基于重构构件的安全协议重构择优技术研究

李 玲 杜学绘 包义保
(解放军信息工程大学 郑州 450004)

摘 要 安全协议可重构实现是提升其安全性能和计算性能的有效方法。在深入分析大量现有安全协议体系结构的基础上,提出了一种基于可重构构件的安全协议高性能实现架构,并且针对该架构中可重构构件库择优优化这一关键问题,提出了一种择优方法。该方法基于改进的带权集合覆盖优化算法,结合启发式优化搜索思想,实现了安全协议可重构实现中优化资源使用与减少重构时间的目的。

关键词 重构构件,安全协议,技术框架,构件择优
中图法分类号 TP309 **文献标识码** A

Research on Optimization Technology of Reconfigurable Security Protocols Based on Reconfigurable Component

LI Ling DU Xue-hui BAO Yi-bao
(PLA Information Engineering University, Zhengzhou 450004, China)

Abstract Reconfigurable implementation of security protocol is an effective way to enhance its computing and safety performance. In this paper, based on analyzing a large number of existing security protocol architecture, we presented a high-performance implementation architecture of security protocol based on reconfigurable component. We presented a best method to optimize the reconfigurable component library for it is a critical issue in this architecture. By improving the optimization algorithms of coverage set with the right and combining ideas of heuristic optimization search, the method achieves a goal of reducing resource and time in the reconfigurable implementation of security protocol.

Keywords Reconfigurable component, Security protocols, Technology architecture, Component optimization

1 引言

当今信息高速公路不断地向世界各地延伸,信息技术革命开辟了前所未有的广阔空间,对信息系统的快速响应与安全需求提出了更大的挑战。随着不同功能、特点的安全协议应运而生,如何让安全系统快速准确地响应形形色色的外部环境的变化,提高安全协议的处理速率必将成为一个不容忽视的主题。而可重构计算的迅猛发展对不断前进的信息系统所面临的困难来说正如天降甘霖。可重构计算(Reconfigurable Computing)是指利用系统中的可重用资源,根据不同应用的需要重新构造一个新的计算平台,计算性能达到或接近专用硬件计算的性能,使之不但具有软件实现的灵活性,同时具有硬件实现的高性能,填补了传统的软、硬件应用实现方案之间的空白。

由此可见,要解决安全协议处理瓶颈的难题,必须打破传统的运行模式,在可重构计算的基础上采用一种资源可共享、功能可重构的面向服务需求的可变的柔性技术体系。已有不少研究利用可重构思想对网络协议进行重构,如 Conte A, Stefan B 等人讨论了应用层协议重构框架[4,5];T. Isobe 给出了可重构硬件实现 SSL 架构[9];Ahmad Salman 等人通过重构技术,给出了一种基于加密算法硬件加速的 IPSec 实现架构[8];Moon J T 研究了基于可重构硬件的物理层协议重构方法[6]。种种数据表明,使用可重构技术实现诸如 IPSec、SSL 等安全协议,其性能可达到甚至超过 ASIC 的能力。

在可重构安全协议方面,目前还缺乏对重构方法特别是基于重构构件的安全协议及重构构件库择优及安全性验证方面的探究。本文借鉴软件工程中的构件化思想[10],将构件的思想引入到可重构安全计算中来,把重构构件类比为软件工程中的构件,通过对不同服务的支持,装配或释放相应的构件来构建新型服务。即允许以不同构件组合方式启用功能模块来完成多种任务。故而提出了基于重构构件的安全协议实现,该系统具有功能模块化、可扩展能力强、便于开发等优点。

然而,当网络环境与服务需求发生改变时,根据具体的需求从重构构件库中调用相应数量与功能的重构构件,进行适当形式的组合以形成所需的功能。如何实现服务能力平滑、快速的升级,达到高效利用系统资源,提高提供多种服务需求的能力,是研究基于重构构件的安全协议必须解决的问题。因此,如何从重构构件库中选择一系列重构构件,并给出其在

