Savin Sergey Vladimirovich – Branch of the Military Academy of Communications (Krasnodar); e-mail: seva_xtime@mail.ru; 4, Krasina, Krasnodar, 350063, Russia; phone: +79189808621; associate postgraduate full-time.

Finko Oleg Anatolievich – e-mail: ofinko@yandex.ru; phone: +79615874848; dr. of eng. sc.; professor.

УДК 621.3.037.3: 004.04

Л.К. Бабенко, Ф.Б. Буртыка, О.Б. Макаревич, А.В. Трепачева

ОБОБЩЕННАЯ МОДЕЛЬ СИСТЕМЫ КРИПТОГРАФИЧЕСКИ ЗАЩИЩЕННЫХ ВЫЧИСЛЕНИЙ*

Рассматривается проблема организации вычислений над зашифрованными данными. Эта задача в последнее время приобрела большую актуальность в связи с развитием парадигмы облачных вычислений и необходимостью обеспечения адекватных мер их защиты. Однако разнообразные примитивы работы с зашифрованными данными, такие как полностью гомоморфное шифрование, функциональное шифрование, многосторонние секретные вычисления, решают свои задачи лишь в ограниченном контексте. Тогда как построение реальной системы защищенных вычислений требует разработки некой общей теории организации защищенных вычислений, использующей системный подход. Предлагается разделить всю функциональность, которую должна поддерживать система защищенных вычислений на несколько уровней, взаимодействие между которыми осуществлялось бы через интерфейсы. Представленная шестиуровневая аналитическая модель под названием «Стек интерфейсов защищенных вычислений» («СИЗВ») призвана стандартизировать и облегчить деятельность исследователей и разработчиков в области систем криптографически защищенных вычислений, т.е. таких систем, в которых обработка конфиденциальных данных ведется недоверенной стороной и, следовательно, ни на одном из этапов обработки информация не может быть расшифрована. Для каждого из уровней очерчивается проблематика, с которой исследователи имеют дело, раскрывается круг вопросов, которые должны быть решены, а также дается краткий обзор работ по данной тематике. Самый верхний уровень является самым абстрактным и предоставляет свой интерфейс прикладному программисту, затем идут два уровня, имеющие дело с внутренним представлением программ, далее – уровень, предназначенный для анализа и синтеза архитектуры виртуальной машины, уровень работы с криптографическими схемами и протоколами и, наконец, уровень аппаратной реализации элементарных операций. Необходимым условием функционирования системы криптографически защищенных вычислений является проработка и реализация в полной мере каждого из этих уровней.

Полностью гомоморфное шифрование; функциональное шифрование; многосторонние секретные вычисления; системы защищенных вычислений; аналитическая модель; защищённые облачные вычисления.

L.K. Babenko, Ph.B. Burtyka, O.B. Makarevich, A.V. Trepacheva

A GENERAL MODEL OF CRYPTOGRAPHICALLY SECURE COMPUTING SYSTEM

The paper deals with the problem of organization the computations over encrypted data. Recently this problem has become increasingly important due to the extension of the cloud computing paradigm and the need for adequate measures to protect them. However, a number of primitives for working with encrypted data, such as a fully homomorphic encryption, functional encryp-

* Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта №15-07-00597 а.

tion, secure multiparty computations and so on solve their problems in a limited context, while building the real secure computing system requires the development of some general theory for organization secure computing, using a systemic approach. We propose to divide all the functionality that the secure computing system must support into the several layers; the interaction between them would be done through the interfaces. Presented six-layer analytical model called "Secure computing interface suite" ("SCIS") is intended to standardize and facilitate the work of researchers and developers in the field of cryptographically secure computing, i.e. such systems in which the untrusted parties processes the sensitive data and therefore the processed information can't be decrypted at the any stage of processing. For each of the layers we outline the problems researchers deal with, reveal a range of issues that must be addressed, and provide a brief overview of the literature on this topic. The top layer is the most abstract and provides interface for application programmer, followed by two layers dealing with the internal representation of programs, then - a layer designed for analysis and synthesis of the virtual machine architecture, the next layer deals with cryptographic schemes and protocols, and, finally, the layer for hardware implementation of elementary operations. A necessary condition for the working of a cryptographically secure computing system is the development and implementation each of these layers.

Fully homomorphic encryption; functional encryption; secure multiparty computations; secure computation system; analytical model; secure cloud computing.

Введение. Возрастающая потребность в надежной защите данных при их обработке в недоверенной среде ставит перед компьютерной индустрией беспрецедентные задачи. Возможно ли гарантировать конфиденциальность данных, которые должны подвергаться сложной обработке?

Недавние успехи криптографии [1] предлагают такие решения проблемы как, например, полностью гомоморфное шифрование (ПГШ), позволяющие проводить вычисления непосредственно над зашифрованными данными, а также функциональное шифрование и многосторонние секретные вычисления. Хотя первые криптосистемы ПГШ были далеки от практики, данная область криптографии быстро развивается и уже сейчас имеет смысл говорить о вопросах практических приложений [2–4].

