Notes from the
Student Wish Course on

Zero-Knowledge

Michael Nüsken

b-it

(Bonn-Aachen International Center for Information Technology)

summer 2007
X = \{0,1\}^n, Y = g(U_{10}), g is not necessarily injective.

Estimate the statistical difference.

(a) Consider \( X = U_{2n} \) versus \( g(U_n) \).

(b) Consider \( g: \{0,1\}^3 \to \{0,1\}^6 \)

\[
\begin{array}{c|c}
X & g(x) \\
\hline
000 & 001100 \\
001 & 001110 \\
010 & 010101 \\
011 & 011011 \\
100 & 101000 \\
101 & 100101 \\
110 & 110010 \\
111 & 110011 \\
\end{array}
\]

Compute the statistical difference between \( g(U_2) \) and \( U_6 \).

(ii) Consider the distinguishing \( D(y) \) that outputs 1 if at most four bits in \( y \) are one. ...

A3 Understand the details of the proof that statistical closeness implies computational indistinguishability.

Proof as a tool:

\[
\frac{1}{2} \sum_x | \text{prob}(X = x) - \text{prob}(Y = x) | \\
= \max_S | \text{prob}(X \in S) - \text{prob}(Y \in S) |
\]

A4 (i) Give a problem / a language in \( \text{NP} \) but not \( \text{NP}-\text{complete} \).

(ii) Give a language in \( \text{NP} \setminus \text{P} \).

(iii) Give a language in \( \text{BPP} \setminus \text{P} \).
\[
\Delta = \frac{1}{2} \sum_{a \in S} \left| \text{prob}(X=a) - \text{prob}(Y=a) \right|
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \max_{S} \left( \frac{\text{prob}(X=S)}{1 - \text{prob}(Y=S)} - \frac{\text{prob}(Y=S)}{1 - \text{prob}(X=S)} \right)
\]

\[
S = \emptyset \quad \alpha = 1 \quad \beta > 0 \quad \delta
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \frac{1}{2} \sum_{a \in S} \left( \text{prob}(X=a) - \text{prob}(Y=a) \right)
\]

\[
\Delta = \max_{S} \left( \frac{\text{prob}(X=S)}{1 - \text{prob}(Y=S)} - \frac{\text{prob}(Y=S)}{1 - \text{prob}(X=S)} \right)
\]

\[
\Delta = \max_{S} \left( \frac{\text{prob}(X=S)}{1 - \text{prob}(Y=S)} - \frac{\text{prob}(Y=S)}{1 - \text{prob}(X=S)} \right)
\]
Assume that \( D \) is a distinguisher, say given as a poly-size circuit \( C_{\alpha} \), or say it doesn't use randomness.

Then let
\[ S = d \alpha \text{ } 1 \text{ } D(\alpha) = 1. \]

\[
\left| \Pr[D(X) = 1] - \Pr[D(Y) = 1] \right|
\]
\[X \in S\]
\[Y \in S\]
\[
= \left| \Pr[X \in S] - \Pr[Y \in S] \right|
\leq \Delta(X, Y).
\]

\( \Delta \)

(i) \( \Delta \left( \frac{9}{8}, \frac{1}{8} \right) = \frac{1}{8} - 2^{-3} \)

(ii) \( S = 1 - \frac{57}{64} \leq \frac{7}{8} \)

\( \frac{7}{64} \)

If \( \text{NP} = \text{P} \):
(i) any \( \text{P} \text{-lang.} \)

(ii) any \( \text{NP} \text{-complete}. \)

(iii) \( \text{BPP} \)
Monday summary

- Randomness?
  - ensemble
  - computational indistinguishable
  - statistically close

Lemmas: shc \implies ci

- ci. by repeated sampling

Thus ci. \iff ci. poly(n)

Proof: hybrid technique.
Proof (Though short becomes long)

Construction of the long generator $G$

Inputs: $s_0$ - $n$-bit string.
Outputs: $g_1, \ldots, g_{p(n)}$ - $p(n)$-bit string.

\[ U_n \rightarrow g_1 \rightarrow g_2 \rightarrow \ldots \rightarrow g_{p(n)} \]

Here, $G_2$ is a pseudo-random generator with expansion $e(n) = n + 1$,
and $p$ some (polynomial) function with $p(n) > n$.

We want to compare $G(U_n)$ with $U_{p(n)}$.
(It is clear that $G$ is poly-time and expandy).
We know that $(G_2(U_n))_n$ and $(U_{p(n)})_n$ are c.i.

