一种改进的基于时空混沌系统的Hash函数构造方法

张向华
(重庆教育学院 重庆 400067)

摘 要 首先介绍了一种以时空混沌为基础的构造Hash函数的典型方案,并分析了它的不足和安全性问题。在此基础上提出了一种改进的Hash函数构造方法。理论和仿真实验分析表明,该算法具有更好的统计特性和更高的效率,有效地弥补了原算法中存在的碰撞漏洞。

关键词 Hash函数,时空混沌系统,耦合映象格子
中图法分类号 TP391 **文献标识码** A

Modified Algorithm for Hash Function Based on the Spatiotemporal Chaotic System

ZHANG Xiang-hua
(Chongqing Education College, Chongqing 400067, China)

Abstract A class scheme of hash function was introduced and the flaws and security holes were pointed out. Then, a modified algorithm for one-way Hash function construction based on the spatiotemporal chaotic system was proposed. Analysis shows that the proposed algorithm has better statistics performance and efficiency, and remedies the security holes of the original algorithm.

Keywords Hash function, Spatiotemporal chaotic systems, Coupled-map lattices

1 引言

近年来对Hash函数碰撞性的研究已取得了突破性的进展,发现诸如MD5,SHA-1和RIPEMD等在抗碰撞性方面的一些以前不为人知的缺陷[1,2]。这也使得Hash函数的研究得到了更多的重视,成为了密码学界当前的一个研究热点。

Hash函数通常具有以下性质[8]:

① 压缩性。输入可以为任意长度的消息,而输出为固定长度的二进值串,比如128比特或256比特。

② 不可逆性。已知输入x,容易计算其Hash值$h(x)$,但已知Hash值h,要找到对应的x在计算上不可行。

③ 抗弱碰撞性。对任意的输入x,找到满足$y\neq x$,且$h(x)=h(y)$的y在计算上不可行。

④ 抗强碰撞性。找到任意满足$h(x)=h(y)$的偶对(x,y)在计算上是不可行的。

K. W. Wong在文献[3]中提出了一种将加密和Hash结合在一起方法。虽然在此方案提出后不久,G. Alvarez等人就指出了其中存在安全漏洞,但是这种将加密和Hash结合在一起的方式,是一种很有特色的思路,具有很好的推广价值。在文献[4]中,Xun Yi将混沌迭代变换与DM(Davies-May)方案[5,6]结合在一起,提出用迭代75次帐篷映射代替块加密方式来产生Hash值。此方案具有与DM方案相同的安全性,并且具有更高的效率。

王继志等在文献[7]中对一类基于混沌映射的Hash函数进行了碰撞性分析,并建议在构造Hash函数时应采用如下的方式:① 最好将明文映射到参数空间;② 分组得到的迭代值应该作为下一分组迭代过程的初值,即不同分组的迭代应该是相关的,而不应完全分离;③ 对分组不足的处理,不能仅仅是单纯地添加某个字符,还需要添加原始明文的信息;④ 对于最后生成Hash值的迭代次数的选择,应尽量对不同的明文选择不同的迭代次数,这样即使最后的迭代序列完全一致,由于迭代次数选择不一样,也可以保证最后的Hash值不同。

与基于简单混沌映射的Hash函数相比,基于时空混沌构造的Hash函数还具有另外两个方面的优势:其一,时空混沌是一个复杂的高维混沌系统,其中存在多个正的Lyapunov指数,能保证所产生的混沌序列的复杂性,能更有效地防止攻击者通过预测混沌序列而实施的攻击;其二,与单个混沌映射相比,由于格子之间是相互耦合的,在已知各格子当前状态值的情况下,计算格子前一状态的值会更加困难,因而系统具有更好的单向性。

本文第2节介绍了一种已有的Hash函数构造方法,并分析了存在的问题;第3节提出了一种改进的Hash函数构造方法,并从理论和仿真实验两个方面分析了该算法的性能;最后是总结。

2 基于调整时空混沌状态的Hash函数构造方案

基于时空混沌设计Hash函数的一种主要思路,是用消息直接控制或调整时空混沌的状态值,然后同样通过多次迭代和从状态值中抽取二进制序列的方式产生Hash值。

到稿日期:2008-08-12 返修日期:2008-09-11 本文受重庆市教委资助项目(KJ071504)资助。
张向华(1969-),男,硕士,副教授,主要研究方向为混沌理论及信息安全。