本文受国家高技术研究发展计划项目(863计划)(2012AA012704)资助。
李 玲(1989—),女,硕士生,主要研究方向为重构安全协议,E-mail:2627670738@qq.com;杜学绘(1968—),女,博士,教授,主要研究方向为网络与信息安全;包义保(1976—),男,博士,副教授,主要研究方向为网络与信息安全。

可编程逻辑器件中的最优组合方式，能在所占资源与重构时间代价最小的情况下提供服务，是值得研究的问题。故本文改进了带权集合覆盖问题的求解算法，并采用启发式算法对择优问题展开研究。

2 重构构件及安全协议高性能实现架构

2.1 重构构件

本节引入重构构件的概念，它是能为特定应用裁剪的，易于实现新的协议且能够快速成形。重构构件可以重用于不同的环境中，能满足应用的特定需求。另外，重构构件能根据需求而改变，既能维持一定的稳定性，又易于扩展。本文基于各类安全协议，深入研究了基于可重构构件的安全协议高性能实现架构的技术。

重构构件即先将服务或任务化分成若干模块，将划分好的模块作为独立的可重用的功能模块，按照既定规则存入重构构件库中。当某一服务或任务到来时，系统便可以从重构构件库中选择符合需求的最优重构构件，根据协议组装方式通过相互关系合作完成此次的服务或任务，显示了极强的灵活性和扩展性。重构构件与一般构件的不同之处在于重构构件的可重构性，即当系统根据服务和重构性能的需求重构出合适的重构构件存入重构构件库中，亦是构件本身的可重构性。目前，对重构构件划分粒度的大小并没有明确的界定，一般情况下，重构构件的粒度越小，协议划分得越细，协议重构构件就越多；反之，亦然。

对于安全协议中重构构件的设计，设计理念的关键在于深入分析协议功能、协议流程、协议要素、实现结构特征等，并在此基础上抽象协议的共同点和差异点。综合分析、归纳出各个协议的共同点、差异点，根据可重构的需求进行合理聚类，设计出重构构件。设计过程分为 3 步：1)将功能单一、差异性较小的基本功能抽象为同一类处理模型；2)将差异性较大但实现难度较小的基本功能分别抽象为不同的处理模型；3)将差异性较大、设计实现难度亦较大的基本功能采用通过其他处理模型的重新划分和定义的方法，达到缩小处理差异性的目的，从而抽象为同一种处理模型。这种对功能性重构构件的提取和聚类，实现了资源共享，进而对现有的服务需求屏蔽不同运行资源的差异。

2.2 安全协议高性能实现架构

通过对重构构件的分析与设计，从系统的功能出发，建立基于重构构件的安全协议高性能实现架构。该架构对系统的环境感知、需求分析、重构实施、安全验证等模块进行了描述，给出图 1 所示的基于可重构构件的安全协议高性能实现架构。从图中可知其实现架构可划分成如下几个层次：第一层，重构信息获取：当可重构安全协议系统的应用需求与外界环境发生变化时，通过对重构环境的感知、监测和评估等来获取有效信息以对应用需求与环境变化进行捕获；第二层，重构决策：分析重构感知层捕获到的信息，进行安全需求分析，匹配重构协议库，生成安全协议逻辑；第三层，重构实施层：根据重构决策层得到的安全协议逻辑，进行软硬件任务划分与调度，启动可重构配置生成、编译、综合、映射等工具，继而从重构构件库中选取合适的重构构件优化设计，生成重构配置，部署到重构平台之中，同时对重构配置与重构结果进行分析和安全性验证；第四层，重构基础平台层：为可重构系统提供基础的软硬件平台，将上层得到的重构配置部署到软硬件平台中，配置模块则根据运行需求动态改变协议的运行参数、运行状态，从而实现对应用需求与环境变化的快速响应。

