5) 终止消除规则 SSPE5:

$$\frac{\Gamma \mapsto m:\bot}{\Gamma \mapsto \{abort^c m\}:C};$$

6) 量词∀消除规则 SSPE6:

$$\frac{\Gamma \mapsto m:\forall x.A}{\Gamma \mapsto \{mt\}:[t/x]A};$$

7) 量词∃消除规则 SSPE7:

$$\frac{\Gamma \mapsto m:\exists x.A;\Gamma,u:[a/x]A \mapsto [a/x]n:C}{\Gamma \mapsto \{let\langle x,u\rangle = \min\ n\}:C}.$$

SSPE 合成规则是由 SSP 基本推理规则推导而来,其完备性证明参见文献[3].

2 面向问题的等级化安全重构元描述

安全重构元作为可重构安全计算系统的基本元素,是搭建系统的"积木".安全重构元描述处于自动推理系统的第 1 个级别,其主要目的是提供一种便于用户理解的方式建立安全重构元和表达安全需求,并且能够高效地转换为第 2 级别上的逻辑语言.本节依据已建立的 SSPE 逻辑系统设计了安全重构元描述方法,以下简称 SRCS(security reconfigurable component specification).

定义 2(安全重构元(security reconfigurable component,简称 SRC)). 安全重构元是具有独立功能、与上下文紧密相关、接口具有统一描述规范、能够提供多种不同等级安全服务、具有逻辑推理能力的软件实体.安全重构元可以形式化地描述为一个四元组⟨*ID*,*MetaInterface*,*Function*,*Detail*⟩,其中:*ID* 是安全重构元的唯一标识;*MetaInterface* 是元接口,描述了安全重构元接口变量和接口变量之间的内部数据依赖关系以及安全重构元之间的绑定关系等,是推理的基本依据;*Function* 是功能模块集合,表示安全重构元数据依赖关系的具体实现;*Detail* 是安全重构元的描述信息,包括功能、安全属性、可靠性等.

定义 3(元接口 *MetaInterface*). *MetaInterface* 是一个五元组⟨*MI*,*InterfaceVariable*|*constant*|*Metavariable*,*Binding*,*Relation*,*SecProperlist*⟩,其中,

- *MI* 是元接口名字;
- *InterfaceVariable* 是接口变量,包括安全重构元的输入变量和输出变量,其描述形式如下:

  InterfaceVariable::=*Indentifier*::*Specifier*.

  其中,*Indentifier* 是接口变量的唯一名字,对应于逻辑中一个命题变量;*Specifier* 是接口变量的类型,类型可以是编程语言中的任意基本类型,也可以是元接口或安全重构元标识:

  Specifier::=*int*|…|*float*|*MetaInterface*|*SRC*.
- *Constant* 是常量,对应于逻辑中的断言.其描述形式:

  Constant::=*InterfaceVariable*{*val*}.

  其中,*val* 是目标程序语言中的常量;
- *Metavarible* 是元变量,元变量是一种特殊的接口变量,元变量的值由安全重构元实例或元接口实例指派,其形式如下:

  Metavarible::=*Metavarible*=(*MetInterfacename*|*SRCname*).

  其中,*MetInterfacename* 是元接口标识,*SRCname* 是安全重构元标识,实例的具体实现则在合成过程中进行计算.元变量可能出现在 *Binding* 和 *Relation* 中;
- *Binding* 是接口变量之间的绑定关系,其描述形式:

  Name::=*Indentifier*⌊.*Name*⌋;

  Binding::=*Name*=*Name*.

  表示两个接口变量的实现必须具有相同的值.其中,*Name* 可以是名字标识 *Indentifier* 也可以是复合名字.例如,*Encrypt.IPpacket* 表示接口变量 *Encrypt* 的接口变量 *IPpacket*;

- *Relation* 表示接口变量之间的数据依赖关系,对应于逻辑中的一个公理.

Relation 描述了安全重构元能够提供的所有的计算能力,通常用蕴含式表示:

$$Relation::=[IndentifierList[,SubtaskList]]\rightarrow Indentifier[|Alternatives]|ERROR\{realization\}.$$

其中,

$$IdentifierList::=Identifier[,IdentifierList];$$
$$Alternatives::=Identifier[|Alternatives];$$
$$SubtaskList::=Subtask[,SubtaskList].$$

蕴含符号左部代表输入参数,可以是接口变量或子任务,*IdentifierList* 是的接口变量列表,*SubtaskList* 代表子任务列表.蕴含符号右部代表输出参数,可以是接口变量 *Identifier* 或其他可选参数 *Alternatives*,也可能是 *ERROR*,*ERROR* 是一个常量,表示错误处理.*realization* 对应一个实现程序.*Relation* 表达了当所有的输入参数已知或是可计算时,它的实现 *realization* 是可用的,输出形式为蕴含式右部的参数列表.

