行政院國家科學委員會專題研究計畫　期中進度報告

子計畫二：擬亂數產生器與編碼及其密碼之應用(1/3)

計畫類別：整合型計畫
計畫編號：NSC91-2213-E-009-102-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學資訊工程學系

計畫主持人：陳榮傑

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 92 年 5 月 23 日

行政院國家科學委員會補助專題研究計畫 □成果報告 ■期中進度報告

理論密碼學與應用-子計畫二：擬亂數產生器與編碼及其密碼之應用 (1/3)

計畫類別：□ 個別型計畫　　整合型計畫
計畫編號：NSC 91－2213－E－009－102－
執行期間：　91 年　8 月　1 日至　92 年　7 月　31 日

計畫主持人：陳榮傑
共同主持人：
計畫參與人員： 胡鈞祥、劉穎駿、劉明宇

成果報告類型(依經費核定清單規定繳交)： 精簡報告 □完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢

□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立交通大學資訊工程學系

中　華　民　國　　92 年　　5 月　　15　日

中文摘要：

在隨機演算法中，如何產生一個亂數，以及亂數對隨機演算法進行去隨機化的動作，是非常重要的課題。

亂度萃取器是一個能夠從微弱的亂度源中萃取亂度出來的演算法，這是我們目前用來對隨機演算法進行去隨機化的主要方法。在[Tre99]中提到，我們可以利用亂數產生器做出亂度萃取器，這些亂數產生器都是要建立在一些非常困難的問題上，所以我們研究的方向之一是探討亂度萃取器與已知的困難問題之間的關聯性，以及如何更有效率的利用亂數產生器。

英文摘要：

In randomized algorithms, it is an important subject how we could generate a random number and de-randomize randomized algorithm with random numbers.

An Extractor is an algorithm that is able to extract randomness from weak random source. It is now the major method on de-randomizing a randomized algorithm. In [Tre99], we know we could make extractors from pseudorandom generators, and these pseudorandom generators are based on some very hard predicate. And one aspect of our research is to explore the relation between extractors and some well-known open hard problems, and the way to use pseudorandom generators more efficiently.

報告內容：

前言

隨機演算法和編碼理論各自都是長久以來已有大量研究的課題，而兩者之間，原本並沒有被認為是相關的，因此兩方的研究人員各自處理各自的問題。但在近來的研究中[4][10]，逐漸地建立起這兩方的橋梁。在隨機演算法中相當重要的亂度萃取器，和編碼理論中各式各樣的糾錯碼，開始被拿來一起研究討論，而且得到了很重要的成果。而因為這些研究，在隨機演算法中開始進行對各式糾錯碼的研究，試圖利用發展得很完整的編碼理論來進行亂度萃取的工作[8]；而另一方面，也有人嘗試使用已知的亂度萃取器來建造更好的糾錯碼[7]。

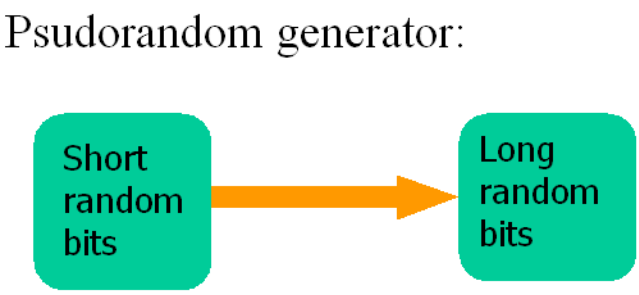
研究目的

隨機計算(Randomized computation)對於許多計算方面的難題而言，是非常有用且可行的方法。如質數檢定(Primality testing)等等，而使用隨機演算法是已知

的有效方法。在計算理論中，常用 BPP 這類的問題來定義這些存在有效隨機演算法的問題。依照目前的研究來說，如果存在一個有效的擬亂數產生器(Pseudorandom generator)，即可以將 BPP 中的問題轉換成 P 的問題，此即為 Derandomize BPP，亦即證明了 BPP=P 的問題。因此凡相關至亂數產生器的研究都是我們關心的課題。

