6.875: Solutions for Homework 5

Yevgeniy Dodis

Problem 1: from PRG to PRF

Recall the "tree" construction of a pseudo-random function from a pseudo-random generator. Let $G : \{0,1\}^k \mapsto \{0,1\}^{2k}$ be a length-doubling PRG. Denote by $G_0(s)$ the first half of $G(s)$, by $G_1(s)$ – the second half. In other words, $G(s) = G_1(s) \circ G_2(s)$, where $|G_1(s)| = |G_2(s)| = k$ and $\circ$ is the concatenation operator. For any string $x = (x_1, \dots x_k)$ we let $G_x(s) = G_{x_k}(\dots G_{x_2}(G_{x_1}(s))\dots)$ and $f_s(x) = G_x(s)$. We showed that the family $\mathcal{F}_k = \{f_s : \{0,1\}^k \mapsto \{0,1\}^k \mid s \in \{0,1\}^k\}$ is a pseudo-random function family. In other words, a function is chosen by randomly filling the root of the tree and, given x, we follow the path specified by x to find $f_s(x)$.

What about the family $\mathcal{G}_k = \{G_s : \{0,1\}^k \mapsto \{0,1\}^k \mid s \in \{0,1\}^k\}$? In other words, a function is chosen by choosing a random path down the tree and, given x, this path is followed with x placed at the root. Either prove that the construction works or provide a counter-example.

Solution: No, $\mathcal{G}_k$ is not necessarily a family of PRFs. Let G be any length-doubling PRG. Define $G'(0^k) = 0^{2k}$ and $G'(x) = G(x)$ for all other $x \neq 0^k$. I claim that G' is also a length-doubling PRG. Indeed, since the seed s to the generator is chosen at random, G' behaves differently from G with probability $1/2^k$, so any distinguisher for G' has at most $1/2^k = negl(k)$ more advantage in a statistical test than it has for G. Thus, since it has negligible advantage for G, the same holds for G'. Now, however, $G'_s(0^k) = 0^k$ for any $s \in \{0,1\}^k$ (this follows by iterating the relation $G'_0(0^k) = G'_1(0^k) = 0^k$). And since a truly random function f has only $1/2^k$ chance of having $f(0^k) = 0^k$, $\mathcal{G}'_k$ is not a PRF family (a statistical test that, given an oracle access to f, accepts iff $f(0^k) = 0^k$, distinguishes a truly random function from a random G'_s, $s \in \{0,1\}^k$, with advantage $1 - \frac{1}{2^k}$).

Problem 2: PRF Candidates

Suppose that $\{F_S : \{0,1\}^k \mapsto \{0,1\}^k \mid s \in \{0,1\}^k\}$ is a pseudo-random family of functions from k-bit input to k-bit output, indexed by a k-bit key ("seed"). We would like to get a new pseudo-random function family in which each function maps k bits to $2k$ bits. Consider the following construction, and for each show whether it is good or bad (namely whether the specified family is pseudo-random or not). Below, "$\circ$" denotes concatenation, and $\bar{x}$ denotes the bitwise negation of x. In all the statistical tests below, let the distinguisher be given access to a function f.

1. $F^1_S(x) = F_S(0^k) \circ F_S(x)$.
   This is bad since $F^1_S(x)$ has the same first k bits for every $x \in \{0,1\}^k$. A distinguishing test that checks if $f(0^k)$ and $f(1^k)$ have the same first k bits, distinguishes with probability $1 - \frac{1}{2^k}$. Alternatively, $F^1_S(0^k) = F_S(0^k) \circ F_S(0^k)$ – a concatenation of 2 identical strings, which happens with probability $1/2^k$ for a random f.

2. $F^2_S(x) = F_S(x) \circ F_S(\bar{x})$.
   This is bad since $F^2_S(\bar{x}) = F_S(\bar{x}) \circ F_S(x)$ can be obtained from $F^2_S(x)$ by flipping the first and the

last k bits, i.e. $F_S^2(\bar{x})$ is determined from $F_S^2(x)$. For a random function, for any x, the probability that this happens is $1/2^{2k}$. Thus, advantage is $1 - \frac{1}{2^{2k}}$.