- *SecProperties* 是安全属性,它描述了安全重构元的提供的安全服务类型和安全等级.其形式如下:

$$SecProperty(type,levelx)$$
$$type\in\{confidentiality,integrity,non\text{-}repudiation,access\ \ control,authentication\}.$$
$$levelx\in[0,n],n\in N$$

根据 ISO7498 的定义,安全服务类型包括机密性(confidentiality)、完整性(integrity)、抗抵赖性(non-repudiation)、访问控制(access control)、认证(authentication)这 5 种,每种类型的安全重构元划分成若干等级,用 *levelx* 表示,其中,$levelx\in[0,n],n\in N$,若干不同种类、不同级别的安全重构元组合而成的安全服务形成多样的安全保障等级.

定义 4(子任务 *Subtask*). *Subtask* 是一个没有实现部分的 *Relation* 函数,子任务不能独立存在,一般作为关系 *Relation* 的前置条件出现在蕴含式的左部.其形式如下:

$$Subtask::=[IdentifierList]\rightarrow Identifier\{\varphi\}.$$

IdentifierList 是输入接口变量列表,可以为空;*Identifier* 是输出接口变量;φ表示子任务未知的实现.

定义 5(重构请求). 重构请求定义了一个待求解的安全重构元组合描述ψ,其形式化描述如下:

$$RR::=MetaInterfaceList\mapsto[Assumptions]\rightarrow[MetaInterfaceList.]Identifier\{\psi\},$$
$$MetaInterfaceList::=MetaInterfaceName[,MetaInterfaceList],$$
$$Assumptions::=[MetaInterfaceName.]Identifier[,Assumptions],$$

其中,$MetaInterfaceList\mapsto[Assumptions]$作为安全重构元组合$\psi$的输入;*MetaInterfaceList* 是元接口标识列表,它保证了参与查询评估的元接口数量的有限性,是推理的前提;*Assumptions* 是参与重构过程的输入参数列表,是已知的或可计算的,参数可以是元接口变量,也可以是子任务 *Subtask*;[*MetaInterfaceList.*]*Identifier* 安全重构元组合ψ的输出是接口变量列表.

定义 6(控制变量 *Control*). 控制变量[6]是一个实现函数,用于表示重构运算过程中的循环、选择、递归等控制操作,其形式如下:

$$Control::=[InputParameterList]\rightarrow[OutputParameterList]\{control\}.$$

其中,

$$InputParameterList::=InputParameter[,InputParameterList],$$
$$OutputParameterList::=OutputParameter[,OutputParameterList].$$

InputParameterList 是控制变量的输出参数列表,可以是接口变量也可以是子任务;*OutputParameterList* 是输出参数列表.

下面我们以上述例 1 中的 IPSec 协议为例,说明安全重构元的描述方法.

以安全重构元 *Encrypt* 为例,其 SRCS 描述如下:

```
Encrypt
{
    IPpacket-R :: Bitsteam
    IPpacket-E :: Bitstream                    } Interfacevariable
    Auth :: Authenticate
    Auth=(Authenticate)                        } Metavariable
    Auth.Ippacket-E=Ippacket-E                 } Binding
    IPPacket-R→IPPacket-E{Encrypt}             } Relation
    SecProperty(confidentiality,level2)        } Secproperty
}
```

其中,*IPpacket-R*,*IPpacket-E* 为接口变量;*Auth* 是元变量,*Auth*=(*Authenticate*)表示为元变量 *Auth* 赋予一个 *Authenticate* 实例;*Auth.IPpacket-E=IPpacket-E* 是绑定关系,表示当前安全重构元 *Encrypt* 与一个 *Authenticate* 实例具有一个共享接口变量 *IPpacket-E*;*IPpacket-R*→*IPpacket-E*{*Encrypt*}是一个数据依赖关系 *Relation*,其含义为"*Encrypt* 是已知可构造的,若命题变量 *IPpacket-R* 可构造,则 *IPpacket-E* 是可计算的";*SecProperty*(*confidentiality*, *level*2)表示安全重构元 *Encrypt* 的安全属性,其安全类型是 *confidentiality*,安全等级为 *level*2.