研究內容

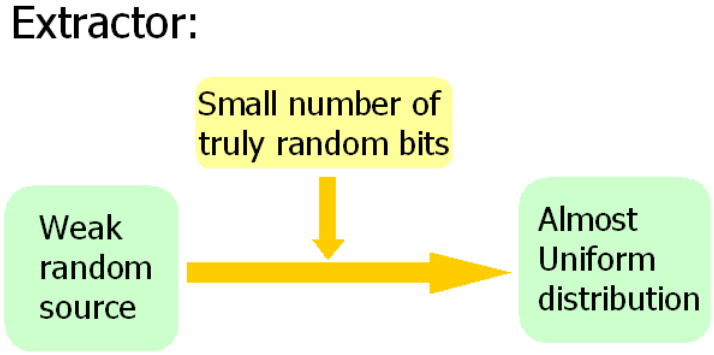
亂數產生器 (Pseudorandom generators)



亂數產生器(Pseudorandom generators)就是一個演算法，可將一個長度較短的亂數(random bits)拉長，使成為長度較長的擬亂數(pseudorandom bits)。擬亂數可以用在去隨機性(derandomizing)：去除原本在進行隨機演算法時所需用到的亂數。如果可產生一個夠好的亂數產生器，我們就可以把屬於 BPP(Bounded Probabilistic Polynomial)中的問題用確定性的(deterministic)演算法來完成。

而所謂一個好的亂數產生器，乃是指其所產生的擬亂數必須和均勻分布(Uniform distribution)為不可分辨的(indistinguishable)。

亂度萃取器 (Randomness Extractors)

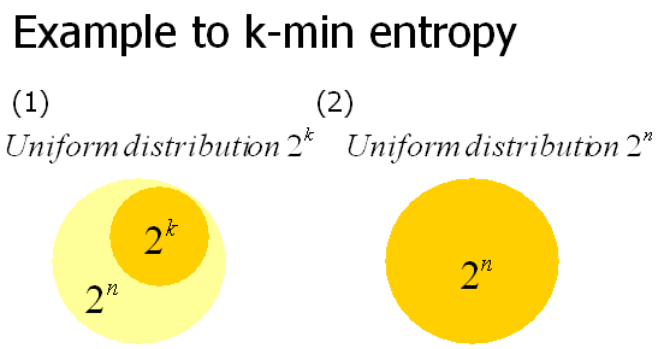


亂度萃取器是一個演算法，可將一個稍具亂度的資料來源(weak source of randomness)，在搭配使用少許長度的真實亂數(truly random bits)後，將其中的亂度萃取(extract)出來，成為一個高亂度的亂數。所以亦可說是另一種產生亂數的方法。

首先，我們先說明一下何謂一個稍具亂度的資料來源。我們稱一個資料來源具有最小亂度為 k(k-min entropy)，即是對於每一個屬於此資料來源的樣本(Sample)，所發生的機率皆小於等於 2 的負 k 次方，其數學表示法如下(假設具有最小亂度 k 的資料來源為 X)：

$$\Pr[X = x] \le \frac{1}{2^k}$$

以下是兩個具有最小亂度 k 的分佈的例子：



其中圖一是具有最小亂度 k 的分佈中最不均勻的例子，所有的機率都集中到 2^k 的個體上，而剩下的機率都是 0，所以符合定義。而圖二是均勻分佈，也就是最均勻的例子。

擁有了符合要求的資料來源後，我們再加入少許的真實亂數(truly random bits)加以催化，便能得到一個接近均勻分布的亂數，這便是亂度萃取器的演算過程，其數學表示法如下(資料來源長度為 n，加入真實亂數的長度為 t，所萃取的亂數長度為 m)：

$$Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$$

亂度萃取器在演算法中有非常多的應用，我們以下舉出一些例子：

1. 模擬 BPP 的演算法：
2. RP 問題的黑盒(black-box)模擬法。
3. 確定性的機率問題的增益放大方法(Deterministic Amplification)
4. 建造遺忘性的取樣者(Oblivious Sampler)
5. Clique 問題的近似解

6. 產生圖論中的超級精練器(Super Concentrators)
7. 產生“高度擴張圖”(Highly expanding Graphs)
8. 建造擬亂數產生器

而 Luca Trevisan 在[10]中也有研究亂度萃取器的建立方法，簡單來看，可以將其看成是一個二分圖(bipartite graph)，而輸入值和輸出值各佔一邊，輸入值域是 2^n，而輸入值域是 2^m。而每個輸入值都有 2^t 個邊對映到輸出，當然，不是一對一映射(one to one mapping)也不是蓋射(on to mapping)。而另外輸入的 t 位元亂數，等於是用來隨機選一個對應，將輸入值對應到輸出值。而其對應關係的建立是利用「設計」(Design)此種數的結構來定義的。

