Part 1: Collision within the Merkle-Damgård Construction

1. $H$ is a random function, hence the output is uniformly distributed, so it should be trivial to see that

$$\Pr[H(x) = H(x')] = \frac{1}{2^n}$$

To extrapolate in more detail, there are $(2^n)^2$ functions $h : \{0, 1\}^N \to \{0, 1\}^n$ and the probability that $H$ is equal to any of these is uniformly distributed. For a given $x, x'$ where $x \neq x'$, we obtain

$$\Pr[H(x) = H(x')] = \sum_h \Pr[H = h] * 1_{h(x) = h(x')} = \frac{1}{(2^n)^2} \sum_h 1_{h(x) = h(x')} = \frac{(2^n)^2 - 1}{(2^n)^2} = \frac{1}{2^n}$$

2. As the IV is fixed and is the same for both inputs $x, x'$, this probability is exactly the same as the probability we computed in the previous section, so

$$\Pr[h_1(IV, x_1) = h_1(IV, x_1')] = \frac{1}{2^n}$$

3. Both messages $x, x'$ have the same length $\ell$, and pad = cst$(N)$ where $N = \ell$ in this case, so we have the same pad for both $x, x'$. Thus, this probability is exactly the same as what we computed in the previous section. In fact,

$$\Pr[H(x) = H(x')|h_1(IV, x_1) \neq h_1(IV, x_1')] = \frac{1}{2^n}$$

4.

$$\Pr[H(x) = H(x')] = \Pr[h_1(IV, x_1) = h_1(IV, x_1')] * \Pr[H(x) = H(x')|h_1(IV, x_1) = h_1(IV, x_1')] + \Pr[h_1(IV, x_1) \neq h_1(IV, x_1')] * \Pr[H(x) = H(x')|h_1(IV, x_1) \neq h_1(IV, x_1')]$$

$$\quad = \frac{1}{2^n - 1} + (1 - \frac{1}{2^n}) * \frac{1}{2^n} = \frac{1}{2^n - 1} - \frac{1}{2^n}$$

for given $x, x'$ where $x \neq x'$.

5. We prove this by induction. For $d = 1$, by the previous section, this result is correct! Assuming this result is correct for $d$, we prove it for $d+1$.

For $d+1$, if the input to $h_{d+2}$ is $(A, x_{d+2})$ and $(A', x'_{d+2})$ for $x, x'$ respectively, we know that $x_{d+2} = x'_{d+2}$ as both messages have the same length. Calling $x_1$ as the message of length $d$ and $x_2$ as the message of length $d + 1$ and $B = H(x_1) = H(x_1')$ and $C = H(x_2) = H(x_2')$ and $D = h_{d+2}(A, \text{pad}) = h_{d+2}(A', \text{pad})$, we have

$$\Pr(C) = \Pr(D|B) * \Pr(B) + \Pr(D'|B) * \Pr(B') = 2^{-n} \sum_{i=0}^{d} (1 - 2^{-n})^i + 2^{-n} \sum_{i=0}^{d} (1 - 2^{-n})^i$$

$$\quad = 2^{-n} \sum_{i=0}^{d+1} (1 - 2^{-n})^i$$
If $d \to \infty$, we have a geometric series which converges, as $1 - 2^{-n} < 1$. So,

$$\Pr[H(x) = H(x')] = \frac{1}{2^n} \cdot \frac{1}{1 - (1 - 2^{-n})} = 1$$

We can conclude that Merkle-Damgård construction is not appropriate for arbitrary large message sizes!

6. We first need to compute the probability that $A = h_1(\mathbb{IV}, x_1) = h_1(\mathbb{IV}, x'_1)$ and $B = h_2(a, x_2) = h_1(a', x'_2)$, we have

$$\Pr(B) = \Pr(B|A) \cdot \Pr(A) + \Pr(B|\overline{A}) \cdot \Pr(\overline{A}) = 2^{-2n} + 2^{-n}(1 - 2^{-n}) = 2^{-n}$$

With similar computations as before, we obtain

$$\Pr[H(x) = H(x')] = 2^{-n} \sum_{i=0}^{d-1} (1 - 2^{-n})^i$$

7. First, look for the largest $j$ such that $x_j \neq x'_j$. Using the previous results, we have

$$\Pr[H(x) = H(x')] = 2^{-n} \sum_{i=0}^{d-j+1} (1 - 2^{-n})^i$$

**Part 2: RSA Variants with CRT Decryption**

1. We need to inverse $e$ modulo $\varphi(n) = (p-1)(q-1)$. This can be performed using the extended Euclid Algorithm.

2. Here, we have to extract the $e$th root of $c$ modulo $n$. Using Chinese Remainder Theorem, this can be obtained by extracting the $e$th root of $c$ modulo $p$ and the $e$th root of $c$ modulo $q$. Let $c_p := c \mod p$, $c_q := c \mod q$ and $d_p := e^{-1} \mod p - 1$, $d_q := e^{-1} \mod q - 1$. We then compute

$$m_p := c_p^{d_p} \mod p \text{ and } m_q := c_q^{d_q} \mod q.$$ 

By inverting the CRT transform on $(m_p, m_q)$, we get the desired plaintext. Note that replacing both $d_p$ and $d_q$ by $d := e^{-1} \mod (p-1)(q-1)$ would lead to the correct result as well.

