1 Fun with Digital Signatures

Let \((KeyGen, Sign, Verify)\) be a secure digital signature scheme for \(k\)-bit messages. For a fixed secret key, let \(\sigma(m)\) be a signature, where the secret key is omitted when it’s clear from the context. You would like to sign a message \(M\), where \(|M| > k\).

Consider the following three signature schemes \(\Sigma\):

1. Let \(M = m_1 \circ m_2 \circ \ldots \circ m_n\), such that each \(m_i\) is of length \(k\). Note that if \(M\) is not a multiple of \(k\), then we will pad the end of \(M\) with extra 0s.

\[\text{Scheme 1: } \Sigma(M) = (\sigma(m_1), \sigma(m_2), \ldots, \sigma(m_n))\]

2. Choose the smallest \(n\) such that \(\lfloor \log_2 (n+1) + \frac{|M|}{n} \rfloor \leq k\). (Assume that \(|M|\) is small enough and \(k\) is large enough to make this is possible.) Then break \(M\) up into \(M = m_1 \circ \ldots \circ m_n\), where each \(m_i\) is such that \(|m_i| = k - \lfloor \log_2 (n+1) \rfloor\), and \(m_n\) is padded with 0s as necessary.

\[\text{Scheme 2: } \Sigma(M) = (\sigma(1 \circ m_1), \sigma(2 \circ m_2), \ldots, \sigma(n \circ m_n))\]

where each index \(i\) is represented using \(\lfloor \log_2 (n+1) \rfloor\) bits.

3. Choose the smallest \(n\) such that \(\lfloor 2 \log_2 (n+1) + \frac{|M|}{n} \rfloor \leq k\). (Assume that \(|M|\) is small enough and \(k\) is large enough to make this is possible.) Then break \(M\) up into \(M = m_1 \circ \ldots \circ m_n\), where each \(m_i\) is such that \(|m_i| = k - \lfloor 2 \log_2 (n+1) \rfloor\), and \(m_n\) is padded with 0s as necessary.

\[\text{Scheme 3: } \Sigma(M) = (\sigma(1 \circ m_1), \sigma(2 \circ m_2), \ldots, \sigma(n \circ n \circ m_n))\]

where each index is represented using \(\lfloor \log_2 (n+1) \rfloor\) bits.

a. Briefly explain a vulnerability that is in \textbf{Scheme 1}, but not in \textbf{Scheme 2} or \textbf{Scheme 3}.

Give a target-message attack using adaptive queries to the signer (i.e. an attack in which \(A\) can pick specific messages to be signed in order to forge on message \(M\) that will succeed on \textbf{Scheme 1}, but not on \textbf{Scheme 2} or \textbf{Scheme 3}. 

Homework 10
b. Briefly explain a vulnerability that is in Scheme 2, but not in Scheme 3.
Give a target-message attack using adaptive queries to the signer that will succeed most of the time on Scheme 2, but not on Scheme 3. If your attack does not always succeed, state which messages it will fail on and explain why.

c. Briefly explain a vulnerability that is in Scheme 3.
Give a target-message attack using adaptive queries to the signer that will succeed on Scheme 3.

d. Give a scheme that is based on Scheme 3, but is not vulnerable to the attack that you gave in part c. (You may NOT use CRHF as part of your construction.) Explain why your scheme fixes the vulnerability that is exhibited by Scheme 3.

e. Show that your scheme from part d. is EUF-CMA.

2 The Schnorr Signature

The Schnorr signature is a signature scheme (\(\text{Gen}, \text{Sign}, \text{Verify}\)) that is secure in the random oracle model under the assumption that the discrete logarithm problem is hard. The scheme is defined as follows:

**Key Generation** First \(\text{Gen}\) picks a group \(G\) as follows. Pick a \(k\)-bit prime \(p = aq + 1\) where \(q\) is a prime number of length \(\Theta(k)\), and pick some \(g \in \mathbb{Z}_p^*\) with order \(q\). Let \(G = \langle g \rangle\) (the subgroup of \(\mathbb{Z}_p^*\) generated by \(g\)). Next it picks a hash function \(H : \{0, 1\}^* \rightarrow \mathbb{Z}_q\) and a secret key \(x \leftarrow \mathbb{Z}_q\). Finally, \(\text{Gen}\) outputs \(\text{VK} = (H, p, q, g, h = g^x)\) and \(\text{SK} = x\).

**Signing** To sign a message \(m\), \(\text{Sign}\) chooses a random \(r \leftarrow \mathbb{Z}_q\) and computes

\[
A = g^r \pmod{p} \\
c = H(m \circ A) \\
s = r + xc \pmod{q}
\]

and outputs the signature \(\sigma = (A, s)\).