Однако для построения реальной информационной системы защищенных вычислений (СЗВ) одного гомоморфного шифрования мало: необходима целая инфраструктура, поддерживающая его работу.

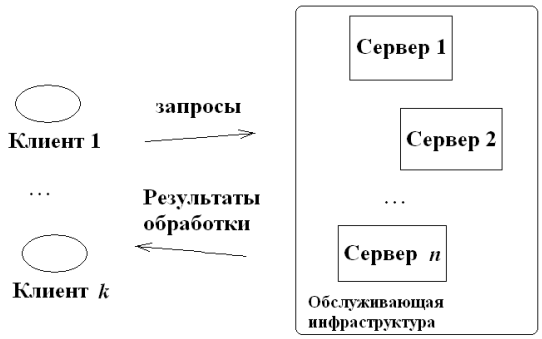

Рис. 1. Общая организация СКЗВ

А именно, необходимо обеспечить инфраструктуру генерации и распределения ключей между пользователями СЗВ в соответствии с их правами доступа. Необходимо определить, какой уровень защищенности необходим пользователям, что и как они хотят защитить, какие вычисления проводить, хотят ли они проверять результаты вычислений, делегировать доступ к какой-то части своих данных другим участникам. Также необходимо поддерживать достаточную производительность, совместимость и надежность СЗВ. Важным вопросом являются способы оценки клиентами *качества обработки их запросов*.

Для рассмотрения всех этих вопросов в той или иной степени предлагаются различные подходы, решающие свои узкие частные задачи в рамках своих моделей. Количество публикаций, освещающих перечисленные вопросы в том или ином ракурсе, стало уже довольно солидным [1–19]. Однако для того, чтобы представить цельную картину и осмыслить ее, необходима общая концепция, связывающая все части воедино.

Такая концепция, предлагаемая нами в данной статье, носит название «Общая модель криптографически защищенных вычислений» (ОМКЗВ) или «Стек интерфейсов защищенных вычислений» (СИЗВ). По мнению авторов, концепция, подобная представленной, должна быть стандартизирована в целях обеспечения лучшего взаимодействия между составными частями систем ЗВ.

Целью работы является разработка модели СЗВ, которая систематизирует информацию, необходимую для построения СЗВ. Такая модель предлагается впервые и дает возможность оценки затрат на создание СЗВ, позволяет ускорить и автоматизировать проектирование, а также формализовать анализ возникающих проблем.

1. Обзор модели. Основной объект нашего рассмотрения – это система криптографически защищенных вычислений (СКЗВ). Под этим термином подразумевается компьютерная система (см. рис. 1), которой пользователи делегируют вычисления, при этом требуя от СКЗВ гарантий конфиденциальности и целостности своих данных и алгоритмов. Примером таких реальных систем могут служить сервисы в парадигме *облачных вычислений*, которые в настоящее время получают все большее распространение.

Рассматриваемая организация СКЗВ включает *сервер* и одного или нескольких *клиентов*, которые подключаются к серверу в отдельные моменты времени для передачи ему вычислительного задания или обмена данными с ним. При этом предполагается, что вычислительная мощность клиентов *намного* меньше мощности сервера. А также предполагается, что сервер взаимодействует с клиентами не в интерактивном режиме, и количество пересылок между клиентом и сервером необходимо минимизировать. Ну и, наконец, самым существенным признаком СКЗВ является то, что *информация вне клиента всегда должна быть зашифрованной.*

Из всего сказанного следуют два требования. Во-первых, *вычислительное задание клиента должно формулироваться кратко*, желательно в виде одной короткой команды. Это выливается в то, что на сервере должны быть готовы алгоритмы, реализующие эти команды в виде последовательности подробных инструкций, а также то, что эти команды должны быть стандартизированы, чтобы клиент мог *единообразно делать запросы к разным сервисам*. А во-вторых, сервер не может играть роль лишь хранилища данных, поскольку тогда каждому клиенту необходимо будет производить вычисления самому, каждый раз расшифровывая принятые с сервера данные. Поэтому *для шифрования клиенту нужно использовать ПГШ* [1–4] и тогда *сервер сможет обрабатывать данные в зашифрованном виде без знания ключа расшифрования*.

Эти требования приводят к необходимости инкапсуляции и многоуровневой структуры. Уровень инкапсуляции определяется требованиями к функциональности СЗВ. Например, для организации защищенного сервиса бухгалтерских вычислений необходимо поддерживать уровень инкапсуляции, отличный от того, который необходим для реализации гомоморфного вычисления программ с циклами.

