安全协议多目标语言代码自动化实现方案

李兴华[1,2]，李帅团[1]，李登[1]，马建峰[1]

（1. 西安电子科技大学 计算机学院，陕西 西安 710071； 2. 南京大学 计算机软件新技术国家重点实验室，江苏 南京 210032）

摘 要：提出了安全协议多目标语言代码自动化实现方案。首先，基于 XML 语言，设计了安全协议描述方法；其次，开发了一个图形用户接口 GUI，用户通过 GUI 配置安全协议，自动化生成协议的 XML 描述文档；最后，设计并实现了一个安全协议编译器，将先前得到的 XML 描述文档作为输入，生成所需的目标语言代码。分析表明同已有方案相比，该方案具有明显的优势。

关键词：安全协议；代码实现；自动化；XML

中图分类号：TP311 **文献标识码：**A **文章编号：**1000-436X(2012)09-0152-08

Multi-language oriented automatic realization method for cryptographic protocols

LI Xing-hua[1,2], LI Shuai-tuan[1], LI Deng[1], MA Jian-feng[1]

(1. School of Computer Science and Technology, Xidian University, Xi'an 710071, China;

2. State Key Lab. for Novel Software Technology, Nanjing University, Nanjing 210032, China)

Abstract: A multi-objective-language-oriented automatic code generation scheme for security protocols was put forward. First, based on XML, a description method for security protocols was designed. Second, a graphic user interface (GUI) was developed, through which the XML description documents could be automatically generated by configuring the security protocol. Finally, a security protocol interpreter was designed, through which the corresponding objective code was generated with the XML description document as input. nalysis indicates that compared with the existing schemes the scheme has obvious advantages.

Key words: security protocol; code realization; automation; XML

1 引言

随着网络应用的飞速发展，接踵而至的是各种纷繁的安全问题，为了能在一个不安全的网络环境下提供安全的通信服务，提出了安全协议(密码协议)的概念。安全协议运用密码学方法实现与安全相关的功能，它定义了 2 个或者多个参与实体之间进行数据通信的规则集，以达到诸如身份认证、密钥分配等目的。

安全协议从最初的设计到最后正式投入使用，一般都会经历如图 1 所示的 3 个阶段：1）设计分析；2）代码实现；3）协议测试。在设计分析阶段，如何保证安全协议的正确性是比较棘手和易错的，有相当数量曾经认为是安全的协议经过若干年后被证明是不安全的。为了降低协议出错的可能性，在过去的几十年里，研究人员在该领域做了大量的工作，并提出了

收稿日期：2011-07-05；修回日期：2012-04-09

基金项目：国家科技部重大专项基金资助项目（2011ZX03005-002）；国家自然科学基金资助项目（U1135002,61072066）；中央高校基本科研业务费基金资助项目（JY10000903001，JY10000901034）

Foundation Items: Major National S&T Program (2011ZX03005-002); The National Natural Science Foundation of China (U1135002, 61072066)；The Fundamental Research Funds for the Central Universities(JY10000903001, JY10000901034)

不少安全协议形式化设计与分析的方法，这些方法既包括 Random Oracle 模型[1]、Canetti-Krawczyk 模型[2]、Universally Composable 安全模型[3]、BAN 逻辑方法[4]和 PCL 模型[5]，也包括许多自动化验证方法，如：Mur[6]、Athena[7]和 ProVerif[8]等。

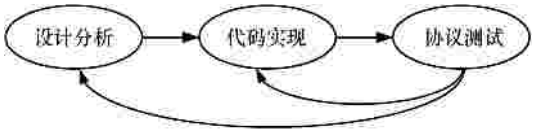

图 1　安全协议的 3 个阶段

在协议测试阶段也有不少方法，既有基于传统软件工程的黑盒测试和白盒测试，还有专门针对安全协议的测试方法，如：基于 TTCN-3[9]的一致性测试工具 TAU/Tester[10]、OpenTTCN3[11]、Danet TTCN-3 tool[12]和安全性测试工具 Protos[13]、Fuzzing Tools[14]等，而且这些方法大部分是靠机器自动化实现的。