Construct hybrids:

\[ H^k_n : \quad U_n' \rightarrow g_1 \rightarrow g_2 \rightarrow \ldots \rightarrow g_{p(n)} \]

\[ H^k_n (U_{p(n)}) \]
Show:

- Extremes:
  \[ H_u^0 = g(U_n) \]
  \[ H_u^{p(n)} = V_p(n) \]

- Neighbors:
  \[ H_u^{k+1} = \sigma_{k+1} \rightarrow U_{n+1} \]
  \[ H_u^k = \sigma_{k+1} \rightarrow G_i(U_n) \]

Suppose \( D_0 \) tries to distinguish \( H_u^k \) and \( H_u^{k+1} \).

Then \( D_x \):

1. Input: \( \sigma_1 \rightarrow \sigma_k \rightarrow U_{n+1} \)
2. Output: \( 0 \) or \( 1 \).
3. Choose \( \sigma_1 \rightarrow \sigma_k \) at random uniformly.
4. Compute \( \sigma_{k+1} \rightarrow \sigma_{p(n)} \) as in the generator starting with seed \( \sigma_{k+1} \).
5. Output \( D_0 (\sigma_1 \ldots \sigma_k \sigma_{k+1} \sigma_{k+2} \ldots \sigma_{p(n)}) \)

Claim: if \( D_0 \) distinguishes \( H_u^k \) and \( H_u^{k+1} \)
then \( D_x \) distinguishes \( G_i(U_n) \) and \( U_{n+1} \).

Warning: this is only heuristic.

\[
\Pr(D_x(g(U_n)) = 1) - \Pr(D_x(V_{n+1}) = 1) = \Pr(D_0(H_u^k) = 1) - \Pr(D_0(H_u^{k+1}) = 1)
\]

Thus: if \( D_0 \) the claim is true.

- \# Hybrids = \( p(n) \) polynomial

\[ L \]
Suppose \( D \) tries to distinguish 
\( G(U_n) \) and \( V_p(n) \).

Define \( D' \)

Input: \( \alpha \in \{0, 1\}^{n+1} \)

Output: 0 or 1.

1. Choose \( k \in \mathbb{R} \left( 0, \ldots, p(n)-1 \right) \) uniformly random.

2. Choose \( g_1, \ldots, g_k \in \mathbb{R} \) uniformly random and independently.

3. While \( \alpha = g_{k+1}, \ldots, g_{2k} \).

4. Compute \( g_{k+2}, \ldots, g_{p(n)} \) as in the long generator.

5. Output \( D \left( g_1, \ldots, g_k, g_{k+1}, g_{k+2}, \ldots, g_{p(n)} \right) \)

\[
\text{prob} \left( D'(G(U_n)) = 1 \right) = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \text{prob} \left( D(H^k) = 1 \right)
\]

\[
= \sum_{k_0=0}^{p(n)-1} \text{prob} \left( D'(G_x(U_n)) = 1 \mid k = k_0 \right) \cdot \text{prob} \left( k = k_0 \right)
\]

\[
= \sum_{k_0=0}^{p(n)-1} \text{prob} \left( D(H^k) = 1 \mid k = k_0 \right)
\]

\[
\text{prob} \left( D'(U_{n+1}) = 1 \right) = \frac{1}{p(n)} \sum_{k_0=0}^{p(n)-1} \text{prob} \left( D(H^k_{n+1}) = 1 \right)
\]
\[ \text{prob } (D' (G,(V_n)) = 1) \quad - \quad \text{prob } D' (V_{n+1}) = 1 \]
\[ = \frac{1}{p(n)} \left( \text{prob } (D(H_n^0) = 1) \quad - \quad \text{prob } (D(H_n^{p(n)-u}) = 1) \right) \]
\[ = G(u_n) \quad U_{p(n)} \]

Thus: if \( D \) distinguishes \( G(V_n) \) and \( U_{p(n)} \)
then \( D' \) distinguishes \( G_n(V_n) \) and \( V_{n+1} \).

**Til & Mathies construction**

\[ G(TM) \]

inputs: \( s \in \{0,1\}^n \)

outputs: \( G \in \{0,1\}^{p(n)} \).

1. \( s_0 \leftarrow s_0 \)
2. \( \text{for } i = 1 \ldots p(n) \text{ do} \)
3. \( s_i \leftarrow G_T(s_{i-1}) \)
4. \( \text{return } s_{p(n)} \).

Then? \( G(TM) \) is a pdf. if \( G_n \) is pdf.

**Ex** Proof this! \( H^k_n = G^{p(n)-n-k} (V_{n+k}) \)

But \( D' \) ? \( \rightarrow H_n^0 = G(TM(V_n)), \quad H^{p(n)-n}_n = U_{p(n)} \).
Proof (Yao, right direction)

Assume that \( A \) is a predictor for the ensemble \( (X_n) \).

(Think of \( X_n = G(U_n) \) for a generator \( G \).

