Recall the Goldreich-Levin theorem. It says that for any OWF $f(x)$, a random parity of the input bits of $x$ is a hardcore bit of $f$. More formally, for $r = r_1 \ldots r_n \in \{0,1\}^n$ and $x = x_1 \ldots x_n \in \{0,1\}^n$, we define their inner product modulo 2 by $r \cdot x = r_1 x_1 + \ldots + r_n x_n \mod 2$. Now, the Goldreich-Levin theorem says that if $x$ and $r$ are chosen randomly from $\{0,1\}^n$, then no PPT adversary $A$ can guess $x \cdot r$ with probability better than $\frac{1}{2} + \text{negl}(n)$, when given $r$ and $f(x)$. Put yet differently, $x \cdot r$ is the hardcore bit of OWF $f'(x, r) = \langle f(x), r \rangle$. On the other hand, the examples of OWF’s we considered so far (e.g., RSA or modular exponentiation) had much simpler hardcore bits. Namely, one of even most of their input bits $x_i$ were hardcore. One may wonder if the Goldreich-Levin theorem could be improved to say: “for any OWF $f$ there exists an input position $i$ such that $x_i$ is a hardcore bit for $f$”. In this note we show that this hope is false in general: there are OWF’s each of whose input bits $x_i$ is easy to predict individually. Hence, Goldreich-Levin theorem says the next best thing: most of the parities of the input bits of $x$ are hardcore bits for any $f$ (even though no one such parity works for all the $f$’s).

We start with an easy construction of $f$ where each bit can be guessed with probability $3/4$. We then show how to extend it to probability $1 - 1/N^{1-\varepsilon}$, for any $\varepsilon > 0$ (here $N$ is the input length of the OWF we construct). The latter construction is more advanced and maybe skipped.

1 Prediction with probability $\frac{3}{4}$

Let $f$ be any OWF (assume it is length-preserving to slightly simplify the notation). Using $f$, we will explicitly construct a one-way function $g$ for all inputs $x$ whose number of bits is a multiple of $3^1$. Consider the following functions $c$ and $d$, mapping $\{0,1\}^3$ to $\{0,1\}$ and $\{0,1\}^2$, respectively:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$c(x)$</th>
<th>$d(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>

The following observations should be clear from the definition of $c$ and $d$:

1. $c$, $d$ are defined on all 3-bit sequences.
2. $c$ and $d$ are uncorrelated, i.e. knowing $c(x)$ does not help in determining $d(x)$ if $x$ is unknown.

\footnote{If $x$ is of length $3n + k$, where $k \in \{1, 2\}$, then we encode the first $3n$ bits using $g$, and append the $k$ remaining bits ‘as is’. It is clear that the function remains one-way.}
3. If we know $c(x)$ then every bit of $x$ can be predicted with probability $\frac{3}{4}$, in fact every bit of $x$ is equal to $c(x)$ with probability $\frac{3}{4}$.

We define $g$ as follows (where $\circ$ corresponds to concatenation):

\begin{align*}
    c_i(x_1x_2 \ldots x_{3n}) &:= c(x_{2i-2}x_{2i-1}x_{3i}) \quad \text{for } i \in \{1, \ldots, n\} \\
    d_i(x_1x_2 \ldots x_{3n}) &:= d(x_{2i-2}x_{2i-1}x_{3i}) \quad \text{for } i \in \{1, \ldots, n\} \\
    g(x) &:= c_1(x) \circ c_2(x) \circ \cdots \circ c_n(x) \circ f \circ d_1(x) \circ d_2(x) \circ \cdots \circ d_n(x), \text{ where } |x| = 3n
\end{align*}

It is clear that $g$ is length preserving if $f$ is. Also, given $g(x)$ we can predict every bit $j \in \{1, \ldots, 3n\}$ of $x$ with probability $\frac{3}{4}$ by simply outputting $c_i_{i/3}(x)$.

