1 Introduction

In this lecture we start discussing the bit security of one way functions. We begin with
eponentiation modulo a prime $p$.

2 Bit Security Of The Discrete Logarithm Function

As we did previously, let $I = \{(p, g) : p$ is prime and $g$ is a generator of $\mathbb{Z}_p^*$\}$ and define

$$\text{EXP} = \{\text{EXP}_{p, g} : \mathbb{Z}_{p-1} \rightarrow \mathbb{Z}_p^* \text{ where } \text{EXP}_{p, g}(x) = g^x \mod p\}_{(p, g) \in I}.$$  

Correspondingly, we also define the discrete logarithm collection of functions as

$$\text{DL} = \{\text{DL}_{p, g} : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_{p-1} \text{ where } \text{DL}_{p, g}(y) = x \in \mathbb{Z}_{p-1} \text{ such that } y \equiv g^x \mod p\}_{(p, g) \in I}.$$  

We will be interested in the most significant bit (MSB) of the discrete logarithm $x$ of $y$ modulo $p$ and we give it a special notation as follows.

**Definition 1** For $(p, g) \in I$ and $y \in \mathbb{Z}_p^*$, let $B_{p, g}(y) = \begin{cases} 0 & \text{if } y = g^x \mod p \text{ where } 0 \leq x < \frac{p-1}{2} \\ 1 & \text{if } y = g^x \mod p \text{ where } \frac{p-1}{2} \leq x < p-1 \end{cases}.$  

We want to show that if for $p$ a prime and $g$ a generator of $\mathbb{Z}_p^*$, $\text{EXP}_{p, g}(x) \equiv g^x \mod p$ is hard to invert, then given $y = \text{EXP}_{p, g}(x)$, $B_{p, g}(y)$ is hard to compute in a very strong sense; that is, in attempting to compute $B_{p, g}(y)$ we can do no better than essentially guessing its value randomly. The proof will be by way of a reduction. It will show that if we can compute $B_{p, g}(y)$ in polynomial time with probability greater than $\frac{1}{2} + \epsilon$ for some non-negligible $\epsilon > 0$ then we can invert $\text{EXP}_{p, g}(x)$ in time polynomial in $|p|$, $|g|$, and $\epsilon^{-1}$. The following is a formal statement of this fact.

**Theorem 1** Let $S$ be a subset of the prime integers. Suppose there is a polynomial $Q$ and a PTM $G$ such that for all primes $p \in S$ and for all generators $g$ of $\mathbb{Z}_p^*$

$$\Pr[G(p, g, y) = B_{p, g}(y)] > \frac{1}{2} + \frac{1}{Q(|p|)}$$
(where the probability is taken over \( y \in \mathbb{Z}_p^* \) and the coin tosses of \( G \)). Then for every polynomial \( P \), there is a PTM \( I \) such that for all primes \( p \in S \), generators \( g \) of \( \mathbb{Z}_p^* \), and \( y \in \mathbb{Z}_p^* \)

\[
\Pr[I(p, g, y) = x \text{ such that } y \equiv g^x \mod p] > 1 - \frac{1}{P(|p|)}
\]

(where the probability is taken over the coin tosses of \( I \)).

**Proof:**

In proving Theorem 1 we will actually show that there is a Las Vegas algorithm \( I \) satisfying the assertion. The proof involves three main steps, each of which is presented as a lemma. Each subsequent lemma builds on the result of the previous lemma. Lemma 1 shows that if we can compute \( B_{p,g}(g^x) \) using a Las Vegas algorithm which runs in expected polynomial time, then there is a Las Vegas expected polynomial time algorithm for inverting \( \text{EXP}_{p,g}(x) \). Lemma 2 shows that if we can compute \( B_{p,g}(g^x) \) in polynomial time with high probability (greater than \( 1 - \frac{1}{2|p|} \)) then we can invert \( \text{EXP}_{p,g}(x) \) in expected polynomial time using a Las Vegas algorithm. Finally, Lemma 3 gives us Theorem 1 by showing that if we can compute \( B_{p,g}(g^x) \) in polynomial time with probability greater than \( \frac{1}{2} + \frac{1}{Q(|p|)} \) for some polynomial \( Q \), then we can compute \( B_{p,g}(g^x) \) in polynomial time with probability greater than \( 1 - \frac{1}{2|p|} \).

