx \frac{n}{2},
\]

\( H_2(X \mid Y = 0) = 0 \), because \( X \mid Y = 0 \) is deterministic, and

\( H_2(X \mid Y = 1) = \log_2(2^n - 1) \), because \( X \mid Y = 1 \) is uniformly distributed.

Hence

\[
H_2(X \mid Y) = \text{prob}(Y = 0) \cdot H_2(X \mid Y = 0) + \text{prob}(Y = 1) \cdot H_2(X \mid Y = 1)
\]

\[
= (1 - 2^{-n/4}) \log_2(2^n - 1) = (1 - 2^{-n/4}) \log_2(2^n(1 - 2^{-n}))
\]

\[
= (1 - 2^{-n/4})(n + \log_2(1 - 2^{-n})) \geq (1 - 2^{-n/4})(n - 2^{-n}/\ln(2))
\]

\[
> (1 - 2^{-n/4})(n - 2^{-n}) > n - 2^{-n/4}n - 2^{-n},
\]

which is approximately twice as large as \( H_2(X) \).

© H. Delfs and H. Knebl
4. prob\( (X = x_i) = \text{prob}(E) \cdot \text{prob}(X = x_i|E) \)
   
   \[
   = \begin{cases} 
   (1-p) \cdot \text{prob}(X = x_i|E) & \text{if } i \neq i_0, \\
   p + (1-p) \cdot \text{prob}(X = x_i|E) & \text{if } i = i_0.
   \end{cases}
   \]

Let \( q_i := \text{prob}(X = x_i|E) \).

\[
\text{H}(X) = - \sum_{i=1}^{m} \text{prob}(X = x_i) \log_2(\text{prob}(X = x_i)) \\
= - \sum_{i=1}^{m} (1-p) \cdot q_i \log_2((1-p) \cdot q_i) \\
- p \log_2(p + (1-p) \cdot q_{i_0}) \\
- (1-p) \cdot q_{i_0} \log_2(p + (1-p) \cdot q_{i_0}) \\
+ (1-p) \cdot q_{i_0} \log_2((1-p) \cdot q_{i_0})
\]

\[
= - (1-p) \sum_{i=1}^{m} q_i \log_2(q_i) \\
+ p \log_2(p + (1-p) \cdot q_{i_0}) \\
+ (1-p) \cdot q_{i_0} \log_2((1-p) \cdot q_{i_0})
\]

\[
\leq - (1-p) \log_2(1-p) + (1-p) \log_2(m) - p \log_2(p) \\
= (1-p) \log_2(m) + h(p).
\]

Now let \( q_i = \text{prob}(X = x_i|E) = 1/m, 1 \leq i \leq m \). Then we can replace

\[
\text{the inequality by the following equality and continue}
\]

\[
= - (1-p) \log_2(1-p) + (1-p) \log_2(m) - p \log_2(p + (1-p) \cdot \frac{1}{m})
\]

\[
- (1-p) \cdot \frac{1}{m} \log_2 \left( 1 + \frac{p}{(1-p) \cdot \frac{1}{m}} \right)
\]

\[
\approx - (1-p) \log_2(1-p) + (1-p) \log_2(m) - p \log_2(p) \text{ for large } m
\]

\[
= (1-p) \log_2(m) + h(p).
\]

Introduction to Cryptography by H. Delfs and H. Knebl
6. From the well known equalities (see, for example, [BroSemMusMüh07, 2.7.2])

\[ \sin^2(x) = \frac{1}{2}(1 - \cos(2x)), \quad \cos^2(x) = \frac{1}{2}(1 + \cos(2x)) \]

we get

\[ \cos^2(x) \cos^2(y) + \sin^2(x) \sin^2(y) = \frac{1}{4}((1 + \cos(2x))(1 + \cos(2y)) + (1 - \cos(2x))(1 - \cos(2y))) = \frac{1}{2}(1 + \cos(2x) \cos(2y)). \]

7. Let \( E \) and \( E' \) be the random variables that capture the quantum states which Eve measures and re-sends, respectively.

