CSA E0 235: Cryptography March 2,2016

Scribe for Lecture 12

Instructor: Arpita Patra *Submitted by: Kuntal Ghosh*

1 One-Way Functions

A one-way function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is easy to compute, yet hard to invert. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a function. Consider the following experiment defined for any algorithm A and any value n for the security parameter:

The inverting experiment $Invert_{A,f}(n)$

1. Choose uniform $x \in \{0,1\}^n$, and compute $y := f(x)$.
2. A is given 1^n and y as input, and outputs x'.
3. The output of the experiment is defined to be 1 if $f(x') = y$, and 0 otherwise.

A need not find the original pre-image x; it suffices for A to find any value x' for which $f(x') = y = f(x)$. Now, let us define what we mean f to be a one-way funcion(OWF):

Definition 1 *A function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is **one-way** if the following two conditions hold:*

- *Easy to compute: $\forall x \leftarrow 0, 1^n, f(x)$ can be computed in $poly(n)$ time.*
- *Hard to invert: For every probabilistic polynomial-time algorithm A, there is a negligible function negl such that*

$$Pr[Invert_{A,f}(n) = 1] \leq negl(n)$$

We can express the second requirement of the definition as follows:

$$Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) \in f^{-1}(f(x))] \leq negl(n)$$

Notice that any function is invertible in principle given enough time/computational power. The assumption of existence of OWF is about computational hardness. Let us take some examples to understand which functions are not one-way.

- Example 1: Consider a function f such that,

$$Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) \in f^{-1}(f(x))] \begin{cases} > \frac{1}{n^{10}} & \text{when n is even} \\ \leq negl(n) & \text{hen n is odd} \end{cases}$$

Clearly, f is not a one-way function. Because, for infinitely many n, f is not hard to invert.

- Example 2: Let $f(x,y) = x.y$. Then,

$$Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) \in f^{-1}(f(x))] \geq 3/4$$

Because, if $f(x,y)$ is even, $(2, xy/2)$ is a pre-image of f. Hence, the probability is $3/4$ (when both x and y are odd, we can't calculate the pre-image efficiently) which is not negligible. So, f is not one-way.

- Example 3: $f(x) = x_1, x_2, \ldots, x_{n-1}$ where $x \in \{0,1\}^n$. Then,

$$Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) \in f^{-1}(f(x))] \geq 1/2$$

which is not negligible. Hence, f is not one-way.

1.1 Candidate One-Way Functions

We do not know how to prove one-way functions exist unconditionally, so we must conjecture or assume their existence. Such a conjecture is based on the fact that several natural computational problems have received much attention, yet still have no polynomial-time algorithm for solving them.
Most famous one-way function is *integer factorization.* Given $f(x,y)$ we need to find two equal length primes such that $f(x,y) = x.y$.
Another candidate one-way function is based on *subset-sum problem* which is defined as,

$$f_{ss}(x_1, x_2, \ldots, x_n, J) = (x_1, x_2, \ldots, x_n, [\Sigma_{j \in J} x_i \bmod 2^n])$$

where each x_i is an n-bit string interpreted as an integer, and J is an n-bit string interpreted as specifying a subset of $\{1, \ldots, n\}$. Inverting f_{ss} on an output $(x_1, x_2, \ldots, x_n, y)$ requires finding a subset $J' \subset \{1, \ldots, n\}$ such that $\Sigma_{j \in J'} x_j = y \bmod 2^n$.
Note that, $P \neq NP$ does not imply existence of OWF. The former suggests every PPT algorithm must fail to solve at least for one input whereas the latter suggests every PPT algorithm must fail to solve *almost always* (for any random input). Believing that OWF exist is much more than believing $P \neq NP$.