3. $F_S^3(x) = F_{0^k}(x) \circ F_S(x)$.
   This is bad since a distinguisher can compute $F_{0^k}(x)$ by himself (i.e. it is independent of S). So by testing that $f(0^k)$ starts with $F_{0^k}(0^k)$, we get advantage $1 - \frac{1}{2^k}$.

4. $F_S^4(x) = F_{S_1}(x) \circ F_{S_2}(x)$, where $S_1 = F_S(0^k)$ and $S_2 = F_S(1^k)$.
   This construction is good and the proof is given below.

5. $F_S^5(x) = F_x(S) \circ F_S(x)$. This *might be* bad as we saw in problem 1. Indeed, if F_S is constructed from a generator G' described in problem 1 by the standard tree construction, then we know that F_S is indeed a PRF family. However, we saw that $F_{0^k}(S) = 0^k$ for all S, so that $F_S^5(0^k)$ starts with 0^k, giving the advantage of $1 - \frac{1}{2^k}$.

Proof for $F_S^4(x)$: Let us denote by $R_1, R_2, R = R_1 \circ R_2$ truly random functions from k to k, k and $2k$ bits, respectively; by $g_1, g_2, g = g_1 \circ g_2$ – functions F_{S_1}, F_{S_2}, $F_{S_1} \circ F_{S_2}$, where S_1 and S_2 are chosen independently and uniformly at random from $\{0,1\}^k$; by $h_1, h_2, h = h_1 \circ h_2$ – functions F_{S_1}, F_{S_2}, F_S^4, where $S_1 = F_S(0^k)$, $S_2 = F_S(1^k)$ and S is chosen at random from $\{0,1\}^k$. Given an algorithm C with a black-box access to a function f chosen according to some distribution, denote by $C(f)$ the random variable denoting the value of the output of C when f is chosen according to this distribution. Say that $f_1 \approx f_2$ if they are distributed according to the same distribution.

Now, we are ready to prove that $\{F_S^4\}$ is a PRF family. Assume not, i.e. there is a distinguisher $A(\cdot)$ for $\{F_S^4\}$ s.t. for infinitely many k,

$$| \ \Pr(A(h) = 1) - \Pr(A(R) = 1) \ | > \frac{1}{poly(k)} \tag{1}$$

Let us obtain a contradiction by constructing a distinguisher B for $\{F_S\}$. Given a black-box access to a function $f : \{0,1\}^k \mapsto \{0,1\}^k$, define $f_1 = F_{f(0^k)} \circ F_{f(1^k)}$, $f_2 = g_1 \circ f$ (where g_1 is simulated by B by choosing random S_1) and $f_3 = f \circ R_2$ (where R_2 is a random function simulated by B). B picks a random $i \in \{1,2,3\}$ and outputs $A(f_i)$.

We note that when $f = F_S$ for a random S, then: $f_1 \approx h$, $f_2 \approx g$ and $f_3 \approx g_1 \circ R_2$; if f is a random function, then $f_1 \approx g$, $f_2 \approx g_1 \circ R_2$ and $f_3 \approx R$. Hence, the advantage of B can be written as:

$$\begin{aligned} p \quad &= \quad \frac{1}{3} \cdot | \ (\Pr(A(h) = 1) - \Pr(A(g) = 1)) + (\Pr(A(g) = 1) - \Pr(A(g_1 \circ R_2) = 1)) + \\ & \quad\quad (\Pr(A(g_1 \circ R_2) = 1) - \Pr(A(R) = 1)) \ | = \frac{1}{3} \cdot | \ \Pr(A(h) = 1) - \Pr(A(R) = 1) \ | \\ &\overset{(1)}{\geq} \quad \frac{1}{3 \cdot poly(k)} \end{aligned}$$

■

Problem 3: "Coin Flipping in the Well"

Write a *formal* definition of a protocol that cryptographically implements coin flipping in the well.

Solution: Instead of directly giving the final definition (see Definition 1 below), let us try to slowly arrive at it by trying to reason logically. You are free to look at the final result right away, ignoring the next page or so. I give the details just so you see how to design the right definition.

