Multisymbol Majority Vote and Hard Core

Wakaha OGATA[1], Keiichi SAKANO[2] and Kaoru KUROSAWA[3]

[1] Himeji Institute of Technology,
2167, Shosha, Himeji-shi, Hyougo, 671-22, Japan
[2] Matsushita Electric Industrial Co., LTD.
1006, Kadoma, Kadoma-shi, Osaka, 571, Japan
[3] Tokyo Institute of Technology
2–12–1 O-okayama, Meguro-ku, Tokyo 152, Japan

Key words: safety/security in digital systems, one-way function, computational complexity

Abstract

This paper describes a weak sense majority-vote technique for a $k-$symbol function $g(z)$ as follows. If a probabilistic polynomial time algorithm A exists such that $\Pr[A(z) = g(z)] \geq 1/k + \epsilon$, then $s \stackrel{\triangle}{=} 2^{k-1} - 1$ probabilistic polynomial time algorithms $\{\hat{A}_1, \ldots, \hat{A}_s\}$ exist such that for $\forall z \in \{0, \ldots, k-1\}^*$, there exists $\hat{A}_i \in \{\hat{A}_1, \ldots, \hat{A}_s\}$ such that $\Pr[\hat{A}_i(z) = g(z)] \geq 1 - |z|^{-c}$, where c is a constant. We apply our technique to obtain a hard core of $k-$symbol one way funcitons.

1 Introduction

In the field of complexity theory, a "$k-$valued" function is defined as a mapping $\phi : \{0,1\}^* \rightarrow \{0,1\}^*$ such that for every x, $\phi(x)$ is a set of j elements, where $1 \leq j \leq k$ [1]. We consider a "$k-$symbol" function such that $g : \{0, \ldots, k-1\}^* \rightarrow \{0, \ldots, k-1\}^*$, where $g(z)$ is not a set, but an element. In BPP, the majority-vote technique plays an important role in increasing the success probability. This technique, however, cannot be used for multisymbol functions. For example, for a 3-symbol function $g(z)$, suppose that a probabilistic polynomial time algorithm A exists such that $\Pr[A(z) = g(z)] \geq 1/3 + \epsilon$. Then it is not possible to increase the success

probability by using the majority-vote technique. Because if we let $p_i \stackrel{\triangle}{=} \Pr[A(z) = g(z) + i \bmod 3]$, it is possible that $p_0 = p_1 > p_2$ in which case p_0 and p_1 are equal. In this case, the majority-vote technique does not work and the success probability cannot be increased.

This paper describes a weak sense majority-vote technique for a $k-$symbol function $g(z)$ as follows. If a probabilistic polynomial time algorithm A exists such that $\Pr[A(z) = g(z)] \geq 1/k + \epsilon$, then $s \stackrel{\triangle}{=} 2^{k-1} - 1$ probabilistic polynomial time algorithms $\{\hat{A}_1, \ldots, \hat{A}_s\}$ exist such that for $\forall z \in \{0, \ldots, k-1\}^*$, there exists $\hat{A}_i \in \{\hat{A}_1, \ldots, \hat{A}_s\}$ such that $\Pr[\hat{A}_i(z) = g(z)] \geq 1 - |z|^{-c}$, where c is a constant. We say that this is a weak sense technique because the guessing algorithm is not unique for all inputs. We apply our technique to obtain a hard core of $k-$symbol one way funcitons. Hard core predicates [2, 3, 4] are fundamental tools in modern cryptology such as pseudorandom generators [2, 4], secure probabilistic encryptions [5] and etc. A predicate $b(x)$, which takes a value in $\{0, 1\}$, is called a hard core of a one-way function $f(x)$ if $b(x)$ is easily evaluated on input x, but hard to be guessed (substantially better than 50–50) given only the value of $f(x)$. Recently, Goldreich and Levin [6] showed that any one-way function, padded to the form $f(x) \circ p$, has a hard core predicate $B(x, p)$, where $\circ$ denotes concatenation. Rackoff[7] greatly simplified Goldreich's and Levin's proof [6]. We combine Rackoff's proof with our result to obtain a $k-$symbol hard core.