2 One-Way Permutations

Definition 2 *A function f is if it has the following properties:*

1. *f is a one-way function.*
2. *f is length-preserving i.e. $\forall x$, $|f(x)| = |x|$.*
3. *f is one-to-one i.e. every $f(x)$ has a unique pre-image x.*

3 Hard-Core Predicates

By definition, a one-way function is hard to invert. One might get the impression that nothing about x can be determined from $f(x)$ in polynomial time. This is not necessarily the case. Indeed, it is possible for $f(x)$ to *leak* a lot of information about x even if f is one-way. For a trivial example, let f be a one-way function and define $g(x_1, x_2) = (x_1, f(x_2))$ where $|x_1| = |x_2|$. It is easy to show that g is also a one-way function, even though it reveals half its input. But, $f(x)$ must hide *something* about x. *Something* that remains hidden about x even given $f(x)$ is captured by hard-core predicates. Hard-core predicates are one bit of info about x that is hard to guess given $f(x)$. We define hard-core predicates for one-way functions as follows:

Definition 3 *A boolean function $hc : \{0,1\}^*$ is a hard-core predicate for a one-way function $f : \{0,1\}^* \to \{0,1\}^*$ if the followings hold:*

1. *Given x, $hc(x)$ can be computed in poly(n) time.*
2. $Pr_{x \leftarrow \{0,1\}^n}[A(f(x), 1^n) = hc(x)] \leq 1/2 + negl(n)$

Hard-core Predicate may exist even for functions that are not one-way.For example, consider a non-OWF $f(x_1, \ldots, x_n) = (x1, \ldots, x_{n-1})$. $hc(x)$ for this function is x_n. For a random x, given $f(x_1, \ldots, x_n)$, $hc(x_1, \ldots, x_n) = x_n$ can be guessed with probability 1/2. But, we are not interested in hard-core predicates for functions which are not one-way.
We can also define hard-core predicates for one-way permutations in a similar way.

Definition 4 *A boolean function $hc : \{0,1\}^*$ is a hard-core predicate for a one-way permutation $f : \{0,1\}^* \to \{0,1\}^*$ if the followings hold:*

1. *Given x, $hc(x)$ can be computed in poly(n) time.*
2. $Pr_{x \leftarrow \{0,1\}^n}[A(f(x), 1^n) = hc(x)] \leq 1/2 + negl(n)$

But, finding a hard-core predicate for a OWF/OWP is not simple. Consider, $hc(x) = \oplus_{i=1}^n x_i$. One might hope that if f can't be inverted then $f(x)$ must hide at least one of the bits x_i for which calculating $hc(x)$ will be hard to compute. Now, construct a new OWF using f. Let, $g(x) = (f(x), \oplus_{i=1}^n x_i)$. Here, $\oplus_{i=1}^n x_i$ is not a hard-core predicate for g. In fact, for a given fixed boolean function $hc(x)$, there always exist a OWF f such that hc is not a hard-core predicate for the function f.

4 Hard-Core Predicates from One-Way Functions

The next step is to show that a hard-core predicate exists for any one-way function. Actually, it remains open whether this is true; we show something weaker that suffices for our purposes. Namely, we show that given a one-way function f we can construct a different one-way function g along with a hard-core predicate of g.

Theorem 1 ***(GoldreichLevin theorem)*** *Let f be a OWP and define g by $g(x, r) = (f(x), r)$ where $x = x_1, \ldots, x_n$ and $r = r_1, \ldots, r_n$. Then the Boolean function $hc(x, r) = r_1 x_1 \oplus \ldots \oplus r_n x_n$ is a hard-core predicate for the function g.*

Due to the complexity of the proof, we divide the proof into three successively stronger results culminating in what is claimed in the theorem.

4.1 A Simple Case

Proposition 1 *Let f and hc be as in Theorem 1. If there exists a polynomial-time algorithm A such that,*

$$Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x),r) = hc(x,r)] = 1$$

for all n, then there exists a polynomial-time algorithm A' such that,

$$Pr_{x \leftarrow \{0,1\}^n}[A'(f(x)) = x] = 1$$

for all n. ◇

Proof We construct A' as follows. $A'(1^n, y)$ computes $x_i := A(y, e^i)$ for $i = 1, \ldots, n$,