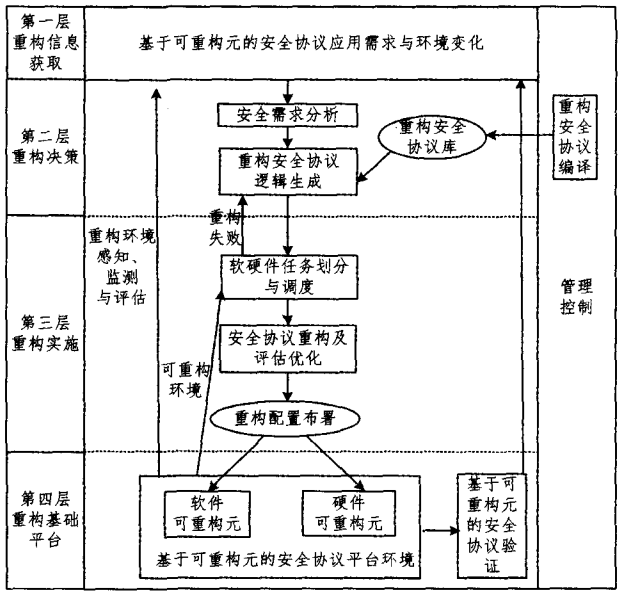

图 1 基于重构构件的安全协议高性能实现架构

从基于重构构件的安全协议高性能实现架构的整体出发，在把握整体框架后，中心是做好重构实施阶段的实现，重点为研究安全协议重构及评估优化，难点是重构过程中可重构构件库择优问题的剖析，完成这一工作是极具现实意义的。

3 重构构件的拆分及其数学模型

通过对重构构件与安全协议系统实现架构的描述中可知，重构构件的拆分是重构实施的基础。本文在深入研究各类安全协议的基础上，提出了安全协议系统中可重构构件拆分的原则及重构构件的数学模型，为后续的研究工作奠定了基础。

3.1 基于重构构件的安全协议功能模块的拆分

在掌握协议和算法原理之后，需要对协议和算法的结构进行“拆分”。“拆分”过程中要把握的一个关键是粒度问题。在第二节重构构件技术中提到重构构件的拆分粒度问题，了解到重构构件的粒度可随服务的需求灵活把握，简单地说，可以将重构构件库中多个细粒度的重构构件组装成一个粗粒度的重构构件，再放回重构构件库中。若要实现细粒度的可重构，则“拆分”的粒度应尽可能小，最小可重构粒度甚至可以是一个逻辑门或触发器，基于此粒度构成的可重构构件具有很高的灵活性，可以满足多种协议和算法的需要。同时也由于这种拆分方式粒度细，控制复杂，使得可重构设计过程速度较慢。故要设计高速可重构系统，“拆分”的粒度不能太小。

对安全协议的“拆分”，下面以 IPSec 为例，以粗粒度的方式进行重构构件的拆分。本文主要依靠功能来拆分重构构件，IPSec 功能可分为数据的接收和发送、网络协议解析、ESP 协议处理、AH 协议处理、策略和 SA 管理包括存储和查询、

加脱密处理等，分成各个模块之后，依照高内聚低耦合的原则，在模块中合适地选择粒度。由此，我们得到基本的处理模块(粒度有待讨论，尚未细分)，如表1所列。

表1 IPSEC协议基本处理模块

协议	IPSEC
基本处理模块	网络协议解析模块
	ESP协议处理模块
	AH协议处理模块
	策略模块
	SA管理模块
	数据的接收和发送模块
	加脱密处理模块
	其它模块

为了更好地理解安全协议中可重构构件的拆分，以安全协议中的加脱密处理模块为研究对象，如本文以对称密码算法中典型的DES和AES算法为例，采用细粒度的划分方式。为了实现快速高效的重构，对算法中完整的运算模块进行“拆分”，通过对算法运作流程与实现进行研究，得到了DES和AES算法的基本运算，如表2所列。