2 Preliminaries

2.1 Notation

Concatination is denoted $\circ$. For a k-symbol string $x = (x_1, \ldots, x_n)$ such that $x_i \in \{0, 1, \ldots, k-1\}$, we write $|x|_k = n$. For two k-symbol strings $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$,

$$x \odot y \stackrel{\triangle}{=} \sum_{i=1}^{n} x_i \cdot y_i \bmod k.$$

$e^j = (0, \ldots, 0, \underset{j}{1}, 0, \ldots, 0)$. k will be omitted if it is clear from the context.

2.2 Hard Core Predicate

Definition 1 *[6] [Hard core predicate] Let f be a one-way function and H be a function of $\{0,1\}^* \to \{0,1\}$. We call H a hard core predicate (HC) of*

f if for every probabilistic polynomial time algorithm $A(\cdot)$ and for every c, there exists N_c such that for all $n > N_c$,

$$Pr_{x,A}[A(f(x)) = H(x)] \ < \ 1/2 + n^{-c},$$

where the probability is taken over $x \in \{0,1\}^n$ and the coin tosses of A.

Proposition 1 *[6] Let f be a one-way function such that $|f(x)| = |x|$. Define f' and H as*

$$f'(w) \stackrel{\triangle}{=} f(x) \circ r, \quad H(w) \stackrel{\triangle}{=} x \odot r,$$

where $w = x \circ r$ and $|x| = |r|$. Then, H is a hard core predicate of f'.

2.3 Rackoff's Proof of Proposition 1 [7]

Assume that H is *not* a hard core predicate of f', i.e. there exist a probabiistic polynomial time algorithm A and constants c_1, N_{c_1} such that for infinitely many $n' > N_{c_1}$,

$$Pr[A(f(x) \circ r) = H(x \circ r)] \ \geq \ 1/2 + (n')^{-c_1} \tag{1}$$

Let x_j be the j-th bit of x. The following algorithm I_j guesses x_j by using $H(x \circ r)$ as an oracle. Let $m \stackrel{\triangle}{=} \lceil \log\{n^{c_2+1}(2^{c_1+1}n^{c_1})^2/4\} \rceil$ for some constant c_2.

[**Algorithm** I_j] Input: $y(= f(x))$. Let $n = |y|$.

(step 1) For $0 \leq i \leq m$, choose $R_i \in \{0,1\}^n$ randomly. By using the oracle, obtain

$$h_i = H(x \circ R_i) \quad \text{for } i = 0, \ldots, m.$$

(step 2) For each $b = (b_1, b_2, \ldots, b_m) \in \{0,1\}^m$, compute

$$\begin{aligned} r_b &= R_0 + (b_1 \times R_1) + \cdots + (b_m \times R_m) \bmod 2 \\ x_b &= A(f(x) \circ (e^j + r_b)) - h_0 - (b_1 \times h_1) - \cdots - (b_m \times h_m) \bmod 2 \end{aligned} \tag{2}$$

(step 3) Let $\hat{x}_j$ be the majority value of $\{x_{0\cdots0}, \ldots, x_{1\cdots1}\}$. Output $\hat{x}_j$.

Let X_b be the random variable induced by x_b.

Claim 2.1 $x_b = A(f(x) \circ (e^j + r_b)) - x \odot r_b \bmod 2$.

Proof: Substitute $h_i = x \odot R_i$ into eq.(2). ¶

Claim 2.2 $\Pr[X_b = x_j] = \Pr[A(f(x) \circ r) = H(x \circ r)]$.