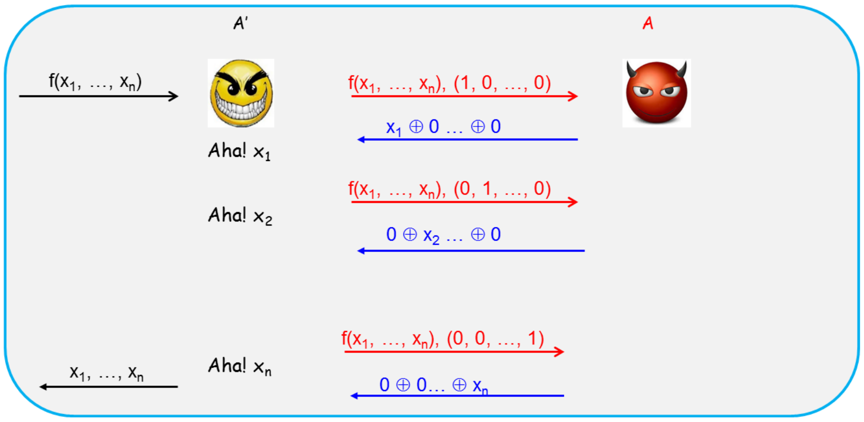
A'
A
f(x1, ..., xn)
f(x1, ..., xn), (1, 0, ..., 0)
x1 ⊕ 0 ... ⊕ 0
Aha! x1
f(x1, ..., xn), (0, 1, ..., 0)
Aha! x2
0 ⊕ x2 ... ⊕ 0
f(x1, ..., xn), (0, 0, ..., 1)
x1, ..., xn
Aha! xn
0 ⊕ 0... ⊕ xn

Figure 1: Simple case

where e^i denotes the n-bit string with 1 in the ith position and 0 everywhere else. Then, A' outputs $x = x_1, \ldots, x_n$. Clearly, A' runs in poly(n) time.
In the execution of $A'(1^n, f(x'))$, the value x_i computed by A' satisfies,

$$x_i = A(f(x'), e^i) = hc(x', e^i) = \oplus_{j=1}^{n} x'_j . e^i_j = x'_i$$

Thus, $x_i = x'_i$ for all i and so A' outputs the correct inverse $x = x'$. ■ If f is one-way, it is impossible for any PPT algorithm to invert f with non-negligible probability. Thus, we conclude that there is no polynomial-time algorithm that always correctly computes $g(x,r)$ from $(f(x), r)$.

4.2 A More Involved Case

Proposition 2 *Let f and hc be as in Theorem 1. If there exists a polynomial-time algorithm A and a polynomial $p(.)$ such that,*

$$Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r) = hc(x,r)] \geq 3/4 + 1/p(n)$$

for infinitely many values of n, then there exists a polynomial-time algorithm A' such that,

$$Pr_{x\leftarrow\{0,1\}^n}[A'(f(x)) = x] \geq 1/4.p(n)$$

for infinitely many values of n. ♢

Proof Notice that the strategy in the proof of Proposition 1 fails here because it may be that A never succeeds when $r = e^i$ (although it may succeed, say, on all other values of r). Furthermore, in the present case A' does not know if the result $A(f(x), r)$ is equal to $hc(x, r)$ or not; the only thing A' knows is that with high probability, algorithm A is correct.
The main observation underlying the proof of this proposition is that for every $r \in \{0,1\}^n$, the values $hc(x, r \oplus e^i)$ and $hc(x, r)$ together can be used to compute the ith bit of x (fig. 2).

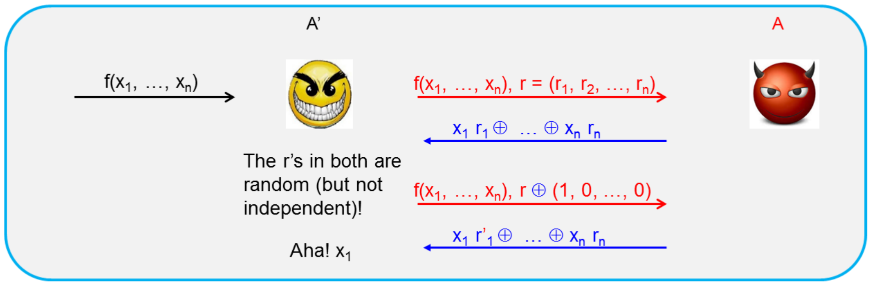
A'
A
f(x1, …, xn)
f(x1, …, xn), r = (r1, r2, …, rn)
x1 r1 ⊕ … ⊕ xn rn
The r's in both are random (but not independent)!
f(x1, …, xn), r ⊕ (1, 0, …, 0)
x1 r'1 ⊕ … ⊕ xn rn
Aha! x1