2.1 算法思想描述

文献[9]提出的 Hash 函数构造算法是用消息直接控制或调整时空混沌的状态值，并通过多次迭代和状态值抽取来产生 Hash 值的典型代表。算法所采用的时空混沌模型为临近耦合映象格子：

$$x_{n+1}(i)=(1-\varepsilon)f(x_n(i))+\varepsilon[f(x_n(i-1))] \quad (1)$$

其中，$x_n(i)$为格子 i 在时刻 n 的状态变量，n 为离散时间坐标，i 为离散空间坐标，$i=1,2,\cdots,N$(N 为耦合映象格子的长度)；$\varepsilon\in(0,1)$为耦合系数。周期边界条件为 $x_n(N+i)=x_n(i)$。非线性函数 f 为 logistic 映射，即 $f(x)=\mu x(1-x)$，$x\in(0,1)$，$\mu\in[3.57,4]$。

其产生 Hash 值的具体过程如下：

第 1 步　待处理消息按对应字节 $C_1C_2\cdots C_N$(C 为消息的 ASCII 码)线性变换为[0,1]范围内的数，这样整个消息变换为一个数值序列，记为 $M_1M_2\cdots M_N$，其中 N 为消息的字节数。线性变换公式如下：

$$M_i=C_i/256 \quad (2)$$

其中，$i=1,2,\cdots,N$。

第 2 步　令 $M_1,M_2,\cdots,M_N$ 为 N 个格子的初值，即

$$X_0(i)=M_i \quad (3)$$

其中，$i=1,2,\cdots,N$。取 $\mu=4$，$\varepsilon=0.8$，按照式(1)进行迭代，此时可得到时空混沌序列 N 组：

$X_n(1),X_n(2),X_n(3),\cdots,X_n(N)$

第 3 步　从迭代结果序列中取出最后一组序列的 $X_R(N),X_{2R}(N),X_{3R}(N)$，且 $R>>N$，将它们线性变换和取整运算映射为两个 40bit 和一个 48bit 的二进制数，合起来作为最后 128bit 的 Hash 值。

2.2 算法的安全性问题

该算法通过格子之间的相互耦合和多次迭代(R 次)，将消息对系统状态值的影响进行不断的扩散和放大，从而使时空混沌系统中最后一个格子的状态值不仅与消息的最后一个字节有关，而且与其他字节密切相关。文献[9]中仿真的结果表明，该算法具有较好的统计性能，也较好地利用了时空混沌的特性。但此 Hash 函数的构造方法存在以下不足：

① 随着处理消息的长度增加，其效率会显著降低。假设待处理消息的长度为 N 个字节，即 CML 中格子的个数为 N。根据式(1)，在 CML 中格子 i 由状态 n 变为状态 $n+1$ 时，需要进行 2 次 Logistic 映射运算，因此整个 CML 由状态 n 变为状态 $n+1$ 时，需要进行 $2N$ 次 Logistic 映射运算。产生最终的 Hash 值时，CML 由状态 0 变为状态 $3R(R>>N)$，总共需要进行 $6RN$ 次 Logistic 映射运算。即使在 $R=N$ 时，产生 Hash 值所需要计算的 Logistic 映射次数也与消息长度的平方成正比。所以当处理消息较长时，计算量会显著增加，从而效率会大大降低。

② 当待处理消息的长度为一个字节时，CML 模型中仅有一个格子，模型退化为单个的 Logistic 映射，不能充分体现 CML 模型的特点。

③ 虽然通过相互之间的耦合，格子能互相作用，但是 Hash 值仅从最后一个格子的状态值中提取的方式没有充分利用模型整体的状态特性。针对此不足，文献[7]提出了一种构造碰撞的方法，对此算法进行了攻击。

④ 在把消息映射为系统初值时，特别当消息为文本时，存在某种缺陷。由于文本中的字符都是一些可见字符，其 ASCII 码值介于 033～126 之间，因此系统初值介于 33/256～126/256 之间，在区间[0,1]上的分布不具有较好的统计特性。

3 改进的基于调整时空混沌状态的 Hash 函数

3.1 时空混沌模型选择

耦合映象格子系统具有多种状态[10,11]，分别为冻结化随机图案状态、图案选择状态、缺陷混沌扩散状态、缺陷湍流状态、图案竞争阵发混沌状态和完全发展混沌状态。由于最终的 Hash 值是从耦合映象格子的状态中抽取，因此自然希望模型处于完全发展的混沌状态。在改进方案中我们仍然选择如下的临近耦合映象格子模型：

$$x_{n+1}(i)=(1-\varepsilon)f(x_n(i))+\frac{\varepsilon}{2}[f(x_n(i-1))+f(x_n(i+1))] \quad (4)$$