Proof: If $x_b = x_j$,

$$\begin{aligned} A(f(x) \circ (e^j + r_b)) &= x_b + x \odot r_b = x_j + x \odot r_b = x \odot e^j + x \odot r_b \\ &= x \odot (e^j + r_b) = H(x \circ (e^j + r_b)). \end{aligned} \quad (3)$$

Conversely, it is easy to see that $x_b = x_j$ if $A(f(x) \circ r) = H(x \circ r)$. Furthermore, note that $e^j + r_b \bmod 2$ is random. Thus, we have this claim. ¶

Claim 2.3 $\{X_b\}$ *are pairwise independent.*

Proof: For simplicity, suppose that $b = (0, b_2, \ldots, b_n)$ and $b' = (1, b_2, \ldots, b_n)$. Then, $r_{b'} = r_b + R_1$ and

$$x_{b'} = A(f(x) \circ (e^j + r_b + R_1)) - x \odot (r_b + R_1) \bmod 2$$

from claim 2.1. In this equation, R_1 is random and independent of r_b. Also, the coin tosses of A are independent of these for x_b. Therefore, $x_{b'}$ is independent of

$$x_b = A(f(x) \circ (e^j + r_b)) - x \odot r_b \bmod 2 \qquad ¶$$

Claims 2.2, 2.3 and eq.(1) imply that $\Pr[I_j(f(x)) = x_j]$ is very large. (The majority vote technique is used here.) By letting $(h_0, \ldots, h_m)$ take $(0, \ldots, 0) \sim (1, \ldots, 1)$, we can obtain a probabilistic polynomial time algorithm which inverts $f(x)$ with sufficiently large probability instead of using the oracle.

3 Multisymbol Majority-Vote Technique

This section describes a weak sense majority vote technique for a $k-$symbol function $g(z)$ as follows. If a probabilistic polynomial time algorithm A exists such that $\Pr[A(z) = g(z)] \geq 1/k + \epsilon$, then $s \stackrel{\triangle}{=} 2^{k-1} - 1$ probabilistic polynomial time algorithms $\{\hat{A}_1, \ldots, \hat{A}_s\}$ exist such that for $\forall z \in \{0, \ldots, k-1\}^*$, there exists $\hat{A}_i \in \{\hat{A}_1, \ldots, \hat{A}_s\}$ such that $\Pr[\hat{A}_i(z) = g(z)] \geq 1 - |z|^{-c}$, where c is a constant.

Definition 2 *We say that $g(z)$ is a k-symbol function if $g : \{0,\ldots,k-1\}^* \to \{0,\ldots,k-1\}$.*

Definition 3 *Let $g(z)$ be a k-symbol function. We say that $g \in k\text{-}BPP$ if there exist a constant $c > 0$ and a probabilistic polynomial time algorithm A such that for every z,*

$$\Pr[A(z) = g(z)] \ \geq \ 1/k + |z|^{-c}, \tag{4}$$

where the probability is taken over the coin tosses of A.

Note that 2-BPP coincides with BPP. Is it possible to increase the success probability ? That is, is there a constant c_1 and a probabilistic polynomial time algorithm $\hat{A}$ such that $\Pr[\hat{A}(z) = g(z)] \ \geq \ 1 - |z|^{-c_1}$? If $k = 2$, the answer is yes because we can use the majority vote technique. For $k \geq 3$, however, the answer is not known because we can no longer use the majority vote technique as shown in the Introduction. Here, we give a weak answer to this problem. Let $K \stackrel{\triangle}{=} \{0,\ldots,k-1\}$.