Figure 2: Computation of x_i

This is true because

$$\begin{aligned} hc(x,r) \oplus hc(x, r\oplus e^i) &= \left(\oplus_{j=1}^n x_j.r_j\right) \oplus \left(\oplus_{j=1}^n x_j.(r_j \oplus e^i_j)\right) \\ &= x_i.r_i \oplus (x_i.\overline{r_i}) = x_i \end{aligned}$$

Unfortunately, A' does not know when A answers correctly and when it does not; A' knows only that A answers correctly with "high" probability. For this reason, A' will use multiple random values of r, using each one to obtain an estimate of x_i, and will then take the estimate occurring a majority of the time as its final guess for x_i.
As a preliminary step, we show that for many x's the probability that A answers correctly for both $(f(x), r)$ and $(f(x), r \oplus e^i)$, when r is uniform, is sufficiently high. This allows us to fix x and then focus solely on uniform choice of r, which makes the analysis easier.

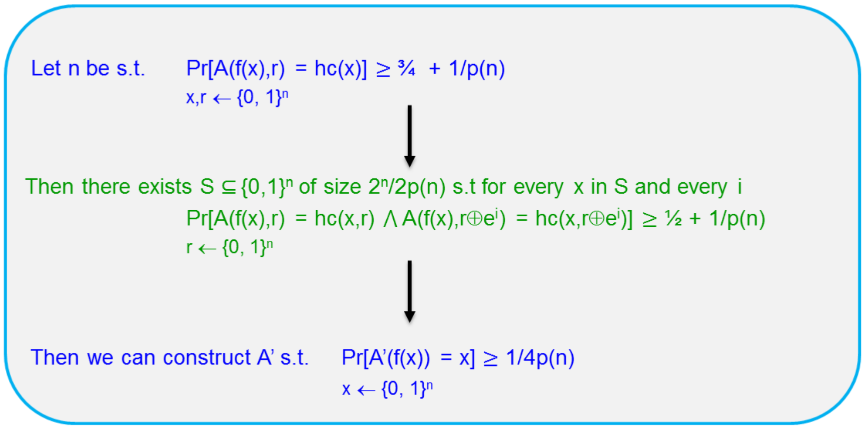
Let n be s.t.
Pr[A(f(x),r) = hc(x)] ≥ ¾ + 1/p(n)
x,r ← {0, 1}^n
Then there exists S ⊆ {0,1}^n of size 2^n/2p(n) s.t for every x in S and every i
Pr[A(f(x),r) = hc(x,r) ∧ A(f(x),r⊕e^i) = hc(x,r⊕e^i)] ≥ ½ + 1/p(n)
r ← {0, 1}^n
Then we can construct A' s.t.
Pr[A'(f(x)) = x] ≥ 1/4p(n)
x ← {0, 1}^n

Figure 3: Proof sketch

Claim 1 *Let n be such that*

$$Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq 3/4+1/p(n)$$

Then there exists a set $S_n\subseteq\{0,1\}^n$ of size at least $\frac{1}{2p(n)}.2^n$ such that for every $x\in S_n$ it holds that

$$Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq 3/4+1/2p(n)$$

$\diamondsuit$

Proof We have

$$\begin{aligned}
Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)] &= \textstyle\sum_{x'\in\{0,1\}^n} Pr_{r\in\{0,1\}^n}\left[A(f(x'),r)=hc(x',r)\right].Pr\left[x=x'\right]\\
&= \tfrac{1}{2^n}\textstyle\sum_{x'\in\{0,1\}^n} Pr_{r\in\{0,1\}^n}\left[A(f(x'),r)=hc(x',r)\right]\\
&= \tfrac{1}{2^n}\textstyle\sum_{x'\in S_n} Pr_{r\in\{0,1\}^n}\left[A(f(x'),r)=hc(x',r)\right]\\
&\quad+\tfrac{1}{2^n}\textstyle\sum_{x'\notin S_n} Pr_{r\in\{0,1\}^n}\left[A(f(x'),r)=hc(x',r)\right]\\
&\leq \tfrac{|S_n|}{2^n}+\tfrac{1}{2^n}\textstyle\sum_{x'\notin S_n}(3/4+1/2p(n))\\
&\leq \tfrac{|S_n|}{2^n}+(3/4+1/2p(n)).
\end{aligned}$$