而对于从安全协议设计分析到测试阶段起桥梁作用的代码实现，却仍然主要停留在手工编码水平，研究人员尚未对此开展广泛和深入的研究。手工编码效率低，存在大量重复性的工作，人工成本较高，这就影响了整个协议设计开发的速度。另外，手工编码难免会引入编程错误，如调用了不安全的函数，缓冲区溢出等。由此可见，依靠传统方法要高效、安全地实现对协议的编码对程序员来说是一项有挑战性的工作。

因此，对于已经证明安全的协议，如果可以设计一个代码自动化生成工具，将加速编码过程，同时结合第 3 阶段协议测试方法，可以提高协议设计开发的效率，缩短整个过程所需时间；对某些协议效率要求较高的应用来说，如果能够很快将设计的协议转化为代码进行部署，就可以快速验证所设计协议的有效性，从而加速设计出满足应用需求的协议；能够节约大量人工成本，从而投入更多的精力专注于安全协议的设计及测试；即使自动化实现的代码不能够被直接使用，也可以给编程人员提供参考，减少编程人员的工作量；同时，在自动化工具设计时可以尽量利用语言本身的安全机制来避免一些代码缺陷，如：避免调用类似 C 语言中的 strcpy 之类的不安全的函数，这在一定程度上提高了代码的安全性；另外，该工具对于那些编程水平不高的人员（如高校学生）以及对代码质量要求不是很高的应用（如学术研究）来说，也是有意义的。

而目前的研究工作存在诸多的不足之处（详见第 2 节），不能够满足应用的需求。为了克服这些缺陷，设计了安全协议多目标语言代码自动化实现方案。以 XML 作为安全协议描述语言，在此基础上开发了一个安全协议代码编译器，该编译器可以将协议的 XML 描述转化为多目标语言实现，如面向对象的 Java、C#等以及非面向对象的 C 语言。同时，为了进一步提高代码生成的速度和降低对使用者的要求，将安全协议 XML 描述这一过程自动化，用户通过 GUI 就能够将安全协议转化为对应的 XML 描述。

2　相关工作

迄今为止所知道的最早涉及代码自动化实现的方案，来自于 Berkeley 大学的 Song 等人参与的 Athena 项目[7]，整个系统包括了 3 个子模块：自动化协议生成器 APG、自动化协议验证器 APV 和自动化代码生成器 ACG。ACG 利用 APG 和 APV 阶段生成的最优协议描述，自动化了协议的 Java 实现。该方法给安全协议的代码自动化实现提供了一种很好的思路，但是由于其侧重点在设计验证阶段，ACG 这一部分存在较大的局限性，它没有涉及到中间对象的状态信息，加解密等计算原语在扩展性方面也存在局限性，甚至没有构造出实际的网络通信，而这些都是安全协议在应用中所不可或缺的。

Millen 与 Muller 在文献[15]中提出了一种基于通用认证协议描述语言（CAPSL, common authentication protocol specification language）和 CIL（casper intermediate language）的安全协议编译器，将协议的 CAPSL 描述通过一个转换器得到 CIL 输出，将所得的 CIL 描述作为代码生成器的输入，得到协议的 Java 代码实现。该代码生成器对常用的安全性操作、算法都给出了限定，不具有可扩展性，如将对称加密算法限定为 DES，消息认证码的算法限定为 SHA1，而且未实现对常用公钥加密算法（如 RSA）的支持。

Abdullah 和 Menascé 在文献[16]中提出了一种模型，对安全协议进行基于有限状态机的 XML 描述，然后设计对应目标语言的 XLST 样式表（XML stylesheet），对 XML 描述进行解析，从而得到安全协议的代码实现。但是该方案在对协议进行 XML

描述的时候，将具体实现函数（例如：某种语言的加密函数的实现算法）放在描述中，增加了描述和实现之间的耦合性，这样对于不同的目标语言就需要重新写 XML 描述。

Oxford 大学的 Didelot 在文献[17]中构建了一个编译器 COSP-J，实现了安全协议 Java 代码的自动化实现。其中使用了模型工具 FDR（failures divergence refinement）对安全协议的高层描述（Casper 输入脚本）进行了安全性分析，之后将这些描述稍作修改作为编译器 COSP-J 的输入，生成以 Java 为目标语言的协议代码。该方案避免了协议实现过程中可能出现的诸如类型缺陷攻击和多协议攻击等，但是存在以下局限性：其实现依赖于 Java 的序列化机制，难以和其他语言（如 C 语言）版本的协议实现交互；生成的代码过于封闭，不方便应用集成；没有提供安全操作具体函数的可选择性。