表2 DES和AES密码算法包含的基本运算模块

算法	DES	AES
基本运算	64 * 64 初始置换	128 位 XOR
	32 * 48 扩展置换	8 * 8S 盒变换
	32 位 XOR	32 位循环移位
	6 * 4S 盒变换	32 位列混合运算
	32 * 32 盒置换	32 位异或
	28 位循环移位	
	56 * 48 压缩置换	

3.2 安全协议重构构件的数学模型

为了便于进行数学描述，用二元组来表示重构构件 $e_i=\langle F_i,S_i\rangle$，其中 $F_i(F_i\subseteq U)$ 是重构构件 e_i 的功能集，$S_i(S_i\subseteq S)$ 为重构构件 e_i 的代价向量。在划分重构构件粒度的同时，涉及到内聚度与耦合度的研究，在此不做重点讨论，显然地各个重构构件 e_i 的功能之间会有重叠，故而设总功能集 U 由相互独立的 m 个原子功能构成，即 $U=\{f_1,f_2,\cdots,f_m\}$，那么重构构件 e_i 的功能集 F_i 就等于功能相互独立的若干个原子功能的 f 的并集。设总的可用资源用向量 $W=[w_1,w_2,\cdots,w_k]$ 表示，在重构构件库中选择 e_i 时花费了 h 种资源，即重构构件 e_i 的代价向量 $w_{e_i}=[w_1,w_2,\cdots,w_h]$。设在重构过程中被选中的重构构件 e_i 中包含目标功能的个数为 k_i。设在重构构件库中选择 e_i 时重构的重构时间为 T_i，则重构构件 e_i 的代价向量 $S_i(S_i\subseteq S)$ 由资源代价与时间代价两部分组成。与此同时，本文考虑到不同的服务需求其资源与时间的权重各不相同，引入代价权重向量。其中选择重构构件 e_i 占用不同资源代价权重用 $\lambda_{w_{e_i}}$ 表示，时间代价权重用 $\lambda_{t_{e_i}}$ 表示。因考虑到资源与重构时间的权重为不失偏颇，在重构构件数学描述中引入 k 来表示重构构件包含目标原子功能的个数。对要素进行数学描述之后，经过分析对重构构件建立数学化模型，提出如下的重构构件数学模型：

$$e_i=\langle F_i,S_i\rangle \tag{1}$$

$$F\subseteq U,U=\{f_1,f_2,\cdots,f_m\}\text{且 }f_1\cap f_2\cap\cdots\cap f_m=\varnothing \tag{2}$$

$$S_i=(w_{e_i}\lambda_{w_{e_i}}+t_{e_i}\lambda_{t_{e_i}})/k_{e_i} \tag{3}$$

式(1)表示一个特定的构件由功能与代价组成；式(2)表示该构件功能集是系统全功能集的一个覆盖；式(3)表示该构件的代价由时间、资源与所含目标构件数等方面共同决定。

4 重构构件库择优分析

4.1 安全协议重构过程简述

从基于重构构件的安全协议实现架构可知，在整个可重构网络构建和重构过程中，，就单从工程实现的角度看，其过程可划分为生成重构需求、拟定重构方案、划分重构任务、实现协议重构满足安全需求。由图2可知，在实现安全协议重构过程中，重构构件的选取与性能评估、优化是不容忽视的问题。本小节就系统如何在重构构件库中寻求最优目标，从占用资源与重构时间双重约束进行研究讨论。