Define a distinguisher \( D \):

- Input: a bit string
- Output: 0 or 1
  - If \( A(\alpha, A) = \text{next}_A(\alpha) \)
    - Then return 1
    - Else return 0.

\[
\text{prob} \left( D \left( X_n \right) = 1 \right) = \text{prob} \left( A(\text{next}_A(X_n)) = \text{next}_A(X_n) \right)
\]

\[
\text{prob} \left( D \left( U_n \right) = 1 \right) = \frac{A}{2}.
\]

\[
\left| \text{prob} \left( D(X_n) = 1 \right) - \text{prob} \left( D(U_n) = 1 \right) \right| = \left| \frac{1}{2} - \frac{A}{2} \right| 
\]

\[
\geq \frac{1}{2} + \frac{1}{\text{poly}(n)} \quad \text{for infinitely many } n \quad \text{and some polynomial } \text{poly}(n)
\]

\[
\geq \frac{1}{\text{poly}(n)}
\]

But \( (X_n) \) and \( (U_n) \) are assumed to be c.i.

Thus c.i. from \( (U_n) \) \( \Rightarrow \) unpredictable.

\[ \square \]
Proof (Yes, unpredictable $\implies$ c.i. for $(U_e\omega)$) 7.8.07

So assume to the contrary that

$D$ is a good distinguisher,

_i.e._  $|\text{prob}(D(X_n) = 1) - \text{prob}(D(U_e\omega) = 1)| > \frac{1}{p(n)}$

for infinitely many $n$ and some polynomial $p$.  

Define the algorithm $A$:

\begin{itemize}
  \item\textbf{Input}:  a bit string of length $e(n)$,
  \item $1^n$.
  \item\textbf{Output}:  $\hat{e}$ a guess for first unread bit of $e$.
\end{itemize}

1. choose $k \in \{0, \ldots, e(n)\}$ uniformly at

2. choose $\beta_{k+1}, \ldots, \beta_{e(n)} \in \{0, 1\}$

\hspace{1cm} uniformly random and independent.

3. if $D((\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_{e(n)})) = 1$

\hspace{1cm} then return $\beta_{k+1}$

\hspace{1cm} else return $\bar{\beta}_{k+1}$

\[ H_n^k = \text{the first } k \text{ bits of } X_n, \ U_{e(n) - k} \cdot \]

Observe:  $H_n^0 = U_{e(n)}$,  $H_n^{e(n)} = X_n$.

Heuristic:  $H_n^{e(n)} \sim H_n^k$ reflects `prophets'.

\# hybrids $= e(n) + 1$ polynomial.
\[ \text{Wlog:} \]
\[ \text{prob} \left( D(X_n) = 1 \right) - \text{prob} \left( D(U_{c(n)}) = 1 \right) > \frac{1}{p(n)} \]

for infinitely many \( n \) and some polynomial \( p \).

(Otherwise flip the output of \( D \).

We want to show that

\[ \text{succ}_A := \text{prob} \left( A(X_n, 1^n) = \text{next}^A(X_n) \right) > \frac{1}{2} + \frac{1}{\text{poly}(n)} \]

First split according to the chosen \( k \).

\[ \text{succ}_A = \frac{1}{\beta(n)} \sum_{k=0}^{p(n) - 1} \text{prob} \left( A(X_n, 1^n) = \text{next}^A(X_n) \right) \]

if \( D \left( X_n^{(k)} \ldots X_n^{(k)} U_{c(n)}^{(k+1)} \ldots U_{c(n)}^{(c(n))} \right) = 1 \)

answers \( U_{c(n)}^{(k+1)} \)

else if \( D \left( X_n^{(k)} \ldots X_n^{(k)} U_{c(n)}^{(k+1)} \ldots U_{c(n)}^{(c(n))} \right) = 0 \)

answers \( 1 - U_{c(n)}^{(k+1)} \)

\[ = \frac{1}{\beta(n)} \sum_k \left( \text{prob} / D \left( H_n^{k_0} \right) = 1 \land \left( H_n^{k_0} \right)^{(k_0)} = X_n^{k_0} \right) \]

\[ + \text{prob} / D \left( H_n^{k_0} \right) = 0 \land \left( H_n^{k_0} \right)^{(k_0)} = X_n^{k_0} \]
The odds would be:

\[ \overline{H}_{n+1}^k = X_n^{(k)} \cdots X_n^{(k+1)} \overline{X}_n^{(k+1)} U_{e_n}^{(k+1)} \overline{U}_{e_n}^{(k+1)} . \]

\[
\text{prob } (D(\overline{H}_{n+1}^k) = 1) \]

\[ = \frac{1}{2} \left( \text{prob } (D(\overline{H}_{n+1}^k) = 1) + \text{prob } (D(\overline{H}_{n+1}^k) = 1) \right) \]