其中各参数、局部映射以及周期边界条件均与式(1)相同。此模型的 Lyapunov 指数谱为[12]：

$$LE_1=\lim_{n\to\infty}\frac{1}{n}\ln\left|\prod_{i=1}^{n}f'(x_i)\right| \quad (5)$$

$$LE_2=LE_1+\ln[1-\varepsilon+\varepsilon\cos(2\pi/L)] \quad (6)$$

$$LE_3=LE_1+\ln[1-\varepsilon+\varepsilon\cos(4\pi/L)] \quad (7)$$

$$\vdots$$

$$LE_L=LE_1+\ln[1-\varepsilon+\varepsilon\cos(2\pi(L-1)/L)] \quad (8)$$

其中，L 表示格子的个数，即 $L=N$。

最大 Lyapunov 指数是表示混沌系统整体混乱程度的一个重要指标。由式(5)可知，此模型的最大 Lyapunov 指数与局部映射的最大 Lyapunov 指数相同，且与系统中格子的个数无关，因此可根据需要来确定格子的个数。此处我们将格子的个数设置为 $N=16$。另外，由于选用的局部映射为 $f(x)=\mu x(1-x)$，$x\in(0,1)$，$\mu\in[3.57,4]$，为了让模型的最大 Lyapunov 指数有较大的值，设置 μ 为 3.999999。

为了保证耦合映象格子处于完全发展的混沌状态，LE_2 应当越大越好，因此 ε 应当取较小的值。同时为了保证格子之间存在必要的耦合强度，加快格子之间信息的扩散，又要求 ε 不应设置得过小，所以在改进的算法中将 ε 设置为 0.1。

3.2 改进的 Hash 函数

① 算法描述

改进后的算法如下：

第 1 步　定义 16 个 8-bit 的初始变量，用十六进制表示为 $IV[1\cdots16]$ = [0F,1E,2D,3C,4B,5A,69,78,87,96,A5,B4,C3,D2,E1,F0]。按照式(9)把它们线性变换为区间[0,1]内的数，并作为 CML 中对应格子的初始状态值。

$$x_0(i)=IV[i]/256,\ i=1,2,\cdots,16 \quad (9)$$

第 2 步　待处理消息按对应字节 $C_1C_2\cdots C_N$(C 为消息的 ASCII 码)线性变换为区间[0,1]内的数，得到的数值序列记为 $M_1M_2\cdots M_N$，N 为消息的字节数。变换公式如下：

$$M_i=(C_i+0.8)/128 \bmod 1,\ i=1,2,\cdots,N \quad (10)$$

第 3 步　对 CML 模型进行如下 N 步操作：

设 $a\in[1,N]$为其中某一步骤。在步骤 a 中，按照式(11)更改 CML 中第 1 个格子的状态值，然后将 CML 迭代 K 次，即将 CML 由状态$(a-1)K$ 转变为状态 aK。

$$x_{a-1}(1)=\{x_{a-1}(1)+2\times(M_a+M_{N-a})\}/5 \quad (11)$$

第 4 步　最终 CML 的状态为 NK。将 CML 中各格子的状态值按式(12)、(13)和式(14)转变为二进制形式[13]，从每个格子的二进制表示中抽取 8 比特(小数点后第 6 位～第 13 位的数)合并在一起构成 128 位的 Hash 值。

一个实数 x 表示成如下的二进制形式：

$$x=0.b_1(x)b_2(x)\cdots b_i(x)\cdots,x\in[0,1],b_i(x)\in\{0,1\} \tag{12}$$

在这个表示形式中，第 i 比特可以表示成：

$$b_i(x)=\sum_{r=1}^{2^i-1}(-1)^{r-1}\Theta_{(r/2^i)}(x) \tag{13}$$

此处，$\Theta_t(x)$ 是一个域值函数，其定义如下：

$$\Theta_t(x)=\begin{cases}0, & x<t\\ 1, & x\geqslant t\end{cases} \tag{14}$$

② 确定每步的迭代次数 K

此处根据 χ^2 检测来确定算法 N 步操作中的迭代次数 K[4,14,15]，按照式(15)计算 CML 中每个格子的 χ^2 值，并取各格子中最大的 χ^2 值作为整个 CML 的 χ^2 值。

$$\chi^2=S(\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{k_{ij}^2}{\sum_{i=1}^{m}k_{ij}\cdot\sum_{j=1}^{m}k_{ij}}-1) \tag{15}$$