It remains to be shown that $g$ is one-way. Assume $g$ were not one-way, and let $A$ be an ‘inverter’ for $g$, i.e. a PPT satisfying for some constant $c$:

$$\Pr[A(g(x)) = x \mid x \leftarrow \{0,1\}^{3n}] \geq \frac{1}{nk}$$

Then the following PPT $B$ inverts $f$ with a non-negligible probability:

1. Given input $y \in \{0,1\}^{2n}$, choose a random $c \in \{0,1\}^n$
2. Let $x := A(c \circ y)$
3. Output $z = d_1(x) \circ d_2(x) \circ \cdots \circ d_n(x)$

The first line makes sure that as $y$ ranges over $g(z)$ for $z \in \{0,1\}^{2n}$, $c \circ y$ ranges over $g(x)$ for $x \in \{0,1\}^{3n}$. This follows from the fact that $c$ and $d$ are independent. Thus in line 2, $A$ with non-negligible probability succeeds in inverting $g$, and then $z \in f^{-1}(y)$ is output (as $y = f(z)$ from the definition of $g$). So $B$ with non-negligible probability inverts $f$, which was supposed to be one-way. The contradiction proves that $g$ is one-way. $\square$

2 Extension to probability $1 - \frac{1}{N^{1-\varepsilon}}$, for any $\varepsilon > 0$

We can improve our construction of $g(x)$ such that for any $\varepsilon > 0$ every bit of an $x$ of length $N$ can be predicted with probability $1 - \frac{1}{N^{1-\varepsilon}}$, and $g$ is still one-way. This is the more or less the strongest possible negative result we can expect, since we can never predict every bit of an input to a OWF with probability any better than $1 - \frac{1}{N} + \text{negl}(N)$. Indeed, if every bit is predicted incorrectly with probability at most $\frac{1}{N} - \frac{1}{\text{poly}(N)}$, then by the union bound we predict at least one of $N$ bits incorrectly with probability at most $1 - \frac{N}{\text{poly}(N)}$, i.e. we correctly invert the function with non-negligible probability at least $\frac{N}{\text{poly}(N)}$, a contradiction. We make use of (binary) Hamming codes (which are examples of perfect codes) which we define next.

Brief Detour to Hamming Codes

The $m$-dimensional hypercube is simply the set $\{0,1\}^m$ where the distance between 2 points $a_1 \ldots a_m$ and $b_1 \ldots b_m$, $\text{dist}(a,b)$, is the number of positions they differ in. This distance is called a hamming
(or \( \ell_1 \)) distance. This distance defines the notion of a “ball” inside the hypercube. A ball of radius \( r \geq 0 \) and center \( a \in \{0,1\}^m \) is defined to be the set of points of distance at most \( r \) from \( a \),

\[
B_m(a,r) := \{ y \mid \text{dist}(a,y) \leq r \}
\]

A (binary) perfect code of length \( m \) and distance \( 2e + 1 \) is a way to disjointly partition the \( m \)-dimensional hypercube \( \{0,1\}^m \) into balls of radius \( e \). In other words, it is a subset \( \mathcal{C} \subseteq \{0,1\}^m \) such that \( B_m(a,e) \cap B_m(b,e) = \emptyset \), for any distinct \( a, b \in \mathcal{C} \) while \( \bigcup_{a \in \mathcal{C}} B_m(a,e) = \{0,1\}^m \).

Very few perfect codes exist. One of the classical examples is the (binary) Hamming code, which corresponds to \( e = 1 \), i.e. we disjointly partition \( \{0,1\}^m \) into balls of radius 1 (let’s denote \( B_m(a,1) \) simply by \( B_m(a) \)). Since each radius 1 ball has \((m + 1)\) points, we must have that \((m + 1)\) divides \( 2^m \), i.e. \( m = 2^k - 1 \) for some \( k > 0 \). In this case \( |\mathcal{C}| = 2^{m-k} \). The converse is also easy to establish and can be found in any book on combinatorics and coding theory.