**Lemma 1** Let \( S \) be a subset of the prime integers. Suppose there is a Las Vegas expected polynomial time algorithm \( G \) such that for all primes \( p \in S \), generators \( g \) of \( \mathbb{Z}_p^* \), and \( y \in \mathbb{Z}_p^* \), \( G(p, g, y) = B_{p,g}(y) \). Then there is a Las Vegas expected polynomial time algorithm \( \text{LOG} \), which uses \( G \) as a subroutine, such that for all primes \( p \in S \), generators \( g \) of \( \mathbb{Z}_p^* \), and \( y \in \mathbb{Z}_p^* \), \( \text{LOG}(p, g, y) = x \text{ such that } y \equiv g^x \mod p \).

**Proof:**

Before we present the algorithm, let us recall a few facts about \( \mathbb{Z}_p^* \) when \( p \) is a prime.

**Fact 1** Let \( a \) be a square modulo \( p \) and let \( g \) be a generator of \( \mathbb{Z}_p^* \). The two solutions to \( a \equiv y^2 \mod p \) are of the form \( y \equiv g^x \) and \( -y \equiv g^{x+\frac{p-1}{2}} \) where \( 0 \leq x < \frac{p-1}{2} \). We will refer to \( g^x \) as the principal square root of \( a \) and to \( -g^x \) as the nonprincipal square root of \( a \).

**Fact 2** There exists a Las Vegas algorithm \( \text{SQRT} \) for computing square roots modulo prime integers. That is, \( \text{SQRT} \) is a probabilistic algorithm running in expected polynomial time and on input a prime \( p \) and a square \( a \in \mathbb{Z}_p^* \), \( \text{SQRT}(p, a) = y \) such that \( a \equiv y^2 \mod p \). Such an algorithm \( \text{SQRT} \) was shown to exist in Lemma 1 of Lecture 5.

**Fact 3** Given \( g^x \mod p \) with \( 0 \leq x < p - 1 \) the least significant bit (LSB) of \( x \) can immediately be determined because

\[
\text{LSB}(x) = \begin{cases} 
0 & \text{if } g^x \text{ is a square mod } p \\
1 & \text{if } g^x \text{ is not a square mod } p 
\end{cases}
\]
or equivalently, by Euler’s Theorem,

\[
\text{LSB}(x) = \begin{cases} 
0 & \text{if } (g^x)^{p-1} = 1 \\
1 & \text{if } (g^x)^{p-1} = -1
\end{cases}.
\]

The idea in the construction of LOG is to compute \( x = \text{DL}_{p,g}(y) \) bit by bit starting from the least significant bit. As seen in Fact 3 we can obtain \( \text{LSB}(x) \) in polynomial time. If \( \text{LSB}(x) = 0 \) compute the square root of \( g^x \) using the algorithm \text{SQRT} mentioned in Fact 2. If \( \text{LSB}(x) = 1 \) compute the square root of \( g^x g^{-1} \) using \text{SQRT}. In either case, let \( z \) denote the value of the square root computed. Note that \( z \) may either be the principal square root or the nonprincipal square root. We are interested in the principal square root. By computing \( B_{p,g}(z) = \text{G}(p, g, z) \) we can determine which is the case and, if \( B_{p,g}(z) = 1 \), replace \( z \) by \( p - z \). Now, we have ensured that \( z \) is the principal square root and so, by continuing the above process \(|x| - 1 \) times, we will obtain all of the bits of \( x \).

Formally, the algorithm LOG runs as follows.

Input: \((p, g, y)\) where \( p \) is a prime, \( g \) is a generator of \( \mathbb{Z}_p^* \), and \( y = g^x \in \mathbb{Z}_p^* \) with \( x \in \mathbb{Z}_{p-1} \).
Output: \( x \).