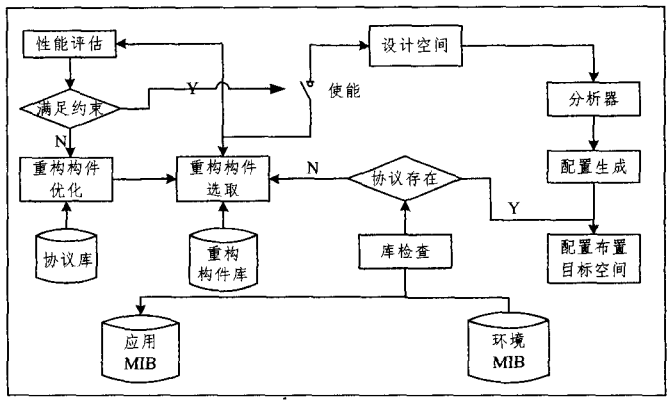

图2 安全协议重构实现过程

重构构件的重构分成两个部分：一是重构构件的选取与优化；二是配置文件的生成。本文主要研究重构构件的选取与优化。

第一阶段是重构构件的选取与优化。首先，应用与环境的变化将触发安全协议可重构系统对协议库进行检查，判定是否进行协议重构。如果协议已存在，则直接在目标空间中处理请求；若协议不存在，则协议重构机制被触发，即在构件库中选取符合条件的重构构件进行性能评估与选优操作。此模块将基于诸如处理能力、资源代价、重构时间等约束来给出一个估计值，从而决定重构构件参与重构操作与否，且评估选优与优化过程是以一种周期的方式执行直到重构构件符合需求为止。

第二阶段为配置文件的生成。符合各种约束的重构构件的规范化形式将被送往设计空间，设计空间以保证逻辑的正确性为前提设计协议的逻辑关系；然后传送给分析器，最终由分析器得到协议的配置文件，将其部署到目标空间中来处理请求。

4.2 安全协议重构构件库择优问题

在图2中可以看出，重构构件选取及评估优化在整个重构过程中占据极其重要的位置。如何从重构构件库中提取符合条件的重构构件并以最佳的方式组合完成安全协议，达到满足服务的同时也更节约资源与重构时间的目的，是研究基于重构构件对安全协议进行重构必须直面的问题。下面就对

安全协议重构过程中的择优问题展开研究。

前面所讨论的重构构件的数学模型是本节中讨论问题的铺垫，容易得到总的资源代价问题 S 可以描述成：

$$\min S=\min\sum S(e_i)$$

其表示重构过程中的优化目标，即让总的硬件资源与重构时间代价最小。一方面，参与运算的 i 表示其对应的重构构件库中的重构构件是被选中的状态，同时由这些选中的重构构件组成的功能集必需覆盖目标组装电路的功能集。另一方面，$\forall i\in l, j\in[1,m]$，$\sum_{i\in l} w_j^i \leqslant w_j$，表示所选择的重构构件的各项资源代价都不能够超出总的可用资源约束。

由上述描述可知，安全协议组装过程中的优化问题是一个带权的集合覆盖问题。所谓集合覆盖问题，可描述如下：

设非空集合 U，$H=(F_1,F_2,\cdots,F_m)$，其中 $F_i\subseteq U$ 且 $F_i\neq\varnothing$，若 $F_1\cup F_2\cup F_3\cup\cdots\cup F_i\cdots\cup F_m=U$，则称 H 是集合 U 的一个覆盖。例如：设 $U=\{4,8,9,15,16,20\}$，定义 U 的如下子集：

$F_1=\{x\mid x\in A$ 且 x 能被 3 整除$\}$

$F_2=\{x\mid x\in A$ 且 x 能被 4 整除$\}$

$F_3=\{x\mid x\in A$ 且 x 能被 5 整除$\}$

根据题意得：$F_1=\{9,15\}$，$F_2=\{4,8,16,20\}$，$F_3=\{15,20\}$。$\{F_1,F_2\}$ 是 U 的划分，也是 U 的覆盖，但 $\{F_2,F_3\}$ 不是 A 的覆盖。

