ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А.Н. Тихонова

Гвасалиа Метью

Перевод статьи Жиля Барте и др. "Формальная спецификация схемы Эль-Гамаля"

Курсовая работа
по специальности 10.05.01 «Компьютерная безопасность»
студента образовательной программы специалитета

Студент

подпись

Гвасалиа Метью

ФИО

Преподаватель
доцент

подпись

Нестеренко А.Ю.

ФИО

Москва, 2021 г.

Оглавление

1 Введение **2**

2 Доказуемая стойкость и техника, основанная на играх **4**

3 Введение в CertiCrypt **7**
3.1 Синтаксис и семантика игр 7
3.2 Анализ игр 9

4 Семантическая безопасность хешированного шифрования Эль-Гамаля **12**
4.1 Стойкость в стандартной модели 13

5 Похожие работы **14**
5.1 Заключение 14

6 Информация о курсовой **16**
6.1 Ссылка на репозиторий 16
6.2 Компилятор 16
6.3 Поля и шрифты 16
6.4 Стилевые пакеты 16
6.5 Кастомная команда и сниппеты 17

Аннотация

CertiCrypt [?] - фреймворк, который позволяет создавать верифицируемые компьютером криптографические доказательства, которые могут быть автоматически проверены третьей стороной. На данный момент, **CertiCrypt** использовался для формального доказательства стойкости хорошо изученных криптогрфических схем, таких как OAEP (Оптимальное асимметричное шифрование с дополнением) и FDH. Цель данной статьи - введение читателя в CertiCrypt на простом и наглядном примере - семантическая стойкость хешированной схемы Эль-Гамаля в моделях со стандартным и случайным оракулом.

Глава 1

Введение

CertiCrypt [?] - фреймворк, который позволяет создавать верифицируемые компьютером криптографические доказательства в стиле доказуемой криптостойкости [?, ?]. Согласно данному стилю, взаимодействие между схемой и злоумышленником должно быть точно описано, и доказательство стойкости в данном случае дожно быть строгим. **CertiCrypt** основан на игровом подходе к криптографическим доказательствам [?, ?, ?]. Данный подход использует техники, которые позволяют понизить сложность криптографических доказательств, представляя доказательства в виде шагов адекватного размера. На данный момент, **CertiCrypt** использовался для формального доказательства стойкости хорошо изученных криптогрфических схем, таких как OAEP (Оптимальное асимметричное шифрование с дополнением) и FDH, и для установления возможности широкого применения данного фреймворка в криптографических доказательствах, например доказательства леммы об эвивалентности стойкости PRP и PRF и фундаментальной леммы игр.

CertiCrypt написан на интерактивном средстве доказательства теорем Coq [?], благодоря которому CertiCrypt обладает высоким уровнем надежности и способностью предоставлять доказательства того, что доказательства верны. Одной из целей проекта является увелечение доверия к криптографическим доказательствам. Что неудивительно, ведь построение доказательства криптостойкости может быть настолько тонким делом, что некоторые схемы известны своими неубедительными доказательствами криптостойкости, которые не могли опровергнуть спустя годы. Ситуация еще хуже, так как есть сомнения в криптографических доказательствах в целом [?, ?]. Возможным решением данной проблемы является предложение Халеви [?] создать и использовать специальные инструменты (для доказательств), он также указал некоторые свойства, которыми данные инструменты должны обладать. В каком-то смысле **CertiCrypt** является первым шагом на пути к завершению программы Халеви, несмотря на то, что CertiCrypt скорее предоставляет автоматизацию, выразительность и высокий уровень надеждности, а не предоставляет интерфейс для наброска доказательств.

Главной сложностью при создании инструмента для верификации криптографических доказательств является то, что данные доказательства обычно содержат широкий набор концепций и рассуждений из мира теории вероятностей, теории груп и теории сложности вычислений. В случае игрового подхода, доказательства еще опираются на семантику языков программирования и на трансформацию програм и их верификацию. Несмотря на то, что все эти аспекты учтены в **CertiCrypt**, впервые мы затрагиваем некоторые необходимые детали, которые возникают при использовании инструмента для построения конкретного доказательства.