Theorem 3.1 *Let W be a random variable over K such that*

$$\Pr[W = 0] \geq \frac{1}{k} + \epsilon. \tag{5}$$

For each $B \subset K$, define a random variable W_B as $W_B \stackrel{\triangle}{=} W - i \bmod k$, where i is chosen from B randomly. Then, there exists $\tilde{B} \subset K$ such that

$$\Pr[W_{\tilde{B}} = 0] \geq \Pr[W_{\tilde{B}} = j \bmod k] + \epsilon/(k-1)^2 \quad \textit{for } j \neq 0.$$

Proof: Let $p_i \stackrel{\triangle}{=} \Pr[W = i]$. Write $\{p_0,\ldots,p_{k-1}\}$ in a decreasing order. That is, $p_{i_0} \geq p_{i_1} \geq \cdots \geq p_{i_{k-1}}$. Then, $p_{i_0} \geq 1/k + \epsilon$ and $p_{i_{k-1}} < 1/k$ from eq.(5). Therefore, there exists l such that $p_{i_l} - p_{i_{l+1}} \geq \epsilon/(k-1)$. For this l, define $\tilde{B}$ as $\tilde{B} \stackrel{\triangle}{=} \{i_0,\ldots,i_l\}$. Then,

$$\begin{aligned}
\Pr[W_{\tilde{B}} = 0] - \Pr[W_{\tilde{B}} = j] \ &= \ \frac{1}{l+1}\sum_{i\in\tilde{B}} p_i - \frac{1}{l+1}\sum_{i\in\tilde{B}} p_{i+j} \\
&\geq \ \frac{1}{l+1}(p_{i_l} - p_{i_{l+1}}) \geq \epsilon/(l+1)(k-1) \geq \epsilon/(k-1)^2 \quad \P
\end{aligned}$$

Theorem 3.2 *Let $Y_1, \ldots, Y_T$ be pairwise independent random variables over K. Let Y be a random variable which takes the majority value of $Y_1, \ldots, Y_T$ as its value. (If the majority value is not unique, take one of them randomly.) Suppose that (1) $\Pr[Y_i = j] = q_j$ for $1 \leq i \leq T$ and (2) $q_0 - q_j \geq \epsilon$ for $j \neq 0$. Then, $\Pr[Y \neq 0] \leq (k-1)/T\epsilon^2$.*

Proof: Let $X_j \stackrel{\triangle}{=} \sum_{l=1}^{T} X_j^l$, where

$$X_j^l \stackrel{\triangle}{=} \begin{cases} 1, & \text{if } Y_l = 0 \\ -1, & \text{if } Y_l = j \\ 0, & \text{otherwise} \end{cases}$$

Then,

$$\begin{aligned} E(X_j^l) &= \Pr[Y_i = 0] \cdot 1 + \Pr[Y_i = j] \cdot (-1) = q_0 - q_j \\ E(X_j) &= \sum_{l=1}^{T} E(X_j^l) = T(q_0 - q_j) \\ V(X_j^l) &= E((X_j^l)^2) - (E(X_j^l))^2 \leq E((X_j^l)^2) \leq 1 \\ V(X_j) &= \sum_{l=1}^{T} V(X_j^l) + 2 \cdot \sum_{l \neq l'} E(X_j^l X_j^{l'}) - 2 \cdot \sum_{l \neq l'} E(X_j^l) E(X_j^{l'}) = \sum_{l=1}^{T} V(X_j^l) \leq T \end{aligned}$$

Therefore,

$$\Pr[Y = j] \leq \Pr[X_j \leq 0] \leq \Pr[|X_j - E(X_j)| \geq E(X_j)] \leq V(X_j)/(E(X_j))^2 \leq 1/T\epsilon^2$$

$$\Pr[Y \neq 0] = \sum_{j=1}^{k-1} \Pr[Y = j] \leq (k-1)/T\epsilon^2. \qquad \P$$

Corollary 3.1 *Suppose that $g \in k\text{-}BPP$. Let $s \stackrel{\triangle}{=} 2^{k-1} - 1$. Then,*

(1) there exist a constant c_1 and s probabilistic polynomial time algorithms $\{A_1, \ldots, A_s\}$ such that $\forall z$, $\exists A_i \in \{A_1, \ldots, A_s\}$, $\forall j \neq 0 \bmod k$,

$$\Pr[A_i(z) = g(z)] \geq \Pr[A_i(z) = g(z) + j \bmod k] + |z|^{-c_1}, \qquad (6)$$