具体方法如下：将单位正方形的边进行 m 等分，划分为 m^2 个面积相等的小正方形，其边长为 $1/m$。然后随机选择一个初始状态，令其为 $x_0(l)(l=1,2,\cdots,16)$和一个微小变量 $\Delta\alpha$，对于每个格子分别从 $x_0(l)$和 $x_0(l)+\Delta\alpha$ 开始迭代 K 次，得到相应的状态变量 $x_K(l)$和 $x_K'(l)$。共进行 S 组这样的测试，则 k_{ij} 表示状态变量 $x_K(l)$和 $x_K'(l)$的值落入第 i 行第 j 列的小正方形的个数。

取 $m=11$，$S=1000$，$\Delta\alpha=0.8/256$，得到 CML 的 χ^2 值与迭代次数 K 之间的关系如图 1 所示。由于当 χ^2 值小于 5%对应的临界值时，认为两个变量无关。此处，自由度为 $(m-1)(m-1)=100$，查 χ^2 表得在 5%临界点时 χ^2_{100}为 124.3。实验结果表明，当 $K>36$ 时，能保证两个迭代后的耦合映象格子系统的状态完全无关。为了安全起见，取 $K=40$。

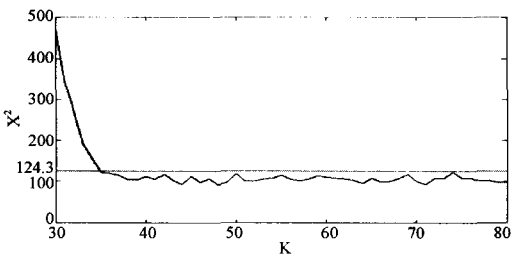

图 1　迭代次数 K 与 χ^2 值的关系

3.3　算法性能分析

① 文本仿真

取文本 1 为“Chaos-based cryptography has been around for more than a decade by now. During this time of foundation and development, it came to mean different things, mostly depending on the implementation. So, we can speak of additive masking, chaos shift keying, two-channel communication, message embedding, etc.”。文本 2 中把初始文本的首字母“C”改为“c”，文本 3 中把初始文本内的“So”改为“so”，文本 4 中把初始文本内的“two-”写成“two/”。计算这些文本的 Hash 值，用十六进制数表示如下：

文本 1：F9640BE0CE8A7AFFA91E5B7D28C4A615

文本 2：A7EFD2DC9E9E40B7F556018BE6B657FB

文本 3：25E4B00659FB1C0B2F9E9E8EA8F88FAD

文本 4：E0B62FDC6698B3D5FBFAACFA7AC54E7F

从仿真的结果看算法对消息具有高度的初值敏感性。

② 混乱与扩散性质统计分析

混乱与扩散是设计加密算法的两个重要标准，它们对设计 Hash 函数仍然有效。对于 Hash 函数，理想情况下的扩散性表现为初值微小的变化都会引起 Hash 值中每比特以 50%的概率变化。对 Hash 函数常进行如下统计分析。

平均变化 bit 数：$\bar{B}=\frac{1}{N}\sum_1^N B_i$

平均变化概率：$P=(\bar{B}/128)\times100\%$

B 的均方差：$\Delta B=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i-\bar{B})^2}$

P 的均方差：$\Delta P=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i/128-P)^2}\times100\%$

其中 N 为统计总次数，B_i 为第 i 次测试时 Hash 值变化的比特数。现随机选取一段明文，计算其 Hash 值。然后随机更改明文中 1 个比特的值，计算其 Hash 值，比较两次的 Hash 值得到 B_i。重复上述过程 1024 次，得到置乱数的分布如图 2 所示。在 $N=1024$ 次测试中，Hash 值(128 比特)平均比特变化数为 63.94，非常接近理想状况下的 64 比特变化数。此外，B_i 主要集中在 60～70 比特之间，即紧靠在理想值 64 比特附近，这表明算法对明文的置乱能力强而且稳定。