Цель данной статьи - введение читателя в CertiCrypt. Мы предоставляем пошаговые доказательства стойкости хешированной схемы Эль-Гамаля, надеясь, что это поможет читателям понять некоторые тонкие детали фреймворка. В продолжение статьи вводной

статьи Шупа о доказательствах, основанных на играх, мы предъявляем доказательства в стандартной модели и в модели со случайным оракулом, предполагая, что хэш функцию нельзя отличить от чисто случайной функции. Эти доказательства обощают наше прошлое доказательство стойкости схемы Эль-Гамаля, которое было кратко описано в [?].

Глава 2

Доказуемая стойкость и техника, основанная на играх

Цель криптографии - достичь определенного уровня криптостойкости. независимо от поведения злоумышленников. Однако нельзя просто перечислить все способы, которые злоумышленники могут использовать для взлома схемы, и создать идеальную криптосхему. Подобная методология обречена на провал, поскольку злоумышленники будут действовать непредсказуемо, чтобы обойти все контрмеры. Именно поэтому правильные доказательства должны установить защищенность от всех потенциальных злоумышленников. Не каждый противник считается потенциальным, на потенциальных противников накладываются определенные ограничения, которые делают возможным доказательство стойкости. В частности, предполагается, что злоумышленник не является всезнающим (например, он не знает какие-то секреты) и всемогущим (не может выполнять трудоемкие вычисления). Оба предположения будут формализованы с помощью управления доступом и ресурсами с одной стороны, и классами сложности с другой.

В игровой технике, которую использует **CertiCrypt**, криптостойкость выражена через вероятностную программу, в которой отражено взаимодействие между криптосистемой и противником. В этой статье мы будем фокусироваться на системах шифрования с открытым ключом и на их семантической стойкости (или $IND-CPA$ стойкость), которая гарантирует неразличимость шифртекста против атак на основе подобранного шифртекста. Проще говоря, система шифрования с открытым ключом семантически стойкая если ни один потенциальный злоумышленник, зная открытый ключ и выбирая пару сообщений (m_0, m_1), не может определить в каком случае ему был дан результат шифрования m_0, а когда - результата шифрования m_1. Очевидно, необходимое условие для криптосхемы, удовлетворяющей данному свойству, - быть вероятностной, потому что в другом случае злоумышленник может просто сравнить результат шифрования m_0 с шифртекстом, который ему предоставили. В игровом стиле, семантическая стойкость описывается следующей программой:

Здесь KG - алгоритм генерации ключей схемы и Enc - алгоритм шифрования, а $\mathcal{A}$ и $\mathcal{A}'$ процедуры представляющие злоумышленника; например в хешированной схеме Эль-

Игра IND-CPA $(sk, pk) \leftarrow KG()$;
(m_0, m_1) $gets\mathcal{A}(pk)$;
$b \xleftarrow{\$} 0, 1$;
$\eta \leftarrow Enc(pk, m_b)$;
$b' \leftarrow \mathcal{A}'(pk, \eta)$;
$d \leftarrow b = b'$

Гамаля злоумышленнику предоставлен доступ к случайному оракулу. Спецификация игры $IND - CPA$ завершается указанием того, что злоумышленник относится к классу вероятностных программ, выполняющихся за полиномиальное время, и что у него есть доступ к глобальной переменной для поддержания состояния, доступ только на чтение pk, но не имеет доступа к sk или b. Есть два способа для управления доступом к переменным: можно определить переменную как локальную, чтобы она была доступна только в области видимости процедуры, или как глобальную, в этом случае доступ будет ограничен явно указанной политикой доступа.

Если схема удовлетворяет свойству $IND-CPA$, вероятность того, что злоумышленник угадает какое сообщение было зашифровано не значительно больше $\frac{1}{2}$. Точное определение включает в себя параметр стойкости η (который определяет параметры схемы) и требует: вероятность того, что $d = 1$ по окончанию игры, или $Pr_{IND-CPA^\eta}[d = 1]$, близка к $\frac{1}{2}$ как функция от η. Формально, функция $v : \mathbb{N} \mapsto \mathbb{R}$ называется незначительной, тогда и только тогда, когда

$$negligible(v) \stackrel{\text{def}}{=} \forall c.\exists n_c.\forall n.n \geq n_c \implies |v(n)| \leq n^{-c}$$

Мы считаем, что функция v незначительно близка к константе k, когда функция $\lambda\eta.|v(\eta) - k|$ незначительна.