控制变量 *loop*1 的 SRCS 描述如下:

$$Loop1::Classify.IPpacket\text{-}C \wedge Encrypt.IPpacket\text{-}R \rightarrow Authenticate.IPpacket\text{-}A\{\varphi\} \rightarrow result\{Loop1\}.$$

其中,蕴含式左部包含一个子任务 *Subtask*::*IPpacket-R*→*IPpacket-A*{φ},*Loop*1 控制该子任务的重复执行.

3 基于映射关系的逻辑语义转换

本节提出了基于映射关系的 SRCS 向 SSPE 的逻辑语义转换方法,详细阐述了各类表达式如接口变量与常量、绑定、关系、安全属性等的转换规则.

为了保证逻辑系统上证明查找算法的有效性以及可终止性,我们假设安全重构元满足以下属性:设 *MI* 是一个安全重构元,若(x_1,x_2,…,x_k)是构成该重构元的基本元素,则 *MI* 是可计算的当且仅当:

$$MI \rightarrow x_1,x_2,\ldots,x_k \wedge x_1,x_2,\ldots,x_k \rightarrow MI.$$

3.1 符号定义

为了阐述方便,先规定一些符号的含义.

- τ表示目标编程语言的有效类型集合;
- *C* 表示组件(名字)集合;
- *S* 表示接口变量和常量的集合;
- *O* 表示对象前缀集合.对象前缀确保不丢失信息的展开元接口,它是一个复合名字,如 *Encrypt.En*.其构造方式如下:假设系统当前对象前缀为 *IPsec*,要展开一个元接口 *Encrypt*::*MI*,则添加名字 *Encrypt* 到当前对象前缀 *IPsec*,使用 *IPsec.Encrypt* 作为对象前缀来展开元接口 *Encrypt*::*MI*,则元接口中接口变量常量转换为逻辑语言时都需要添加当前对象前缀 *IPsec.Encrypt*,如 *IPsec.Encrypt.En*,从而保证了每一个接口变量常量在展开描述过程中具有一个唯一的名字;
- *a*,*b* 表示描述语言 SRCS 的合法语法结构,其具体语法依赖于它们出现的上下文,既可以是简单名字也可以是复合名字,还可以是一个子任务描述或是公理主蕴含式的左部或右部;
- *v* 表示接口变量和常量的名字的集合;
- *r* 表示接口变量和常量的实现集合;
- *Val* 表示目标编程语言的常量集合;
- *M* 表示由公理实现的模型的名字集合;
- *SP* 表示安全属性集合;
- *id* 表示一个恒等函数,这个函数是可以重载的,适合需要的类型.

3.2 映射函数

定义 7(语义函数 L^P). 设 Sexp 是一个有效的 SRCS 描述表达式,O 是对象前缀的集合,设 *LForm* 代表逻辑语言 SSPE 的有效公式.SRCS 描述表达式的逻辑语义由以下函数定义:

$$L^P: sexp \to (O \to LForm).$$

该公式表示从 SRCS 描述表达式到逻辑语言 SSPE 的映射关系.

定义 8(展开函数 *Unfold*). *Unfold* 是关于一个元接口名子 *MI* 和对象前缀 O 的函数,用于产生一个 SSPE 逻辑公式集合,*Unfold* 函数将语义函数 L^P 应用于给定元接口的每一个描述表达式 *sexp*:

$$Unfold(MI,O)=\{L^P[\![sexp]\!](O) sexp \in sp(MI)\}.$$

其中,*sp* 是一个辅助函数,用于产生元接口 *MI* 的 SRCS 描述表达式集合.

3.3 逻辑转换规则

下面具体定义 SRCS 中各类表达式的逻辑语义.

转换规则 1. 设 $o,o\in O$ 是当前对象前缀,v 是接口变量,且 $v::S$,若 $S\in\tau$,则 $r_s:o.v$,r_s 表示 $o.v$ 的实现,":"代表实现;若 $S\notin\tau$,则 $r_s:o.v\cup Unfold(S,o.v)$,$Unfold(S,o.v)$表示元接口 S 的 SRCS 描述表达式使用新的对象前缀 $o.v$,其形式如下:

$$L^P[\![v::S]\!](o)=\begin{cases} r_s:o.v, & \text{if } S\in\tau \\ r_s:o.v\cup unfold(S,o.v), & \text{otherwise} \end{cases}.$$