a. From Lemma 10.18 we get that

\[ \text{prob}(E = |w_E\rangle | A = |v\rangle) = \text{prob}(E = |v_E\rangle | A = |w\rangle) = \sin^2(\theta), \]

\[ \text{prob}(B = |v\rangle | E' = |w_E'\rangle) = \text{prob}(B = |w\rangle | E' = |v_E'\rangle) = \sin^2(\theta'), \]

hence

\[ \text{prob}(E = |v_E\rangle | A = |v\rangle) = \text{prob}(E = |w_E\rangle | A = |w\rangle) = \cos^2(\theta), \]

\[ \text{prob}(B = |v\rangle | E' = |v_E'\rangle) = \text{prob}(B = |w\rangle | E' = |w_E'\rangle) = \cos^2(\theta'). \]

Applying the addition theorem from Exercise 6 (and Proposition B.3), we now obtain

\[ \text{prob}(B = |w\rangle | A = |w\rangle) = \text{prob}(B = |v\rangle | A = |v\rangle) = \cos^2(\theta) \cos^2(\theta') + \sin^2(\theta) \sin^2(\theta') \]

and hence

\[ \text{prob}(B = |v\rangle | A = |w\rangle) = \text{prob}(B = |w\rangle | A = |v\rangle) = \frac{1}{2}(1 - \cos(2(\theta) \cos(2\theta')). \]

From Lemma 10.18 and the subsequent remark we get that
prob(\(E = |w_E\rangle | A = |v\rangle\)) = \frac{1}{2} (1 - \cos(\delta) \sin(2\theta)),
\prob(B = |v\rangle | E' = |w_E\rangle) = \prob(B = |v\rangle | E' = |v_E\rangle)
= \frac{1}{2} (1 - \cos(\delta') \sin(2\theta')).

\text{hence}
\prob(\(E = |v_E\rangle | A = |v\rangle\)) = \prob(\(E = |w_E\rangle | A = |w\rangle\))
= \frac{1}{2} (1 + \cos(\delta) \sin(2\theta)),
\prob(\(B = |v\rangle | E' = |v_E\rangle\)) = \prob(\(B = |v\rangle | E' = |w_E\rangle\))
= \frac{1}{2} (1 + \cos(\delta') \sin(2\theta')).

\text{We now obtain}
\prob(B = |w\rangle | A = |w\rangle) = \prob(B = |v\rangle | A = |v\rangle)
= \frac{1}{2} (1 + \cos(\delta) \sin(2\theta)) \cdot \frac{1}{2} (1 + \cos(\delta') \sin(2\theta'))
+ \frac{1}{2} (1 - \cos(\delta) \sin(2\theta)) \cdot \frac{1}{2} (1 - \cos(\delta') \sin(2\theta'))
= \frac{1}{2} (1 + \sin(2\theta) \sin(2\theta') \cos(\delta) \cos(\delta')).

\text{Hence}
\prob(B = |w\rangle | A = |v\rangle) = \prob(B = |v\rangle | A = |w\rangle)
= \frac{1}{2} (1 - \sin(2\theta) \sin(2\theta') \cos(\delta) \cos(\delta')).

Alice selects the encoding basis randomly. Therefore, the error probability is
\(\epsilon = \epsilon(\theta, \theta', \delta, \delta')\)
= \frac{1}{2} (\prob(B = |w\rangle | A = |v\rangle) + \prob(B = |w\rangle | A = |v\rangle'))
= \frac{1}{4} (2 - \cos(2\theta) \cos(2\theta') - \sin(2\theta) \sin(2\theta') \cos(\delta) \cos(\delta')).

At this point, we see that \(\epsilon = \frac{1}{4}\) if Eve uses the same basis for
measurement and re-encoding, i.e., \(\theta = \theta', \delta = \delta', \) and \(\gamma = \alpha\) or
\(\gamma = \alpha + \pi,\) i.e., \(\delta = 0\) or \(\delta = \pi\) and hence \(\cos^2(\delta) = 1.\)
b. The error rate $\varepsilon$ depends on the differences $\delta, \delta'$ of the phase factors. We are interested in the minimum and maximum values of $\varepsilon$. The partial derivates with respect to $\delta$ and $\delta'$ vanish at these points. Therefore, we consider the partial derivatives

$$\frac{\partial \varepsilon}{\partial \delta} = \frac{1}{4} \sin(2\theta) \sin(2\theta') \sin(\delta) \cos(\delta'),$$

$$\frac{\partial \varepsilon}{\partial \delta'} = \frac{1}{4} \sin(2\theta) \sin(2\theta') \cos(\delta) \sin(\delta').$$