Sisto 等[18]在 p 演算的基础上，扩展出可以描述常用安全操作的 Spi 演算，根据规格说明文件，将 Spi 演算中术语映射到特化的 Java 类，最后将用 Spi 演算描述的协议作为 Spi2Java 的输入，在前面规格说明文件辅助下生成协议的 Java 代码。该方案存在以下缺陷：加解密算法选择、参数设置、数据表示等操作不够灵活，给用户提供的可选择性较小；其引入的分层类型映射可能是不完善的，并且动态确定对象类型的映射可能带来不确定性。

Kiyomoto 等人在文献[19]中采用 XML 作为协议描述语言，设计了一个产生协议 C 语言实现的编译器，其目的是提出一种安全协议 C 语言的实现方案，弥补安全协议在该目标语言上自动化实现的空白，扩大协议使用的范围。其缺陷是：在进行消息描述的时候，过多的考虑了其实现；在 XML 中对具体操作也进行了描述，导致了 XML 描述复杂、难以理解。

通过以上分析可以看出，已有工作存在如下缺陷。

1) 在协议描述语言上，大多数方案的代码生成功能是在协议安全性形式化分析的基础上来进行设计的，并非直接以代码自动化实现为目的，不能够给用户提供选择安全操作及算法的能力，因此并不完全适合代码的自动化实现。

2) 大多数方案依赖于某种形式化分析方法，该分析方法仅仅适用于某类安全协议（如 CAPSL[16] 仅适合于认证协议的代码生成），因此他们的应用范围有限，并不具有通用性。

3) 在具体实现上，安全操作及算法的可扩展性比较差，限制了方案的使用范围。

4) 要求使用者必须掌握某些形式化方法，如协议安全分析方法或 XML，其自动化程度有限。

5) 仅仅以某一种特定语言（如 Java）作为安全协议实现目标语言，影响了生成代码的应用范围。

3 代码自动化实现方案

为了克服已有工作的不足，实现多目标语言代码的自动化实现，本文主要做了以下工作：首先，基于 XML 语言，设计了一种新的安全协议形式化描述方法；其次，开发了一个图形用户接口（GUI），通过 GUI 配置安全协议，自动化生成安全协议的 XML 描述文档；最后，设计并实现一个安全协议编译器，将先前得到的 XML 描述文档作为输入，生成协议的目标语言代码。

系统的整体框架如图 2 所示，它主要由 3 个部分组成。1）安全协议 XML 描述自动化生成模块。通过 GUI 配置安全协议，自动化生成安全协议的 XML 描述文档。2）编译器模块。编译器模块负责对安全协议的 XML 描述进行解析并生成相应的协议代码。3）底层支撑模块（安全类库，通信类库）为编译器模块的代码生成提供支撑。

