Cryptography: Answers to Homework 3

Vishvas Vasuki

May 4, 2009

Remark. No Collaborators unless specifically acknowledged in individual solutions.

1 Answer to question 1

The answer is simply the Lamport one time signature scheme, about which I read in the Wikipedia.

Some notation

Let the message space be contained in $\{0,1\}^k$. Alternatively, you can think of this space as the space to which all messages are hashed to by a collision resistant hash function, prior to signing.

Let G be the prime order group of order p. Let g be some generator of G.

The one-time signature scheme Sig

- Setup(l): For $i \in \{1, ..k\}$ and $j \in 0, 1$, pick numbers $y_{i,j}$ uniformly at random from G. Compute numbers $z_{i,j} = g^{y_{i,j}} \mod p$. The verfication key $VK = (z_{i,j})$. Signing key $SK = (y_{i,j})$.
- Sign(m): Let m_i represent the ith bit of m. Then, signature $s = (y_{1,m_1} .. y_{k,m_k})$.
- Verify(m, s): The following checks are done for every number in the vector s: $g^{y_{i,m_i}} \mod p \stackrel{?}{=} z_{i,m_i}$. If all tests succeed, the s is considered to be a valid signature for m.

Proof of unforgeability under one-time signature scheme

Consider the following security game. Let A be the Discrete log assumption challeger. Let B be the Sig challenger/ Discrete log assumption attacker. Let C be an attacker against the Sig scheme.

- A challenges B to find the a in $b = g^a \mod p$, given g, p and b.

- B runs the setup algorithm described above. B then replaces a randomly picked $z_{i,j}$ with b. So, $z_{i,j} = b$ now.
- B tells C the VK.
- C asks B to sign some message of its choice m'.
- B returns the signature s', if $m'_i \neq j$. Here j is one of the indices of the number $z_{i,j}$ which was replaced with b. Otherwise, B aborts. As $z_{i,j}$ was randomly chosen, this happens with probability 1/2.
- C forges a valid signature s for message $m \neq m'$ with some non negligible probability p.
- m differs from m' by atleast 1 bit. If one of the changed bits is i, and if s is valid, B obtains the solution for $b = g^a \mod p$, $y_{i,j} = a$ from the ith number in s. As i, j were randomly chosen, this happens with probability at least 1/k.
- In such a case, B can respond to A's challenge by sending it a.

Thus, B successfully responds to A's challenge with probability at least $\frac{p}{2k}$, which is non negligible.

2 Answer to question 2

We want to show a separation between weak signatures and existentially unforgeable signatures. In the weak definition, the attacker must tell the challenger all the messages he wants signed before seeing the public key.

It is obvious that every existentially unforgeable signature is also a weak signature. To show the separation between these two classes, we will therefore construct a weak signature scheme Sig-W which is not existentially unforgeable.

Take any standard signature scheme, Sig, with the algorithms Setup, Sign and Verify.

Scheme 1

Acknowledgement. This scheme emerged during a discussion with Rashid Kaleem.

We build the algorithms of Sig_W as follows:

- $Setup_W(\lambda, m_1, ..m_n)$:

  As this is the setup function of a weak signature scheme, we are assuming that it will know beforehand the messages $m_1, ..m_n$, whose signatures the adversary will see.

  Call Setup(λ). This returns the public key VK and the secret key SK. Set $VK_W = VK, SK_W = (SK, m_1, ..m_n, m')$, where m' is picked randomly from the message space such that $m' \notin (m_1, ..m_n)$.

- $Sign_W(m, SK_W)$:

  If $m \in (m_1, ..m_n)$, return Sign(m, SK). Otherwise, return Sign(m', SK).

- $Verify_W(m, s, VK_W)$:

  Simply return the result of $Verify(m, s, VK)$

Sig_W is a weak signature.

To see this, suppose that there existed an attacker C which is able to break Sig_W under the weak signature security game. We will see that C can be used to build an attacker B which is successful against the strong signature scheme, Sig. Consider the following security game, where A is the challgenger for Sig.

- A runs Setup, and announces VK to B.
- B simulates the functioning of the weak signature scheme Sig_W.

  It gathers from C the messages it wants signed: $(m_1, ..m_n)$.

  B announces $VK_W = VK$ to C.
- B gets signatures for $(m_1, ..m_n)$ from A and passes it on to C.
- C produces a forgery on message $m \notin (m_1, ..m_n)$ with some non negligible probability p.
- B then transmits C's attempt at forgery to A. Thus, B succeeds with non negligible probability p.

Sig_W is not existentially unforgeable.

Under Sig_W, all $m \notin (m_1, ..m_n)$ have the same signature s. Under the definition of existential unforgeability, the attacker is allowed to make a polynomial number of signature queries, after seeing the verification key VK. So, an attacker C would simply need to see the signatures of n+2 distinct messages. Then, C could trawl through the signatures and find the signature s which appears atleast twice. Then, it picks a message m at random from the message space. Then, if M is the size of the message space, with probability at least $\frac{n+2}{M}$, s is a forged signature for m.

A potential problem

Is the proof that Sig_W is a weak signature valid? It may not be, because the attacker C is not guaranteed to work on such a weird distribution.

3 Answer to question 3

A precise definition for a pseudorandom bit generator

Let n be the security parameter. For making a precise argument, we will use k(n) and l(n) instead of k and l mentioned in the question.

$G : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ for $l(n) > k(n)$ is called pseudorandom if, for all polynomial time (possibly randomized) algorithms A, and for all polynomials q(n), $\exists n_0 : n > n_0 \implies |Pr_{x \in \{0,1\}^{k(n)}}(A(G(x))) - Pr_{y \in \{0,1\}^{l(n)}}(A(y))| \leq \frac{1}{q(n)}$.

The generator G'

Consider the function $G' : \{0,1\}^{2k(n)} \to \{0,1\}^{2l(n)}$ which is evaluated by executing G on the first k(n) input bits and then on the next k(n) input bits. We will show that G' is also pseudorandom bit generator.

Proof

Note that this proof implicitly, but not explicitly, uses the hybrid technique.

Suppose, for the sake of contradiction, that G' is not a pseudorandom bit generator. Consider any polynomial time algorithm A.

For each i, let t_i denote the probability that A(y)=1 when the first i bits of y are the first i bits of G'(x), evaluated over a random input x; and the remaining bits of y are truly random.

If G' were not a pseudorandom generator, $1/q(n) \leq |t_0 - t_{2l(n)}| \leq |t_0 - t_{l(n)}| + |t_{l(n)} - t_{2l(n)}|$ for some polynomial q(n) and for arbitrarily large n. As G is a pseudorandom generator, $|t_0 - t_{l(n)}| \leq (2q(n))^{-1}$ for sufficiently large n. So, we have: $(2q(n))^{-1} \leq |t_{l(n)} - t_{2l(n)}|$ for arbitrarily large n.

With A, one can build a randomized algorithm B which can tell if it has been supplied truly random bits, or bits from G. This is done as follows: Given $x \in 0, 1^{l(n)}$, B first runs G by supplying it k(n) truly random bits. It prepends the l(n) bits it gets from G to x, and gets the bit-string y. B then runs A on y. As $(2q(n))^{-1} \leq |t_{l(n)} - t_{2l(n)}|$, B behaves noticably differently on truly random x, and on x generated from G.

The existence of B then leads us to a contradiction of our assumption that G is a pseudorandom bit generator. Thus, B cannot exist. Thus, G' is a pseudorandom bit generator if G is.