The derivative $\frac{\partial \varepsilon}{\partial \delta}$ is 0 if and only if

(a) $\theta = k \cdot \frac{\pi}{2}, k \in \mathbb{Z}$, or $\theta' = k \cdot \frac{\pi}{2}, k \in \mathbb{Z}$, or
(b) $\delta = k \cdot \pi, k \in \mathbb{Z}$, or
(c) $\delta' = \frac{\pi}{2} + k \cdot \pi, k \in \mathbb{Z}$.

The derivative $\frac{\partial \varepsilon}{\partial \delta'}$ is 0 if and only if

(a) $\theta = k \cdot \frac{\pi}{2}, k \in \mathbb{Z}$, or $\theta' = k \cdot \frac{\pi}{2}, k \in \mathbb{Z}$, or
(b') $\delta' = k \cdot \pi, k \in \mathbb{Z}$ or
(c') $\delta = \frac{\pi}{2} + k \cdot \pi, k \in \mathbb{Z}$.

If $a$ or $(c)$ or $(c')$ occurs, then $3/4 \geq \varepsilon = \frac{1}{4}(2 - \cos(2\theta) \cos(2\theta')) \geq 1/4$.

In the remaining case where $(b)$ and $(b')$ occur, we have $\cos(\delta) \cos(\delta') = \pm 1$.

If $\cos(\delta) \cos(\delta') = 1$, then $\varepsilon = \frac{1}{4}(2 - \cos(2\theta) \cos(2\theta') \sin(2\theta)(\sin(2\theta')) = \frac{1}{4}(2 - \cos(2(\theta - \theta'))),$ hence $3/4 \geq \varepsilon \geq 1/4$, with equality on either of the sides if and only if $\theta = \theta' + k \frac{\pi}{2}$. If $\cos(\delta) \cos(\delta') = -1$ then $\varepsilon = \frac{1}{4}(2 - \cos(2\theta) \cos(2\theta') + \sin(2\theta)(\sin(2\theta')) = \frac{1}{4}(2 - \cos(2(\theta + \theta'))),$ hence $3/4 \geq \varepsilon \geq 1/4$, with equality on either of the sides if and only if $\theta = -\theta' + k \frac{\pi}{2}$.

© H. Delfs and H. Knebl
11. Provably Secure Digital Signatures

1. We can use a pair of claw-free one-way permutations to construct a collision-resistant compression function \( \{0,1\}^{l(k)} \rightarrow \{0,1\}^{g(k)} \), as in Section 11.2. The collision resistance can be proven as in the proof of Proposition 11.7. Here, the prefix-free encoding is not necessary, since all strings in the domain have the same binary length. Then, we can derive a provably collision-resistant family of hash functions by applying Merkle's meta method.

2. Let \( K \) be the key generator of \( H \).

Let \( A(i,y) \) be a probabilistic algorithm with output in \( \{0,1\}^{\leq l_i} \) which successfully computes pre-images, i.e., there is a positive polynomial \( P \), such that

\[
\Pr(A(i,h_i(x)) \in h_i^{-1}(h_i(x)) : i \leftarrow K(1^k), x \leftarrow \{0,1\}^{\leq l_i}) \geq \frac{1}{P(k)}
\]

for \( k \) in an infinite subset \( K \subseteq \mathbb{N} \).

By \( D_k \) we denote the subset \( \{(i,x) \mid i \in I_k, x \in \{0,1\}^{\leq l_i}, h_i^{-1}(h_i(x)) = \{x\}\} \) of those \((i,x)\) where \( x \) is the only pre-image of \( h_i(x) \) and by \( D_i \) be the set of elements of \( D_k \) with key \( i \). \( h_i \) maps \( D_i \) injectively to \( \{0,1\}^{g(k)} \). Thus \( D_i \) contains at most \( 2^{g(k)} \) elements. Moreover, we have \( l_i \geq g(k) + k \) by assumption, thus

\[
\Pr(D_k) \leq \sum_{i \in I_k} \Pr(i) \cdot \frac{2^{g(k)}}{2^{l_i+1} - 1} \leq \sum_{i \in I_k} \Pr(i) \cdot \frac{1}{2^k} = \frac{1}{2^k}.
\]

In the computation, observe that the number of bit strings \( \leq l_i \) is equal to \( \sum_{j=1}^{l_i} 2^j = 2^{l_i+1} - 1 \).