下面主要对安全协议 XML 描述方法、底层支撑模块和编译器模块分别进行详细介绍。

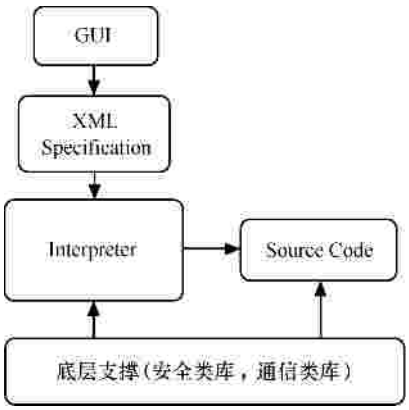

图 2 系统整体框架

3.1 安全协议 XML 形式化描述方法

通过对相关工作的研究可以看出，协议代码自动化实现首先需要解决协议的形式化描述问题，即选择合适的形式化语言来清晰、准确、灵活地描述

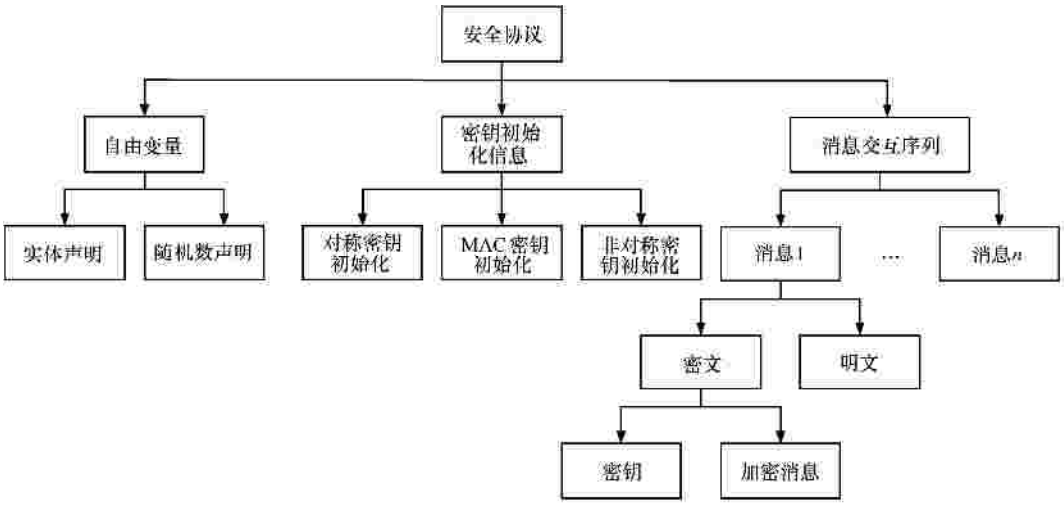

图 3　安全协议的树状结构

协议安全属性以及通信序列。

基于 XML 语言设计了一种安全协议描述方法。之所以采用 XML 描述，主要是因为 XML 作为独立于协议形式化分析语言具有简单、灵活、易扩展和应用范围广的特性，用户可以根据需求扩展 XML 标记和属性；而且 XML 语言具有强大的数据描述功能、高度的结构性和可验证性，解析和处理 XML 技术成熟，适用于协议代码的自动化实现。

3.1.1　安全协议结构

安全协议是实体之间若干消息顺序交互，它主要有 3 部分组成（如图 3 所示）：自由变量、密钥初始化信息和消息交互序列。因此，所设计的 XML 描述方法也分为 3 个部分：自由变量描述、密钥初始化信息描述和消息交互序列描述。

3.1.2　语法规则

为了便于安全协议的 XML 描述，本文设计了 XML 节点及其属性来描述图 3 中安全协议的各组成成分，下面用 XML 来描述图 3 中安全协议 3 个主要组成部分：自由变量、密钥初始化信息及消息序列。

1) 自由变量描述

每个 XML 描述文档只有一个 FreeVariable 节点，其子节点又由多个 Agent 节点和 Nonce 节点组成，描述协议中所涉及的实体和随机数变量。FreeVariable 架构如下所示：

<FreeVariable>

<AgentAddress=" … "actualName=" … "actual ="…">…</Agent>——实体声明<Nonce owner= “…”> …<Nonce>——随机数声明

<FreeVariable>

2) 密钥初始化信息描述

每个 XML 描述文档只有一个 KeyInit 节点，用于描述安全协议所用到密钥的初始化信息。如图 3 所示，密钥初始化信息主要由 3 类：对称密码、非对称密钥以及 MAC 算法密钥，分别用节点 Symcrypto、Pkcrypto、MACcrypto 描述。节点 Knowledge 比较特殊，用于描述在公钥密码体制中，实体的公钥知识。如：初始化时，实体 A 需要知道实体 B 的公钥，可以用一个节点 Knowledge 描述。KeyInit 架构如下所示：

<KeyInit>

<Symcrypto algorithm=" … "share1=" … "share2="…"SymKeyPathname="…">…

</Symcrypto>——对称密码描述

<Pkcrypto algorithm=" … "owner=" … "PKPathname="…"PRPathname="…">…

</Pkcrypto>——非对称密码描述

<Hash algorithm="…"/>——哈希算法描述

<MACcrypto algorithm=" … "share1=" … "share2="…"SymKeyPathnam="…">…

</MACcrypto>——消息认证码描述

<Knowledge subject="…"object="…">——实体的公钥知识

</KeyInit>

3) 消息交互序列描述

节点 Message 描述一条消息，其属性 source、destination、sequence 分别表示消息的发送者、接收者和消息序列号。消息交互序列可以由如下表达式