Предлагаемая нами «эталонная» модель системы вычислений над зашифрованными данными имеет 6 уровней (табл. 1). Название «Стек интерфейсов защищенных вычислений» (СИЗВ) означает, что интерфейс, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Хотя в

данном случае сетевое соединение и присутствует, стек не носит такой характер как, скажем, стеки сетевых протоколов в модели OSI: любая реализация объекта данной модели может взаимодействовать *только с соседними уровнями*, а именно, использовать интерфейс нижележащего уровня для доступа к его объектам и предоставлять свой собственный интерфейс на вышележащий уровень.

Таблица 1

Сущности (объекты) ОМКЗВ (СИЗВ)

Типы данных и операций	Уровень (Layer)	Функциональность
Типы данных в используемом ЯП	Прикладной уровень	Написание программы на ЯП высокого уровня для выполнения прикладной задачи
Лексемы языка программирования	Уровень трансляции с высокоуровнего ЯП	Корректное преобразование высокоуровневых конструкций в низкоуровневые
Запросы к виртуальной памяти	Уровень потока данных и потока управления	Оптимальное использование выбранной архитектуры
Виртуальные машинные команды	Уровень виртуальной архитектуры	Обеспечение возможности вычислить программу необходимой структуры
Криптографический протокол, криптосхема	Уровень криптографических протоколов и систем	Обеспечение необходимой криптостойкости и коммуникационной эффективности
Биты, гейты	Уровень реализации элементарных операций	Работа с аппаратурой, аппаратными архитектурами для эффективной реализации атомарных операций

Далее мы рассмотрим каждый уровень более подробно.

2. Прикладной уровень. Начнем рассмотрение с прикладного уровня, поскольку именно приложениями, в конечном счете, формируются требования к СКЗВ. Предполагается, что прикладная вычислительная задача, которую надо решать клиенту и которую он хочет делегировать СКЗВ может быть описана на некотором языке программирования высокого уровня (ЯВУ). Это может быть как ЯВУ общего назначения, такой как C++ или Java, так и специализированный язык, такой как SQL/PSQL или Haskell. Основным требованием, предъявляемым на данном уровне к системе, является степень инкапсуляции – такая, что при работе с ней прикладной программист не должен знать как подробности работы с шифрованием, *так и сам факт того, что его программа будет выполняться в недоверенной среде*. То есть программа пишется так, как будто она будет выполняться на локальных вычислительных мощностях клиента. Отметим, что впервые идея отделения программной среды от платформы выполнения для СКЗВ на основе гомоморфного шифрования была озвучена в работе [5], однако в ней предполагается специализированный ЯВУ EDSL (надстройка к Haskell), который обязывает *явно* отмечать данные, подлежащие шифрованию, а также в котором был доступен *только поток*

управления, не зависящий от данных (data-independent control flow). Есть и другие специализированные ЯВУ, такие как SMC [6], FAIRPLAY [7], SIMAP [8], которые разработаны для поддержки написания кода защищенных приложений с использованием протокола многосторонних секретных вычислений. Трансляция на такие ЯВУ с обычных является предметом рассмотрения прикладного уровня.

3. Уровень трансляции с высокоуровнего ЯП. Обычно компилятор осуществляет преобразование программы с исходного ЯВУ в некоторое внутреннее представление, анализируя которое он может оптимизировать производительность программы или искать ошибки программиста. Используемый на данном уровне высокоуровневый ЯВУ является специализированным (например, такой как в работе [5]). Итак, исходной точкой для дальнейшей разработки СКЗВ будет являться программа на ЯВУ. Анализ этой программы позволит определить требования к СКЗВ *полностью*. Эти требования могут определяться автоматизированно на основе некоторых признаков. Одним из подходов может быть анализ графов потока управления и потока данных программы.

4. Уровень потока данных и потока управления. В простейшем случае граф потока управления не содержит циклы и зависимости по данным, т.е. мы имеем дело с линейной программой [9]. Однако если программа содержит условные переходы, косвенную адресацию и циклы, то её выполнение требует платформы со специальной архитектурой. Причиной этого служит то, что появляются зависимости потока управления по данным (data-dependent control flow), однако данные зашифрованы и возникает необходимость принимать так называемые скрытые решения (hidden decisions).

Как это отражается на зашифрованных вычислениях? Если данные зашифрованы, а при этом необходимо принимать решение об остановке программы в зависимости от данных, то возникает *проблема остановки*: *СКЗВ не в состоянии определить, когда вычисление программы завершится и завершится ли оно*. Данную проблему можно решить с помощью специальных протоколов или на более нижнем уровне с помощью специальных архитектур. Одним из решений данной проблемы, предложенным в работе [10], является передача недоверенной исполняющей стороне (серверу) вместе с программой верхней оценки на число итераций цикла. При этом программа заранее должна быть преобразована так, что выполнение дополнительных лишних итераций циклов не влияет на результат, необходимый клиенту. Сама программа в [10] представлена в виде набора базовых блоков $B_1,...,B_k$, каждый из которых представляет собой упорядоченную последовательность инструкций и имеет одну «точку входа» и одну «точку выхода». Каждый базовый блок вычисляется «автоматически» – поток управления входит сверху базового блока и все инструкции на протяжении базового блока выполняются линейно одна за другой в установленном порядке. Каждый базовый блок имеет открытым параметром фиксированный адрес, который соответствует уникальному программному счетчику, обозначаемому как PC, передача которому адреса j-го базового блока $PC(B_j)$ означает переход потока управления к базовому блоку B_j и начало его выполнения. Далее в [10] предлагается разбить поток управления программы на «ярусы» (см. пример на рис. 2) с целью минимизации количества переходов и вычислений условий и как следствие оптимизации выполнения программы.