First of all, we do not want too specific of a definition. We simply want to say that any general protocol P satisfying certain properties is a good implementation of "coin flipping in the well" (shortly, cfw). Before thinking about the definition I want to tell about the following set of rules that greatly simplify (almost make it axiomatic) the design of a robust definition of what it means to have a two-party (this also generalizes to many parties) protocol implementing certain task T (e.g. cfw). We split our task into 2 subtasks:

- *What do we want to achieve if everyone follows the protocol correctly?* This part of the definition is typically called **correctness** since it specifies what happens when everything is as planned. Here we do not worry about any misbehaviors. The correctness for two-party case typically goes like this: "there *exist* (honest) parties A and B such that if they follow the protocol, the following conditions hold: ...".
- *What are the things we do not want a possibly malicious party to be able to do?* Here we say that even when one of the parties is malicious (while the other is honest as specified in the correctness part), this party can not achieve certain tasks. We do not worry what happens when both parties misbehave since in this case there is nothing we can do. We also do not worry about what good thing we want to happen – correctness takes care of this. This part of the definition is typically called **security**, since it protects an honest party from a (possibly) dishonest one. For two-parties, this is usually split into 2 parts: what is honest B protected against dishonest $\tilde{A}$ and the same for honest A against dishonest $\tilde{B}$. A typical condition reads like this: "for any $\tilde{A}$, $\tilde{A}$ can not do this and that to honest B".

Now, let us first informally apply this to our problem. So what do we want to happen if everyone is honest? The first thing to notice is that our protocol should have 2 stages: generation stage at the end of which Alice learns the bit, and revealing stage when Alice reveals the correct bit to Bob. Correctness simply says that honest Alice (A) is able to construct some bit c at the end of stage 1 (generation), and that in stage 2 she is able to convince honest Bob (B) that the bit generated was c (with probability 1, say). That's it for correctness.

The trickier part is security. So assume we have dishonest Alice ($\tilde{A}$) talking to honest Bob (B). What are the bad things $\tilde{A}$ can do? Seems like we need a few things. At the end of stage 1 $\tilde{A}$ should "generate a *random* bit c". But a closer look will reveal that this statement *does not make sense.* $\tilde{A}$ can do whatever she wants at the end of stage 1 since she is not yet proving anything to B. We only know that *honest* A would actually (try to) generate something, $\tilde{A}$ might not have any bit c in mind. The only thing we can protect B against is what $\tilde{A}$ should *not* be able to prove to B in stage 2. Let the output of B at the end of stage 2 be $c' \in \{0, 1, fail\}$. $c' = fail$ means that B caught $\tilde{A}$ cheating and decided to reject, which is reasonable to happen when $\tilde{A}$ is dishonest.

A naive attempt would be to say that we want both probabilities that $c' = 0$ and $c' = 1$ to be very close to $\frac{1}{2}$. However, this is clearly unachievable. $\tilde{A}$ can just refuse to talk, so that $c' = fail$ all the time. A second attempt would be to say that $\Pr[c' = 0] \approx \Pr[c' = 1]$, so that if B is happy ($c' \neq fail$) c' is roughly random. However, this is again unreasonable, since here is a possible behavior of $\tilde{A}$. Follow honest A until the end of stage 1 and generate the correct bit c. If $c = 1$, honestly convince B that $c = 1$ (so that $c' = 1$). If $c = 0$, shut up (so that $c' = fail$). Here $\Pr[c' = 1] = \Pr[c' = fail] = \frac{1}{2}$. This tells us that the only thing that makes sense is to say that $\tilde{A}$ cannot bias towards some particular value, i.e. $\Pr[c' = 0] < \frac{1}{2} + negl(k)$ and $\Pr[c' = 1] < \frac{1}{2} + negl(k)$. This exactly captures the fact that $\tilde{A}$ can not bias the outcome too much. And another look tells us that this is the only thing we need to (and can) protect Bob against.