定义：

Message::=Plaintext|Encrypt|MAC|Hash

Plaintext::=Agent|Nonce

Encrypt::=(Message, usedKey)

MAC::=(Message, usedKey)

Hash::=(Message)

usedKey::=PKAgent|PRAgent|KAgentAgent…|MKAgentAgen…

Message List::=Message(|Message,Message List)

其中，PKAgent、PRAgent 分别表示实体的公钥和私钥，如实体 A 的公钥和私钥可以分别表示为 PKA、PRA；KAgentAgent…表示 2 个或多个实体之间的共享密钥，如：实体 A 和 B 之间的共享密钥可表示为 KAB；MKAgentAgent…表示 2 个或多个实体之间所用的 MAC 算法密钥，如实体 A 和 B 之间的共享 MAC 算法密钥可表示为 MKAB。

从表达式可以看出，消息内容包括明文（Nonce、Agent）、MAC、Hash 和密文，密文含有该消息加密所使用的密钥和被加密的消息，被加密的消息是消息的递归调用。

图 4 给出了一条消息 B->S:{Na,{Nb}MK_{AB},A}K_{BS}, B 示例。Message 节点属性 source、destination 和 sequence 表示此消息由实体 B 发送至实体 S，序列号为 1（表示该消息是协议的第一条消息）。此消息由 2 部分组成：密文 Encrypt 和实体 B 标识。Encrypt 节点属性 usedKey=“KBS”表示此加密消息块采用实体 B 和实体 S 之间的共享密钥进行加密；MAC 节点属性 usedKey=“MKAB”则表示实体 A 和实体 B 之间共享的 MAC 算法密钥。

```
<Message source="B"destination="S"sequence = "1">
<Encrypt usedKey="KBS">
<Nonce>Na</Nonce>
<MAC usedKey="MKAB">
<Nonce>Nb</Nonce>
</MAC>
<Agent>A</Agent>
</Encrypt>
<Agent>B</Agent>
</Message>
```

图 4 一条消息示例

3.1.3 通过 GUI 实现 XML 的自动生成

为了进一步提高代码的生成速度和减少方案对使用者的要求，本文设计了一个图形用户接口，用户通过 GUI 输入协议的相关信息后它能够自动生成该协议的 XML 描述文档。

通过 GUI，用户可以选择目标语言，添加协议的参与方和输入协议的交互序列。通过配置每一个参与实体 Agent 节点的属性（如实体的身份标识和地址）以及 Nonce 节点的属性（如生成者）。根据这部分的配置信息，可以生成自由变量部分 FreeVariable 的 XML 描述。另外，通过初始密钥信息（对称密钥、非对称密钥和 MAC 算法密钥）可以生成 KeyInit 部分的 XML 描述。在此基础上，消息交互序列中的 XML 描述也可以生成。

3.2 底层支撑模块

底层支撑部分主要包括安全类库以及通信类库等，它们在安全协议代码生成以及协议安全运行中起支撑作用。为了使最终自动化得到的协议实现清晰、简洁，在底层支撑功能的设计上，采用了面向对象的方法，用一个派生类来实现最终的协议，设计的类关系如图 5 所示（为了简洁起见，省略类中参数和方法的描述），其中，类 Implementation 是最后得到的派生类，是生成协议的实现类。它只需要执行相关功能的调用（即函数调用）即可完成协议的任务，而这些功能的实现都是由底层支撑部分提供，这使得最后的协议源码实现简单易懂；同时，该方法为外部提供的仅仅是功能调用接口，该特点也为协议多目标语言（该语言应为面向对象语言）实现奠定了基础：在不同的目标语言实现时，只需要将具体的函数用对应的语言实现即可，而其他部分不需做修改。