Суть игрового метода заключается в том, что свойство стойкости, такое как $IND - CPA$, доказывается через преобразования оригинальной игры-атаки. Если быть точнее, доказательства, которые следуют игровому методу, организованы как последовательности переходов вида $G, A \to G', A'$, где G и G' - игры, а A и A' - события. Цель состоит в том, чтобы установить для каждого перехода неравенство $Pr_G[A] \leq f(Pr_{G'}[A'])$, где f - некоторая монотонная функция. Совмещая последовательно неравенства, полученные из каждого перехода, можно извлечь из игрового доказательства неравенство $Pr_{G_0}[A_0] \leq f(Pr_{G_n}[A_n])$. Таким образом, если G_0, A_0 обозначают исходную игру-атаку и атаку, можно получить оценку $Pr_{G_0}[A_0]$ из оценки $Pr_{G_n}[A_n]$.

Во многих случаях переходы $G, A \to G', A'$ удовлетворяют условию: $Pr_G[A] = Pr_{G'}[A']$. Такие переходы называются промежуточными этапами, они включают в себя преобразования программы, сохраняющие семантику. Формально сохранение семантики определяется с помощью вероятностного невмешательства [8], поскольку нас интересует только сохранение наблюдаемого поведение игр. Однако во многих случаях сохранение семантики контекстно-зависимо; для учета таких случаев необходимо прибегнуть к реляционной логике, которая обобщает вероятностное невмешательство и позволяет рассуждать о до- и пост- условиях.

Доказательства на основе игрового метода часто опираются на сбои, которые помогают ограничить потерю вероятности при переходах с помощью вероятности появления сбоя. Одним важным инструментом для анализа сбоев является так называемая Фундаментальная лемма: для двух игр G_1 и G_2, код которых отличается только после поднятия определенного флага (т.е. после присваивания $bad \leftarrow true$, где bad изначально имеет значение false и всегда остается поднятым после установки), можно сделать вывод, что для любого события A, $Pr_{G_1}[A \wedge \neg bad] = Pr_{G_2}[A \wedge \neg bad]$. Это, в свою очередь, подразумевает, что

$$|Pr_{G_1}[A] - Pr_{G_2}[A]| \leq Pr_{G_1}[bad] = Pr_{G_2}[bad]$$

при условии, что обе игры заканчиваются с одинаковой вероятностью.

Наконец, некоторые переходы оправданы предположениями. Например, доказательство в разделе 4.2 основано на предположении Диффи-Хеллмана о принятии решения (?) или сокращенно предположении DDH. Для семейства конечных циклических групп это предположение гласит, что ни один эффективный алгоритм не может отличить тройки вида (g^x, g^y, g^{xy}) от тройки вида (g^x, g^y, g^z), где x, y, z равномерно распределены на $\mathbb{Z}_q$,

где q - (простой) порядок группы, а g - генератор. Одной из характеристик игровых доказательств является формулировка этих предположений через игры; предположение DDH формулируется следующим образом

Определение 1 (предположение DDH). *Рассмотрим игры*

Игра DDH_0 :
$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^{xy})$

Игра DDH_1 :
$x, y, z \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^z)$

и определим

$$\epsilon_{DDH}(\eta) \stackrel{\text{def}}{=} |Pr_{DDH_0^\eta}[d = 1] - Pr_{DDH_1^\eta}[d = 1]|$$

Тогда для каждого PPT злоумышленника $\mathcal{B}$, ϵ_{DDH} является незначительной функцией. Обратите внимание, что семантика вышеупомянутых игр (и, в частности, порядок q группы) зависит от параметра безопасности η.

Глава 3

Введение в CertiCrypt

Цель данного раздела заключается в том, чтобы дать краткий обзор фреймворка. Сначала мы покажем синтаксис и семантику языка, используемого для описания игр, а затем инструменты, которые фреймворк предоставляет для их анализа.