Let \( D'_i := \{0,1\}^{\leq l_i} \setminus D_i \) be the complement of \( D_i \).

Lemma B.5 tells us that

\[
\Pr(A(i,h_i(x)) \in h_i^{-1}(h_i(x)) : i \leftarrow K(1^k), x \leftarrow \{0,1\}^{\leq l_i}) \geq \frac{1}{P(k)} - \frac{1}{2^k} \geq \frac{1}{2P(k)}
\]

for \( k \) in an infinite subset \( K' \subseteq \mathbb{N} \).

Now let \( \tilde{A}(i) \) be the following algorithm:

a. Randomly choose \( x \leftarrow \{0,1\}^{\leq l_i} \).

b. Return \((x, A(i,h_i(x)))\).

If \( x \neq A(i, h_i(x)) \) and \( A(i, h_i(x)) \in h_i^{-1}(h_i(x)) \), then \( \tilde{A} \) returns a collision of \( H \). (In fact, the algorithm computes a second pre-image. Therefore, our proof will even show that second-pre-image resistance implies the one-way property.)

We compute the probability of this event.

Introduction to Cryptography by H. Delfs and H. Knebl
\[ \text{prob}(x \neq A(i, h_i(x)), A(i, h_i(x)) \in h_i^{-1}(h_i(x)) : \]
\[ i \leftarrow K(1^k), x \leftarrow \{0, 1\}^{\leq l} \]
\[ \geq \text{prob}(x \neq A(i, h_i(x)), A(i, h_i(x)) \in h_i^{-1}(h_i(x)) : \]
\[ i \leftarrow K(1^k), x \leftarrow \{0, 1\}^{\leq l} \]
\[ = \text{prob}(A(i, h_i(x)) \in h_i^{-1}(h_i(x)) : i \leftarrow K(1^k), x \leftarrow \{0, 1\}^{\leq l}) \cdot \text{prob}(x \neq A(i, h_i(x)) | A(i, h_i(x)) \in h_i^{-1}(h_i(x)) : \]
\[ i \leftarrow K(1^k), x \leftarrow \{0, 1\}^{\leq l} \]
\[ \geq \frac{1}{2P(k)} \cdot \frac{1}{2} - \frac{1}{2^k} \]
\[ \geq \frac{1}{5P(k)} \]

for infinitely many \( k \in \mathcal{K}' \). This is a contradiction, since \( \mathcal{H} \) is assumed to be collision-resistant. Note that for \( x \in \mathcal{D}_i \) the fibre \( h_i^{-1}(h_i(x)) \) contains more than one element. So, for a randomly chosen \( x \), the probability that \( x \neq A(i, h_i(x)) \) is \( \geq 1/2 \).

3. a) RSA:
The attacks discussed in Section 3.3.1 are key-only attacks against the RSA one-way function which may result in the retrieval of secret keys. Forging signed messages \((m^e, m)\) (Section 3.3.4) is an existential forgery by a key-only attack. The “homomorphism attacks” can be used for universal forgery by chosen-message attacks.

b) ElGamal:
The retrieval of secret keys is possible, if the random number \( k \) is figured out by the adversary in a known-signatures attack (see Section 3.4.2). Existential forgery by a key-only attack is possible (loc.cit.). If step 1 in the verification procedure is omitted, then signatures can be universally forged by a known-signature attack, as Bleichenbacher observed (loc.cit.).

c. For a given \( m \), about \( n \) of the \( n^2 \) pairs \((s_1, s_2)\) are solutions of \( m = s_1^2 + ys_2^2 \). Choosing a pair randomly (and uniformly), the probability that it is a solution is about \( n^{-1} \approx 2^{-|m|} \) and hence negligible.