转换规则 2. 设 $o,o\in O$ 是当前对象前缀,v 是一个接口常量,且 $v::S\{val\}$,则:

$$L^P[\![v::S\{val\}]\!](o)=L^P[\![v::S]\!](o)\cup L^P[\![\to v\{val\}]\!](o).$$

转换规则 3. 设 $o,o\in O$ 是当前对象前缀,(C)是一个元变量,l 表示公理蕴含式的左部,r 表示公理蕴含式的右部,则:

$$L^P[\![(C)]\!](o)=\begin{cases} \forall wC(w), & \text{if } p=l \\ \exists wC(w), & \text{if } p=r \end{cases}.$$

转换规则 4. 设 $o,o\in O$ 是当前对象前缀,*flalsity* 表示错误处理,则:

$$L^P[\![falsity]\!](o)=F.$$

转换规则 5. 设 $o,o\in O$ 是当前对象前缀,[*control*]是控制变量,则:

$$L^P[\![[control]]\!](o)=control.$$

转换规则 6. 设 $o,o\in O$ 是当前对象前缀,$a=b$ 是元接口的一个绑定关系,*id* 是一个恒等函数,则:

$$\begin{aligned} L^P[\![a=b]\!](o) &= L^P[\![a\to b\{id\}]\!](o)\cup L^P[\![b\to a\{id\}]\!](o) \\ L^P[\![a\to b\{id\}]\!](o) &= id: L^P[\![a\to b]\!](o) \end{aligned}.$$

转换规则 7. 设 $o,o\in O$ 是当前对象前缀,m 是一个公理的实现,则:

$$L^P[\![a\to b\{m\}]\!](o)=\begin{cases} m: L^P[\![a\to b]\!](o), & \text{if } m \text{ is constructor} \\ (r_S^o,m): o\wedge L^P[\![a\to b]\!](o), & \text{otherwise} \end{cases}.$$

公理蕴含式也满足下列转换规则:

$$\begin{aligned} &L^P[\![MI\mapsto a\to b]\!](o)=L^P[\![a\to b]\!](o.MI) \\ &L^P[\![\to a]\!](o)=T\supset L^P[\![a]\!](o) \\ &L^P[\![a\to b]\!](o)=L^l[\![a]\!](o)\supset L^r[\![b]\!](o) \\ &L^P[\![a\to b\{val\}]\!](o)=val: L^P[\![a\to b]\!](o) \\ &L^P[\![a,b]\!](o)=L^P[\![a]\!](o)\wedge L^P[\![b]\!](o) \\ &L^P[\![a\mid b]\!](o)=L^P[\![a]\!](o)\vee L^P[\![b]\!](o) \end{aligned}.$$

转换规则 8. 设 $o,o\in O$ 是当前对象前缀,*SecProperty*(*type*,*levelx*)是元接口安全属性,则:

$$L^P[\![SecProperty(type,levelx)]\!](o)=o.Secproperty(type,levelx).$$

举例:以上述例 1 中 *IPsec* 协议安全重构元 *Encrypt* 为例,将其转换为逻辑语言 SSPE 如下:

// *Extend axioms*
r : *Encrypt*
id : $\forall x.Encrypt(x) \supset Encrypt$
id : $Encrypt \supset \exists x.Encrypt(x)$
// *Interface variables*
X_1 : *Encrypt.IPpacket - R*
X_2 : *Encrypt.IPpacket - E*
X_3 : *Encrypt.Auth*
// *Metavariables* : *Auth* = (*Authenticate*)
Id : $\forall x.Authenticate(x) \supset Encrypt.Auth$
Id : $Encrypt.Auth \supset \exists x.Authenticate(x)$
// *Binding* : *Auth.IPpacket - E* = *IPpacket - E*
Id : *Encrypt.Auth.Ippacket - E* ⊃ *Encrypt.IPpacket - E*
id : *Encrypt.IPpacket - E* ⊃ *Encrypt.Auth.Ippacket - E*
// *Relation* : *IPpacket - R* → *IPpacketE*{*Encrypt*}
Encrypt : *Encrypt.IPpacketR* → *Encrypt.IPpacket - E*
// *SecProperty* : *SecProperty*(*confidentiality*,*level*2)
Pr : *Encrypt.SecProperty*(*confidentiality*,*level*2)