3.1 Синтаксис и семантика игр

Самый низкий уровень CertiCrypt - формализация вероятностного языка программирования с вызовами процедур. При заданном наборе $\mathcal{V}$ переменных и наборе $\mathcal{P}$ имен процедур, команды могут быть определены индуктивно с помощью выражений:

$\mathcal{I} ::= \mathcal{V} \leftarrow \mathcal{E}$	присваивание
$\mid \mathcal{V} \xleftarrow{\$} \mathcal{D}$	случайное присваивание
$\mid$ if $\mathcal{E}$ then $\mathcal{C}$ else $\mathcal{C}$	условный оператор
$\mid$ while $\mathcal{E}$ do $\mathcal{C}$	цикл while
$\mid \mathcal{V} \leftarrow P(\mathcal{E}, \cdots, \mathcal{E})$	вызов процедуры
$\mathcal{C} ::= nil$	nop
$\mid \mathcal{I} \; ; \mathcal{C}$	последовательность

где $\mathcal{E}$ - набор выражений, а $\mathcal{D}$ - набор распределений, откуда значения могут быть выбраны во время случайных присваиваний. Есть часто встречающиеся типы данных и операторы, но для адаптации к различным настройкам синтаксис может быть расширен пользователем: пользователи могут определять новые типы данных и операции, предоставляя адекватную интерпретацию в терминах Coq. Кроме того, синтаксис типизован, так что операторы и выражения имеют тотальную семантику.

Игры состоят из основной команды и среды, которая сопоставляет идентификатор процедуры с ее объявлением, состоящим из списка формальных параметров, тела и return выражения (однако при написании игр мы используем явный return)

$$declaration \stackrel{\text{def}}{=} \{params : V^*; body : \mathcal{C}; re : \mathcal{E}\}$$

Формально, тип игр: $\mathcal{C} \times (\mathcal{P} \rightarrow declaration)$. Семантика игр определена с помощью монады меры $M(X)$ Адеба и Полин [9]; конструктор типа, оператор подъема и операция

связывания определены следующим образом:

$$\begin{aligned} M(X) &\stackrel{\text{def}}{=} (X \to [0,1] \\ unit : X \to M(X) &\stackrel{\text{def}}{=} \lambda x.\lambda f.fx \\ bind : M(X) \to (X \to M(Y)) \to M(Y) & \\ &\stackrel{\text{def}}{=} \lambda\mu.\lambda M.\lambda f.\mu(\lambda x.Mxf) \end{aligned}$$

Эта монада может рассматриваться как специализация монады продолжения и позволяет обеспечить семантику игр в непрерывном стиле. Очевидно, элемент в $M(X)$ может быть интерпретирован как математическое ожидания (суб)распределения вероятностей на X. Таким образом, денотация игры связывает начальную память с оператором ожидания (суб) вероятностного распределения конечных воспоминаний, которое является результатом ее выполнения. Денотационная семантика игр определяется внутри с помощью семантики малых шагов, которая использует фреймы для обработки вызовов процедур. Однако с точки зрения пользователя эти детали можно игнорировать, не мешая пониманию; формальное определение семантики малых шагов можно найти в [1]. Обозначение игр представлено на рис. 1; на рисунке мы представляем память m как пару $(m.loc, m.glob)$, явно указывая ее локальные и глобальные компоненты. Выражения детерминированы и их семантика задается функцией $[\![\cdot]\!]_\epsilon$, которая вычисляет выражение для заданной памяти и возвращает значение. Семантика распределений в $\mathcal{D}$ задается другой функцией $[\![\cdot]\!]_{\mathcal{D}}$; мы приводим в качестве примеров семантику равномерного распределения на $\mathbb{B}$ и на целых интервалах $[0..n]$. На рисунке мы опустили среду процедуры E для удобства чтения. В дальнейшем мы часто не будем делать различий между игрой $G = (c, E)$ и ее основной командой c, когда среда, в которой она вычисляется, либо не важна, либо ясна из контекста. **CertiCrypt** предоставляет альтернативное, более удобное правила записи циклов