В случае необходимости скрытия алгоритмов необходимо воспользоваться методами обфускации программ [12]. Одним из основных методов обфускации программ является скрытие условия перехода (например, с помощью «затемнённых предикатов»).

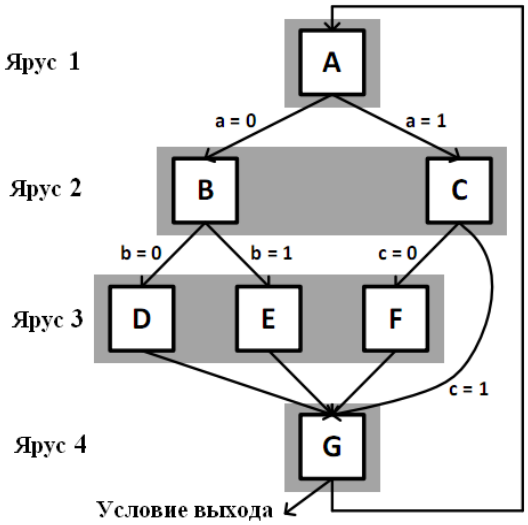

Рис. 2. Диаграмма потока управления четырехъярусного цикла

К вопросам, относящимся к данному уровню, относятся вопросы организации циклов, рекурсии. В некоторых случаях программу удается преобразовать так, чтобы исключить зависимости потока управления по данным, в частности циклы с зависимостями по данным (data-dependent loops). Интересно отметить, что задача исключения таких циклов уже рассматривалась исследователями в контексте распараллеливания программ [20] (для эффективного распараллеливания необходимо знать количество итераций цикла до начала выполнения программы).

5. Уровень команд виртуальной машины (уровень архитектуры). На этом уровне возможно формальное описание архитектур для исследования их возможностей и ограничений. Как известно, две основные архитектуры компьютера – это принстонская и гарвардская (архитектура фон Неймана): принстонская архитектура предполагает разделять «физически» хранение и передачу данных от хранения и передачи инструкций, гарвардская же, напротив, предполагает хранение инструкций и данных в одном месте и передачу их по одним каналам.

Традиционная гарвардская архитектура СКЗВ не позволяет решить проблему остановки зашифрованных вычислений в общем виде (т.е. для всех программ). Например, в [13] Бреннер с соавторами предлагают концепцию виртуальной вычислительной машины (гарвардской архитектуры) под названием ShapeCPU, способной выполнять зашифрованные операции над зашифрованными данными на основе полностью гомоморфного шифрования битов. Они описывают архитектуру ShapeCPU и предоставляют первую реализацию. Основные «строительные блоки» этого подхода – это зашифрованный доступ к памяти, зашифрованное АЛУ, зашифрованное CU, в которых все команды выражаются через гомоморфные операции криптосхемы. Разработанный зашифрованный доступ к памяти обеспечивает доступ к зашифрованным ячейкам памяти через зашифрованный адрес без утечки какой-либо информации об используемых данных или их местоположении. Каждый цикл ShapeCPU переписывает всю память, в том числе программные инструк-

ции и данные. Поэтому ShapeCPU поддерживает (хотя и крайне неэффективным образом) нелинейные программы с динамическими параметрами и самомодифицирующийся код.

В статье [11], например, предлагается архитектура виртуальной машины под названием VME, которая, хотя и не решает проблему остановки, но позволяет осуществить ускоренный доступ к памяти по зашифрованному адресу с помощью специальных методик (похожих на private information retrieval [1]). Это модифицированная гарвардская архитектура, в которой память разделена на «хранилище для инструкций», состоящее из n блоков и позволяющее только чтение, и хранилище для данных, состоящее из двух блоков, в первом из которых хранятся «указатели на данные», т.е. индексы, а во втором блоке – сами данные (на которые ссылаются указатели из первого блока). Такая архитектура снижает сложность вычислений при отсутствии необходимости в самомодифицирующемся коде.

Для вычисления над зашифрованными данными сложных компьютерных программ необходимо выразить их в виде последовательности элементарных машинных команд, таких, например, как команды языка assembler в архитектуре IA-32. В [11] реализованы такие стандартные команды как add, mult, cmp для обработки многоразрядных данных (всего 8 команд, табл. 2).