(2) there exist a constant c_2 and s probabilistic polynomial time algorithms $\{\hat{A}_1, \ldots, \hat{A}_s\}$ such that $\forall z \in \{0, \ldots, k-1\}^*$, $\exists \hat{A}_i \in \{\hat{A}_1, \ldots, \hat{A}_s\}$,

$$\Pr[\hat{A}_i(z) = g(z)] \geq 1 - |z|^{-c_2}. \qquad (7)$$

Proof: (1) Consider c and A which satsify eq.(4). For each $B \subset \{1, 2, \ldots, k-1\}$, define a probabilistic polynomial time algorithm A_B as follows.

- For input z, A_B chooses $i \in B$ at random. Then A_B outputs $A_B(z) = A(z) - i \bmod k$.

¿From Theorem 3.1, we have (1) for $|z|^{c_1-c} \geq (k-1)^2$. (Note that $\{z \mid |z|^{c_1-c} < (k-1)^2\}$ is finite. Therefore, A_i can memorize all the values of $g(z)$ for such finite z.)
(2) ¿From $\{A_1, \ldots, A_s\}$ of (1), define $\{\hat{A}_1, \ldots, \hat{A}_s\}$ as follows. For $1 \leq i \leq s$,

- For input z, $\hat{A}_i$ runs A_i T times independently, where $T = (k-1)|z|^{2c_1+c_2}$. Let the outputs be $A_i^1(z), \ldots, A_i^T(z)$. Let y be the majority value of $A_i^1(z), \ldots, A_i^T(z)$. $\hat{A}_i$ outputs y.

Then, (2) is obtained from Theorem 3.2. ¶

4 Multisymbol Hard Core

In this section, we show a multisymbol hard core. The proof is based on Theorem 3.1 and Theorem 3.2.

4.1 Theorem on Multisymbol Hard Core

Definition 4 *A function $f : \{0, \ldots, k-1\}^* \to \{0, \ldots, k-1\}^*$ is a k-symbol one-way function if (1) $f(x)$ can be computed in polynomial time of $n = |x|_k$. (2) For every probabilistic polynomial time algorithm $A(\cdot)$ and for every constant $c > 0$, there exists an N_c such that for all $n > N_c$:*

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

where the probability is taken over $x \in \{0, \ldots, k-1\}^n$ and the coin tosses of A.

Definition 5 *Let f be a k-symbol one-way function and H_k be a function of $\{0, \ldots, k-1\}^* \to \{0, \ldots, k-1\}$. We call H_k a k-symbol hard core function (k-HC) of f if for all polynomial time algorithm $A(\cdot)$ and for all c, there exists N_c such that for all $n > N_c$*

$$\Pr_{x,A}[A(f(x)) = H_k(x)] < 1/k + n^{-c},$$

where the probability is taken over $x \in \{0, \ldots, k-1\}^n$ and the coin tosses of A.

Theorem 4.1 *Let f be a k-symbol length-preserving one-way function. Define f' as*

$$f'(x \circ r) \stackrel{\triangle}{=} f(x) \circ r, \tag{8}$$

where $|r|_k = |x|_k$. Then $H_k(x \circ r) \stackrel{\triangle}{=} x \odot r$ is a k-symbol hard core function of f'.

4.2 Proof of Theorem 4.1

We apply Theorem 3.1 and Theorem 3.2 to Rackoff's proof. Assume that H_k is *not* a hard core function of f', i.e. there exist a probablistic polynomial time algorithm A and constants c_1, N_{c_1} such that for infinitely many $n' > N_{c_1}$,

$$Pr_{w,A}[A(f(x) \circ r) = H_k(x \circ r)] \ \geq \ \frac{1}{k} + (n')^{-c_1} \tag{9}$$

where the probability is taken over $w(= x \circ r) \in \{0, \ldots, k-1\}^{n'}$ and the coin tosses of A.