設計(Design)，又稱為分裝(packing)，為符合條件的一群集合。我們稱一群集合 $S_1, S_2, \ldots, S_m$ 為一個(m,t,l,a)設計，即共 m 個集合，每個集合的元素(element)皆由數字 1~t 中不重複選出，且每個集合 S_i 的大小皆為 l，而任兩個集合之間的交集個數小於等於 a 個。由數學式來表示如下：

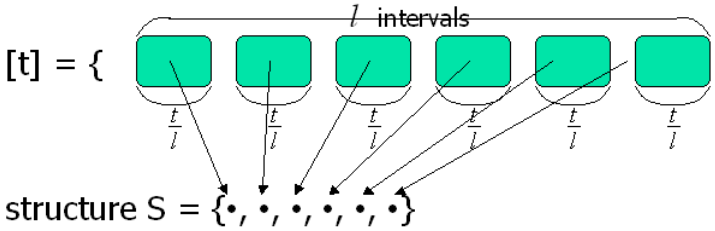
$$(m,t,l,a)-design$$

$$S_i \subseteq [t]$$

$$|S_i| = l$$

$$\forall i \neq j \in [m], \quad |S_i \cap S_j| \leq a$$

而如何產生一個設計呢？最簡單的作法，便是先將 1~t 的數字分為 l 塊，每一塊含有 t/l 個數字，並且每個集合 S_i 從皆從每一塊中選取一個數字，如此剛好為 l 的大小，而可由 Chernoff bound 證明得存在符合設計條件的集合元素選法。



而在[10]中，即是利用[13]的亂數產生器，套用設計(Design)的性質，產生出一個建立亂度萃取器的演算法，並在剛開始建立的步驟中，加入了糾錯碼(Error-correcting codes)的概念。

在從亂數產生器建立亂度萃取器的過程中，加入了糾錯碼的應用，是主要是用於在證明其定理的正確性，而主要的亂度來源還是來自於擬亂數產生器。其中，所用到的糾錯碼叫二元碼(binary code)。所謂二元碼，是指具有串列解碼性質(list decoding property)的編碼方法。所謂的串列解碼性質是指在編碼空間中任

半徑為 1/2-α的漢明距離的球(Hamming ball)內最多只可能有 $O(1/\alpha^2)$個編碼，換句話說，其實主要是利用糾錯碼的特性，經過編碼過後的字串，將會有比較均勻的亂度分佈。而將輸入字串經過編碼後再拿來使用比直接拿來使用有更好的效果。

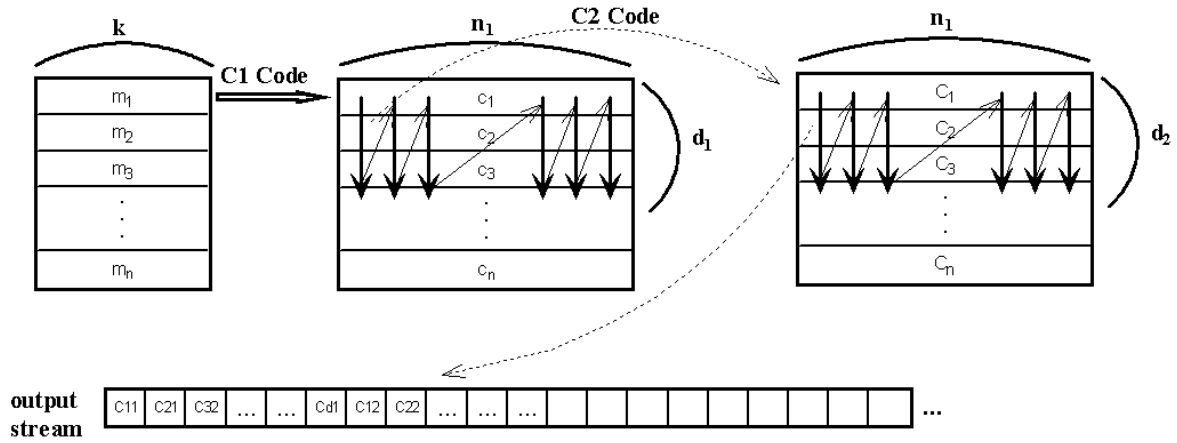
如何使用糾錯碼來建立亂度萃取器呢？所使用到的糾錯碼是 Reed-Muller code，這是在編碼理論中非常有用而且常被其他的領域利用的一種編碼。這裡所用到的 Reed-Muller code 編碼的定義形式如下：