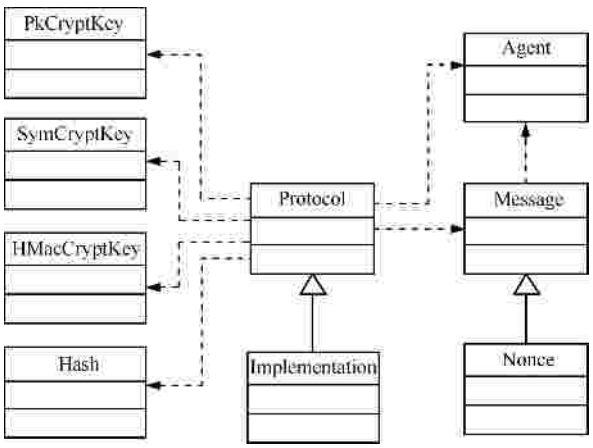

图 5 底层支撑类

对于非面向对象的语言，如 C 语言，同样借鉴如图 5 所示的框架来实现，具体来说用结构体来模拟对应类的实现。

在图 5 中，Protocol 类是整个框架中的核心类，

协议的每个参与实体的代码实现类都需要继承 Protocol 类。Agent 类和 Nonce 类分别对应于协议 XML 描述中 FreeVariable 的 Agent 子节点和 Nonce 子节点，它们的对象存储着这些子节点的相关属性。PkCryptKey 类、SymCryptKey 及 HMacCryptKey 分别对应于协议 XML 描述中的 PKcrypto、MACcrypto、Hash 子节点，它们的对象存储着这些子节点的相关属性。类 Implementation 代表实体所实现的类，该类继承了 Protocol 类，且实现了参与者对某个给定协议的操作序列。

3.3　编译器模块的设计

在实现中采用了结构化的 XML 语言来描述协议，编译器（compiler）模块负责解析协议 XML 描述文档并生成实现代码。解析过程基于文档中相关节点（如 Agent、Nonce、Pkcrypto、Message、MAC 等）及其属性来进行处理，具体来说是对关键字的解析。编译器模块由类 CGenerator 实现，该类包括 FreeVarGenerator、AlgorithmNegoGenerator、KeyInfoGenerator、BuildMsg、ParseMsg 及 Generator 等方法，这些方法完成对 XML 描述文档的解析和相应代码的生成。该过程如图 6 所示。

以下是类 CGenerator 中这几个主要的成员函数的简要介绍。

1) FreeVarGenerator 函数负责对 XML 文档中 FreeVariable 节点的协议实体、随机数变量等信息进行解析。

2) AlgorithmNegoGenerator 函数负责对协议 XML 文档中的 KeyInit 节点进行解析，收集协议用到的算法信息，并生成协议算法协商阶段代码（如收集协议参与实体自身支持的各类算法集、与对端进行交互）。

KeyInfoGenerator 函数负责对协议 XML 文档中 KeyInit 节点的密钥初始化信息进行解析，生成实体相关密钥（如对称密钥、非对称密钥以及 MAC 算法密钥）的初始化代码。

3) BuildMsg 函数负责 XML 文档中 Message 节点的解析，根据描述中每条待发送消息的结构构造消息，即生成一条消息的构造代码；ParseMsg 函数对收到的消息按照 Message 节点的结构进行解析，生成一条消息的解析代码。

4) Generator 函数通过调用 FreeVarGenerator、KeyInfoGenerator、BuildMsg 和 ParseMsg 函数，生成每个参与方实体代码，即生成参与方实体的代码实现类（即图 5 中所示的类 Implementation）。

4　方案的特点和优势

同手工编码相比，本文的方案能够改进安全协议设计及开发的效率，节约大量的劳动力资源，减少程序员的工作量。同时，在一定程度上能够避免代码的缺陷，提高其安全性。

同已有方案相比，所提方案具有以下特点和优势。

1) 在协议描述语言的选择上，本文采用了 XML 语言，XML 语法简单，描述简洁清晰，适合作为信息交换的载体。与那些主要用于协议分析的形式化描述方法相比，XML 更加精确、灵活，能