Таблица 2

Коды операций виртуальной машины VME

Код операции	Описание	Выражение на языке C
000	Условный переход	if a goto label
001	Побитовая операция XOR	a = a ^ b
010	Побитовая операция AND	a = a & b
011	Побитовый сдвиг влево	a = a << b
100	Побитовый сдвиг вправо	a = a >> b
101	Команда присвоения (основная)	a = b
110	Команда сравнения	a = a < b
111	Команда сложения	a = a + b

На данном логическом уровне ОМКЗВ рассматриваются RISC и CISC наборы команд, команды для операций с плавающей точкой и остальные стандартные операции, не работающие с косвенной адресацией в основной памяти.

Для некоторых целей (установления некоторых свойств) на данном уровне работа системы может описываться и в виде «ограниченной машины Тьюринга» (принстонской архитектуры), как, например, в [10]. Как было описано в предыдущем разделе, программы состоят из набора базовых блоков. Программный код в таком случае представляется в виде последовательности команд двух типов: вычитание и условный переход (разветвление). Вычитание, обозначаемое как sub a,b, действует по формуле

$$M[b] \leftarrow M[b] - M[a],$$

где M – это набор ячеек памяти («лента машины Тьюринга»), доступ к которым осуществляется по адресу с помощью индекса. А условный переход, обозначаемый инструкцией branch a, BBX, BBY, действует как:

если $M[a] < 0$, то $\mathrm{PC} \leftarrow PC(BBX)$

иначе $\mathrm{PC} \leftarrow PC(BBY)$

где $PC(BBX)$ обозначает «адрес базового блока BBX». Вместе эти две команды полны по Тьюрингу.

6. Уровень криптографических протоколов и систем. Для организации КЗВ может быть использовано множество различных криптографических примитивов: гомоморфное шифрование, протоколы многосторонних секретных вычислений, проверяемое шифрование, функциональное шифрование и т.д.

В частности, на данном уровне определяется количество клиентов, серверов, условия взаимодействия между клиентами (возможность передачи прав доступа и/или модификации данных и алгоритмов).

Скрытие не только данных, но и вычисляемых функций предъявляет свои требования «сверху» к криптографическим протоколам, однако дополнительные требования, возникающие на данном уровне – это модель безопасности (модель нарушителя). Самая слабая модель сервера-нарушителя – «пассивная» – так называемая «честный, но любопытный» (honest-but-curious), предполагает, что СКЗВ выполняет все инструкции клиента в точности, не внося изменений в алгоритмы обработки или данные клиента.

Основные вопросы, решаемые на данном уровне – это вопросы обеспечения достаточной криптостойкости и вопросы, связанные с пространствами открытых текстов. Сюда же относятся вопросы обеспечения проверяемости вычислений и вопросы многопользовательского взаимодействия. И наконец, многие принципиальные вопросы эффективности могут быть решены на этом уровне. Например, может быть принято решение о необходимости повышения эффективности за счет использования параллельной пакетной обработки зашифрованных данных [4] – при такой обработке пространство открытых текстов является как бы «составным».

К уровню криптографических протоколов и систем также относятся программные библиотеки функций гомоморфного шифрования такие как HELib [1] или другие реализации [13].

7. Уровень реализации элементарных операций. При фиксированных криптографических протоколах и криптосистемах, как правило, становится известным, из каких операций состоит вычисление и взаимодействие. Соответственно, для оптимизации эффективности работы всей системы необходимо оптимизировать выполнение этих элементарных операций.

В большинстве случаев такая оптимизация подразумевает аппаратное ускорение или аппаратную реализацию, например, реализацию с помощью ПЛИС. Большинство исследовательских работ на данном уровне посвящено так называемому полностью гомоморфному шифрованию с использованием целых чисел и, соответственно, аппаратным способам ускорения операций с целыми числами [15–18]. Однако для других криптосистем может быть актуально уметь эффективно проводить операции с полиномами от одной или многих переменных, матрицами или другими математическими объектами. Обзор различных способов ускорения полностью гомоморфных криптосхем можно найти в [19].

8. Верификация ОМКЗВ. Итак, чем же полезна описанная модель? Прежде всего тем, что каждый уровень позволяет формально поставить какую-то задачу и решить ее в рамках собственной модели (вместе с возможностью проверки правильности решения!). Сложная задача подразделяется на ряд элементарных этапов (по принципу «разделяй и властвуй») и становится более простой (см. рис. 3).

Возникает также вопрос: зачем каждый раз проектировать новую архитектуру, не легче ли взять во всех случаях самую универсальную. Однако здесь возникает противоречие «Эффективность-универсальность»: чем больше универсальность архитектуры, тем меньше её эффективность.