同时定义一个权值 W，带权集合覆盖问题的目标是找出一个 H 的子集 E，使 E 覆盖整个 U，并使权值最小，即 $\min S=\min\sum S(e_i)$。带权集合覆盖问题是 NP 难解问题，因此难以找到多项式时间的求解算法。本文对解决此问题的覆盖优化算法进行改进并采用启发式算法对优化问题进行研究。具体算法考虑如下，U 为相互独立的目标原子功能的集合。每次选重构构件 $e_i\cap U\neq\varnothing$，其资源与重构时间代价分别 w_{e_i}，t_{e_i}，其权值分别为 $\lambda_{w_{e_i}}$，$\lambda_{t_{e_i}}$，使得选择函数 $select(e_i)$ 最小，然后将 $e_i\cap U$ 中的元素从 U 中去掉，并取下一个集合，直到 U 被完全覆盖为止。算法设计如下：

Input：目标功能集 U

Output：优化结果集 E

1. E=∅

   While U≠∅ do

   Select $e_i\cap U\neq\varnothing$;

   that minimize

$$select(e_i)=\frac{\dfrac{w_{e_1}w_{e_2}\cdots w_{e_n}}{\sum_{i=1}^{n}\dfrac{w_{e_1}w_{e_2}\cdots w_{e_n}(w_{e_i}\lambda_{w_{e_i}}+t_{e_i}\lambda_{t_{e_i}})}{k_{e_i}}}\cdot\dfrac{(w_{e_i}\lambda_{w_{e_i}}+t_{e_i}\lambda_{t_{e_i}})}{k_{e_i}}$$

2. U 为相互独立的原子功能的集合。在重构构件库中选一个含有目标功能的元素 e_i

3. U←U−$e_i\cap$U

4. E←E∪{e_i}

5. Return E

从算法中可以注意到，包含元素多的集合被选中的概率较大，而在每一轮循环中，算法以较大概率选择权重较小的集合。算法结束输出的结果 E 已经是目标功能集的一个覆盖了，带权覆盖问题的分析就到此结束，但是不难发现由此得到的集合 E 内可能有冗余的元素。下面本文将结合启发式算法的思想对结果集合 E 进行操作，所完成的功能是去除不相关和冗余的重构构件，从而最大限度地降低目标电路的实现代价，算法设计如下：

Input：目标功能集 U，优化结果集 E

Output：优化结果 E′

1. ∀i∈[1,m]，counter[i]←0，flag[i]←0

2. For i=1 to n

3. If e[i]==1 Then

4. For j=1 to m

5. If $f_j\in F_i$ then

6. counter[j]←counter[j]+1

7. For i=1 to m

8. If $f_i\in$U then flag[i]←1

9. 对集合 E={e_i | e[i]=1}中的重构构件，按|F_i|的大小进行降序排列；

10. For each $e_i\in$E

11. For j=1 to m

12. If f_j 属于 e_i 的功能集，则 f_vec[j]←1，否则 f_vec[j]←0，

13. If counter−f_vec≥flag then 将重构构件 e 从集合{e_i | e[i]=1}中删除

    counter←counter−f_vec;

14. 若 e 在重构构件库中的序号为 i，则 e[i]←0；

15. Return E

上述算法中，功能计数器 counter 和目标功能标志 flag 长度取为 m（相互独立的原子功能的个数），对于某一重构构件集 E，将其所代表的重构构件选择方案中各个重构构件的各项原子功能累加到功能计数器 counter 中。目标功能标志 flag 表示所有 m 项原子功能中哪些是目标电路所要求具备的功能。本文将带权集合覆盖求解问题与启发式算法相结合，在完成预期功能的基础上减少了重构构件的冗余，从而优化了安全协议的重构代价。

