1 Security Interference

We consider a zero-knowledge proof of knowledge $\pi$ in which a prover $P(x, w)$ holding a witness $w$ for an instance $x$ can convince a verifier $V(x)$ that he knows $w$ such that the relation $R(x, w)$ holds.

We construct a mutual-authentication protocol $\pi'$ in which two participants $A(x, w)$ and $B(x, w)$ share the secret $w$ for the instance $x$. The protocol $\pi'$ runs as follows:

1: $A$ and $B$ execute $\pi$: $A$ runs $P(x, w)$ and $B$ runs $V(x)$
2: if $V(x)$ accepted for $B$, $B$ sends $w$ to $A$
3: $A$ accepts if and only if $w$ is correct

Q.1 Show that there is an algorithm $E^{C^*}$ calling $C^*$ as a subroutine such that, for every input $z$ and every malicious algorithm $C^*(x, z)$, if $C^*(x, z)$ interacts with $B(x, w)$ and $B(x, w)$ accepts, then $E^{C^*}(x, z) = w'$ such that $R(x, w')$ holds.

If $B(x, w)$ accepts, it must be during the execution of $\pi$. So, $C^*(x, z)$ can make $V(x)$ execute $\pi$ and accept. We know that $\pi$ is a sound proof of knowledge. So, we can use the extractor $E^{C^*}$ and extract a valid witness $w'$.

Q.2 Show that there is an algorithm $S^{C^*}$ calling $C^*$ as a subroutine such that, for every input $z$ and every malicious algorithm $C^*(x, z)$, if $C^*(x, z)$ interacts with $A(x, w)$ and $A(x, w)$ accepts, then $S^{C^*}(x, z) = w'$ such that $R(x, w')$ holds.

WARNING: $S$ does not know $w$, a priori.

We can consider $C^*$ as a malicious verifier who produces a final output $w'$. If $A(x, w)$ accept, it must be that $w'$ is a valid witness for $x$ (which is actually $w$). We know that $\pi$ is zero-knowledge. So, we can use the simulator $S^{C^*}$ and extract some $w'$ which is indistinguishable. The distinguisher checking $R(x, w')$ must have a negligible advantage. So, $w'$ must be a valid witness for $x$. 
Q.3 Show that $\pi$ and $\pi'$ do not compose: even though a malicious verifier learns nothing from $P(x, w)$ and a malicious Alice learns nothing from $B(x, w)$, in a network where $P(x, w)$ and $B(x, w)$ are two honest participants, show that a malicious participant can extract $w$.

| The malicious participant relays messages between $P(x, w)$ and $B(x, w)$. Clearly, $B(x, w)$ accepts and sends $w$ as his last message and the attack stops. The adversary has learnt $w$. |

|
2 Distance Bounding

We consider a distance-bounding protocol, in which there is a prover $P$ and a verifier $V$ sharing a secret $x$. The protocol starts with an initialization phase which consists of setting up a matrix $a \in \{0, 1\}^{n \times 2}$ to be shared between $P$ and $V$. (We will see later how this initialization phase works.) Then, we have $n$ rounds of time-critical challenge-response exchanges: in the $i$th round, $V$ sends a random $c_i \in \{1, 2\}$ to which $P$ answers by $r_i = a_{i,c_i}$. $V$ accepts the response if it is correct and if the elapsed time between sending $c_i$ and receiving $r_i$ is at most $\frac{2B}{C}$, where $B$ is a distance bound and $C$ is the speed of light. We say that the protocol succeeds if $V$ accepts the response in all rounds. We assume that the time used to compute is negligible against the time of flight of messages. So, a honest prover within a distance up to $B$ can pass all rounds. We want the protocol to resist to two types of threats:

- In a concurrent setting with several honest provers using key $x$ and several honest verifiers using key $x$, including a target verifier $V$, if there is no prover within a distance up to $B$ to $V$, no malicious participant $A$ can make a protocol with $V$ succeed. If this holds, we say the protocol is secure.
- A malicious prover within a distance larger than $B$ to the verifier cannot make the protocol succeeds. In what follows we call this threat a distance fraud.

We stress that the above malicious participant starts by ignoring $x$ while the malicious prover in distance fraud knows $x$.

Q.1 (General security upper bound.)

We assume that the initialization phase is such that $a$ computed by a honest verifier is a uniformly distributed matrix no matter any malicious environment.

Q.1a We consider a honest verifier $V$ and a malicious participant $A$ with no other participant. Show that $A$ can make the protocol succeed with probability $2^{-n}$.

| A can just send a random response. It passes with probability $\frac{1}{2}$. So, the protocol succeeds with probability $2^{-n}$. |

Q.1b We consider a man-in-the-middle $A$ between a honest prover $P$ and a honest verifier $V$ who are within a distance larger than $B$.