Turning this upside down:

\[
\text{prob } (D(\overline{H}_{n+1}^k) = 1) = 2 \text{prob } (D(H_n^k) = 1) - \text{prob } (D(H_n^k) = 1) \]

Thus

\[
\text{succ}_a = \frac{1}{2} \sum \text{prob } (D(H_n^k) = 1) \overline{U}^{(k+1)} + \text{prob } (D(H_n^k) = 0) U \neq X_n^{(k+1)} \]

\[ = \frac{1}{2} \sum \left( \text{prob } (D(H_n^k) = 1) \overline{U}^{(k+1)} \right) + 1 - \text{prob } (D(H_n^k) = 1) \]

\[ = \frac{1}{2} \sum \left( \text{prob } (D(H_n^k) = 1) \overline{U}^{(k+1)} + 1 - 2 \text{prob } (D(H_n^k) = 1) \right) \]

\[ = \frac{1}{2} + \frac{1}{2} \sum \left( \text{prob } (D(H_n^k) = 1) - \text{prob } (D(H_n^k) = 1) \right) \]

\[ > \frac{1}{2} + \frac{1}{2} \sum (\text{prob } (D(X_n) = 1) - \text{prob } (D(U_{e_n}) = 1)) \]
Tuesday summary

Def pseudo-random
  pseudo-random generator

\( (X_n) \xrightarrow{n} (V_{\ell(n)}) \)
\( \forall c.i. \)
+ poly-time
+ \( \ell(n) > n \).

Thus short becomes long [Hybrids]

Def unpredictable

Then unpredictable \( \iff \) c.i. [Hybrids].

Outlook Wednesday (morning)

Connection to one-way functions

Examples
Proof

$f: \{0,1\}^n \to \{0,1\}^n$

Supposing that $G$ is a PRG with $\ell(n) = 2^n$.

$1 \times 1 = 1 \times 1$

Claim: $f$ is easy to compute.

So: assume $f$ is hard to invert.

Then there is an attacker (inverter) $A$ with non-negligible success, i.e. there is a polynomial $p(n)$ and infinitely many $n$ with

$$\text{prob}(A(f(x), 1^n) \in f^{-1}(f(y))) > \frac{1}{p(n)}$$

Define Algorithm D

**Import:** $x \in \{0,1\}^n$

**Outputs:** 0 or 1

1. Call the attacker to obtain a preimage $(x, y)$ with $G(x) = \alpha$, $f(x, y)$

2. If the attacker fails let $(x, y) \leftarrow \{0^n\}$.

3. If $G(x) = \alpha$ (attacker successful)
   - Then Return 1
   - else Return 0.
\[
\begin{align*}
\text{prob}(\ D(G(V_n)) = 1) \\
&\quad \frac{\binom{n}{k}}{f(k)} (\frac{1}{u_2}, u_3') \\
&= \text{prob}\left( A\left( \frac{f(u_n u'_n)}{u_2} \right) \in f'f(u_n u'_n) \right) \\
&> \frac{1}{p(n)} \quad \text{for infinitely many } n.
\end{align*}
\]

\[
\begin{align*}
\text{prob}(\ D(V_{2n}) = 1) &< \frac{\binom{2n}{2}}{2^{2n}} = \frac{1}{2^{2n} - 2^n}.
\end{align*}
\]

\[
\begin{align*}
\# \{ x \in \mathbb{R}^n \leq 2^n \} &\leq 1 \times 1 \times 1 \times 1 \times 1 \times 1. \\
\# \{ x \in \mathbb{R}^{2n} \} &\leq 2^{2n}.
\end{align*}
\]

\[
\begin{align*}
\text{prob}(\ D(G(V_n)) = 1) - \text{prob}(\ D(V_{2n}) = 1) \\
&> \frac{1}{p(n)} - \frac{1}{2^n} \\
&> \frac{1}{2p(n)}.
\end{align*}
\]

\[\begin{array}{c}
G \sim \text{pg.}
\end{array}\]
Choose \( p, q \) \( n \)-bit primes, let \( N = p \cdot q \), \( L = (p-1) \cdot (q-1) \), \( e < L \) coprime \( \phi(L) \).

Then

\[
f(x) = x^e \pmod{N}.
\]

Consider \( b(x) = \text{least significant bit of } x \)
\[
= \text{bit}_0(x)
\]

Claim: If you have an algorithm to compute \( b(x) \) then given \( f(x) \) then you can compute \( x \)!

Sketch: we have \( y = f(x) \).
By assumption we can obtain \( x_0 = b(x) \).

Case \( x_0 = 0 \)
Then \( x = 2 \cdot x' \)
Thus \( y = x^e = (2x')^e = 2^e \cdot x'^e \) for some \( e \)
and so \( y' := x'^e = y / 2^e \).
Now ask \( b(x') \).
But that is \( \text{bit} 1 \) of \( x \)!