What about the honest A and dishonest $\tilde{B}$? Clearly, we need to protect A only at the end of the generation stage since after that A is not learning anything new. As before, assume A outputs $c' \in$

$\{0, 1, fail\}$ at the end of stage 1. As before, it is too much to expect that $\Pr[c' = b] \approx \frac{1}{2}$, where $b \in \{0, 1\}$, since $\tilde{B}$ can shut up immediately. However, now it makes perfect sense to say that if A is happy ($c' \neq fail$), it better be the case that c' is a "random" bit. The difference between this and the previous case with $\tilde{A}$ is that $\tilde{A}$ convinces B after she (might have) learned the correct c, while $\tilde{B}$ is not supposed to learn c, so we may as well require that c' is equally likely to be 0 and 1. A technical way to say it to condition on c' being a bit and require that $|\Pr[c' = 0 \mid c' \in \{0, 1\}] - \frac{1}{2}| < negl(k)$ (this implies that $|\Pr[c' = 1 \mid c' \in \{0, 1\}] - \frac{1}{2}| < negl(k)$ as well).

We remark that all the probabilities above are taken over random coins of both the place (where the important part are the coin tosses of the *honest* player).

You might ask whether we needed some more properties from correctness, e.g. that honest A generates c that is 0 with probability close to 1/2. But we do not need it since secrecy properties hold against *any* $\tilde{A}$ and $\tilde{B}$, in particular, for $\tilde{A} = A$ and $\tilde{B} = B$. This is another part where one would avoid redundancy: do not include the conditions that follow from security into correctness. Now we can put all the pieces together and we get the *formal* definition:

Definition 1 *Let k be a security parameter. A two-stage protocol between A and B implements* coin flipping in the well *if (all probabilities are over the random coins of both players. Hence c and c' below are random variables):*

- **Correctness:**
  - *At the end of stage 1, A generates a bit c.*
  - *At the end of stage 2 A can convince B with probability* 1 *that the bit she generated was c.*
- **Security:**
  - *for any $\tilde{A}$, if B outputs $c' \in \{0, 1, fail\}$ at the end of stage 2, then for $b = 0, 1$:*

$$\Pr[\ c' = b\ ] < \frac{1}{2} + negl(k)$$

  - *for any $\tilde{B}$, if A outputs $c' \in \{0, 1, fail\}$ at the end of stage 1, then*

$$\left| \Pr[\ c' = 0 \mid c' \neq fail\ ] - \frac{1}{2} \right| < negl(k)$$

Problem 4: Good or Bad?

For each of the following protocols and the corresponding hardness assumption, EITHER *prove* that the protocol implements coin flipping in the well (according to your formal definition and based on the specified assumption), OR show that it does not, suggest a simple fix and *prove* that the fix works.

1. Assumption: Factoring Blum integers is hard. Protocol:
   Stage 1:
   - Bob: Choose random k-bit primes p_1, p_2 s.t. $p_1 \equiv p_2 \equiv 3 \bmod 4$. Let $n = p_1 p_2$. Send n to Alice.
   - Alice: Choose random $x \in \mathbb{Z}_n^*$. Let $y = x^2 \bmod n$. Send y to Bob.
   - Bob: Pick random $b \in \{0, 1\}$. Send b to Alice.
   - Alice: Compute $c = b \oplus \left(\frac{x}{n}\right)$ (where $\left(\frac{x}{n}\right)$ is the Jacobi symbol of x modulo n).

Stage 2:

- Alice: Reveal x to Bob.
- Bob: Check that $y = x^2 \bmod n$. Compute $c = b \oplus \left(\frac{x}{n}\right)$.