其中,扩展公理 *Extend Axioms* 用于引入一个 *Encrypt* 安全重构元实例,*id*:$\forall x.Encrypt(x)\supset Encrypt$ 和 *id*:$Encrypt\supset \exists x.Encrypt(x)$是等价关系 *Encrypt*=(*Encrypt*)的展开式;X_1,X_2 和 X_3 分别是接口变量 *Encrypt.Ippacket-R*,*Encrypt.IPpacket-E* 和 *Encrypt.Auth* 的别名;元变量 *Auth*=(*Authenticate*)引入一个 *Authenticate* 安全重构元实例;绑定关系 *Auth.IPpacket-E=IPpacket-E* 展开为两个等价式 *Id*:*Encrypt.Auth.Ippacket-E*⊃*Encrypt.IPpacket-E* 和 *id*:*Encrypt.IPpacket-E*⊃*Encrypt.Auth.Ippacket-E*;*Encrypt*:*Encrypt.IPpacket-R*→*Encrypt.IPpacket-E* 是公理展开式,*Encypt* 是数据依赖关系的具体实现,对应编程语言中一个函数模块;*SecProperty*(*confidentiality*,*level*2)是安全属性.

4 动态自动重构配置生成

所谓动态自动重构,是指给定一个重构请求,系统可以通过逻辑的智能推理,自动地计算出满足重构请求的配置,这个过程不需要人工介入.*Q* 是安全重构元逻辑描述集合,要得到一个由 *Q* 可计算的安全需求请求 $Q\mapsto X_1\wedge\ldots\wedge X_n\rightarrow Y_1\wedge\ldots\wedge Y_m$,首先,用户对问题通过 SRCS 进行建模和表达,逻辑语言转换器自动将 SRCS 转换为逻辑语言 SSPE,产生安全重构元逻辑描述集合 *Q*,推理系统将安全需求逻辑描述和 *Q* 转换成逻辑语言 SSPE 下的直觉主义逻辑公理集合 *T*,若公式 $T\mapsto\exists\psi(X_1\wedge\ldots\wedge X_n\rightarrow Y_1\wedge\ldots\wedge Y_m)$对于每一个可解问题是可以构造证明的,说明存在一个合法重构配置可以满足安全需求,其中,ψ即是要求解的重构配置.

下面以上述 IPSec 协议重构为例,给出可重构安全计算系统重构目标自动合成过程:

设重构目标为 *IPpacket*→*result*{φ}.该例中有 3 个安全重构元 *Classify*,*Encrypt* 和 *Authenticate*,*loop*1 是控制变量,其 SRCS 描述如图 3 所示.

将各安全重构元按照第 3 节所述方法展开得到逻辑表达式集合 *Q*,*Q* 与安全需求(重构请求)及推理系统规则构成直觉主义逻辑公理集合 *T*,限于篇幅,这里我们只给出与推理有关的公理集(*),如下:

$$\left.\begin{array}{l} Ippacket\text{-}R\supset Ippacket\text{-}E\{Encrypt\} \\ Ippacket\text{-}E\supset Ippacket\text{-}A\{Authenticate\} \\ Ippacket\text{-}R\supset Ippacket\text{-}A\{\varphi\} \\ (Ippacket\text{-}R\rightarrow Ippacket\text{-}A\{\varphi\})\wedge Ippacket\text{-}C\supset Result\{loop(\varphi)\} \\ IPpacket\supset Ippacket\text{-}C\{classify\} \end{array}\right\}(*)$$