©H. Delfs and H. Knebl
5. a. We have $f_{[m],i}(\sigma(i, x, m)) = x = f_{[m'],i}(\sigma(i, x, m'))$. Let $[m] = m_1 \ldots m_r$ and $[m'] = m'_1 \ldots m'_{r'}$. Let $l$ be the smallest index $u$ with $m_u \neq m'_u$. Such an index $l$ exists, since neither $[m]$ is a prefix of $[m']$ nor vice versa. We have $f_{m_1 \ldots m_r,i}(\sigma(i, x, m)) = f_{m'_1 \ldots m'_{r'},i}(\sigma(i, x, m'))$, since $f_{0,i}$ and $f_{1,i}$ are injective. Then $f_{m_1+\ldots+m_r,i}(\sigma(i, x, m))$ and $f_{m'_1+\ldots+m'_{r'},i}(\sigma(i, x, m'))$ are a claw of $(f_{0}, f_{1})$.

b. A successful existential forgery by a key-only attack computes a valid signature $\sigma(i, x, m)$ from $(i, x)$, for some message $m$. Let $b$ be the first bit of $[m]$, i.e., $[m] = b m'$. Then $f_{m',i}(\sigma(i, x, m)) = f_{b,i}^{-1}(x)$. Thus, a pre-image of $x$ may be computed from $(i, x)$, for $f_{0,i}$ or $f_{1,i}$. We obtain a contradiction to the one-way property of $f_{0}$ and $f_{1}$.

c. Adaptively-chosen-message attack means in a one-time signature scheme that the adversary knows the signature $\sigma$ of one message $m$ of his choice and tries to forge the signature $\sigma'$ for another message $m'$.

Assume that a successful forger $F$ exists performing an adaptively-chosen-message attack. Then, we can define an algorithm $A$ which computes claws of $f_{0}, f_{1}$ with a non-negligible probability.

On input $i \in I_k$, $A$ works as follows:

i. Randomly choose a message $\tilde{m} \leftarrow \{0, 1\}^{\lceil \log_2(k) \rceil}$.

ii. Randomly choose $x \leftarrow D_i$ and compute $z := f_{[\tilde{m}],i}(x)$.

iii. Call $F(i, z)$ with the key $(i, z)$.

Note that $z$ is also uniformly distributed in $D_i$, since $x$ was chosen uniformly and $f_{[\tilde{m}],i}$ is bijective.

iv. $F(i, z)$ requests the signature $\sigma$ for a message $m$. If $m = \tilde{m}$ (which happens with probability $\geq 1/k^c$), then $\sigma = x$ is supplied to $F$.

Otherwise, $A$ returns with a failure.

v. If $F(i, z)$ now returns a valid forged signature $\sigma'$ for a message $m' \neq m$, then $A$ easily finds a claw of $f_{0}, f_{1}$, as shown in a).

$A$’s probability of success is greater than or equal to $F$’s probability of success multiplied by $1/k^c$, hence non-negligible. This is a contradiction.

d. We do not know how to simulate the legitimate signer and provide the forger with the requested signature, with a non-negligible probability. The approach of c), simply to guess the message in advance, does not work, if there are exponentially many messages.

6. a) The verification procedure for a signature $\sigma = (s, \tilde{m})$ for $m$ is:

1. Check whether $\tilde{m}$ is well-formed, i.e., $\tilde{m} = [\tilde{m}_1] \ldots ||[\tilde{m}_r]$ with messages $m_j \in \{0, 1\}^\ast$. 

Introduction to Cryptography by H. Delfs and H. Knebl
We now define an algorithm ˜
\[ m, (s, ˜m) \]
for k (m, valid signed message (encoding given in Section 11.2, and assume that an adversary learns a The first step cannot be omitted, in general. Take, e.g., the prefix-free f
\[ b \]
either finds a claw of
\[ i, x, F \]
when supplied with
\[ l, x \]
that were generated before by the user with key (i, x) (l = l(k) a polynomial function). Recall that the messages of user (i, x) are generated by the probabilistic polynomial algorithm M(i). Assume that F is successful. This means that there is a positive polynomial such that
\[
\text{prob}(F(i, x, m_1, \sigma_1, \ldots, m_l, \sigma_l) = (\tilde{m}, \tilde{\sigma}), \\
\text{Verify}(\tilde{m}, \tilde{\sigma}) = \text{accept} : i \leftarrow I_k, x \leftarrow D_i(, m_1, \ldots, m_l) \leftarrow M(i)) > \frac{1}{P(k)},
\]
for k in an infinite subset K ⊆ N.