Since $3/4+1/p(n)\leq Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]$, straightforward algebra gives $|S_n|\geq\frac{1}{2p(n)}.2^n$. ■

Now, it is easy to prove the highlighted(green) statement in figure 3.
Claim 2 Let n be such that

$$Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq 3/4+1/p(n)$$

Then there exists a set $S_n \subseteq \{0,1\}^n$ of size at least $\frac{1}{2p(n)}.2^n$ such that for every $x \in S_n$ and every i it holds that

$$Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r) = hc(x,r) \wedge A(f(x),r\oplus e^i) = hc(x,r\oplus e^i)] \geq 1/2 + 1/p(n)$$

$\diamondsuit$

Proof We know that for any $x \in S_n$ we have

$$Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r) \neq hc(x,r)] \leq 1/4 - 1/2p(n)$$

Fix $i \in \{1,\ldots,n\}$. If r is uniformly distributed so is $r \oplus e^i$; thus

$$Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r\oplus e^i) \neq hc(x,r\oplus e^i)] \leq 1/4 - 1/2p(n)$$

Hence, A is correct on both $hc(x,r)$ and $hc(x,r\oplus e^i)$ is at least

$$1-(1/4-1/2p(n)+1/4-1/2p(n)) = 1/2+1/p(n).$$

$\blacksquare$

Now from the previous claim we construct a probabilistic polynomial-time algorithm A' that inverts $f(x)$ with probability at least $1/2$ when $x \in S_n$ (3^{rd} statement of figure 4).

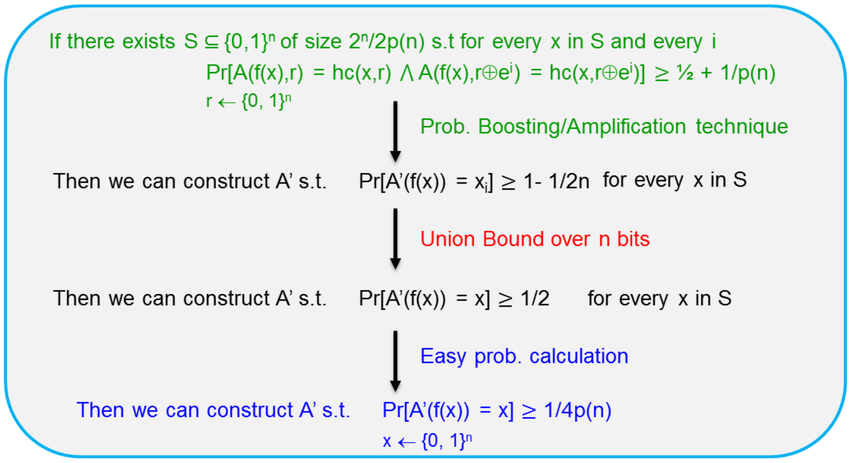
If there exists S ⊆ {0,1}ⁿ of size 2ⁿ/2p(n) s.t for every x in S and every i
Pr[A(f(x),r) = hc(x,r) ∧ A(f(x),r⊕eⁱ) = hc(x,r⊕eⁱ)] ≥ ½ + 1/p(n)
r ← {0, 1}ⁿ
Prob. Boosting/Amplification technique
Then we can construct A' s.t. Pr[A'(f(x)) = xᵢ] ≥ 1- 1/2n for every x in S
Union Bound over n bits
Then we can construct A' s.t. Pr[A'(f(x)) = x] ≥ 1/2 for every x in S
Easy prob. calculation
Then we can construct A' s.t. Pr[A'(f(x)) = x] ≥ 1/4p(n)
x ← {0, 1}ⁿ