5 仿真实验

仿真软件平台采用 MATLAB 2012b，所采用的 PC 机主频为 2.5GHz，内存为 4GB。对基于重构构件的覆盖优化算法与启发式算法进行仿真实验。设原子功能总数取 $m=40$，重构构件库中的重构构件个数取 $n=20$，资源总数取 $k=4$。由于没有特殊的服务需求，实现假定资源代价权重为 0.5，重构时间代价权重为 0.5。重构构件功能的生成方法如下：首先，对于任意重构构件 e_i，生成一个在区间[1,10]内服从均匀分布的随机数 x 作为重构构件的原子功能数；然后，从原子功能中随机选择彼此不同的 x 项作为 e_i 的原子功能；最后，考虑到某些原子功能可能没有被任何元构件选择到，可再将这些原子功能随机指派给某些重构构件，从而保证各项原子功能至少能够被一个重构构件所拥有。系统随机生成重构构件的时间代价精确到小数点后千分位。对于各项原子功能，随机生成其各项硬件资源的代价值；重构构件的硬件资源代价值为其所拥有的原子功能的代价之和。

以下是第一次运行仿真实验的实例，随机产生的重构构件库如下：

$e_1=\{f_7,f_{10},f_{15},f_{28},f_{32},f_{34},f_{38}\}$

$e_2=\{f_{22},f_{26},f_{31}\}$

$e_3=\{f_{12},f_{30},f_{36}\}$

$e_4=\{f_{11},f_{15},f_{20},f_{22},f_{24},f_{26}\}$

$e_5=\{f_5,f_7,f_8,f_{18},f_{27},f_{29},f_{31},f_{33},f_{35}\}$

$e_6=\{f_{27}\}$

$e_7=\{f_{11},f_{12},f_{18},f_{23},f_{26},f_{30},f_{32},f_{37}\}$

$e_8=\{f_5,f_{14},f_{24},f_{29},f_{36}\}$

$e_9=\{f_{17},f_{40}\}$

$e_{10}=\{f_4,f_{17},f_{36}\}$

$e_{11}=\{f_{15},f_{19},f_{21},f_{23}\}$

$e_{12}=\{f_5,f_9,f_{12},f_{19},f_{26}\}$

$e_{13}=\{f_1,f_7,f_8,f_{14},f_{19},f_{20},f_{21},f_{24},f_{33},f_{37}\}$

$e_{14}=\{f_1,f_3,f_{12},f_{14},f_{21},f_{23},f_{32},f_{36}\}$

$e_{15}=\{f_{11},f_{13},f_{14},f_{34},f_{39}\}$

$e_{16}=\{f_9,f_{23},f_{31}\}$

$e_{17}=\{f_{16},f_{18},f_{29}\}$

$e_{18}=\{f_1,f_6,f_9,f_{21},f_{39}\}$

$e_{19}=\{f_2,f_3,f_4,f_6,f_{10},f_{11},f_{14},f_{22},f_{32},f_{33}\}$

$e_{20}=\{f_1,f_{20},f_{25},f_{27},f_{29},f_{35}\}$

目标功能集为 $U=\{f_1,f_5,f_7,f_{14},f_{18},f_{21},f_{23},f_{29},f_{31},f_{35},f_{36}\}$。此功能的最优解为 $E=\{e_5,e_{14}\}$，总代价为 18.21。这里重构代价表示在特定的系统下代价的比值情况，如本实验将算法代价与目标代价在本系统中的相对值做一个量化分析，故无需单位。对上述重构构件库运行覆盖优化算法，生成优化解 $E=\{e_5,e_8,e_{14}\}$，总代价为 23.45。对上述解运行启发式算法，生成优化解为 $E=\{e_5,e_{14}\}$，总代价为 18.21。由此可见，求解最优解时此算法是有效的。图 3 给出了 10 次仿真实验中目标最优解与算法优化解的比较，实验发现本文提出的择优算法是行之有效的。