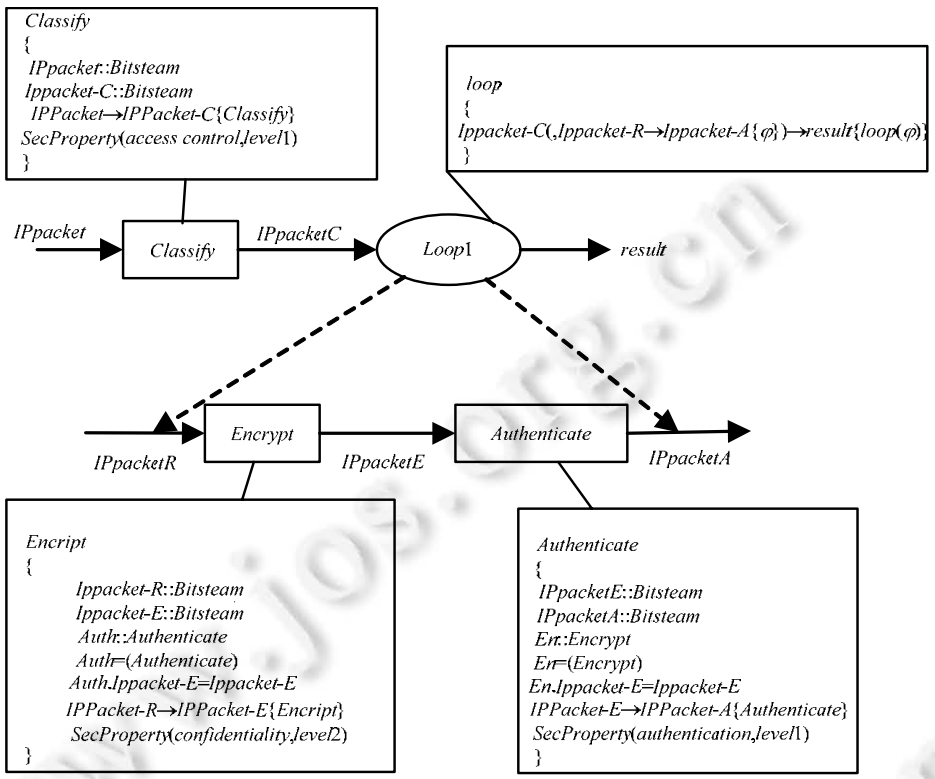

Fig.3 SRCS description of Outbound processing in IPSec protocol

图 3 IPSec 协议 Outbound 处理过程的 SRCS 描述

根据 SSPE 推理规则,推导过程如下:

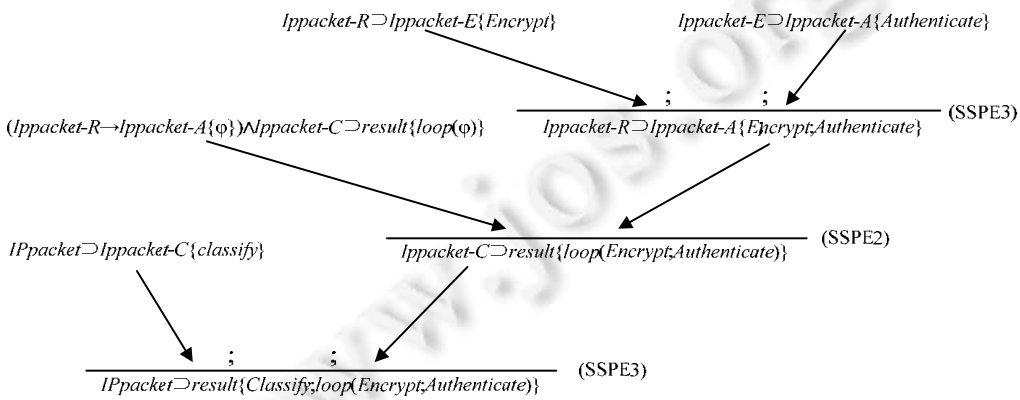

Fig.4 Reconfiguration generation instance of Outbound processing in IPSec protocol

图 4 IPSec 协议 Outbound 处理重构配置生成实例

从图 4 我们看到,推理过程构成一颗推理树,其中,叶子结点代表公理,中间结点代表推理规则,重构配置在推导过程中逐步生成.首先得到{*Encrypt*;*Authenticate*},然后得到{*loop*(*Encrypt*;*Authenticate*)},最后得出:

$$\psi=\{Classify;loop(Encrypt;Authenticate)\}.$$

合成方法能够计算一种应用的多种不同构型,也能计算一个安全重构元集合上能够被满足的所有重构请求,即这个集合上的构型变化能力,这些工作会在下一步工作中深入研究.

5 相关研究

目前,能够提供高灵活性、适应性和可扩展性安全服务的可重构安全计算系统,已成为安全研究领域的热点问题[7].一些学者从可重构安全计算系统软硬基础平台出发研究重构方法,麻省理工大学 Eddie 博士提出了基于数据转发层面的可重构路由开发平台 Click 系统[8],Click 系统配置文件完全由手工编写,不能适应新网络安全环境下动态多变的安全需求.该系统通过用户手工编写配置文件将系统中各重构组件有机结合起来,实现路由器的功能重构.Paul 等人[9]提出一种可重构 SSL/TLS 安全处理体系,所有的加密、哈希函数、密钥交换算法以比特文件的形式存储在 SD 卡上,利用 FPGA 平台的部分重构能力,可动态地执行嵌入式系统中的应用.该方法只对安全系统的局部实施重构,重构能力非常有限.文献[10]提出一种高性能的实现 IPSec 协议和 SSL 协议加速的网络安全处理器系统体系 NSP.使用基于特定域描述指令集的可编程并行密码引擎阵列,控制流将数据包分段并将其配置到密码引擎阵列,有效提高系统的吞吐率.这些研究说明在有限的可重构软硬件资源上实现复杂的安全系统是有可能的.