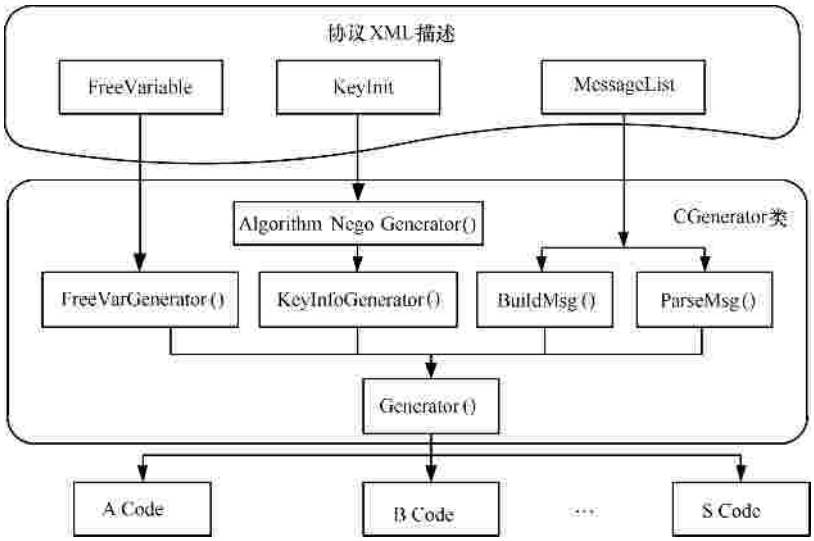

图 6　代码生成过程

表 1 相关方案比较

方案名称	通用性	算法可选择性	可扩展性	无需形式化知识	多目标语言	支持算法协商
Song 方案[7]	×	×	×	×	×	×
Millen 方案[15]	×	×	×	×	×	×
Abdullah 方案[16]	√	N	N	×	√	×
Didelot 方案[17]	×	×	×	×	×	×
Sisto 方案[18]	×	×	√	×	×	√
Kiyomoto 方案[19]	√	√	√	×	×	×
本文的方案	√	√	√	√	√	√

注：N 代表文中未提及

够在协议描述中给用户提供选择安全操作及算法的能力。而基于协议形式化安全分析（如 Spi 演算、串空间模型和 Casper/FDR）的方案并不是为代码生成来设计的，其描述能力有限，过于抽象，不能够具体到算法这一层次，用户不能够指定某个操作或算法。

2) 所提方案同协议的形式化安全分析过程相互独立，不管协议设计分析阶段使用什么样的方法，都可以采用本文的方案，因此它具有一定的通用性。

3) 现实中协议的多样性（如身份认证协议、公平交易协议等）带来了安全操作的多样性，为了尽可能扩大方案的适用范围，在实现中尽可能多地提供了安全操作的可扩展性，具体来说通过扩充 XML 节点来实现。

4) 不要求使用者掌握 XML 或任何安全协议形式化分析方法。已有的代码生成方案大部分基于某种协议形式化分析方法，使用者要掌握这些方法具有相当大的难度。

5) 充分考虑协议使用环境（如操作系统）的多样性，设计了多目标语言生成器，用户可根据具体环境的需要来选择目标语言。这样，即使对于处于不同环境的双方，也可以使用不同语言来进行协议的交互。在已有方案中，只有文献[17]支持多目标语言生成，但要生成不同的目标语言，需要重写 XML 文档。而本文的方案可以用同一个 XML 文档生成不同的目标语言。

6) 在底层支撑模块的设计上，采用了面向对象的设计方法，使得最终生成的协议代码类作为一个派生类出现，这样就屏蔽了底层支撑模块的实现细节，使得最后的协议实现清晰简洁；同时，该特点使得协议的操作与具体的语言实现分离，这为多目标语言生成提供了有利条件：在多目标语言（该语言需支持面向对象）生成中，只需要将图 6 中具体的函数功能用目标语言实现即可，对其他部分均不需做修改。

7) 所提方案支持密码算法的协商，因此那些具有算法协商功能的协议（如 TLS）也能够由本文的方案来实现。在已有方案中，大部分并不支持该功能（可以参见表 1）。

8) 在目标语言代码生成时，尽量利用语言本身的安全机制，避免调用那些容易导致协议安全隐患（如类型缺陷，缓冲区溢出等）的函数（如 C 语言中的 strcpy）来提高代码的安全性。这样可以减少手工编码中不经意的类似错误，提高了协议执行的安全性。

本文将所提方案同已有方案在通用性、算法可选择性、可扩展性、用户是否需要形式化知识、多目标语言支持及支持算法协商等方面做了比较，如表 1 所示。由此可以看出，本文的方案同已有方案相比具有明显的优势，很好地解决了它们存在的缺陷。

