University of Maryland
CMSC858K — Introduction to Cryptography
Professor Jonathan Katz

---

# Problem Set 4

1. (Here, let $p$ be an arbitrary prime.) If $h$ is a quadratic residue modulo $p$, then $h = g^2 \bmod p$ for some $g \in \mathbb{Z}_p^*$. Then

$$h^{(p-1)/2} = \left(g^2\right)^{(p-1)/2} = g^{p-1} = 1 \bmod p.$$

For the other direction, let $g$ be a generator of $\mathbb{Z}_p^*$ and say $h^{(p-1)/2} = 1 \bmod p$. We know that $h = g^x \bmod p$ for some $x \in \mathbb{Z}_{p-1}$, and so $g^{x(p-1)/2} = 1 \bmod p$. This implies that $x(p-1)/2 = 0 \bmod (p-1)$. This cannot be satisfied for any odd value of $x$, and so we conclude that $x$ must be even. But then

$$h = g^x = \left(g^{x/2}\right)^2,$$

and $h$ is a quadratic residue.

The previous problem shows that given $h$ and $p$ it is possible to determine in polynomial time whether $h$ is a quadratic residue or not. But if $h = g^x \bmod p$ then $h$ is a quadratic residue iff $x$ is even, which is true iff the least significant bit of $x$ is 1.

(b) Given $(g, y_1, y_2, y_3)$, where $g$ is a generator, do the following: let $b_1, b_2, b_3$ equal 1 if $y_1$, $y_2$, and $y_3$, respectively, are quadratic residues (and 0 otherwise). Then if $b_1 \cdot b_2 = b_3$ output 1, and output 0 otherwise.

If $(g, y_1, y_2, y_3)$ is a Diffie-Hellman tuple (i.e., $y_1 = g^x$ and $y_2 = g^y$ for some $x, y$, and $y_3 = g^{xy}$), then $b_1 \cdot b_2 = b_3$ always holds. (To see this, note that $xy \bmod (p-1)$ is even iff at least one of $x$ or $y$ is even.) So given a Diffie-Hellman tuple the above algorithm always outputs 1.

On the other hand, if $(g, y_1, y_2, y_3)$ is a random tuple (i.e., $y_1 = g^x$ and $y_2 = g^y$ and $y + 2 = g^z$ for random $x, y, z$), then the probability that $b_3 = b_1 \cdot b_2$ is exactly 1/2.

Taken together, this means we have a polynomial-time algorithm that distinguishes with non-negligible probability $1 - \frac{1}{2} = \frac{1}{2}$.

(c) If the decisional Diffie-Hellman assumption holds in $\mathbb{G}$ then the computational Diffie-Hellman (CDH) assumption holds in this group also. We show now that if the computational Diffie-Hellman assumption does *not* hold in $\mathbb{Z}_p^*$ then it does not hold in $\mathbb{G}$.

Let $A$ be a polynomial-time algorithm solving the CDH problem in $\mathbb{Z}_p^*$ with probability $\delta(n)$, where this probability is taken over *randomly-chosen* generator $g$ and $y_1, y_2 \in \mathbb{Z}_p^*$, (i.e., $A(g, y_1, y_2)$ outputs $\mathsf{CDH}_g(y_1, y_2)$ with probability $\delta(k)$ over random choice of $g, y_1, y_2$). Note that to use $A$ effectively in a reduction we must provide it with inputs chosen according to the same distribution.

When $p = 2q + 1$ and $q$ is odd, then $-1 \in \mathbb{Z}_p^*$ is not a quadratic residue (and so no in $\mathbb{G}$). So if $g$ is a generator of $\mathbb{G}$ then we can decompose $\mathbb{Z}_p^*$ as

$$\mathbb{Z}_p^* \cong \mathbb{G} \times \mathbb{Z}_2 \cong \langle g \rangle \times \langle -1 \rangle.$$

The key observation is that a random element of $\mathbb{G}$ times a random element of $\langle -1 \rangle$ gives a random element of $\mathbb{Z}_p^*$; furthermore, a random generator of $\mathbb{G}$ times $-1$ gives a random generator of $\mathbb{Z}_p^*$.