一些学者对可重构安全计算系统的体系结构展开研究,文献[11]基于 SDN(软件自定义网络)/NFV(网络功能虚拟化)体系架构,提出通过组合虚拟安全服务模块来构建安全服务链(security service chain,简称 SSC)的思想,但是并没有给出安全服务链接组合的具体方法.文献[12]提出 FRESCO 安全架构,其以安全模块组合的方式生成安全服务,但仅给出了功能上的验证.文献[13]提出通过网络功能单元的组合实现节点对应用需求的自适应匹配.

还有一些学者在重构机理方面展开研究,文献[14]提出了基于构件的层次化可重构网络体系结构以及基于进程代数的重构原则、模型和功能实体的重构方法,实现了构件过程的自主性、反应性和主动性.但是该体系采用静态功能候选集模板实现重构配置的生成,其灵活性依赖于功能候选集模板的配置,重构构件之间不能实现灵活的自动组合,主动性具有局限性.文献[15]将可重构系统定义为构件及构件上的运算集合,建立可重构系统代数模型,提出了可重构安全服务之间相互转换的代数运算方法.该方法能够计算特定构件集合上的构型变化能力和对构件的可替换性进行验证,但该方法没有考虑安全需求的变化性,不能根据需求变化灵活计算重构构型.文献[16]提出一种可同时支持软件构件和硬件构件的参数化可重构构件模型,基于此模型,提出了混合可重构系统上构件设计空间的快速搜索算法.该方法同样基于静态功能候选集模板,其灵活性受限制.文献[17]提出一种软件自定义网络的安全服务链动态组合机制,建立了基于向量空间和整数规划的组合模型,利用人工智能方法实现快速的配置生成.但是该方法的安全服务链是根据安全服务功能编排模板构建的,缺乏灵活性.

以上这些研究虽然能够提高系统的灵活性、适应性和扩展性,但是可重构系统的配置生成方法主要采取手动配置或者是基于重构配置模板的半自动匹配方式来实现,而人工介入方式易增加系统的脆弱性.可重构安全计算系统作为一种灵活的主动防御机制,应该具有动态自动适应安全需求和环境参数变化的能力,避免人工介入的导致脆弱性,同时考虑安全服务等级和效能属性,为当前安全需求提供最合适的安全服务,而这方面当前鲜少有人研究.为此,本文重点研究了可重构安全计算系统的动态自动配置生成方法,主要借鉴了程序自动合成理论.Mints 与 Tyugu 提出一种基于直觉主义逻辑[3]的程序自动合成理论 SSP,将属性与属性之间的函数约束关系描述为直觉主义逻辑公式,利用逻辑推理规则能够自动合成程序,并将 SSP 应用于仿真软件、工程计算软件和安全监测等领域[18].文献[19]将直觉主义逻辑的推理能力应用于语义 Web 领域,实现了 Web 服务的自动组合.我们有理由相信,基于直觉主义逻辑的程序自动合成理论同样可以应用于可重构安全计算系统的重构配置自动生成.

本文的主要贡献在于提出了一种有效的可重构安全计算系统的建模和表达方法,能够动态自动生成适应安全需求和效能属性变化的重构配置,为用户提供具有高灵活性、适应性和可扩展性的安全服务,为进一步研究可重构安全计算系统构型变化能力和验证安全服务的正确性和安全性打下基础.

6 总 结

本文对可重构安全系统的自动重构机理进行了深入的理论研究.在程序自动合成理论 SSP 的基础上,给出

了一种从面向安全重构元的问题描述语言到可执行的重构配置程序的有效方式,提出了一种改进的逻辑语言SSPE和安全重构元建模语言SRCS,以及SRCS到SSPE的逻辑语义转换规则.SSPE与SSP不同的是:SSP理论的研究对象是程序的最小数据类型,而SSPE的研究对象是安全重构元.SSP理论不能完全适应可重构安全计算系统的建模和表达,因此我们对SSP中的公式,即条件和非条件计算声明语句以及推理规则进行了改进,引入了分支结构和错误处理的消除规则,并通过实例证明了SSPE的可行性和正确性.SSPE存在一个主要问题是安全重构元展开后可能产生的公理数量庞大,需要在设计高效的自动查询评估算法方面做进一步研究.

References:

[1] Heyting A. Mathematische Grundlagenforschung Intuitionismus Beweistheorie. 4th ed., Berlin: Springer-Verlag, 2013. 10−60.