5 结束语

本文提出了一种安全协议多目标语言代码自动化实现方案。首先，根据安全协议的结构，设计了基于 XML 的安全协议描述方法；在此基础上设计并实现一个安全协议编译器，将先前得到的 XML 描述文档作为输入，根据需要生成对应的目标语言代码。同时，为了进一步提高自动化实现程度和降低对使用者的要求，开发了一个图形用户接口，用户通过 GUI 配置安全协议，自动化生成协议的 XML 描述文档。分析表明同已有的方案相比，该方案具有明显的优势。

参考文献：

[1] BELLARE M, ROGAWAY P. Random oracles are practical: a paradigm for designing efficient protocols[A]. First Annual Conference on Computer and Communications Security[C].1993.

[2] CANETTI R, KRAWCZYK H. Analysis of key-exchange protocols and their use for building secure channels[A]. Eurocrypt'01[C]. 2001.

[3] CANETTI R, KRAWCZYK H. Universally composable notions f key exchange and secure channels[A]. Eurocrypt'02[C]. 2002.

[4] BURROWS M, ABADI M, NEEDHAM R. A logic of authentication[J]. ACM Transactions on Computer System,1990, 8(1):18-36.

[5] DATTA A, DEREK J, MITCHELL C, *et al*. A derivation system for security protocols and its logical formalization[A]. Proceedings of 16th IEEE Computer Security Foundations Workshop[C]. 2003. 109-125.

[6] HE C, MITCHELL J C. Analysis of the 802.11i 4-way handshake[A]. Proceedings of the 3rd ACM Workshop on Wireless Security[C]. 2004.

[7] SONG D X. Athena: a new efficient automatic checker for security protocol analysis[A]. Proceedings of the 12th IEEE Computer Security Foundations Workshop[C]. 1999. 192-202.

[8] BLACNCHET B. A computationally sound mechanized prover for security protocols[A]. Proceedings of the 27th IEEE Sy ium on Security & Privacy[C]. 2006. 140-154.

[9] European Telecommunications Standards Institute (ETSI). The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. ETSI standard ES 201 873-1 V3.2.1(2007-03) [S]. 2007.

[10] Telelogic TAU/Tester[EB/OL]. http://www.telelogic.com.

[11] OpenTTCN[EB/OL]. http://www.openttcn.com.

[12] Danet's TTCN-3 Toolbox[EB/OL]. http://www.danet.com.

[13] Oulu university secure programming group[EB/OL]. http://www.ee.oulu.fi/research/ouspg/protos /index.html, 2002.

[14] FuzzingTools[EB/OL].http://www.scadasec.net/secwiki/FuzzingTools.

[15] MULLER F, MILLEN J. Cryptographic protocol generation m CAPSL[R]. SRI International, SRI-CSL-01-07, 2001.

[16] ABDULLAH I S, MENASCE D A. Protocol specification and uto-matic implementation using XML and CBSE[A]. Proceedings of Int Conf on Communications, Internet and Information Tech[C]. 2003. 17-19.

[17] DIDELOT X, HALL L M. COSP-J: a compiler for security protocols[EB/OL].http://www.comlab.ox.ac.uk/gavin.lowe/Security/Casper/ COSPJ/secu.pdf. 2003.

[18] POZZA D, SISTO R, DURANTE L. Spi2java: automatic cryptographic protocol java code generation from spi calculus[A]. Proceedings of the International Conference on Advanced Infor tion Networking and Applications[C]. 2004. 400-405.

[19] KIYOMOTO S, OTA H, TANAKA T. A security protocol compiler generating C source codes[A]. Proceedings of the 2008 International Conference on Information Security and Assurance[C]. 2008. 20-25.

作者简介：

李兴华（1978-），男，河南南阳人，博士，西安电子科技大学副教授，主要研究方向为网络与信息安全等。

李帅团（1987-），男，河南许昌人，西安电子科技大学硕士生，主要研究方向为网络与信息安全等。

李登（1987-），男，湖北襄樊人，西安电子科技大学硕士生，主要研究方向为网络与信息安全等。

马建峰（1963-），男，陕西西安人，博士，西安电子科技大学计算机学院院长、教授、博士生导师，主要研究方向为网络与信息安全等。