Using this observation, construct the following algorithm $A'$ that takes $g, y_1, y_2 \in \mathbb{G}$ as input: pick random $b_1, b_2 \in \{0,1\}$ and set

$$\begin{aligned}\hat{g} &= g \cdot (-1) \bmod p \\ \hat{y}_1 &= y_1 \cdot (-1)^{b_1} \bmod p \\ \hat{y}_2 &= y_2 \cdot (-1)^{b_2} \bmod p.\end{aligned}$$

Then run $A(\hat{g}, \hat{y}_1, \hat{y}_2)$ to obtain output $\hat{h}$. Output $h = \hat{h} \cdot (-1)^{b_1 b_2} \bmod p$. We claim that $A'$ succeeds with probability exactly $\delta(n)$. This follows from the facts that: (1) When the inputs to $A'$ are a random generator $g \in \mathbb{G}$ and random $y_1, y_2 \in \mathbb{G}$, then the inputs to $A$ are a random generator $\hat{g} \in \mathbb{Z}_p^*$ and random $\hat{y}_1, \hat{y}_2 \in \mathbb{Z}_p^*$; and (2) whenever $A$ succeeds, so does $A'$. Filling in the details is left to the reader.

2. (a) The definition I was looking for was the following: the adversary $A$ outputs two equal-length messages $m_0, m_1$; then a random bit $b$ is chosen; the parties run the protocol to send the message $m_b$; the adversary is given the entire transcript and outputs $b'$. The adversary succeeds if $b' = b$ and security requires that for all PPT $A$ the probability of success is at most $\frac{1}{2} + \mathsf{negl}(n)$.

   (b) Fix an adversary $A$ attacking the interactive encryption scheme in the above sense. Say the success probability of $A$ is $\delta(n)$. Construct the following adversary $A'$ attacking the key exchange protocol: $A'$ is given as input a transcript $\mathsf{trans}$ of an execution of the key-exchange protocol, along with a key $k$ that is either the key corresponding to the given transcript or a random key. It runs $A$ to obtain $m_0, m_1$, chooses a random bit $b$ on its own, gives $\mathsf{trans}$ and $\mathsf{Enc}_k(m_b)$ to $A$, and obtains $A$'s output $b'$. If $b = b'$, then $A'$ outputs 1; it outputs 0 otherwise.

   When $k$ is the actual key then the simulation provided for $A$ is perfect, and so $A'$ outputs 1 in this case with probability exactly $\delta(n)$. On the other hand, by security of the key exchange protocol we know that the probability that $A'$ outputs 1 when given a random key is at least $\delta(n) - \mathsf{negl}(n)$. I.e., even when $k$ is random (and uncorrelated with $\mathsf{trans}$), $A$ guesses the bit $b$ correctly with at least this probability.

   Now consider the following adversary $A''$ attacking the encryption scheme. $A''$ runs $A$ to obtain messages $m_0, m_1$. It outputs these messages, and is given in return a ciphertext $c$. Then it runs, on its own, an execution of the key-exchange protocol to obtain a transcript $\mathsf{trans}$ (it ignores whatever key is computed); it then gives $\mathsf{trans}$ and $c$ to $A$. Finally, $A''$ outputs whatever is output by $A$.

   In the above, the key used to generate ciphertext $c$ is chosen at random independent of $\mathsf{trans}$. But as observed earlier, we know that $A$ succeeds with probability at least $\delta(n) - \mathsf{negl}(n)$. But security of the encryption scheme implies that this is at most $\frac{1}{2} + \mathsf{negl}(n)$. Putting everything together shows that $\delta(n)$ is at most $\frac{1}{2} + \mathsf{negl}(n)$.

3. I give the construction, but the proof that it is secure is left to the reader. The observation is that $y^r$ already "looks random" so we may use it directly as an encryption key instead of first using it to "blind" another random value $k$. This means encryption is done as:

$$\langle g^r, \mathsf{Enc}_{y^r}(m) \rangle.$$

(Technically speaking, we need to worry about the fact that $y^r$ is a group element and not a random string. This can be dealt with as discussed in class with regard to Diffie-Hellman key exchange.)

Another way to think of the above is that it is just a non-interactive scheme that results from "collapsing" the interactive encryption protocol of the previous problem (assuming Diffie-Hellman is used for key exchange).