Definition 6 (Good x) *We say that x is* good *if*

$$Pr_{r,A}[A(f(x) \circ r) = H_k(x \circ r)] \ \geq \ \frac{1}{k} + \frac{1}{2}(2n)^{-c_1}, \tag{10}$$

where $n = |x| = |r| = n'/2$ (See eq.(9)). The probability is taken over $r \in \{0, \ldots, k-1\}^n$ and the coin tosses of A.

Claim 4.1 $\Pr_x[x \text{ is good}] \ \geq \ (2n)^{-c_1}/2$, *where $n = |x|$.*

Proof: $Pr_{w,A}[A(f(x) \circ r) = H_k(x \circ r)]$

$$\begin{aligned}
&= \Pr_x[x \text{ is good}] \Pr_{r,A}[A(f(x) \circ r) = H_k(x \circ r) | x \text{ is good}] \\
&\quad + \Pr_x[x \text{ is not good}] \Pr_{r,A}[A(f(x) \circ r) = H_k(x \circ r) | x \text{ is not good}] \\
&\leq \Pr_x[x \text{ is good}] \cdot 1 + (1 - \Pr_x[x \text{ is good}]) \left(\frac{1}{k} + \frac{1}{2}(2n)^{-c_1} \right) \\
&\leq \Pr_x[x \text{ is good}] + \frac{1}{k} + \frac{1}{2}(2n)^{-c_1}
\end{aligned}$$

From eq.(9),

$$\begin{aligned}
\frac{1}{k} + (2n)^{-c_1} &\leq \Pr_x[x \text{ is good}] + \frac{1}{k} + \frac{1}{2}(2n)^{-c_1} \\
\Pr_x[x \text{ is good}] &\geq \frac{1}{k} + \frac{1}{2}(2n)^{-c_1} \qquad \P
\end{aligned}$$

Claim 4.2 *Let $s = 2^{k-1} - 1$. Then, there exist s probabilistic polynomial time algorithms $\{A_1, \ldots, A_s\}$ such that, $\exists c_2, \exists N_{c_2}$, for infinitly many $n \geq N_{c_2}$, for any good x such that $|x| = n$, $\exists A_i \in \{A_1, \ldots, A_s\}$, $\forall j \neq 0 \bmod k$,*

$$\Pr_r[A_i(f(x) \circ r) = H_k(x \circ r)] \geq \Pr_r[A_i(f(x) \circ r) = H_k(x \circ r) + j \bmod k] + (2n)^{-c_2} \tag{11}$$

Proof: Similar to the proof of Theorem 3.1 (1). ¶

For any constant $c_3 > 0$, let $m = \lceil \log_2(k-1)(2n)^{2c_2+c_3} \rceil$. Let x_l be the l-th bit of x. We show an algorithm $I_{l,j}$ which guesses x_l by using $H_k(x \circ r)$ as an oracle and A_j as a subroutine, where $1 \leq j \leq s$.

[**Algorithm** $I_{l,j}$] Input: $y(= f(x))$. Let $n = |y|$.

(step 1) For $0 \leq i \leq m$, choose $R_i \in \{0, \ldots, k-1\}^n$ randomly. By using the oracle, obtain

$$h_i = H_k(x \circ R_i) \text{ for } i = 0, \ldots, m.$$

(step 2) For each $b = (b_1, b_2, \ldots, b_m) \in \{0,1\}^m$, compute

$$\begin{aligned} r_b &= R_0 + (b_1 \times R_1) + \cdots + (b_m \times R_m) \bmod k \\ x_b &= A_j(f(x) \circ (e^l + r_b)) - h_0 - (b_1 \times h_1) - \cdots - (b_m \times h_m) \bmod k \end{aligned}$$

(step 3) Let $\hat{x}_l$ be the majority value of $\{x_{0\ldots0}, \ldots, x_{1\ldots1}\}$. Output $\hat{x}_l$. (If the majority value is not unique, take one of them randomly.)