Show that $A$ can make the protocol succeed with probability $\left(\frac{3}{4}\right)^n$.

HINT: assume that $A$ can make a challenge-response exchange with $P$ before he receives the first challenge from $V$.

| After the initialization phase where $A$ passively relays messages between $P$ and $V$, we make $A$ send random challenges to $P$ and get his responses $r_i$. When a challenge $c_i$ is received from $V$, $A$ sends $r_i$. Clearly, if $A$ has picked $c_i$ as the $i$th challenge sent to $P$ (this happens with probability $\frac{1}{2}$), the round passes. Otherwise (with another probability $\frac{1}{2}$), the response $r_i$ is accepted with probability $\frac{1}{2}$. So, the round passes with probability $\frac{1}{2} \times 1 + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}$. Hence, the protocol succeeds with probability $\left(\frac{3}{4}\right)^n$. |

Q.2 (General distance fraud.)

We make the same assumption on $a$. 
Q.2a Show that a far-away malicious prover who sends random \( r_i \)'s can make a distance fraud with probability \( 2^{-n} \).

**HINT:** assume that the malicious prover can predict when \( c_i \) will be sent by the verifier.

If the prover predicts that \( c_i \) will be sent at time \( t \), he sends a random \( r_i \) between time \( t - \frac{d}{C} \) and time \( t + \frac{2B-d}{C} \) (where \( d \) is the distance between \( A \) and \( V \)) so that it reaches the verifier after time \( t \) and before time \( t + \frac{2B}{C} \). The response is correct with probability \( \frac{1}{2} \). So, the protocol succeeds with probability \( 2^{-n} \).

Q.2b Find another strategy so that the distance fraud works with probability \( \left( \frac{3}{4} \right)^n \).

He sends a random \( r_i \) selected in \( \{a_{i,1}, a_{i,2}\} \). It is always correct if \( a_{i,1} = a_{i,2} \). Otherwise, it passes with probability \( \frac{1}{2} \). Since \( a_{i,1} = a_{i,2} \) with probability \( \frac{1}{2} \), the probability to pass a round is \( 1 \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4} \). So, the protocol succeeds with probability \( \left( \frac{3}{4} \right)^n \).

Q.3 (Distance fraud for a dedicated protocol.)

We consider a protocol with the following initialization phase: The verifier selects a nonce \( N_V \in \{0,1\}^n \) and sends it to the prover. The prover selects a nonce \( N_P \in \{0,1\}^n \) and sends it to the verifier. Both compute \( a_{i,1} = \text{PRF}_x(N_V) \) by using a pseudorandom function \( \text{PRF} \) and \( a_{i,2} = a_{i,1} \oplus N_P \).

Make a distance fraud which succeeds with probability 1.

A malicious prover could take \( N_P = 0 \) so that \( a_{i,2} = a_{i,1} \). This way, the correct response in round \( i \) would always be equal to \( a_{i,1} \) no matter the challenger. So, a malicious prover could send the correct response before receiving the challenge so that it will reach the verifier on time.

Q.4 (Security of a dedicated protocol.)

We now modify the initialization phase by having \( a_{i,1} = \text{PRF}_x(N_P, N_V) \) and \( a_{i,2} = a_{i,1} \oplus x \).

Q.4a Show that a malicious man-in-the-middle between \( P \) and \( V \) (who are within a distance up to \( B \)) can extract \( x \).

**HINT:** assume that the adversary can see if the protocol succeeded on the side of \( V \).

We consider a man-in-the-middle who passively relay messages except the challenge \( c_i \) which is flipped: if the challenge \( c_i \) is received from \( V \), the challenge \( 3 - c_i \) is sent to \( P \). The response \( r_i \) is relayed.

We note that \( r_i = a_{i,3-c_i} \) while the verifier expect \( a_{i,c_i} \). The difference between the two is \( x_i \). Since all other challenge must be accepted, the protocol succeeds if and only if \( x_i = 0 \). So, by seeing whether the protocol succeeds, the man-in-the-middle can deduce \( x_i \).

Q.4b In a setting with \( n \) provers and \( n + 1 \) verifiers, show that the protocol is insecure: we can have an attack succeeding with probability 1.