**Verification** To verify a signature \((m, \sigma)\) with \(\sigma = (A, s)\), \(\text{Verify}\) computes \(c = H(m \circ A)\) and checks that \(g^s = Ah^c\).

a. Prove that the Schnorr signature scheme is correct (i.e. any signature generated by the signing algorithm will pass verification).

b. Prove that the Schnorr signature scheme is existentially unforgeable against an adaptive attack in the random oracle model (i.e. your reduction may implement its ideal hash function to answer the adversary’s queries). Consider what the reduction gives
the adversary as input. How does it answer signature queries? How does it answer queries to the hash function?

**Hint:** The reduction can rewind the adversary to a specific point and rerun it from there. If it does so, it may answer future queries to the hash function differently the second time through. Also note that for a fixed $A$ and two different values $c_1, c_2$, if we have the correct values $s_1, s_2$ such that $g^{s_i} = Ah^{c_i}$, then we can efficiently compute $x$.

### 3 Merkle Trees

Let $(G, H(\cdot))$ be a family of collision-resistant hash functions where $H_{pk} : \{0, 1\}^* \rightarrow \{0, 1\}^k$ for $pk \in G(1^k)$.

For $n = 2^\ell$, the Merkle hash $\text{Merkle}_{pk,n}(x_0, \ldots, x_{n-1})$ defined by $(G, H(\cdot))$ is an algorithm that takes as input a public key $pk$ of the hash function $H_{pk}$ and $n$ binary strings $x_0, \ldots, x_{n-1}$. For every binary string $s$ of length at most $\ell$, compute its label $u_s$ as follows: if $|s| = \ell$, then $u_s = H_{pk}(x_s)$; otherwise $u_s = H_{pk}(u_{s0} \circ u_{s1})$. Output $u_\varepsilon$ (where $\varepsilon$ is the empty string).

![Merkle Tree Diagram](image)

The authenticating path of $x_i$ in $\text{Merkle}_{pk,n}(x_0, \ldots, x_{n-1})$ is the set of values $(u_{t_1}, \ldots, u_{t_\ell})$ where each $t_j$ is a string of length $j$ where the first $j - 1$ bits agree with the first $j - 1$ bits of $i$, and the last bit is different (so for example if $i = 00111$, then $t_1 = 1$, $t_2 = 01$, $t_3 = 000$, $t_4 = 0010$, $t_5 = 00110$). These are just the labels of the nodes whose siblings are on the path from the $i$th leaf to the root.

To verify an authenticating path, that is, to verify that a given string $x$ is the $i$th string in the set of strings that was hashed together to obtain the value $v$, $\text{MVerify}_{pk,n}(v, x, i, (u_{t_1}, \ldots, u_{t_\ell}))$
proceeds as follows: first compute $u_i = H_{pk}(x)$. For $j$ from $\ell - 1$ to 0: Let $I_j$ be the $j$-bit prefix of the binary string $i$. Compute $u_I = H_{pk}(u_{Ij0} \circ u_{Ij1})$ (if the $j + 1$st bit of $i$ is 0, then $u_{Ij0} = u_{Ij1}$ computed in the previous iteration, while $u_{Ij1}$ was given as part of the authenticating path; if the $j + 1$st bit of $i$ is 1, then it’s the other way around). Finally, accept if $u_e = v$, and reject otherwise. Intuitively, this procedure iteratively computes the label on each node on the path from the $i$th leaf to the root from the labels of its children, and accepts if the label it computed for the root is $v$.

a. What are the domain and range of $\text{Merkle}_{pk,n}$? Show that $(G, \text{Merkle}_{(\cdot),n})$ is a family of collision-resistant hash functions for these domain and range, assuming that $(G, H_{(\cdot)})$ is a collision-resistant hash function family.

b. Prove that, assuming that $(G, H_{(\cdot)})$ is a collision-resistant hash function family, $\text{MVerify}_{pk,n}$ is sound. Namely, no $A$ can produce an authenticating path for the same Merkle root $v$ but conflicting $i$th documents $x$ and $x'$. More formally, show that, for every probabilistic polynomial time $A$ there exists a negligible $\nu$ such that

$$\Pr[pk \leftarrow G(1^k); (v, x, x', i, a, a') \leftarrow A(pk) : x \neq x' \land \text{MVerify}_{pk,n}(v, x, i, a) \land \text{MVerify}_{pk,n}(v, x', i, a')] = \nu(k)$$