We now define an algorithm A which, with non-negligible probability, either finds a claw of f_0, f_1 or inverts f_0 resp. f_1. This will be the desired contradiction. A works on input i, x as follows:

a. \( (m_1, \ldots, m_l) := M(i) \). Let \( m := [m_1] \ldots [m_l] \).
b. Let \( z := f_{m,i}(x) = f_{[m_1],i}(f_{[m_2],i}(\ldots f_{[m_l],i}(x) \ldots)) \).
c. Generate the signatures
\[
\sigma(i, z, m_1) = (f_{[m_2],i}(\ldots f_{[m_l],i}(x), \epsilon) \\
\sigma(i, z, m_2) = (f_{[m_3],i}(\ldots f_{[m_l],i}(x), [m_1]) \\
\ldots \\
\sigma(i, z, m_l) = (x, [m_1] \ldots [m_{l-1}]).
\]
Here, \( \epsilon \) denotes the empty string.
d. \( (\tilde{m}, \tilde{\sigma}) := F(i, z, m_1, \sigma(i, z, m_1), \ldots, m_l, \sigma(i, z, m_l)) \).
We have \( \tilde{\sigma} = (s, \tilde{m}) \) and \( \tilde{m} \neq m_j, 1 \leq j \leq l \). If \( (\tilde{m}, \tilde{\sigma}) \) does not pass the verification procedure, we return some random element and stop. Otherwise, if \( (\tilde{m}, \tilde{\sigma}) \) is a valid signature, i.e., \( f_{\tilde{m}[\tilde{m}],i}(s) = z \), we continue.
e. \( \tilde{m}[\tilde{m}] \) is not a prefix of m, because otherwise \( \tilde{m} \) would be equal to one of the \( m_j \). Here, note that by step 1 in the verification procedure, \( \tilde{m} \) is well-formed with respect to the prefix-free encoding.

The algorithm now distinguishes two cases.
f. Case 1: $m$ is not a prefix of $\tilde{m}||\tilde{m}$.
We have $\sigma(i, z, m_l) = (x, \ldots)$. Since $f_{m, i}(x) = z = f_{\tilde{m}||\tilde{m}}(s)$, we immediately find a claw of $f_{0, i}, f_{1, i}$, as in Exercise 6 a). We return this claw.

g. Case 2: $m$ is a prefix of $\tilde{m}||\tilde{m}$.
Let $\tilde{m}||\tilde{m} = m||u$. Note that $u \neq \varepsilon$, because $\tilde{m} \neq m_l$. Let $u = bu', b \in \{0, 1\}$. Since $f_{u, i}(s) = f_{m, i}^{-1}(z) = x$, we can immediately compute $f_{b, i}^{-1}(x) = f_{u', i}(s)$. We return this pre-image.

We have

$$\text{prob}(A(i, x) \text{ is a claw or one of the pre-images } f_{0, i}^{-1}(x), f_{1, i}^{-1}(x):$$

$$i \leftarrow I_k, x \leftarrow D_i) = \text{prob}(F(i, f_{m, i}(x), m_1, \sigma(i, f_{m, i}(x), m_1), \ldots, m_l, \sigma(i, f_{m, i}(x), m_l)))$$

$$= (\tilde{m}, \tilde{\sigma}),$$

$$\text{Verify}(\tilde{m}, \tilde{\sigma}) = \text{accept : }$$

$$i \leftarrow I_k, (m_1, \ldots, m_l) \leftarrow M(i), x \leftarrow D_i) = \text{prob}(F(i, z, m_1, \sigma(i, z, m_1), \ldots, m_l, \sigma(i, z, m_l)) = (\tilde{m}, \tilde{\sigma}),$$

$$\text{Verify}(\tilde{m}, \tilde{\sigma}) = \text{accept : } i \leftarrow I_k, (m_1, \ldots, m_l) \leftarrow M(i), z \leftarrow D_i)$$

$$> \frac{1}{P(k)},$$

for infinitely many $k$, a contradiction to the claw-freeness and the one-way property of $f_0, f_1$.

Note that $f_{m, i}$ is a permutation of $D_i$ and hence $z = f_{m, i}(x), x \leftarrow D_i$ is the uniform distribution $z \leftarrow D_i$.