Case \( x_0 = 1 \) Then \( -x = N - x \) is even...
Proof ( \( \exists \) one-way perm \( \Rightarrow \) \( \exists \) prg )

we have: \( f \) is one-way, \( f(x) = 1x, f \) bijective.

\( b \) is a hard-core predicate for \( f \).

Define \( G(s) = \frac{f(s)}{u \ b(s)} \). \( \leq \) bit.

Claim: \( G \) is a prg.

Clearly: \( L(u) = u + 1 > u \).

\( G \) is poly-time.

Now suppose \( G \) is not a prg.

Then \( f \) is predictable, i.e., there exists a graphed \( A \) such that

\[ \text{suc}_{A} = \text{prob} \left( A \left( G(V_n) \right) = \text{next}_A \left( G(V_n) \right) \right) > \frac{1}{p(n)} + \frac{1}{2} \]

for \( \infty \) many \( n \) and some poly \( p \).

Assume that \( A \) predicts not the last bit:

\[ \text{prob} \left( A \left( \frac{s(V_n)}{u_n} \right) = \text{next}_A \left( f(V_n) \right) \right) = \frac{1}{2} \ 2^{1 + \frac{1}{2}} \leq \frac{1}{p(n)} \]
Thus $A$ predicts the last bit:

$$\text{swc}_{q} = \text{prob}\left( A(f(U_{n}), q) = b(U_{n}) \right)$$

$$> \frac{1}{2} + \frac{1}{p(n)}$$

for some $n$ and some $p$.

$b$ is hard-core for $f$. □
(B1) Examples: \( g: 10.13^3 \rightarrow 30.15^6 \)

<table>
<thead>
<tr>
<th>x</th>
<th>000</th>
<th>001100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>001</td>
<td>001110</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>0101101</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>011011</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1010000</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>100101</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>110010</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>110011</td>
</tr>
</tbody>
</table>

(iii) Design a prophet for \( g \).

(iv) Transform it to a good distinguisher.

(v) Use the distinguisher \( D(y) = \mathbb{X}(w^t(y) \leq y) \) to derive a prophet as in the proof of Yao's theorem and compute its expected and its actual advantage.

(B2) (i) Repeat the proof that three-way permutations implies \( 3 \text{ progs} \) for one of the three examples.

(ii) (do) for one-way functions that are \((\geq 1\text{-})\) 10-1 and length-preserving.

(B3) Complete the proof that the least significant bit of \( x \) is hardcore for \( f(x) = x^b \text{ in } \mathbb{Z}_N \).

(i) assuming \( \text{prob}(\text{...}) = 1 \) for the algorithm output b.

(ii) assuming less...

(B4) Make a mind map of the major topics of the course.
(iii) \[ P_1(x_0, x_1, x_2, x_3, x_4, x_5) = x_0 \oplus x_1 \oplus x_2 \]

\[ \text{prob}(P_4(g(U_3))) = \text{next}_2(U_3)) = 1 \]

- Look at each of the 8 cases.

\[ P_2(\alpha) = \begin{cases} \text{Wait for a run of length } 2, \\ \text{predict the opposite:} & j = \min \{ i \mid \alpha_i = \alpha_{i+1} \} \\ \text{if } j < \infty \text{ then Return } \overline{\alpha_j} \\ \text{else Return a random bit.} \end{cases} \]

\[ \text{prob}(P_2(g(U_3))) = \text{next}_2(U_3)) = \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{2} + 0 + \frac{1}{2} = \frac{13}{16} \]

(iv) \[ D_1(\alpha) = \chi(P_1(\alpha) = x_3) \]

\[ D_2(\alpha) = \chi(P_2(\alpha) = \text{next}_2(\alpha)) \]

\[ \text{adv}_{D_1} = 1 - \frac{1}{2} = \frac{1}{2} \]

\[ \text{adv}_{D_2} = \frac{13}{16} - \frac{3}{64} = \frac{52 - 32}{64} = \frac{20}{64} = \frac{5}{16} \]

(v) Algorithm \( P_4 \)

- **Input:** \( \alpha \in \{0,1\}^6 \)
- **Output:** \( 0 \) or \( 1 \).