0. If \( y = 1 \) then output 0.
1. Let \( I = \epsilon \) (the empty string), \( z = g^x \).
2. Calculate \( b = \text{LSB}(\text{DL}_{p,g}(z)) \). Let \( I \leftarrow b \circ I \) and \( z \leftarrow \text{SQRT}(p, zg^{-b}) \).
3. If \( G(p, g, z) = 1 \) then \( z \leftarrow p - z \).
4. If \( z = 1 \) then output \( I \). Otherwise, repeat from step 2.

Note that at the end of step 2, \( \text{DL}_{p,g}(z) \circ I = \text{DL}_{p,g}(y) = x \) is invariant and since with each iteration of the loop, \(|I|\) increases by 1, the loop is executed at most \(|x| \leq |p| \) times. Therefore, LOG runs in polynomial time.

\[ \text{Lemma 2} \text{ Let } S \text{ be a subset of the prime integers. Suppose there is a polynomial time algorithm } G \text{ such that for all primes } p \in S, \text{ for all generators } g \text{ of } \mathbb{Z}_p^*, \text{ and for all } y \in \mathbb{Z}_p^*, \]

\[ Pr[G(p, g, y) = B_{p,g}(y)] > 1 - \frac{1}{2|p|} \]

(where the probability is taken over the coin tosses of \( G \)). Then there is a Las Vegas expected polynomial time algorithm \( I \) such that for all primes \( p \in S \), for all generators \( g \) of \( \mathbb{Z}_p^* \), and for all \( y \in \mathbb{Z}_p^* \), \( I(p, g, y) = x \) such that \( y \equiv g^x \mod p \).
Proof:
Run the same algorithm LOG that was used in the proof of Lemma 1. If on each iteration of the loop, \( G(p, g, y) \) is calculated correctly at step 3 then LOG is certainly successful. However, we are only guaranteed a success probability of at least \((1 - \frac{1}{2|p|})^{|x|}\). Therefore, \( \Pr[\text{LOG}(p, g, y) = x \text{ such that } y \equiv g^x \pmod{p}] > (1 - \frac{1}{2|p|})^{|p|} \) (because \(|x| < |p|\)) \geq \frac{1}{2} \) for every value of \(|p|\).

Note that one can verify in polynomial time that \( \text{LOG}(p, g, y) \) is indeed correct by simply comparing \( g^{\text{LOG}(p, g, y)} \pmod{p} \) to \( y \). To obtain a Las Vegas algorithm from LOG, run LOG until it outputs the correct result. Because \( \Pr[\text{LOG}(p, g, y) = x \text{ such that } y \equiv g^x \pmod{p}] > \frac{1}{2} \), we expect to have to run LOG twice.

**Lemma 3** Let \( S \) be a subset of the prime integers. Suppose there is a polynomial \( Q \) and a polynomial time algorithm \( G \) such that for all primes \( p \in S \) and for all generators \( g \) of \( \mathbb{Z}_p^* \)

\[
\Pr[\text{G}(p, g, y) = B_{p,g}(y)] > \frac{1}{2} + \frac{1}{Q(|p|)}
\]

(where the probability is taken over \( y \in \mathbb{Z}_p^* \) and the coin tosses of \( G \)). Then there is a Las Vegas expected polynomial time algorithm \( I \) such that for all primes \( p \in S \), for all generators \( g \) of \( \mathbb{Z}_p^* \), and for all \( y \in \mathbb{Z}_p^* \), \( I(p, g, y) = x \text{ such that } y \equiv g^x \pmod{p} \).

Proof:
For brevity, let \( \epsilon = \frac{1}{Q(|p|)} \). The goal in proving this lemma is to find a way to compute for a given input \((p, g, y)\), the quantity \( B_{p,g}(y) \) with a sufficiently high probability of success so that Lemma 2 can be applied. We will develop an algorithm \( G' \) which makes calls to \( G \) and estimates \( B_{p,g}(y) \) with probability greater than \( 1 - \frac{1}{2|p|} \). Then, as in Lemma 2, we use the algorithm LOG, which will now make calls to \( G' \) instead of to \( G \).