Claim 4.3 *There exists $\exists N_{c_3}$ such that for infinitly many $n \geq N_{c_3}$, for any good x such that $|x| = n$, for A_j which satisfies eq.(11),*

$$\Pr_{I_{l,j}}[I_{l,j}(f(x)) \neq x_l] \leq (2n)^{-c_3}, \tag{12}$$

Proof: Similar to claim 2.2, we can obtain

$$\Pr[X_b = x_l + i \bmod k] = \Pr[A_j(f(x) \circ r) = H_k(x \circ r) + i \bmod k]$$

Furthermore, like claim 2.3, $\{X_b\}$ are pairwise independent. Now, we can apply Theorem 3.2 to $\{X_b\}$. Then, $\Pr_{I_l}[I_l(f(x)) \neq x_l] \leq (k-1)(2n)^{c_2}/2^m \leq (2n)^{-c_3}$. ¶

Now we present an algorithm I which guesses the whole x. Instead of using the oracle in $I_{l,j}$, we let $(h_0, \ldots, h_m)$ take on all possible values.

[**Algorithm** I] Input: $y(= f(x))$. Let $n = |y|_k$.

(step 1) For $0 \leq i \leq m$, choose $R_i \in \{0, \ldots, k-1\}^n$ randomly.

(step 2) For $1 \leq j \leq s$, do the following steps.

(step 3) For each $(h_0, \ldots, h_m) \in \{0, \ldots, k-1\}^{m+1}$, do the following steps.

(step 4) For all $l \in \{1, \ldots, n\}$, obtain $\hat{x}_l$ by running step 2 $\sim$ step 3 of algorithm $I_{l,j}$.

(step 5) Let $\hat{x} = (\hat{x}_1 \cdots \hat{x}_n)$. If $f(\hat{x}) = y$, then output $\hat{x}$ and stop. Otherwise, goto step 3.
If $f(\hat{x}) \neq y$ for all $(h_0, \ldots, h_m)$, goto step 2.

(step 6) If $f(\hat{x}) \neq y$ for all j, choose $\hat{x}$ randomly and output $\hat{x}$.

It is clear that I runs in polynomial time because $m = \lceil \log_2(k-1)(2n)^{2c_2+c_3} \rceil$. From claim 4.3, for a good x,

$$Pr_I[I(f(x)) \neq x] \leq Pr_I[\exists j \ I_j \text{ fails}] \leq \sum_{i=1}^{n} Pr_{I_j}[I_j \text{ fails}] \leq n \times (2n)^{-c_3}$$

Then, from claim 4.1,

$$Pr_{x,I}[I(f(x)) \in f^{-1}(f(x))] \ \geq \ (1/2)(2n)^{-c_1}(1 - n \cdot (2n)^{-c_3}) \geq (1/4)(2n)^{-c_3}$$

This shows that f is not one-way. Therefore, H_k is a hard core function of f'.

Acknowledgement

We would like to thank Prof. Osamu Watanabe of Tokyo Institute of Technology.

References

[1] Ker-I Ko. "complexity theory of real functions ". In *Birkhauser Boston*, page 178, 1991.

[2] M. Blum and S. Micali. "How To Generate Cryptographically Strong Sequences of Pseudo Random Bits". In *SIAM Journal on Computing, vol.13*, pages 850–864, 1984.

[3] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. "RSA and Rabin functions: certain parts are as hard as the whole ". In *SIAM Journal on Computing, vol. 17*, pages 194–209, 1988.

[4] A.C. Yao. "Theory and applications of trapdoor functions ". In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 80–91, 1982.

[5] S. Goldwasser and S. Micali. "Probabilistic encryption ". In *Journal of computer and system sciences, vol.28*, pages 270–299, 1984.

[6] O. Goldreich and A. Levin. "A hard core predicate for all one way functions ". In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[7] C. Rackoff. "Simplified proof for hard core ". In *Private communication*, 1991.