Solution: Let us first see if the protocol satisfies the definition and if not, where it fails. The correctness is trivial, since both b and $\left(\frac{x}{n}\right)$ are (supposed to be) random. Let's check for dishonest $\tilde{A}$ first. Assume $\tilde{A}$ convinces B in the bit c' which is biased away from 1/2 towards a fixed bit $e \in \{0, 1\}$ (i.e. $\Pr[c' = e] > 1/2 + 1/poly(k)$). Then we have: $\tilde{A}$ first produces some value y, then learns a *random* b, and then with probability much better than 1/2 produces the value x in stage 2 such that $x^2 = y$ and $\left(\frac{x}{n}\right) = b \oplus e$. Since $b \oplus e$ is a random bit and we can find x s.t. $\left(\frac{x}{n}\right)$ equal to this random bit with probability better than 1/2, it must be the case that with non-negligible probability $\tilde{A}$ can find x_1, x_2 s.t. $x_1^2 = x_2^2 = y$ while $\left(\frac{x_1}{n}\right) \neq \left(\frac{x_2}{n}\right)$. Since n is a Blum integer, $\left(\frac{-1}{n}\right) = 1$, so $x_1 \neq \pm x_2$. Therefore, $gcd(x_1 - x_2, n)$ is a non-trivial factor of n, so using $\tilde{A}$ we can factor a random Blum integer with non-negligible probability, a contradiction to our assumption.

Now, let us check against dishonest $\tilde{B}$. Unfortunately, it is possible to cheat, since we do not ask Bob to prove that n is a Blum integer. So here is a way for $\tilde{B}$ to cheat against A. Instead of picking $p_1 \equiv p_2 \equiv 3 \bmod 4$, make $p_1 \equiv p_2 \equiv 1 \bmod 4$ and $n = p_1 p_2$. Now it is easy to see that for any quadratic residue y, *all 4 square roots of y have the same Jacobi symbol!* (since $\left(\frac{-1}{p_1}\right) = \left(\frac{-1}{p_2}\right) = 1$). Therefore, when $\tilde{B}$ learns y, he can find any square root x' of y and *know* that $\left(\frac{x}{n}\right) = \left(\frac{x'}{n}\right)$, so he determines $\left(\frac{x}{n}\right)$. Now he can force c to be 0, for example, by setting $b = \left(\frac{x}{n}\right)$, contradicting privacy.

An elementary fix to that would be some kind of a proof by Bob that the n he gives is indeed a Blum integer. In this case, each quadratic residue has half (2) of its roots with Jacobi symbol 1. Hence, since the square root x is random, $\tilde{B}$ is unable to tell $\left(\frac{x}{n}\right)$ with probability better than 1/2, so he can not bias c from 1/2 *at all* (otherwise, he is able to predict $\left(\frac{x}{n}\right)$ with probability better than 1/2). So Alice needs to insist that Bob gives her a "zero-knowledge" proof that n is a Blum integer. It has to be zero-knowledge since we can not afford Alice to know the factorization of n, since otherwise she can proof both $c = 0$ and $c = 1$ by taking the appropriate square root. So Bob should convince Alice that n is a Blum integer *without giving away the factorization of n.* This is indeed possible to do using the heavy machinery of zero-knowledge proof systems that we will touch upon later in the course. However, I know no a *simple* way to do it. Thus, at least on the surface, this cfw protocol is not very good!

This example shows that a seemingly great protocol might be totally insecure and justifies careful examination of protocols.

2. Assumption: Quadratic Residuosity is hard. Protocol:
   Stage 1:

   - Alice: Choose random k-bit primes p_1, p_2, set $n = p_1 p_2$. Choose random $x \in \mathbb{Z}_n^*$ with Jacobi symbol 1, $\left(\frac{x}{n}\right) = 1$. Send n and x to Bob.
   - Bob: Pick random $b \in \{0, 1\}$. Send b to Alice.
   - Alice: Compute $c = b \oplus \chi(x)$ (where $\chi(x)$ is the quadratic character of x modulo n).

   Stage 2:

   - Alice: Reveal p_1, p_2 to Bob.

- Bob: Check that $n = p_1p_2$. Let $\chi(x) = 1$ iff $\left(\frac{x}{p_1}\right) = \left(\frac{x}{p_2}\right) = 1$. Compute $c = b \oplus \chi(x)$.

Solution: Again, the correctness is trivial. Let us now start with dishonest $\tilde{B}$. The only way $\tilde{B}$ can bias $c = b \oplus \chi(x)$ is to learn $\chi(x)$ with probability non-trivially better than 1/2, only based on the value of x. This is exactly the quadratic residuosity assumption, so we are indeed secure against any dishonest Bob.