**Assumption For The Remainder Of The Proof**
For the duration of this proof we will assume for an input \((p, g, g^x \pmod{p})\) to \( I \) that \( 2x \) is small where we define small to mean that \( 0 \leq 2x \pmod{p} - 1 \leq \frac{p}{2} \). Here \( t \) is a parameter affecting the running time of \( I \) and which will be specified later. We will show that Lemma 3 is true for an input \((p, g, g^x \pmod{p})\) to \( I \) for which \( 2x \) is small. Observe that in the algorithm LOG, if the index \( 2DL_{p,g}(z) \) is small at the outset (that is, \( 2x \) is small) then \( 2DL_{p,g}(z) \) will remain small throughout the algorithm because in each iteration of LOG, the principal square root is taken and so, the index is at least halved.

We can make the above assumption because we can run \( t \) copies of the algorithm \( I \) in parallel on the inputs \((p, g, g^{x/r_j} \pmod{p}) = (p, g, g^{x/r_j} \pmod{p}) \) where \( r_j = \left\lceil \frac{p-1}{2t} \right\rceil \) for
0 ≤ j < t. Certainly, 2(x − r_j) = 2x − 2r_j is small for some value of j and if the algorithm is successful for the corresponding input, then we can recover x from x − r_j.

**Majority Vote**

We wish to map the question concerning the value of B_{p,g}(y) to a random question about the function B_{p,g} which we can ask of G. One idea is to randomly choose k integers r_1, . . . , r_k such that 0 ≤ r_i < p−1 2 (here k is to be determined) and compute G(p, g, yg^{r_i}) for i = 1, . . . , k. Then, for the guess of the value of B_{p,g}(y), take the majority of the outcomes over the integers r_i. Let us denote this guess by maj_{i=1}^k(G(p, g, yg^{r_i})). Note that if y = g^x is such that 2x is small then either 0 ≤ x ≤ p−1 2 or p−1 2 ≤ x ≤ p−1 + p−1 2t. The value of B_{p,g}(yg^{r_i}) is different from the value of B_{p,g}(y) only when wrap around occurs. Therefore, for at least a 1 − 1 t proportion of the choices for r_i, we will have B_{p,g}(yg^{r_i}) = B_{p,g}(y). In other words, Pr_{0≤r_i<p−1 2}[B_{p,g}(yg^{r_i}) = B_{p,g}(y)] ≥ 1 − 1 t.

However, this procedure may not yield the correct value for B_{p,g}(y) with sufficiently high probability for any y = g^x such that 2x is small because the successes of G may not occur in a uniform fashion. For instance, if Pr_{0≤x<p−1 2}[G(p, g, g^x) = B_{p,g}(g^x)] = 2ε and Pr_{p−1 2≤x<p−1}[G(p, g, g^x) = B_{p,g}(g^x)] = 1 then Pr_{0≤x<p−1}[G(p, g, g^x) = B_{p,g}(g^x)] = 1 2 + ε as required by the hypothesis of Lemma 3, and yet for 0 ≤ x < p−1 2, taking the majority of the outcomes G(p, g, g^{x+r_i}) for i = 1, . . . , k will yield the wrong value for B_{p,g}(g^x) with very high probability. Specifically, since 0 ≤ x < p−1 2 (that is, B_{p,g}(g^x) = 0), it follows from the Markov Inequality (see Appendix A) that

\[
\Pr[\text{maj}_{i=1}^k(G(p, g, g^{x+r_i})) \text{ is correct}] = \Pr[\text{maj}_{i=1}^k(G(p, g, g^{x+r_i}) = 0] = \Pr[\sum_{i=1}^{k}(1 − G(p, g, g^{x+r_i})) > k/2] = \frac{\text{E}[(1 − G(p, g, g^{x+r_i}))]}{1} ≤ \frac{2\epsilon k}{k/2} = 4\epsilon
\]