1. Choose \( k \in \{0, \ldots, 5\} \).
2. Choose \( \beta_1, \ldots, \beta_5 \in \{0,1\} \).
3. If \( \text{wt}(\alpha_0, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_5) \leq 4 \)
   then Return \( \overline{\beta_{k+1}} \)
   else Return \( \beta_{k+1} \).
\[ \text{prob} \left( P_4 \left( g(U_3) \right) = \text{next}_{P_4} \left( g(U_2) \right) \right) \leq \frac{1}{2} + \frac{1}{6} \cdot \frac{7}{64} = \frac{199}{384} \approx 0.518 \ldots \]
one-way functions
hard-core predicate 1 bit
3prg \implies \text{one-way fn.}

\text{Example: } \text{RSA one-way? A permutation, with specific hard-core bit.}
\text{Example: } \text{Blum, Blum, Shub (perm Based)}

\text{One-way permutation } \implies \exists \text{prg.}
\text{OPEN QUESTION: } \text{One-way fn. } \implies \text{One-way perm.}
\text{DEEP Thm: } \text{One-way } \implies \exists \text{prg.}

Protocol \text{ SUDOKU}

\rightarrow \text{Round 1: Provers and verifiers were honest}
\rightarrow \text{Round 2: Verifier honest, but provers arbitrary}
\rightarrow \text{Round 3: Provers honest, but verifier arbitrary}
\rightarrow \text{Verifier builds its simulator.}
\[ BPP \subseteq \mathsf{IP} \]

Say \( \Pi \) is a \( \mathsf{BPP} \)-machine accepting \( L \in \mathsf{BPP} \).

Then

\[
\begin{array}{c}
\text{Paula} \overset{x}{\rightarrow} \text{Victor} \\
\text{\text{-}} \quad \Pi(x)
\end{array}
\]

\[ \mathsf{NP} \subseteq \mathsf{IP} \] say \( L \in \mathsf{NP} \).

Then there exists a relation \( R \subseteq \mathsf{P} \)

consisting of pair \((x, w)\) where

\[ \text{\text{\text{\text{\text{\text{\text{$\\mathbf{w}$}}}}} \leq \text{poly}(1|x|) \quad \text{and} \quad x \in L} \]

there is a det. poly-time machine \( \Pi \)

that accepts exactly elements of \( R \).

\[
\begin{array}{c}
\text{Paula} \overset{x}{\rightarrow} \text{Victor} \\
\text{Finds w such that } (x, w) \in R \quad w \rightarrow \text{ Runs } \Pi \text{ to see whether } (x, w) \in R \\
\text{and accepts if so}
\end{array}
\]

Let \( x \in L \):

\[
\begin{align*}
\text{prob} ( \langle P, V \rangle (x) = 1 ) &= 1 \\
\text{prob} ( \langle P, V \rangle (x) \neq 1 ) &= 0
\end{align*}
\]

Perfect completeness.

Let \( x \notin L \):

\[
\begin{align*}
\text{prob} ( \langle B, V \rangle (x) \neq 0 ) &= 0
\end{align*}
\]

Perfect soundness.
Proof (Protocol for Graph isomorphism is perfect zero-knowledge)

Given \( P \), find a way to verify \( \mathcal{V}^* \) and \( \mathcal{V} \).

What should the simulator \( \mathcal{M} \) do?

Look at the protocol:

First idea for \( \mathcal{M} \):
- Select \( s = 0 \), \( \psi \) a permutation of \( V_{G_0} \)
- Create \( G' = \psi G_0 \)

Now, \( \text{prob}\left( \mathcal{M}(x) = (G', G, \psi) \right) \)

\[ = \begin{cases} \text{prob}\left( \mathcal{M}(x) = (G', 0, \psi) \right) & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases} \]

Better:
- \( \text{prob}\left( \mathcal{M}(x) = (G', 1, \psi) \right) = 0 \)
- \( \text{prob}\left( \langle P, \mathcal{V} \rangle(x) = (G', 1, \psi) \right) \geq 0 \) if \( G' = \psi G_0 \).

Second trial:

**Algorithm \( \mathcal{M} \)**

**Input:** \( G_0, G \), two (isomorphic) graphs.

**Output:** view of a conversation.

1. Choose \( s \in \{0, 1\} \),
2. Choose \( \psi \in \mathcal{R} : V_{G_0} \rightarrow V_{G_0} \) randomly,
3. Let \( G' = \psi G_0 \). Then \( \psi : G_0 \rightarrow G' \) is iso.
4. Output \( (G', G, \psi) \).

Now, \( \text{prob}\left( \mathcal{M}(G_0, G_1) = (G', G, \psi) \right) = \text{prob}\left( \langle P, \mathcal{V} \rangle(x) = (G', 1, \psi) \right) \geq 0 \).
Now:  \( \text{prob} ( H(6',6) = (6,6,\psi) ) \)
\[ = \text{prob} ( \langle P,V \rangle (6,6') = (6',6,\psi) ) =: pv. \]

\[ m = \frac{1}{2} \frac{1}{(2,4)!} \quad \text{say} \quad V_0 = V_x = 142 < n. \]

\[ = \frac{1}{2n!} \]

whenever \((6',6,\psi)\)
is a legal protocol,

\[ pv = \frac{1}{n!} \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{2n!} \]
i.e. \( \psi : G_0 \rightarrow G' \text{ iso.} \)

For short: our protocol is - what we call -
perfect honest-verifier zero-knowledge.