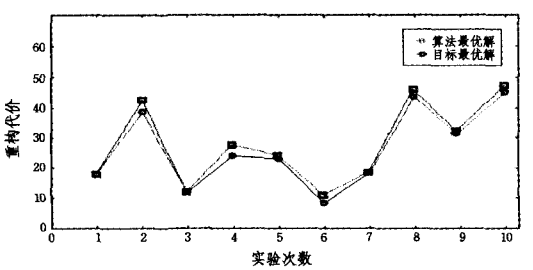

图 3　算法优化解与目标最优解的比较

为了描述优化解的质量，使用 G 来表示优化过程中优化解与最优解的相对偏差，即 $\Delta G=\left|\frac{G_E-G_{OP}}{G_{OP}}\right|$。其中 G_E 为算法求得的优化解，G_{op} 为目标最优解。从而得到上述 10 次仿真实验的算法优化解与最优解的相对偏差，如图 4 所示，算法优化解与最优解的相对偏差值约为0.061。由此可知，算法优化解与最优解相差不大，能达到预期的效果。

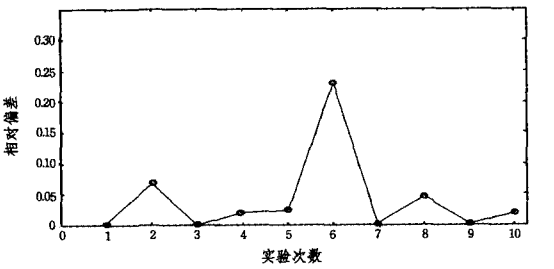

图 4　算法优化解与最优解的相对偏差

结束语　本文在重构安全协议的基础上引入了重构构件的概念，将重构构件作为重构的基本元素，并对基于重构构件的安全协议高性能实现架构进行研究，从整体上把握了安全协议的重构过程。同时提出了重构构件的数学描述，并在此基础上对重构过程中的关键技术，如重构构件的拆分问题、重构构件库择优问题进行讨论，采用覆盖优化算法与启发式算法相结合的方案解决以节约资源与时间为目的的优化问题。由实验可知，通过本文中提出的优化方法确实能找到最优解，达到了资源与时间的双赢。

参 考 文 献

[1] Chen Hui, Zhou Chun-jie, et al. Management of the Reconfigurable Protocol Stack Based on SDL for Networked Control Systems[J]. Information Technology Journal, 2010, 9(5): 849-863

[2] Niamanesh M, Sabetghadam S, Yousefzadeh R, et al. Design and implementation of a dynamic-reconfigurable architecture for protocol stack[J]. International Symposium on Fundamentals of Software Engineering, 2007, 4767: 396-403

[3] 刘强，汪斌强，徐恪. 基于构件的层次化可重构网络构建及重构方法[J]. 计算机学报，2010，33(9)

[4] Conte A, Anquetil L P. Design for application protocol stack framework [C]// IEEE Int Conf on Communications. New Orleans, 2000: 565-570

[5] Stefan B, Ken M, Brian W. Application-compliant networking on embedded systems [C] // Proc of 5th IEEE Int Workshop on Networked Appliances. Manchester, 2002: 53-58

[6] Moon J T, Kim J S, Kim J B, et al. A hardware implementation of distributed network protocol[J]. Computer Standards and Interfaces, 2005, 27(3): 221-232

[7] Casado R, Bermudez A, Quiles F J, et al. Influence of network size and load on the performance of reconfiguration protocols [C]// IEEE Int Symposium on Network Computing and Applications. Cambridge, 2001: 46-57

[8] Salman A, Rogawski M, Kaps J-P. Efficient Hardware Accelerator for IPsec Based on Partial Reconfiguration on Xilinx FPGAs [C] // Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs (RECONFIG). IEEE Computer Society, Washington DC, USA, 2011: 242-248

[9] Isobe T, Tsutsumi S, et al. 10 Gbps implementation of TLS/SSL accelerator on FPGA[C]// IEEE 18th International Workshop on Quality of Service(IWQOS). 2010: 1-6

[10] 毛晓光，邓勇进. 基于构件软件的可靠性通用模型[J]. 软件学报，2004，15(1)：27-32