Now, let us take dishonest $\tilde{A}$. Unfortunately, it is possible to cheat because B does not check the primality of p_1 and p_2 in the last step. $\tilde{A}$ chooses primes q_1, q_2, q_3, q_4 and lets $n = q_1q_2q_3q_4$. $\tilde{A}$ also chooses $x \in \mathbb{Z}_n^*$ such that $\left(\frac{x}{q_1}\right) = \left(\frac{x}{q_2}\right) = 1$, $\left(\frac{x}{q_3}\right) = \left(\frac{x}{q_4}\right) = -1$ (one of many an easy ways to do it is to let $q_1 \equiv q_2 \equiv 1 \bmod 4$, $q_3 \equiv q_4 \equiv 3 \bmod 4$ and $x = -1 \bmod n$). Then $\tilde{A}$ can prove at will both $c = 0$ and $c = 1$ to B (thus, she can always prove $c = 0$, for example, violating Definition 1). Let's assume that $b = 0$ (symmetric trick works for $b = 1$). If she wishes to prove $c = 1$, she pretends $p_1 = q_1q_2$, $p_2 = q_3q_4$. Then $\left(\frac{x}{p_1}\right) = \left(\frac{x}{q_1}\right)\left(\frac{x}{q_2}\right) = 1 \cdot 1 = 1$, $\left(\frac{x}{p_2}\right) = \left(\frac{x}{q_3}\right)\left(\frac{x}{q_4}\right) = (-1) \cdot (-1) = 1$, so that B sets $\chi(x) = 1$, and $c = \chi(x) = 1$. If $\tilde{A}$ wishes to prove $c = 0$ instead, she pretends $p_1 = q_1q_3$, $p_2 = q_2q_4$. Then $\left(\frac{x}{p_1}\right) = \left(\frac{x}{q_1}\right)\left(\frac{x}{q_3}\right) = 1 \cdot (-1) = -1$, $\left(\frac{x}{p_2}\right) = \left(\frac{x}{q_2}\right)\left(\frac{x}{q_4}\right) = 1 \cdot (-1) = -1$, so that B sets $\chi(x) = 0$, and $c = \chi(x) = 0$.

A trivial fix is for B to check the primality of p_1 and p_2 in the last step. We claim that then we correctly implement cfw. We don't need to redo the proof against dishonest $\tilde{B}$ since we only modified stage 2. And now $\tilde{A}$ can not cheat at all, since p_1, p_2 and $\chi(x)$ are deterministic functions of x and n which B computes correctly at stage 2 (since he checks the primality of p_1 and p_2 now). More precisely, let $c' \in \{0, 1, fail\}$ be the output of B. Since B chooses b at *random* and *after* $\tilde{A}$ is committed to n and x (and thus to $\chi(x)$), we have that $c = b \oplus \chi(x)$ is a random bit and if $c' \neq fail$, then $c' = c$. Hence, $\Pr[c' = 0] \leq \Pr[c = 0] = \frac{1}{2}$ and $\Pr[c' = 1] \leq \Pr[c = 1] = \frac{1}{2}$, completing the proof of privacy.

3. Assumption: Discrete Log Problem is hard. Protocol:
   Stage 1:

   - Bob: Choose random k-bit prime p, a generator g of $\mathbb{Z}_p^*$ and a random $R \in \mathbb{Z}_p^*$ s.t. $R \neq 1$. Send p, g, R to Alice.
   - Alice: Check that p is prime. Choose random $a \in \{0, 1\}$ and random $z \in \mathbb{Z}_p^*$. If $a = 0$, let $y = g^z \bmod p$, else $y = g^z R \bmod p$. Send y to Bob.
   - Bob: Pick random $b \in \{0, 1\}$. Send b to Alice.
   - Alice: Compute $c = a \oplus b$.

   Stage 2:

   - Alice: Reveal z to Bob.
   - Bob: Check that either $y = g^z \bmod p$ (then set $a = 0$) or $y = g^z R \bmod p$ (then set $a = 1$). Compute $c = a \oplus b$.