as claimed. (Note that \Pr[G(p, g, g^{x+r_i}) = 0 | x + r_i ≥ p−1 2] = 0 and that \Pr[G(p, g, g^{x+r_i}) = 0 | x + r_i < p−1 2] = 2ε. Hence,

\[
\Pr[G(p, g, g^{x+r_i}) = 0] = \Pr[G(p, g, g^{x+r_i}) = 0 | x + r_i ≥ p−1 2] \Pr[x + r_i ≥ p−1 2] + \Pr[G(p, g, g^{x+r_i}) = 0 | x + r_i < p−1 2] \Pr[x + r_i < p−1 2]
\]

≤ 2ε.

Therefore, \text{E}[\sum_{i=1}^{k}(1 − G(p, g, g^{x+r_i}))] = k \Pr[G(p, g, g^{x+r_i}) = 0] ≤ 2\epsilon k.)

**Estimating Probabilities**

5
Consider the following strategy. Let $\alpha = \Pr[G(p, g, y) = 0 \mid B_{p,g}(y) = 0]$ and let $\beta = \Pr[G(p, g, y) = 0 \mid B_{p,g}(y) = 1]$. Let $\gamma_y = \Pr_{0 \leq r < \frac{n+1}{2}}[G(p, g, yy^r) = 0]$. If $B_{p,g}(y) = 0$ and $2DL_{p,g}(y)$ is small then $\gamma_y$ is close to $\alpha$ (where close means that $|\alpha - \gamma_y| < \frac{1}{2}$). If $B_{p,g}(y) = 1$ and $2DL_{p,g}(y)$ is small then $\gamma_y$ is close to $\beta$. (again where close means that $|\beta - \gamma_y| < \frac{1}{2}$). We will find good estimates $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}_y$ for $\alpha$, $\beta$, and $\gamma_y$, respectively. Then, if $\hat{\gamma}_y$ is close to $\hat{\alpha}$, we conclude that $B_{p,g}(y) = 0$ and if $\hat{\gamma}_y$ is close to $\hat{\beta}$, we conclude that $B_{p,g}(y) = 1$. We will then show that using this strategy the probability of arriving at the right conclusion can be made larger than $1 - \frac{1}{2^{2\beta}}$.

First observe that $\frac{1}{2} + \epsilon < \Pr[G \text{ is correct}]$

$$= \frac{1}{2} \Pr[G(p, g, g^y) = 0 \mid B_{p,g}(y) = 0] + \frac{1}{2} \Pr[G(p, g, g^y) = 1 \mid B_{p,g}(y) = 1]$$

$$= \frac{1}{2}(\alpha + (1 - \beta)) + \frac{1}{2}(\alpha - \beta) + \frac{1}{2}.$$ 

Hence, $\alpha - \beta > 2\epsilon$. Thus, if $|\gamma_y - \alpha| < \epsilon$, then $\gamma_y$ is closer to $\alpha$ than to $\beta$ and if $|\gamma_y - \beta| < \epsilon$, then $\gamma_y$ is closer to $\beta$ than to $\alpha$.

We can estimate the value of $\alpha$ by randomly choosing a sufficient number of integers $r$ such that $0 \leq r < \frac{n+1}{2}$ (that is, such that $B_{p,g}(r) = 0$) and computing the proportion for which $G(p, g, g^r) = 0$. The Weak Law of Large Numbers (see Appendix A) tells us how many values for $r$ we need to choose in order to get a good estimate for $\alpha$. Similarly, we can estimate the value of $\beta$ and the value of $\gamma_y$.

We are now ready to formally state algorithm $G'$.

Input: $(p, g, y)$ where $p$ is a prime, $g$ is a generator of $\mathbb{Z}_p^*$, and $y = g^x \in \mathbb{Z}_p^*$ with $x \in \mathbb{Z}_{p-1}$ and $0 \leq 2x \leq \frac{n+1}{2}$.

Output: 0 or 1.