Figure 4: Proof sketch

This suffices to prove Proposition 2 since then, for infinitely many values of n,

$$Pr_{x\leftarrow\{0,1\}^n}[A(1^n,f(x)) \in f^{-1}(f(x))]$$

$$\geq Pr_{x\leftarrow\{0,1\}^n}[A(1^n,f(x)) \in f^{-1}(f(x))|x \in S_n]Pr_{x\leftarrow\{0,1\}^n}[x \in S_n]$$

$$\frac{1}{2}\cdot\frac{1}{2p(n)} = \frac{1}{4p(n).}$$

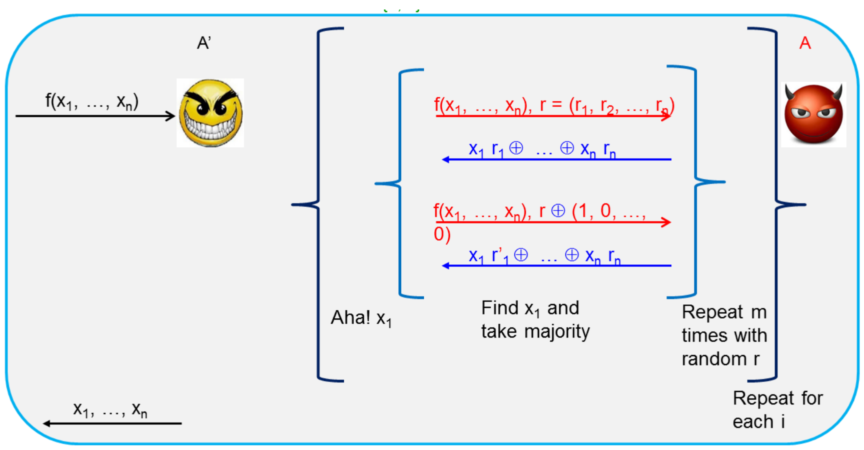
A'
A
f(x1, …, xn)
f(x1, …, xn), r = (r1, r2, …, rn)
x1 r1 ⊕ … ⊕ xn rn
f(x1, …, xn), r ⊕ (1, 0, …, 0)
x1 r'1 ⊕ … ⊕ xn rn
Aha! x1
Find x1 and take majority
Repeat m times with random r
Repeat for each i
x1, …, xn

Figure 5: Prob. Boosting/Amplification technique

Algorithm A', given as input 1^n and y, works as follows (figure 5):

1. For $i = 1, \ldots, n$ do:
   - Repeatedly choose a uniform $r \in \{0,1\}^n$ and compute $A(y,r) \oplus A(y, r \oplus e^i)$ as an estimate for the i^{th} bit of the preimage of y. After doing this sufficiently many times (say m), let x'_i be the "estimate" that occurs a majority of the time.
2. Output $x' = x'_1, \ldots, x'_n$.

We sketch an analysis of the probability that A' correctly inverts its given input y. Say, $y = f(x)$ and for $x \in S_n$ the previous claim ensures that for $Pr_{r \leftarrow \{0,1\}^n}[A(y,r) \oplus A(y, r \oplus e^i) = hc(x,r)] \geq 1/2 + 1/p(n)$. Now, if we take sufficiently many estimates(say m) and letting x'_i be the majority value, by *Chernoff bound* we can ensure that

$$Pr_{r \leftarrow \{0,1\}^n}(x'_i \neq hc(x, e^i)) \leq e^{(-m/2p(n)^2} = 1/2n$$

We can calculate m from the above equation and repeat the experiment m times. A union bound thus shows that A' is incorrect for some i with probability at most $n.\frac{1}{2n} = \frac{1}{2}$. That is, A' is correct for all $i$and thus correctly inverts $y$with probability at least $\frac{1}{2}$. This completes the proof of Proposition 2. ■

Full proof of theorem 1 is out of the scope of this course.

References

[1] Katz, Jonathan, and Yehuda Lindell. *Introduction to modern cryptography.*. CRC Press, 2014.

[2] Arpita Patra. *http://drona.csa.iisc.ernet.in/arpita/Cryptography16.html.* Course Materials.