General case? We have to deal with \( V^\star \).

Algorithm \( V^\star \)

| Input: \( G_0, G' \) two (isomorphic) graphs. |
| Output: \( \$ \) possible output of \( V^\star \). |
1. Choose \( z \in \mathbb{G}_s \) as a guess for \( \psi^\star_k \).
2. Choose \( \psi : V_x \rightarrow V_z \) randomly.
3. Let \( G' = \psi G_z \) so that \( \psi : G_z \rightarrow G' \text{ iso.} \).
4. Ask \( V^\star(G') \rightarrow G \).
5. If \( z \neq G \) Return \(-1\).
6. Return \( V^\star(G',6',\psi) \).

Details: \( \exists \psi \)?
Prove Proposition 3.9.

As a starter, amplify the interactive proof for GRAPH NONISOMORPHISM to soundness error.

(i) \(1/3\)

(ii) \(\frac{1}{\log n}\) where \(n = \# \text{ vertices in the graphs } G_0, G_1, G_2\).

(iii) Do for the full proof of Proposition 3.9.

(Found)

(C2) Find a zero-knowledge proof for SQUARENESS, assuming that factoring is difficult.

\[
L = \{ (N, z) \mid N = p \cdot q, \quad p, q = \text{prime}, \quad z = x^2 \mod N \text{ for some } x \}\]

Hint: the prover first tells the verifier a square \(b = a^2\).

(C3) Find a (perfect) zero-knowledge proof for \(\text{ZEROSQUARENESS}\), assuming ...

\[
L = \{ (N, z) \mid N = p \cdot q, \quad p, q = \text{prime}, \quad z \neq x^2 \mod N \text{ for any } x \}\]

---

(C2 partial answer)

\(N, z\)

\(P\)

\(\begin{align*}
\text{Choose } a &\in \mathbb{Z}_N^\times, \\
\text{Compute } b &= a^2,
\end{align*}\)

\(\quad \leftarrow b\)

\(\text{Compute } w &= x^5 a, \\
\quad \leftarrow w\)

\(\quad \rightarrow \text{Check } w^2 = z^5 b\)

and accept if true.
Thursday summary

- Interactive TTT & joint computation
- Interactive proofs (generalized)
  - Completeness error
  - Soundness error
  - $\text{BPP} \subseteq \text{NP} \subset \text{JP}$

\[
\begin{array}{c}
\text{Simulator}(x') \\
\text{BPP}(x) \\
\end{array}
\quad
\begin{array}{c}
\text{NP-Check}(x,w)
\end{array}
\]

- $\text{GRAPH-NON-ISOMORPHISM}$
  - Interactive proof ($ce = 0$, $se = \frac{1}{2}$)
  - Public-coins vs. private coins ($AM$)
  - Perfect zero-knowledge by simulation
  - Statistical
  - (Computational)

- $\text{GRAPH-ISOMORPHISM}$
  - $2k$ proof ($ce = 0$, $se = \frac{1}{2}$)

---

Quality of the simulator:

Knowledge Tightness: $P_2$

\[ t_{M^*}(x) - P_1(\|x\|) < P_2(1\|x\|) \]

\[ t_{V^*}(x) \leq P_1 \]

For some polynomial $P_1$. 
$P \leq BPP \leq \text{SZK} \leq \text{EL} \leq \text{IP} \subseteq \text{PSPACE}$

Believe: *? * ? * = * = *

Simplify.

Need auxiliary input to model that information is passed from one phase to the next phase during a multiple execution of a protocol.

Suppose we consider a $Q$-fold repetition of a pair $(P, V)$ and finally the new verifier will accept if in a $\alpha$-fraction of the $Q$ phases the original verifier accepted:

$$< P_Q, V_Q^{+\alpha} >$$

Wen completeness error?

$x \in L$:

$$\text{prob} \left( < P_Q, V_Q^{+\alpha} > (x) \neq 1 \right)$$

$$= \text{prob} \left( \exists s \in \{0, \ldots, Q-1\} : \forall i \in S : < P, V > (x) = 1 \right)$$

$$= \sum_{S} \prod_{i \in S} \text{prob} \left( < P, V > (x) = 1 \right) \prod_{i \notin S} \text{prob} \left( < P, V > (x) \neq 1 \right)$$
\[ \sum_{s = 0}^{x \xi Q} (\xi) (1 - ce)^s \cdot ce^{Q-\xi} \cdot \xi^{Q-\xi} \]

\[ \exp\left( -2 \cdot Q \cdot (1 - ce - \alpha)^2 \right) \]

Use Hoeffding inequality

New soundness error?

\[ \text{prob} \left( \langle P_\xi, V_\xi^{+\xi} \rangle (x) \neq 0 \right) \]

\[ \leq \exp\left( -2Q \cdot (\alpha - se)^2 \right) \]

Must have 2 both to be small:

\[ 1 - ce - \alpha > \frac{1}{P_\xi(n)} \]

\[ \alpha - se > \frac{1}{P_\xi(n)} \]

This leads to a proof of Prop. 3.9

that we can amplify interactive proofs.