Solution: As usual, correctness is clear. Let's check dishonest $\tilde{A}$ first. Assume $\tilde{A}$ convinces B in the bit c' which is biased away from 1/2 toward a fixed bit $e \in \{0, 1\}$ (i.e. $\Pr[c' = e] > 1/2 + 1/poly(k)$). Then we have: $\tilde{A}$ first produces some value y, then learns a *random* b, and then with probability much better than 1/2 produces the value z in stage 2 such that $y = g^z$ if $b \oplus e = 0$ and $y = g^z R$ if

$b \oplus e = 1$, i.e. we either find a discrete log of y, or of y/R. Since $b \oplus e$ is a random bit, it must be the case that with non-negligible probability $\tilde{A}$ can find both discrete logs, i.e. z_1, z_2 s.t. $g^{z_1} = g^{z_2}R = y$. But then $R = g^{(z_1 - z_2) \bmod (p-1)}$, so that we can use $\tilde{A}$ to find the discrete log of a random $1 \neq R \in Z_p^*$ with non-negligible probability, contradicting the discrete log assumption.

Now, let's check against dishonest $\tilde{B}$. And again we can see that $\tilde{B}$ can cheat. The point is that $\tilde{B}$ does not prove that g is a generator of $\mathbb{Z}_p^*$. So here is what $\tilde{B}$ can do. Generate a random prime p s.t. you know the factorization of $p-1$. We mentioned in class that we can do it (result of Eric Bach). Let $q_1, \ldots, q_r$ be all distinct prime factors of $p-1$. The following is the useful theorem (that we covered before):

Theorem 1 *$g \in \mathbb{Z}_p^*$ is a generator of $\mathbb{Z}_p^*$ if and only if $g^{p-1/q_i} \neq 1 \bmod p$, for all $i = 1 \ldots r$.*

Using this theorem and the fact that there are plenty of generators for $\mathbb{Z}_p^*$ (exactly $\varphi(p-1) \approx p/\operatorname{loglog} p$), $\tilde{B}$ finds a generator h of $\mathbb{Z}_p^*$ and sets $g = h^{p-1/q_1}$ (so that $g \neq 1$ and $g^{q_1} = 1$) and $R = h$. He gives A these p, g and $R = h$. A gives back y. $\tilde{B}$ computes $w = y^{q_1}$. If $a = 0$, then $y = g^z$, so that $w = y^{q_1} = (g^z)^{q_1} = (g^{q_1})^z = 1$. If $a = 1$, then $y = g^z h$, so that $w = y^{q_1} = (g^z h)^{q_1} = h^{q_1} \neq 1$ (since h is a generator). Therefore, $\tilde{B}$ will learn a with probability 1, violating privacy.

What is the easy fix? Since the discrete log does not have a trapdoor, we do not have a problem of the first protocol (where we could not give Alice the factorization of the Blum integer). So we can ask Bob to augment his message containing p, q, R with: proof that p is prime (not necessary since A does it anyway, but does not hurt to reduce A's complexity) and the proof that g is a generator. By the theorem above, this proof will just consist of the factorization of $p-1$ (by Bach's result, Bob can generate it together with p), so that Alice can check that $g^{p-1/q_i} \neq 1$, $i = 1 \ldots r$. Now I claim that the protocol becomes secure.

First, let's see that $\tilde{B}$ can not cheat now. Since g is a generator now, both distributions of g^z and Rg^z are uniform over $\mathbb{Z}_p^*$, so that $\tilde{B}$ gets no information about a upon receiving y. Thus, he can not cheat by sending a bad b (i.e. b is independent of a, so $a \oplus b$ is a random bit). However, to complete the proof we have to redo the security against $\tilde{A}$, since we changed the protocol by giving more information to Alice. But in this case it is easy to see that nothing changes because the original *proof*. The only thing that does change is the *underlying hardness assumption*. While before we just needed the hardness of finding discrete logs given a prime p, now we need hardness of finding discrete logs when both p and the factorization of $p-1$ are known. However, this is the standard assumption, so we are indeed fine.