**Multi-Prime RSA**

3. This probability corresponds to the ratio

$$\left| \mathbb{Z}_n^* \right| / \varphi(n) = \frac{(p-1)(q-1)(r-1)}{pqr} = \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) \left(1 - \frac{1}{r}\right).$$

Hence, this probability is very close to 1 for primes $p$, $q$, or $r$ of classical cryptographic size.

4. As in classical RSA, the exponent $e$ should be coprime with $\varphi(n)$. With this modulus, this corresponds to the condition $\gcd(e,(p-1)(q-1)(r-1)) = 1$. The decryption exponent is $d = e^{-1} \mod (p-1)(q-1)(r-1)$.

5. We extract an $e$th root componentwise on $(c_p, c_q, c_r)$ in $\mathbb{Z}_p \times \mathbb{Z}_q \times \mathbb{Z}_r$. To this end, we first compute $d_p := e^{-1} \mod p - 1$, $d_r := e^{-1} \mod r - 1$, $d_r := e^{-1} \mod r - 1$. The plaintext is retrieved by evaluating

$$\Psi^{-1}(c_p^{d_p} \mod p, c_q^{d_q} \mod q, c_r^{d_r} \mod r).$$
6. $e_p$ is an integer such that it is a multiple of $q$ and $r$. So, we can write $e_p$ of the form $kqr$, where $k$ is any integer. Since, $e_p$ must be congruent to 1 modulo $p$, it remains to choose $k$ to be the inverse of $qr$ modulo $p$. Applying a similar reasoning for $e_q$ and $e_r$ gives us

$$(e_p, e_q, e_r) = (qr \cdot ((qr)^{-1} \mod p), pr \cdot ((pr)^{-1} \mod q), pq \cdot ((pq)^{-1} \mod r)).$$

Finally, using the linearity with respect to the scalar multiplication, we get

$$\Psi^{-1}(x_p, x_q, x_r) = x_pe_p + x_pe_p + x_pe_p = x_p(qr \cdot ((qr)^{-1} \mod p) + x_q(pr \cdot ((pr)^{-1} \mod q) + x_r(qr \cdot ((pq)^{-1} \mod r)).$$

7. The complexity is mainly due to the modular exponentiations. With the classical RSA modulus, we need to perform 2 modular exponentiations modulo a number of size $s/2$. The second variant requires 3 modular exponentiations modulo a number of size $s/3$. So, the respective asymptotic complexities are within the order of magnitude of $2(s/2)^3$ and $3(s/3)^3$. So, the second variant is faster of a multiplicative factor of $9/4$.

**Multi-Power RSA**

8. We generate two prime numbers $p$ and $q$ of a given size by picking numbers at random until the Miller-Rabin test outputs “pseudo-prime”. We set $n = pq$. Then, we select a public exponent $1 \leq e \leq \varphi(p^2q)$ such that $\gcd(e, \varphi(p^2q)) = \gcd(e, p(p-1)(q-1)) = 1$. The decryption exponent is obtained by computing $d := e^{-1} \mod p(p-1)(q-1)$. The public key is $(n, e)$ and the secret key is $(n, d)$. We encrypt a message $m \in \mathbb{Z}_n^*$, by computing $m^e \mod n$. The decryption is performed as follows $c^d \mod n$.

9. We need to find a $k$ satisfying

$$(x_1 + kp)^e \equiv y_1 + \ell p \pmod{p^2}.$$

From this, we get

$$x_1^e + ekp \equiv y_1 + \ell p \pmod{p^2}$$

and

$$k = \left(\frac{x_1^e - y_1 \mod p^2}{p}\right) e^{-1} \mod p.$$

10. Let $c$ be a given ciphertext. We first compute $c_p := c \mod p^2$ and $c_q := c \mod q$. In order to extract an $e$th root of $c_p$ modulo $p^2$, we extract this root modulo $p$ and apply the technique of the previous question to retrieve this root modulo $p^2$. So, we compute $m_0p := d_p c_p^d \mod p$, where $d_p := e^{-1} \mod p - 1$ ($d$ would be correct as well, but less efficient!). Then, using the previous technique, we retrieve $m_p \in \mathbb{Z}_{p^2}^*$ such that $m_p^e \equiv c_p \pmod{p^2}$. We also compute $m_q := d_q c_q^d \mod q$, where $d_q = e^{-1} \mod q - 1$. Finally, inverting the CRT transform on the pair $(m_p, m_q)$ allows to retrieve the plaintext.

11. The complexity of the above method is mainly due to 2 modular exponentiations modulo a number of size $s/3$. Hence, the asymptotic complexity is within the order of magnitude $2(s/3)^3$. If we compare with the classical RSA with CRT, get a ratio of

$$\frac{2(s/2)^3}{2(s/3)^3} = \frac{27}{8}.$$