(h,D) Reed-Muller code 分佈於 Fq：

1. 訊息：D 變數分佈於 Fq 的多項式 f，總合維度(total degree)最大為 h
2. 碼文：訊息所代之多項式的值。分佈於 Fq^D 。
3. 碼文長度最大為 q^D。
4. 訊息維度(h+D)！/h！D！。

建立亂度萃取器的過程，是先把輸入值用 Reed-Muller code 編碼後，再用二元碼編碼。像這種多重編碼的技巧，在編碼理論中是常見也很有用的。舉例來說，在編碼理論中討論到建造能抵擋大量鄰近而突發的錯誤的糾錯碼時，為了防止大量的錯誤同時打到某一個訊息而導致無法更正，就必需將編碼後的字串再加以打散，讓鄰近的錯誤分散到每一個訊息之上，之後再加更正即可。而將字串加以打散的方法通常會需要亂數的參與，而此時所用的方法是用多重的糾錯碼加以疊加、重複編碼，之間用交錯的方式加以結合。方法如下：

存在兩個糾錯碼 C1=[n1, k1, d1], C2 = [n2, k2, d2], 先將訊息(message)利用 C1 編碼成長度為 n1 的碼文(codeword)，再利用深度為 k2 的內插傳送法，當成中介訊息，而後再將這些長度為 k2 的中介訊息，利用 C2 編碼成長度為 n2 的碼文。此時再將這些長度為 n2 的碼文以深度為 k1-1 的內插傳送法傳送過去。此時 C2 利用本身具有的 d2-1 錯誤偵測能力，而使得 C1 會具有 n1-d1+1 的錯誤更正能力。



而結果證實，利用這種方法可以非常有效的進行打亂的動作，而這個結果也隱含了編碼和亂數之間存在的關係。我們可以嘗試由已知的編碼方式，來建構出一種不同的亂度萃取器演算法，或是利用編碼理論改善其亂度萃取器演算法。

目前進度:

1. 探討並設計新的 Extractor
2. 探討 list coding 在密碼學的應用
3. Extractor code 在記憶裝置的應用

參考文獻：

[1]. L. Blum, M. Blum, M. Shub, "A simple unpredictable random number generator", *SIAM Journal on Computing*, 15(1986), pp. 364-383, 1986.

[2]. M. Blum, S. Micali, "How to generate cryptographically strong sequence of pseudo-random bits", *SIAM Journal on Computing*, 13(1984), pp. 850-864, 1984.

[3]. J. Boyar, "Inferring sequences produced by pseudo-random number generators", *Journal of Association for Computing Machinery (JACM)*, 36(1989), pp. 129-141, 1989.

[4]. N. Nisan and A. Ta-Shma. Extracting randomness : A survey and new constructions. Journal of Computer and System Sciences, 1998. To appear. Preliminary versions in [Nis96, TS96]

[5]. Noam Nisan and Avi Wigderson. Hardness vs randomness. Journal of Computer and System Sciences, 49(2):149-167, October 1994.

[6]. R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. In Proceedings of the 31st ACM Symposium on Theory of Computing, pages 149-158, 1999.

[7]. A. Ta-Shma and D. Zuckerman. "Extractor codes", In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001.

[8]. A. Ta-Shma, D. Zuckerman, S. Safra. "Extractors from Reed-Muller codes", Electronic Colloquium on Computational Complexity, Report No. 36 (2001).

[9]. R. Shaltiel and C. Umans, "Simple Extractors for All Min-Entropies and a New Pseudo-Random Generator", IEEE Symposium on Foundations of Computer Science (FOCS'01)

[10]. Luca Trevisan. Construction of extractors using pseudo-random generators. In Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing. Pages 141-148, Atlanta, Georgia, 1-4 May 1999.

[11]. Andrew C.Yao. Theory and applications of trapdoor functions(extended abstract).

In 23rd Annual Symposium on Foundations of Computer Science, pages 80-91,Chicago,Illinois,3-5 November 1982.IEEE.

[12]. N. Nisan. Extracting randomness: How and Why. In Proceedings of the 11th IEEE Conference on Computational Complexity,page 44-58, 1996.

[13]. N. Nisan and A. Wigdersn. Hardness vs randomness. Journal of Computer and System Sciences, 49149-167, 1994. Preliminary version in Proc. Of FOCS'98.