Модель предназначена для установления взаимосвязей между требованиями к СКЗИ и *требованиями к каждой из её частей*. Предложена поэтапная схема проектирования СКЗВ.

Согласно общепринятым канонам модель должна удовлетворять следующим свойствам: *адекватность* (соответствие модели исходной реальной системе и учет прежде всего наиболее важных качеств, связей и характеристик), *точность* (степень совпадения полученных в процессе моделирования результатов с характеристиками реального объекта), *универсальность* (применимость модели к анализу ряда однотипных систем в одном или нескольких режимах функционирования), *целесообразность* (точность получаемых результатов и общность решения задачи должны увязываться с затратами на моделирование). Рассмотрим более подробно, как ОМКЗВ удовлетворяет этим свойствам.

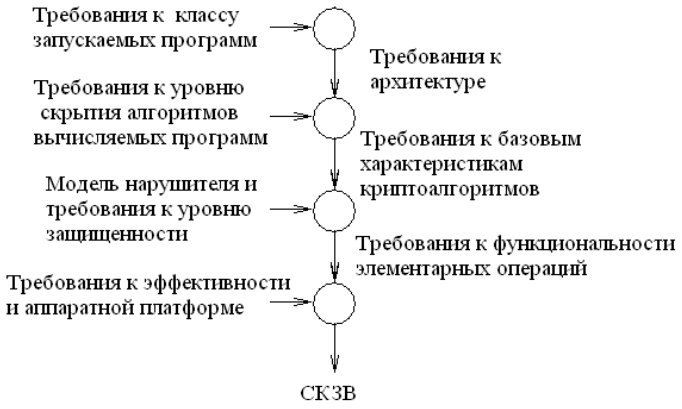

Рис. 3. Процесс проектирования СКЗВ с помощью предложенной модели

Адекватность. Разработанная нами модель позволяет адекватно представить все составляющие, необходимые для функционирования СКЗВ, определить требования к ним так, чтобы они работали согласованно для обеспечения заданных характеристик всей СКЗВ, а также оценить затраты на проектирование СКЗВ.

Точность. Точность представленной модели характеризуется достаточной дифференциацией уровней для отделения различных характеристик СКЗИ.

Универсальность. Представленная модель подходит для всех видов вычислений над зашифрованными данными и любых возможных видов угроз.

Целесообразность. Для определения взаимосвязей между частями СКЗВ представленная модель является необходимой.

Заключение. Предложена шестиуровневая информационная модель организации СКЗВ и рассмотрены объекты, которые представляет каждый уровень. Приведен краткий обзор с классификацией работ по уровням предложенной модели. Основное достоинство модели – возможность обнаружить взаимосвязь требований к СКЗИ в целом и требований к её составным частям, служить в качестве эталона и плана при проектировании и разработке реальной СКЗИ.

Подытожив, можно сказать, что предлагаемая модель СКЗИ обладает принципиально новыми свойствами:

1) известные модели решают свои задачи в рамках узкого контекста ситуации, в то время как предлагаемая модель позволяет судить о системе в целом, дает комплексное представление об СКЗИ;
2) предлагаемая модель позволяет упростить проектирование СКЗИ вплоть до его автоматизации, в отличие от известных моделей;
3) известные модели не позволяют связать общие требования к СКЗИ и требования к её частям;

4) известные модели не позволяют выбрать криптографические средства, отвечающие наилучшим образом на риски и угрозы информационной безопасности, тогда как предлагаемая модель принципиально позволяет решить эту задачу.