$$\begin{aligned} &[\![nil]\!]m && = unit\ m \\ &[\![i; c]\!]m && = bind([\![i]\!]m)[\![c]\!] \\ &[\![x \leftarrow e]\!]m && = unit\ m\{[\![e]\!]_\epsilon \frac{m}{x}\} \\ &[\![x \xleftarrow{\$} d]\!]m && = bind([\![d]\!]_{\mathcal{D}} m)(\lambda v.unit\ m\{\frac{v}{x}\}) \\ &[\![x \leftarrow f(e)]\!]m && = bind([\![E(f).body]\!](\{[\![e]\!]_{\mathcal{D}} \frac{m}{E(f).params}\}, m.glob)) \\ &[\![\text{if } e \text{ then } c_1 \text{ else } c_2]\!]m && = \begin{cases} [\![c_1]\!] \text{ if } [\![e]\!]_\epsilon m & = true \\ [\![c_2]\!] \text{ if } [\![e]\!]_\epsilon m & = false \end{cases} \end{aligned}$$

Рис. 3.1: Денотационная семантика игр

while:

$$[\![while\ e\ do\ c]\!]m\ f = sup\{[\![[while\ e\ do\ c]_n]\!]m\ f\ :\ n \in \mathbb{N}\}$$

где $[while\ e\ do\ c]_n$ - n-ый шаг цикла, то есть

$$\begin{aligned} &[while\ e\ do\ c]_0 && = nil \\ &[while\ e\ do\ c]_{n+1} && = if\ e\ then\ c;\ [while\ e\ do\ c]_n \end{aligned}$$

Обратите внимание, что функция $[\![\cdot]\!]$ отображает $\mathcal{M}$ в $M(\mathcal{M})$, однако очень просто определить функцию $[\![\cdot]\!]'$ из $M(\mathcal{M})$ в $M(\mathcal{M})$ с помощью оператора связывания монады: $[\![G]\!]' \stackrel{\text{def}}{=}$

bind μ $[\![G]\!]$. Главным преимуществом использования монады $M(\mathcal{M})$ является то, что вероятность события A, представленная как булевый предикат на памяти, может быть легко определен через характеристическую функцию $\mathbb{I}_A$ вместо A:

$$Pr_{G,m}[A] \stackrel{\text{def}}{=} [\![G]\!]\ m\ \mathbb{I}_A \tag{3.1}$$

Далее мы иногда будем опускать начальную память m; в таком случае можно спокойно предполагать, что изначально память отображает переменные в значения по умолчанию соответствующего типа.

3.2 Анализ игр

В доказательствах, основанных на игровом методе, переходные этапы в некотором смысле соответствуют преобразованиям, сохраняющим семантику; они используются для повторного определения способа вычисления определенных величин, чтобы подготовить почву для последующего преобразования. Следовательно, на этапе перехода от G, A к G', A' цель состоит в том, чтобы установить $Pr_{G,m}[A] = Pr_{G',m}[A']$. Если мы посмотрим на определение (3.1), это равносильно доказательству того, что $[\![G]\!]\ m\ \mathbb{I}_A = [\![G']\!]\ m\ \mathbb{I}_{A'}$, или обобщению этого на пару начальных запоминающих устройств m_1, m_2 и произвольных функций f, $g : \mathcal{M} \mapsto [0,1]$, что $[\![G]\!]\ m_1\ f = [\![G']\!]\ m_2\ g$.

Основным инструментом, предоставляемым **CertiCrypt** для установления таких равенств, является реляционная логика PRHL, которая обобщает реляционную логику Хоара [**?**] на вероятностный язык. Суждения в pRHL имеют вид $\models G_1 \sim G_2 : \Psi \Rightarrow \Phi$, где G_1 и G_2 - игры, а Ψ и Φ - отношения над детерминированными состояниями. Суждение справедливо $\models G_1 \sim G_2 : \Psi \Rightarrow \Phi$ т.т.т., когда для каждой пары начальных запоминающих устройств m_1, m_2 такое, что $m_1 \Phi m2$,$[\![G_1]\!]\ m_1 \sim_\Phi [\![G_2]\!]\ m_2$ выполняется. Отношение $\sim_\Phi$ является подъемом Φ к мерам. Если Φ - PER, то определение $\sim_\Phi$ довольно-таки интуитивно:

$$\mu_1 \sim_\Phi \mu_2 \stackrel{\text{def}}{=} \forall a. \mu_1\ \mathbb{I}_{[a]} = \mu_2\ \mathbb{I}_{[a]}$$