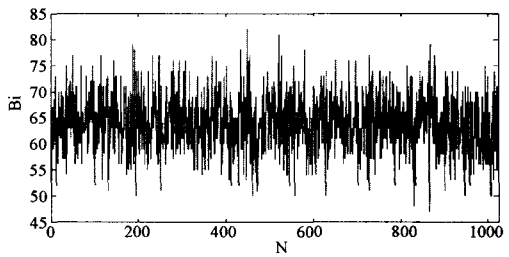

图 2　置乱数分布

另外，在 $N=256,512,1024,2048$ 的情况下分别进行测试，得到明文 1 比特变化时的各项统计值，如表 1 所列。

表 1　算法的统计性能

N =	256	512	1024	2048
$\bar{B}$	64.09	64.11	63.94	64.12
ΔB	5.701	5.787	5.785	5.740
P(%)	50.13	50.12	49.94	50.14
ΔP(%)	4.466	4.638	4.452	4.415
B_{max}	82	83	82	81
B_{min}	52	46	48	46

由表 1 可知，改进算法的 B_i 和 P 都非常接近理想状况下的 64 比特和 50%的变化概率。算法很充分且均匀地利用了密文空间、明文的任何细微变化。密文从统计上看在密文空间中都是接近等密度的均匀分布，因而攻击者从中得不到任何密文分布的有用信息。另外 ΔB，ΔP 为反映 Hash 混乱与扩散稳定性的指标，它们越接近 0，稳定性就越好。改进算法的 ΔB，ΔP 都很小，能有效保证混乱与扩散能力的稳定。

③ 碰撞性分析

碰撞是指虽然消息不相同但其 Hash 值却相同，即多对一映射。随机选择一段明文，将其 Hash 值保存为 ASCII 符的形式。然后随机改变明文中 1 个比特的值，将其 Hash 值也保存为 ASCII 符的形式。按照文献[3]对两个 Hash 值进行比较，计算它们在相同位置上 ASCII 符相同的个数，并按照式(16)计算两者之间的绝对差异度：

$$d=\sum_{i=1}^{N}|t(e_i)-t(e_i')| \tag{16}$$

其中 e_i 和 e_i'分别为两个 Hash 值中的第 i 个 ASCII 符，函数 $t(.)$表示将ASCII 符转换为对应的数值。重复上述过程 2048

次，得到的最大、最小和平均绝对差异度如表 2 所列。

表 2　两个 Hash 值之间的绝对差异度

	最大值	最小值	平均	每字符平均
绝对差值	2139	677	1384	86.675

④ 运算速度分析

在改进的 Hash 算法中，当 CML 中每个格子由状态 n 变为状态 $n+1$ 时，需要进行 40×3 次 Logistic 映射运算。当输入文本包含 N 个字符时，则需进行 $40\times3N$ 次运算。由于 CML 有 16 个格子，因此需要计算的 Logistic 映射总次数为 $1920N$。这表明产生 Hash 值时需要计算的 Logistic 映射次数与消息长度之间为线性关系，能有效避免消息较长时计算量急剧增加的问题。

另外，当消息长度仅为 1 个字符时，由于设置了初始状态变量 $IV[1\cdots16]$，本文算法仍然可以保持时空混沌的优良特性，不会退化为单个的 Logistic 映射。

同时，在改进后的算法中，产生 Hash 值时，对于文献[7]中的攻击，当消息被重复扩展时，消息中的每个 ASCII 码值，加上它所对应的重复扩展形式，都会被用于调整格子的状态值，因此在产生 Hash 值时，它们迭代的次数是不相同的，产生的 Hash 值也不会相同，从而能够有效防止这种形式的碰撞攻击。

综上所述，改进的算法根据 Lyapunov 指数谱和 CML 的状态来确定时空混沌中的各参数值，有效地保证了系统处于完全发展的混沌状态，从而使算法具有很好的单向性和安全性。同时，改进后的算法的时间复杂度与消息的长度呈线性关系，随着消息的增加能有效减少 Hash 值的计算时间。

结束语　本文的研究对象是基于时空混沌的 Hash 函数构造方法。首先介绍了一种以时空混沌为基础的构造 Hash 函数的典型方案，并分析了它的不足和安全性问题。然后，提出了一种改进的 Hash 函数构造方法。改进后的算法具有很好的安全性和灵活性。同时与原算法相比，还具有更好的统计特性和更高的效率，有效地弥补了原算法中存在的碰撞漏洞。

参 考 文 献

[1] Wang X, Feng D, Lai X, et al. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD[C]// Rump Session of Crypto'04 E-print. 2004