1. Choose $t$ such that $\frac{1}{t} < \frac{\epsilon}{2}$. Set $n = \frac{1}{4(\frac{3}{4})^2}t$.

2. Determine an estimate $\hat{\alpha}$ for $\alpha$ and an estimate $\tilde{\beta}$ for $\beta$.

   (i) Randomly choose $r_1, r_2, \ldots, r_n$ and $s_1, s_2, \ldots, s_n$ such that $0 \leq r_i < \frac{n-1}{2}$ and $s_i < p - 1$ for $i = 1, 2, \ldots, n$.

   (ii) Set $\hat{\alpha} = \frac{1}{n} \sum_{i=1}^{n} (1 - G(p, g, g^{r_i})) = 1 - \frac{1}{n} \sum_{i=1}^{n} G(p, g, g^{s_i})$.

   (iii) Set $\tilde{\beta} = \frac{1}{n} \sum_{i=1}^{n} (1 - G(p, g, g^{s_i})) = 1 - \frac{1}{n} \sum_{i=1}^{n} G(p, g, g^{s_i})$.

3. Map questions about $y$ to random places and ask $G$ about those places.
(i) Randomly choose \( u_1, u_2, \ldots, u_n \) such that \( 0 \leq u_i < \frac{p-1}{2} \) for \( i = 1, 2, \ldots, n \).

(ii) Let \( \gamma_y = 1 - \frac{1}{n} \sum_{i=1}^{n} G(p, g, yg^u) \).

4. If \( |\gamma_y - \tilde{\alpha}| < \frac{\epsilon}{4} + \frac{1}{7} \) then output 0. If \( |\gamma_y - \tilde{\beta}| < \frac{\epsilon}{2} + \frac{1}{7} \) then output 1. If neither case occurs, then randomly output 0 or 1.

Claim 1 Let \((p, g, y)\) be the input to \( G \). If \( x = DL_{p,g}(y) \) is such that \( 0 \leq 2x < \frac{p-1}{t} \) then \( \Pr[G'(p, g, y) = B_{p,g}(y)] > 1 - \delta \).

Proof:

By the Weak Law of Large Numbers, the estimate of \( \tilde{\alpha} \) for \( \alpha \) determined in step 2 satisfies \( \Pr[|\alpha - \tilde{\alpha}| < \frac{\epsilon}{4}] > 1 - \frac{\delta}{2} \) and the estimate of \( \beta \) for \( \beta \) satisfies \( \Pr[|\beta - \tilde{\beta}| < \frac{\epsilon}{2}] > 1 - \frac{\delta}{2} \). Further, recall that if \( 2DL_{p,g}(y) \) is small then \( \Pr[B_{p,g}(yg^u)] = B_{p,g}(y) \geq 1 - \frac{1}{t} \). Thus, if \( B_{p,g}(y) = 0 \) then \( \gamma_y \) estimates \( \alpha \) except for a possible error of at most \( \frac{1}{7} \); that is, \( \Pr[|\gamma_y - \alpha| < \frac{\epsilon}{4} + \frac{1}{7}] > 1 - \frac{\delta}{2} \) and if \( B_{p,g}(y) = 1 \) then \( \gamma_y \) estimates \( \beta \) except for a possible error of at most \( \frac{1}{7} \); that is, \( \Pr[|\gamma_y - \beta| < \frac{\epsilon}{4} + \frac{1}{7}] > 1 - \frac{\delta}{2} \). Therefore, if \( 0 \leq 2DL_{p,g}(y) < \frac{p-1}{t} \) then