**HINT:** use the previous question!

We use \( n \) times the previous attack for \( i = 1, \ldots, n \), at different locations with one prover, one verifier, and one man-in-the-middle in each of these locations. Then, all men-in-the-middle send their \( x_i \) to a malicious participant \( A \) sitting close by a verifier \( V \). Clearly, he can impersonate a honest prover by simulating \( P(x) \), and make a protocol succeed for \( V \) even though there is no prover within a distance up to \( B \).
3 On a Weak Fiat-Shamir Transform

This exercise is inspired from Bernhard-Pereira-Warinschi, How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios, Asiacrypt 2012, LNCS vol. 7658, Springer.

Throughout this exercise, we consider some \((G, q, g)\) depending on a security parameter \(t\), where \(G\) is a group, \(q\) is a prime number, and \(g\) is an element of \(G\) of order \(q\). We assume that \(q > 2^t\), that the size of \(q\) is polynomially bounded, and that we can make basic operations (multiplication, inversion, comparison) in \(G\) in polynomial time.

We consider the Schnorr \(\Sigma\)-protocol for the relation \(R\) defined by

\[ R(y, x) \iff g^x = y \]

In the \(\Sigma\)-protocol, the prover picks \(k \in \mathbb{Z}_q\) and sends \(r = g^k\). The verifier picks \(e \in \{1, \ldots, 2^t\}\) and sends it to the prover. The prover answers by \(s = ex + k \mod q\). The verifier checks that \(ry^e = g^s\). In the weak Fiat-Shamir transform constructs a non-interactive proof system by using a random oracle \(H\) as follows:

\begin{align*}
\text{Proof}(y, x; k) :& \quad \text{compute } r = g^k, \ e = H(r), \ s = ex + k \mod q. \ \text{The output is } (r, s). \\
\text{Verify}(y, (r, s)) :& \quad \text{check that } ry^e = g^s. \ \text{If this passes, the output is accept. Otherwise, the output is reject.}
\end{align*}

We assume that the random oracle \(H\) returns elements of \(\mathbb{Z}_q\) which are uniformly distributed. A proof \((r, s)\) for \(y\) is aimed at producing evidence that the algorithm which forged \((r, s)\) knows \(x\) such that \(g^x = y\).

Q.1 Construct an efficient algorithm \(A^H\) invoking \(H\) and producing a triplet \((y, r, s)\) such that \(y \neq 1\), \(y\) is spanned by \(g\), and \(\text{Verify}(y, (r, s)) = \text{accept}\), with probability larger than \(1 - 2^{-t}\).

We consider an algorithm picking \(r\) and \(s\) at random then calling \(H(r)\), then computing \(y = (r^{-1}g^s)^{\frac{1}{q^t}} \mod q\). Except for \(H(r) = 0\), which occurs with probability lower than \(2^{-t}\), \((r, s)\) is a valid proof for \(y\).

Q.2 In the (strong) Fiat-Shamir construction, the query to \(H\) is \(y||r\) instead of \(r\) alone. In this case, say why the previous attack does not work.

In the previous attack, \(y\) is not determined when we call \(H(r)\). Now, to query \(H\) we must commit to some \(y\). So, the previous attack does not work in the strong Fiat-Shamir construction.

Q.3 We let \(y \neq 1\) spanned by \(g\) be fixed.

Let \(A^H\) be an algorithm invoking \(H\). We consider the following experiment:

1: pick \(\rho\) and \(H\)
2: set \((r, s) = A^H(\rho)\)
3: set \(\text{Out} = \text{Verify}(y, (r, s))\)
The goal of this question is to show that there is a generic transform \( T \) such that for any polynomially bounded algorithm \( A^H \) such that \( \Pr[\text{Out} = \text{accept}] \geq 1 - 2^{-t} \) (over the distribution of \( \rho \) and \( H \)) \( B = T(A) \) is a polynomially bounded algorithm producing the discrete logarithm of \( y \).

Q.3a Let \( E \) be the event that during the computation of \( A \), a query to \( H \) was made with the final value \( r \) of the proof. Show that \( \Pr[E] \geq 1 - 2^{-t} \).

HINT: first show that \( \Pr[\text{Out} = \text{accept} \mid E] \leq 2^{-t} \).

\[ \text{We have } \text{Out} = \text{accept} \iff y^{H(r)} = r^{-1}g^s. \text{ If } E \text{ does not hold, } H(r) \text{ is completely independent from } (r, s). \text{ Since } y \text{ is generated by } g \text{ and is not 1, it has order } q. \text{ So, } \Pr[\text{Out} = \text{accept} \mid \neg E] = \frac{1}{q} \leq 2^{-t}. \]

Then,

\[ \Pr[E] \geq \Pr[\text{Out} = \text{accept}] - \Pr[\text{Out} = \text{accept} \mid \neg E] \geq 1 - 2 \times 2^{-t} \]