[2] Wang X, Lai X, Feng D, et al. Cryptanalysis of the Hash Functions MD4 and RIPEMD[C]// Proceedings of Eurocrypt'05. Aarhus, Denmark, 2005: 1-18

[3] Wong Kwok-Wo. A combined chaotic cryptographic and hashing scheme[J]. Physics Letters A, 2003, 307: 292-298

[4] Yi Xun. Hash function based on chaotic tent maps [J]. IEEE Transactions on circuits and systems-II, 2005, 52(6): 354-357

[5] Davies R W, Price W L. Digital Signature-an Update[C]// Proceedings International Conference on Computer Communications. Sydney, Elsevier, North-Holland, 1984: 843-847

[6] Matyas S M, Meyer C H, Oseas J. Generating Strong One-way Functions with Cryptographic Algorithm [J]. IBM Technical Disclosure Bulletin, 1985, 27(10): 5658-5659

[7] 王继智，王英龙，王美琴. 一类基于混沌映射构造 Hash 函数方法的碰撞缺陷[J]. 物理学报，2006，55(10)：5048-5054

[8] Stallings W(美)著. 密码编码学与网络安全：原理与实践[M]. 刘玉珍，等译. 北京：电子工业出版社，2004

[9] 张瀚，王秀峰，李朝晖，等. 基于时空混沌系统的单向 Hash 函数构造[J]. 物理学报，2005，54：4006-4011

[10] 杨维明. 时空混沌和耦合映象格子[M]. 上海：上海科技教育出版社，1994

[11] Kaneko K. Pattern dynamics in spatiotemporal chaos : pattern selection, diffusion of defect and pattern competition intermittency[D]. 1989, 34: 1-41

[12] Ding Mingzhou, Yang Weiming. Stability of synchronous chaos and on-off intermittency in coupled map lattices[J]. Physical Review E, 1997, 56: 4009-4025

[13] Kohda T, Tsuneda A. Statistics of chaotic binary sequences[J]. IEEE Transactions on Information Theory, 1997, 43(1): 104-112

[14] Habutsu T, Nishio Y, Sasase I, et al. A secret key cryptosystem by iterating a chaotic map[C]// Advances in Cryptology-EuroCrypt'91, Lecture Notes in Computer Science 0547. Berlin: Spinger-Verlag, 1991: 127-140

[15] Bhattacharyya G K, Johnson R A. Statistical Concepts and Methods[M]. Tronto: John Wiley and Sons, 1997

[16] 赵德勤，刘曾荣. 基于追踪控制的混沌系统广义同步[J]. 重庆邮电大学学报：自然科学版，2007，19(6)：759-761

(上接第 187 页)

结束语　本文提出了改进的非线性 KPCA 方法应用于化工过程的故障检测。一方面，所提出的方法结合了小波变换与 KPCA，对 KPCA 的性能进行了扩展。另一方面，引进特征向量选择算法，降低了计算费用，减少了耗时。通过非线性过程仿真表明，所提出的方法在故障检测方面优于 KPCA 方法。该方法具有有效性和实用性。

参 考 文 献

[1] SchölkopfF B, Smola A J, Mjuller K. Nonlinear component analysis as a kernel eigenvalue problem[J]. Neural Computation, 1998, 10(5): 1299-1399

[2] Cui Peiling, Li Junhong, Wang Guizeng. Improved kernel principal component analysis for fault detection[J]. Expert System with Application, 2008, 34(2): 1210-1219

[3] Choi S W, Lee I B. Nonlinear dynamic process monitoring based on dynamic KPCA[J]. Chemical Engineering Science, 2004(59): 5897-5908

[4] Choi S W, Lee C, Lee J M, et al. Fault detection and identification of nonlinear processes based on KPCA[J]. Chemometrics and Intelligent Laboratory Systems, 2005(75): 55-67

[5] Kosanovich K A, Plovoso M J. PCA of wavelet transformed process data for monitoring[J]. Intelligent Data Analysis, 1997(1): 85-99

[6] Donoho D L. Denosing by soft-thresholding [J]. IEEE Transactions on Information Theory, 1995, 41(3): 613-627

[7] Baudat G, Anouar F. Kernel - based methods and function approximation[C]// Proceedings of international conference on neural networks. Washington, DC. 2001: 1244-1249

[8] Downs J H, Viogel E F. plant-wide industrial process control problem[J]. Computers Chemical Engineering, 1993, 17(3): 245-255