В дальнейшем планируется создать более формальную модель, на основе которой возможно было бы построение экспертной системы. Такая экспертная система, имея базу знаний по криптографическим протоколам, реализациям, программным библиотекам и т.д. и получая на вход некоторое формальное описание пользовательских требований, могла бы предложить конкретные протоколы и методы из числа имеющихся, которые бы оптимально отвечали этим требованиям либо установить, что имеющимися средствами невозможно создать СКЗИ, отвечающую этим требованиям.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Guellier A.* Can Homomorphic Cryptography ensure Privacy?: diss. – Inria; IRISA; Supélec Rennes, équipe Cidre; Université de Rennes 1, 2014.
2. *Burtyka P., Makarevich O.* Symmetric Fully Homomorphic Encryption Using Decidable Matrix Equations // Proceedings of the 7th International Conference on Security of Information and Networks. – ACM, 2014. – P. 186.
3. *Буртыка Ф.Б.* Симметричное полностью гомоморфное шифрование с использованием неприводимых матричных полиномов // Известия ЮФУ. Технические науки. – 2014. – № 8 (157). – С. 107-122.
4. *Буртыка Ф.Б.* Пакетное симметричное полностью гомоморфное шифрование на основе матричных полиномов // Труды ИСП РАН. – 2014. – Т. 26, № 5. – С. 99-115.
5. *Bain A., Mitchell J., Sharma R., Stefan D., Zimmerman J.* A domain-specific language for computing on encrypted data // 31st International Conference on Foundations of Software Technology and Theoretical Computer Science. – 2011. – P. 6.
6. *Malkhi D., Nisan N., Pinkas B., Sella Y.* Fairplay – Secure Two-Party Computation System // USENIX Security Symposium. – 2004. – Vol. 4.
7. *Bogetoft P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen T.P., Krøigaard Nielsen J.D., Nielsen J.B., Nielsen K., Pagter J., Schwartzbach M., and Toft T.* Secure multiparty computation goes live // Financial Cryptography and Data Security. – Springer Berlin Heidelberg, 2009. – P. 325-343.
8. *Nielsen J.D., Schwartzbach M.I.* A domain-specific programming language for secure multiparty computation // Proceedings of the 2007 workshop on Programming languages and analysis for security. – ACM, 2007. – P. 21-30.
9. *Mitchell, J. C., Sharma, R., Stefan, D., Zimmerman, J.* Information-flow control for programming on encrypted data //Computer Security Foundations Symposium (CSF), 2012 IEEE 25th. – IEEE, 2012. – P. 45-60.
10. *Fletcher C. W., Dijk M., Devadas S.* Towards an interpreter for efficient encrypted computation // Proceedings of the 2012 ACM Workshop on Cloud computing security workshop. – ACM, 2012. – P. 83-94.
11. *Zhuravlev D., Samoilovych I., Orlovskyi R., Bondarenko I., Lavrenyuk Y.* Encrypted Program Execution // Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. – IEEE, 2014. – P. 817-822.
12. *Варновский Н. П., Захаров В.А., Кузюрин Н.Н., Шокуров А.В.* Современное состояние исследований в области обфускации программ: определения стойкости обфускации // Труды ИСП РАН. – 2014. – Т. 26, № 3. – P. 167-198.
13. *Brenner M., Perl H., Smith M.* How practical is homomorphically encrypted program execution? An implementation and performance evaluation // Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on. – IEEE, 2012. – P. 375-382.
14. *Cousins, D. B., Rohloff, K., Peikert, C., Schantz, R.* SIPHER: Scalable implementation of primitives for homomorphic encryption–FPGA implementation using Simulink // High Performance Extreme Computing Conference. – 2011.

15. *Moore C., O'Neill M., Hanley N., O'Sullivan E*. Accelerating integer-based fully homomorphic encryption using Comba multiplication // Signal Processing Systems (SiPS), 2014 IEEE Workshop on. – IEEE, 2014. – P. 1-6.
16. *Doröz Y., Öztürk E., Sunar B*. A million-bit multiplier architecture for fully homomorphic encryption // Microprocessors and Microsystems. – 2014. – Vol. 38, No. 8. – P. 766-775.
17. *Wang, W., Hu, Y., Chen, L., Huang, X., & Sunar, B*. Accelerating fully homomorphic encryption using GPU // High Performance Extreme Computing (HPEC), 2012 IEEE Conference on. – IEEE, 2012. – P. 1-5.
18. *Moore C., Hanley N., McAllister J., O'Neill M., O'Sullivan E., Cao X*. Targeting FPGA DSP slices for a large integer multiplier for integer based FHE // Financial Cryptography and Data Security. – Springer Berlin Heidelberg, 2013. – P. 226-237.
19. *Moore C., O'Neill M., O'Sullivan E., Doröz Y., & Sunar B*. Practical homomorphic encryption: A survey // Circuits and Systems (ISCAS), 2014 IEEE International Symposium on. – IEEE, 2014. – P. 2792-2795.
20. *Curatelli F., Mangeruca L*. A method for computing the number of iterations in data dependent loops // Real-Time Systems. – 2006. – Vol. 32, No. 1-2. – P. 73-104.