\[
\Pr[G'(p, g, y) = B_{p,g}(y)] = \Pr[G'(p, g, y) = 0 \mid B_{p,g}(y) = 0] \Pr[B_{p,g}(y) = 0] + \Pr[G'(p, g, y) = 1 \mid B_{p,g}(y) = 0] \Pr[B_{p,g}(y) = 1] \\
= \frac{1}{2} \Pr[|\gamma_y - \tilde{\alpha}| < \frac{\epsilon}{4} + \frac{1}{7} \mid B_{p,g}(y) = 0] + \frac{1}{2} \Pr[|\gamma_y - \tilde{\beta}| < \frac{\epsilon}{2} + \frac{1}{7} \mid B_{p,g}(y) = 1] \\
\geq \frac{1}{2} \Pr[|\gamma_y - \alpha| < \frac{\epsilon}{4} + \frac{1}{7} \text{ and } |\alpha - \tilde{\alpha}| < \frac{\epsilon}{4}] + \frac{1}{2} \Pr[|\gamma_y - \beta| < \frac{\epsilon}{2} + \frac{1}{7} \text{ and } |\beta - \tilde{\beta}| < \frac{\epsilon}{2}] \\
> \frac{1}{2}(1 - \frac{\delta}{2})^2 + \frac{1}{2}(1 - \frac{\delta}{2})^2 \\
> 1 - \delta
\]

as required.

Thus, if we take \( \delta \) such that \( \delta < \frac{1}{24p} \), then we obtain a PTM \( G' \) as in the hypothesis of Lemma 2. Hence, we can run the PTM \( LOG \) from Lemma 1, as was done in the proof of Lemma 2 but with \( G' \) instead of \( G \), to prove Lemma 3. Note that by the choice of \( \delta \), \( n \) is polynomial in \( |p| \) and so \( G' \) runs in polynomial time. Note further that we have also allowed \( t \) to be chosen of polynomial size and so we need run only polynomially many copies of \( LOG \) in parallel.

As mentioned at the outset, Lemma 3 immediately gives us Theorem 1 and therefore, the proof of the theorem is complete.
3 Simultaneous Security of Many Bits

It can be shown that $O(\log \log p)$ of the most significant bits of $x \in \mathbb{Z}_{p-1}$ are hidden by the function $\text{EXP}_{p,g}(x)$. We state this result here without proof.

**Theorem 2** For a PTM $A$, let

$$\alpha = \Pr[A(p, g, g^x, x_{\log \log p} \ldots x_0) = 0 \mid x = x_0 \ldots x_{p-1}]$$

(where the probability is taken over $x \in \mathbb{Z}_n^*$ and the coin tosses of $A$) and let

$$\beta = \Pr[A(p, g, g^r, r_{\log \log p} \ldots r_0) = 0 \mid r_i \in_R \{0, 1\}]$$

(where the probability is taken over $x \in \mathbb{Z}_n^*$, the coin tosses of $A$, and the bits $r_i$). Then under the Discrete Logarithm Assumption, we have that for every polynomial $Q$ and every PTM $A$, $\exists k_0$ such that $\forall k > k_0$, $|\alpha - \beta| < \frac{1}{Q(k)}$.

**Corollary 1** Under the Discrete Logarithm Assumption we have that for every polynomial $Q$ and every PTM $A$, $\exists k_0$ such that $\forall k > k_0$ and $\forall k < \log \log p$

$$\Pr[A(p, g, g^x, x_k \ldots x_0) = x_{k+1}] < \frac{1}{2} + \frac{1}{Q(k)}$$

(where the probability is taken over the primes $p$ such that $|p| = k$, the generators $g$ of $\mathbb{Z}_p^*$, $x \in \mathbb{Z}_p^*$, and the coin tosses of $A$).

For further information on the simultaneous or individual security of the bits associated with the discrete logarithm see [2].

A Inequalities From Probability Theory

**Proposition 1** (Markov’s Inequality) If $Z$ is a random variable that takes only non-negative values, then for any value $a > 0$, $\Pr[Z \geq a] \leq \frac{E[Z]}{a}$.

**Proposition 2** (Weak Law of Large Numbers) Let $z_1, \ldots, z_n$ be independent 0-1 random variables (Bernoulli random variables) with mean $\mu$. Then $\Pr[|\frac{1}{n}\sum_{i=1}^{n}z_i - \mu| < \epsilon] > 1 - \delta$ provided that $n > \frac{1}{\delta^2}$.

References