But what about zero-knowledge?
Then the protocol for G3C is (perfect) zero-knowledge interactive proof (if boxes exist).

**Completeness:**
\[ \text{prob}\left( \langle p, v \rangle(x) \neq 1 \right) = 0 \]
So we even have perfect completeness.

**Soundness:**
Given any cheating prover B consider a not 3-colorable graph on \( n \) vertices.

\[ \text{prob}\left( \langle b, v \rangle(x) \neq 0 \right) = 1 - \text{prob}\left( \langle b, v \rangle(x) = 0 \right) \]
\[ \leq 1 - \frac{4}{n^2} \leq 1 - \frac{1}{n^2} \]
Repeating \( n^2 \) times brings it down to
\[ (1 - \frac{1}{n^2})^{n^2} \approx e^{-1} < \frac{1}{2} \]

**(Perfect) Zero-Knowledge:**
Simulator M*:

- Guess an edge \( (u', v') \)
- Choose keys and boxes \( C_u \) with \( c_u \rightarrow \)
- and \( C_v \) with \( c_v \)
- and choose \( (a, b, c) \) boxes \( C_v \) with 0 for \( v \in V \setminus \{u, v', \} \)
- Ask if \( V^*(\langle C_u \rangle, \langle u', v' \rangle) \neq \langle u, v \rangle \) then retry (at most poly-times)
- Return \( V^*(\langle C_u \rangle, \langle u, v \rangle, \langle 0, c_u \rangle, \langle 0, c_v \rangle) \)
we would have to prove 3-colorable graph

\[ \Pr( M^*(x, \pi) = \alpha ) \]

aux. input to \( V^* \)

\[ = \Pr( \langle P \rangle, V^*(x) > (x) = \alpha ) \]

\( \checkmark \) case the boxes are perfect...

Sheekh (\( f \) injective one-way \( \Rightarrow \) bid comminhd)

\( \checkmark \) everying is randomized poly-time.
\( \checkmark \) perfectly binding: need only injectivity here!

Receiver sees \( \langle f(s), b(s) \oplus v \rangle \).

\( f(s) \) determines \( s \) (though that cannot

Thus \( b(s) \) is determined (be found fast)

and so \( v \) is determined.

\( \checkmark \) computationally hiding:

Distinguishing \( \langle f(s), b(s) \rangle \)

from \( \langle f(s), b(s) \oplus 1 \rangle \)

means predicting \( b(s) \) from \( f(s) \).

That cannot be done efficient

because \( b \) is hard-core.
Proof of completeness: \( c.e. = 0 \)

Soundness: Suppose \( G \) is not 3-colorable.

\[
\text{prob}\left( B, V > (G) = 1(\neq 0) \right) \leq 1 - \frac{4}{nE < 1 - \frac{4}{n^2}}.
\]

Perfect binding!

Zero knowledge

Note that the simulator either

- can find a 3-coloring
- or the commitments encode a non-3-coloring, which in a actual transcript with the honest prover would never happen.

So assuming that 63C is difficult

the simulator not always outputs wrong transcript.

Yet, we only need that they are indistinguishable from real ones.

We will need hybrids to break down distinguishing simulation from conversation to a bit commitment to 0 from a bit commitment to 1.
First: Assume L E NP

Then there exists a poly-time computable function f such that

\[ f(x) \in G3C \iff x \in L. \]

Let's try to set up a ZK proof for L.

\[ \begin{array}{c}
\text{P} \\
\xrightarrow{\text{w}} \\
\downarrow \\
\text{V} \\
\xrightarrow{\text{x}} \\
\text{f}(x)
\end{array} \]

Claim: all standard transformations

(say from \( L \) to \( \text{SAT} \), \( \text{3SAT} \), \( \text{G3C} \))

allow the required witness transform \( g \).

Poly-time completeness: even perfect completeness.

Soundness: soundness error \( \leq \varepsilon \) \((G3C)\)\( \leq \varepsilon \)

Zero knowledge: simulator is just as before + first calculate \( f(x) \) from x.

But a distinguisher would tell us only that \( \langle P, (x, w) \rangle \) and \( \textbf{1}_L (w) \) are different, yet we need that it see a difference between \( \langle g(x, w), \text{simulated} \rangle \).