REFERENCES

1. *Guellier A.* Can Homomorphic Cryptography ensure Privacy?: diss. Inria; IRISA; Supélec Rennes, équipe Cidre; Université de Rennes 1, 2014.
2. *Burtyka P., Makarevich O*. Symmetric Fully Homomorphic Encryption Using Decidable Matrix Equations, *Proceedings of the 7th International Conference on Security of Information and Networks*. ACM, 2014, pp. 186.
3. *Burtyka F.B.* Simmetrichnoe polnost'yu gomomorfnoe shifrovanie s ispol'zovaniem neprivodimykh matrichnykh polinomov [Symmetric fully homomorphic encryption using irreducible matrix polynomials], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2014, No. 8 (157), pp. 107-122.
4. *Burtyka F.B.* Paketnoe simmetrichnoe polnost'yu gomomorfnoe shifrovanie na osnove matrichnykh polinomov [Batch fully symmetric homomorphic encryption based on matrix polynomials], *Trudy ISP RAN* [Proceedings of ISP RAS], 2014, Vol. 26, No. 5, pp. 99-115.
5. *Bain A., Mitchell J., Sharma R., Stefan D., Zimmerman J*. A domain-specific language for computing on encrypted data, *31st International Conference on Foundations of Software Technology and Theoretical Computer Science*, 2011, pp. 6.
6. *Malkhi D., Nisan N., Pinkas B., Sella Y.* Fairplay – Secure Two-Party Computation System, *USENIX Security Symposium*, 2004, Vol. 4.
7. *Bogetoft P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen T.P., Krøigaard Nielsen J.D., Nielsen J.B., Nielsen K., Pagter J., Schwartzbach M., and Toft T.* Secure multiparty computation goes live, *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2009, pp. 325-343.
8. *Nielsen J.D., Schwartzbach M.I*. A domain-specific programming language for secure multiparty computation, *Proceedings of the 2007 workshop on Programming languages and analysis for security*. ACM, 2007, pp. 21-30.
9. *Mitchell, J. C., Sharma, R., Stefan, D., Zimmerman, J*. Information-flow control for programming on encrypted data, *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*. IEEE, 2012, pp. 45-60.
10. *Fletcher C. W., Dijk M., Devadas S*. Towards an interpreter for efficient encrypted computation, *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*. ACM, 2012, pp. 83-94.
11. *Zhuravlev D., Samoilovych I., Orlovskyi R., Bondarenko I., Lavrenyuk Y*. Encrypted Program Execution, *Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on.* IEEE, 2014, pp. 817-822.
12. *Varnovskiy N. P., Zakharov V.A., Kuzyurin N.N., Shokurov A.V.* Sovremennoe sostoyanie issledovaniy v oblasti obfuskatsii programm: opredeleniya stoykosti obfuskatsii [The current state of research in the field of obfuscation programs: determination of resistance obfuscation], *Trudy ISP RAN* [Proceedings of ISP RAS], 2014, Vol. 26, No. 3. pp. 167-198.

13. *Brenner M., Perl H., Smith M*. How practical is homomorphically encrypted program execution? An implementation and performance evaluation, *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on.* IEEE, 2012, pp. 375-382.
14. *Cousins, D. B., Rohloff, K., Peikert, C., Schantz, R*. SIPHER: Scalable implementation of primitives for homomorphic encryption–FPGA implementation using Simulink, *High Performance Extreme Computing Conference*, 2011.
15. *Moore C., O'Neill M., Hanley N., O'Sullivan E*. Accelerating integer-based fully homomorphic encryption using Comba multiplication, *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 2014, pp. 1-6.
16. *Doröz Y., Öztürk E., Sunar B*. A million-bit multiplier architecture for fully homomorphic encryption, *Microprocessors and Microsystems*, 2014, Vol. 38, No. 8, pp. 766-775.
17. *Wang, W., Hu, Y., Chen, L., Huang, X., & Sunar, B*. Accelerating fully homomorphic encryption using GPU, *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*. IEEE, 2012, pp. 1-5.
18. *Moore C., Hanley N., McAllister J., O'Neill M., O'Sullivan E., Cao X*. Targeting FPGA DSP slices for a large integer multiplier for integer based FHE, *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2013, pp. 226-237.
19. *Moore C., O'Neill M., O'Sullivan E., Doröz Y., & Sunar B*. Practical homomorphic encryption: A survey, *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 2792-2795.
20. *Curatelli F., Mangeruca L*. A method for computing the number of iterations in data dependent loops, *Real-Time Systems*, 2006, Vol. 32, No. 1-2, pp. 73-104.

Статью рекомендовал к опубликованию д.т.н., профессор Н.И. Витиска.

Бабенко Людмила Климентьевна – Южный федеральный университет; e-mail: blk@fib.tsure.ru; 347928, г. Таганрог, ул. Чехова, 2, корпус "И"; тел.: 88634312018; кафедра безопасности информационных технологий; профессор.

Буртыка Филипп Борисович – e-mail: bbfilipp@ya.ru; тел.: +79081948371; кафедра безопасности информационных технологий; программист.

Макаревич Олег Борисович – e-mail: mak@tsure.ru; кафедра безопасности информационных технологий; профессор.

Трепачева Алина Викторовна – e-mail: alina1989malina@ya.ru; тел.: +79085196604; кафедра безопасности информационных технологий; программист.

Babenko Lyudmila Klimentevna – Southern Federal University; e-mail: blk@fib.tsure.ru; Block "I", 2, Chehov street, Taganrog, 347928, Russia; phone: +78634312018; the department of information technologies security; professor.

Burtyka Philipp Borisovich – e-mail: bbfilipp@ya.ru; phone: +79081948371; the department of information technologies security; programmer.

Makarevich Oleg Borisovich – e-mail: mak@tsure.ru; the department of information technologies security; professor.

Trepacheva Alina Viktorovna – e-mail: alina1989malina@ya.ru; phone: +79085196604; the department of information technologies security; programmer.