[2] Kleene A. Introduction to Metamathematics. New York: van Nostrand, 1952. 341−343.

[3] Mints G, Tyugu E. Justifications of the structural synthesis of programs. Science of Computer Programing, 1982,2(3):215−240.

[4] Grigorenko P, Saabas A, Tyugu E. Cocovila-Compiler-Compiler for visual languages. Electronic Notes in Theoretical Computer Science, 2005,141(4):137−142.

[5] Mints G, Tyugu E. The programming system PRIZ. Journal of Symbolic Computation, 1988,5(3):359−375.

[6] Matskin M, Maigre R, Tyugu E. Compositional logical semantics for business process languages. In: Proc. of the 2nd Int'l Conf. on Internet and Web Applications and Services (ICIW 2007). New York: IEEE, 2007. 38−38.

[7] Paul S, Pan J, Jain R. Architectures for the future networks and the next generation Internet: A survey. Computer Communications, 2011,34(1):2−42.

[8] Kohler E, Morris R, Chen B, *et al*. The click modular router. ACM Trans. on Computer Systems (TOCS), 2000,18(3):263−297.

[9] Paul R, Chakrabarti A, Ghosh R. Multi core SSL/TLS security processor architecture and its FPGA prototype design with automated preferential algorithm. Microprocessors and Microsystems, 2016,40:124−136.

[10] Wang H, Bai G, Chen H. A GBPS IPSec SSL security processor design and implementation in an FPGA prototyping platform. Journal of Signal Processing Systems, 2010,58(3):311−324.

[11] Lee W, Choi YH, Kim N. Study on virtual service chain for secure software defined networking. Advanced Science and Technology Letters, 2013,29(13):177−180.

[12] Shin S, Porras P, Yegneswaran V. FRESCO: Modular composable security services for software-defined networks. In: Proc. of the 20th Annual Network and Distributed System Security Symposium (NDSS). San Diego: Internet Society, 2013. 1−16.

[13] Cheng GZ, Chen HC, Chen SQ. How to make network nodes adaptive? IEEE Communications Letters, 2014,18(3):515−518. [doi: 10.1109/LCOMM.2014.011714.132622]

[14] Liu Q, Wang BQ. Construction and reconfiguration scheme of the hierarchical reconfiguration network based on the components. Chinese Journal of Computers, 2010,33(9):1557−1568 (in Chinese with English abstract).

[15] Yuan B, Wang BQ. Algebraic model of reconfigurable system based on component operation. Ruan Jian Xue Bao/Journal of Software, 2012,23(10):2735−2745 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4170.htm [doi: 10.3724/SP.J.1001.2012.04170]

[16] Li F. Research on the unified component model and DSE for hybrid reconfigurable systems [MS. Thesis]. Zhengzhou: PLA Information Engineering University, 2010 (in Chinese with English abstract).

[17] Xiong G, Hu YX, Duan T, *et al*. A dynamic composition mechanism for the security service chain oriented software defined networking. Journal of Electronics & Information Technology, 2016,38(5):1234−1241 (in Chinese with English abstract).

[18] Tyugu E, Matskin M, Penjam J. Applications of structural synthesis of programs. In: Proc. of the Formal Methods (FM'99). 1999. 551−569.

[19] Maigre R, Küngas P, Matskin M, *et al*. Dynamic service synthesis on a large service models of a federated governmental information system. Int'l Journal on Advances in Intelligent Systems, 2009,2(1):181−191.

附中文参考文献:

[14] 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.计算机学报,2010,33(9):1557−1568.

[15] 袁博,汪斌强.基于构件运算的可重构系统代数模型.软件学报,2012.23(10):2735−2745. http://www.jos.org.cn/1000-9825/4170.htm [doi: 10.3724/SP.J.1001.2012.04170]

[16] 李烽.混合重构系统构件的模型设计及空间搜索算法研究[硕士学位论文].郑州:解放军信息工程大学,2010.

[17] 熊钢,胡宇翔,段通,等.一种软件定义网络的安全服务链动态组合机制.电子与信息学报,2016,38(5):1234−1241.

肖玮(1979－),女,湖南桃江人,讲师,主要研究领域为网络与信息安全,可重构安全计算.

陈性元(1963－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为网络与信息安全.

杜学绘(1968－),女,教授,博士生导师,主要研究领域为信息安全.

李海玉(1974－),男,副教授,主要研究领域为计算机应用.

陈宇涵(1993－),男,硕士生,主要研究领域为可重构安全计算.

www.jos.org.cn