Q.3b We consider a simulator for \( A \) and \( H \). The simulation of \( H \) is done following the lazy sampling technique (i.e., fresh random coins are flipped only when needed). The simulation defines a tree of the partial views of the simulator, where each node corresponds to the view when a fresh call to \( H \) is made, and the \( q \) sons of the node correspond to the possible coin flips to respond to the query. A leaf \( \lambda \) corresponds to the end of the execution of \( A \). The event \( \text{Succ}(\lambda) \) holds if \( A \) outputs some \((r, s)\) making the verification accept and \( r \) was queried to \( H \). If \( \text{Succ}(\lambda) \) holds, we let \( \text{dist}(\lambda) \) be the ancestor of \( \lambda \) corresponding to the \( H(r) \) oracle call. Otherwise, we let \( \text{dist}(\lambda) = \emptyset \).

We let \( p \) be the probability that a random descent in the tree ends to a leaf \( \lambda \) such that \( \text{Succ}(\lambda) \) holds. We let \( d \) be the expected length of a random descent. Given a node \( \nu \) in the tree, we let \( Y \) be a random leaf obtained by a random descent starting from \( \nu \). We let \( f(\nu) = \Pr\{\text{Succ}(Y), \text{dist}(Y) = \nu\} \). We let \( X \) be a random leaf obtained by a random descent from the root. We let \( Y \) be a random leaf obtained by a random descent from \( \text{dist}(X) \). The Forking Lemma says that \( E(f(\text{dist}(X))) \geq \frac{p^2}{2d} \).

Show that if \( d \) is polynomially bounded, we can make a polynomial-time algorithm walking in this tree and producing with probability at least \( \frac{p^2}{2d} - (1 - p) - 2^{-t} \) two leaves \( X \) and \( Y \) such that \( \text{Succ}(X) \) and \( \text{Succ}(Y) \) hold, \( \text{dist}(X) = \text{dist}(Y) \), and with \( X \) and \( Y \) in different subtrees connected to \( \text{dist}(X) = \text{dist}(Y) \).
Q.4 The previous reduction works for attacks \(\mathcal{A}\) in which \(y\) is determined at the beginning. Assuming that now \(y\) is not determined and we consider an attack producing valid \((y, r, s)\) triplets. Assume that for each such attack \(\mathcal{A}\), there exists an algorithm \(\mathcal{B}\) such that for each View, if \(\mathcal{A}(\text{View}) = (y, r, s)\) such that \(\text{Verify}(y, (r, s))\) accepts, \(\mathcal{B}(\text{View}) = x\) such that \(y = g^x\).

Show that we can solve the discrete logarithm problem: we can construct a polynomial-time algorithm \(\mathcal{C}\) such that given \(z\) as input, it outputs \(\mathcal{C}(z)\) such that \(g^{\mathcal{C}(z)} = z\).

HINT: Construct some \(\mathcal{A}\) like in Q.1 but with \(r = z\).

---

Q.3c Show that by using \(\mathcal{A}^H\) as a subroutine we can make a polynomial-time algorithm \(\mathcal{B}\) which outputs \(x\) such that \(g^x = y\) with a probability which is not negligible.

---

We let \(\mathcal{B}\) use the simulator in the previous question and produce \(X\) and \(Y\) in polynomial time, with a probability which is not negligible. Let \((r, s)\) be the output of \(\mathcal{A}\) in the first descent \(X\) and \((r', s')\) be the output of \(\mathcal{A}\) in the second one \(Y\). Since both have the same distinguished ancestor, we have that \(r = r'\).

By construction, both output are accepted, so \(r y^H(r) = g^s\) and \(r y^{H'(r)} = g^{s'}\).

We let \(H\) be the oracle function in the first descent and \(H'\) be the oracle function in the second one. By construction, \(H(r) \neq H'(r)\).

Therefore, \(x = \frac{s-s'}{H(r)-H'(r)} \mod q\) is such that \(g^x = y\). This is the final output of \(\mathcal{B}\).

---

Let \(z\) be a value for which we want to compute the discrete logarithm. We construct an algorithm \(\mathcal{A}\) taking \(r = z\), picking \(s\), calling \(H(r)\), and \(y = (r^{-1} g^s)^{\frac{1}{H(r)}}\) to output \((y, r, s)\). The view of \(\mathcal{A}\) is \((z, H(z); s)\).

By hypothesis, there is an algorithm \(\mathcal{B}\) such that \(\mathcal{B}(z, H(z); s) = x\) such that \(y = g^x\). Since \(zy^{H(z)} = g^s\), we deduce \(z = g^{s-xH(z)}\). So, we can compute \(s - xH(z)\) which is the discrete logarithm of \(z\).