**Theorem 1** Hamming code \( H_k \) exists for any \( m = 2^k - 1 \).

**Continuation of the Proof**

Let’s go back to the proof. Given a perfect code \( \mathcal{C} = H_k \) for string length \( m = 2^k - 1 \), we construct functions \( c(x) \) and \( d(x) \) for \( x \in \{0,1\}^m \) as follows. Let \( y(x) \) be the unique codeword from \( \mathcal{C} \) such that \( x \in B_m(y(x)) \). Then \( c(x) \) is the position of \( y(x) \) in the lexicographical ordering of the code \( \mathcal{C} \), and \( d(x) \) is the position of \( x \) in the lexicographical ordering of the unit sphere \( B_m(y(x)) \). Hence, \( |c(x)| = m - k = 2^k - k - 1, |d(x)| = k \). We remark that encoding/decoding of the Hamming code can be done very efficiently, so \( y(x), c(x) \) and \( d(x) \) are efficiently computable.

In other words, the knowledge of \( c(x) \) is equivalent to the knowledge of \( y(x) \), i.e. that \( x \in B_m(y(x)) \), but we do not know where in \( B_m(y(x)) \) our \( x \) resides. The knowledge of \( d(x) \) tells us where \( x \) lies inside its sphere \( B_m(y(x)) \) but does not tell which is the right sphere. Clearly, \( c(x) \) and \( d(x) \) are independent as before. In fact, we see that this exactly generalizes the previous construction with \( m = 3, k = 2, \mathcal{C} = \{000, 111\} \). We now define \( g \) the same way as before for inputs of length \( mn \).

\[
\begin{align*}
[x]_i &:= x_{(i-1)+1} \circ x_{(i-1)+2} \circ \ldots \circ x_m \\
c_i(x) &:= c([x]_i) \quad \text{for } i \in \{1, \ldots, n\} \\
d_i(x) &:= d([x]_i) \quad \text{for } i \in \{1, \ldots, n\} \\
g(x) &:= c_1(x) \circ c_2(x) \circ \ldots \circ c_n(x) \circ f(d_1(x) \circ d_2(x) \circ \ldots \circ d_n(x)) , \text{ where } |x| = mn
\end{align*}
\]

It is easy to see that every bit of \( j \in \{1, \ldots, mn\} \) of \( x \) can be guessed correctly with probability \( 1 - \frac{1}{m+1} = 1 - \frac{1}{2^k} \) (recall, \( m = 2^k - 1 \)). We simply look at \( c_{i,j/m-1} \) \( (x) \), which tells us the center \( a \) of the sphere containing \( [x]_{i,j/m-1} \), and output the \( t := (1 + j \text{ mod } m)^{th} \) bit of \( a \). Since our balls are of radius 1, exactly 1 out of \( m+1 \) points of the sphere around \( a \) will have bit \( t \) different from \( t'-th \) bit of the center \( a \). In other words, the knowledge of the ball (which comes from \( c \)) even without the knowledge of the point inside the ball, still allows us to predict any particular coordinate of this (unknown) point with probability \( 1 - \frac{1}{m+1} \); simply take the corresponding coordinate of the (known) center.

Let \( N = mn = (2^k - 1)n \) be the size of \( x \). Using the independence of \( c \) and \( d \), essentially the same argument as before shows that \( g \) is one-way as long as \( f \) is applied to an input of size at

1-3
least $N^\varepsilon$. This is because to obtain a contradiction with one-wayness of $f$, we need to apply it to inputs of length non-negligible in the length of $x$. We apply $f$ to inputs of length $nk = \frac{N^{1-\varepsilon}}{k}$, since $\|d(\cdot)\| = k$. Hence, it suffices to have

\[ N^\varepsilon \leq N \frac{k}{2^k} - 1 \Rightarrow \frac{2^k - 1}{k} \leq N^{1-\varepsilon} \Rightarrow k \approx (1 - \varepsilon) \log N \]

With this $k$, we get error probability $\frac{1}{2^k} = \frac{1}{N^{1-\varepsilon}}$, for any $\varepsilon > 0$, as needed. \qed