где $\mathbb{I}_{[a]}$ характеристическая функция класса эквивалентности a. Определение $\sim_\Phi$ для произвольных отношений менее очевидное и вовлекает квантор существования:

$$\begin{aligned} range\ P\mu \quad &\stackrel{\text{def}}{=} \forall f.\ (\forall a.\ Pa \Rightarrow\ f\ a = 0)\ \Rightarrow \mu\ f = 0 \\ \mu_1 \sim_\Phi \mu_2 \quad &\stackrel{\text{def}}{=} \exists \mu.\ \pi_1(\mu) = \mu_1 \wedge \pi_2(\mu) = \mu_2 \wedge range\ \Phi\ \mu \end{aligned}$$

где проекции μ определены следующим образом

$$\pi_1(\mu) \stackrel{\text{def}}{=} bind\ \mu\ (\lambda p.unit\ (fst\ p)),\ \pi_2(\mu) \stackrel{\text{def}}{=} bind\ \mu(\lambda p.unit\ (snd\ p))$$

Это определение следует из работ о вероятностных бисимуляциях и обощает подъем до произвольных соотношений. Оба определения совпадают для PER [**?**].

Для того, чтобы анализировать pRHL соотношения, **CertiCrypt** предоставляет множество выведенных правил и частное (слабое) исчисление предусловий. Правила могут быть найдены в статье [**?**]. Важное следствие из pRHL суждения $\models G_1 \sim G_2 : \Psi \Rightarrow \Phi$, что если две функции f и g не могут отличить членов памяти в соотношении Φ, то есть

$$\forall m_1, m_2.\ m_1\ \Phi\ m_2 \Rightarrow f\ m_1 = g\ m_2$$

тогда

$$\forall m_1, m_2.\ m_1\ \Phi\ m_2 \Rightarrow [\![G_1]\!]\ m_1\ f = [\![G_2]\!]\ m_2\ g$$

В частности, если Φ - равенство свободных переменных булевого предиката A, мы получаем, что $Pr_{G_1,m_1}[A] = Pr_{G_2,m_2}[A]$. Это свойство продолжается до $\leq$ соотношения.

Специализируя pHRL суждения в предикаты равенства на множестве переменных, можно получить вероятностное невмешательство: при заданном X - множестве переменных, определяем

$$m_1 =_X m_2 \stackrel{\text{def}}{=} \forall x \in X, m_1\ x = m_2\ x$$

Вероятностное невмешательство по отношению к набору I входных переменных и набору O выходных переменных определяется как $\models \cdot \sim \cdot :=_I \Rightarrow =_O$, мы используем $\models \cdot \simeq_O^I \cdot$ в качестве сокращения.

CertiCrypt предоставляет несколько инструментов для обоснования невмешательства. В частности, **CertiCrypt** реализует несколько тактик, которые помогают установить невмешательство или свести его к более простой цели. Например, тактика **eqobs_in** реализует процедуру полурешения для суждений вида $\models c, E \simeq_O^I\ c, E'$. Другие тактики, такие как **eqobs_hd, eqobs_tl, eqobs_ctxt, deadcode** и **swap**, упрощают цель, используя функции, которые принимают игры c_1, E_1 и c_2, E_2 и наборы переменных I, O, и возвращают c_1', c_2' и I', O' такие как

$$\models c_1', E_1 \simeq_{O'}^{I'} c_2', E_2 \Rightarrow \models c_1, E_1 \simeq_O^I c_2, E_2$$

Тактика отличается своей стратегией вычисления c_1', c_2' и I', O'. Тактика **eqobs_tl** ищет максимальный общий префикс c такой как $c_1 = c; c_1'$ и $c_2 = c; c_2'$, **eqobs_hd** аналогичным образом ищет максимальный суффикс и **eqobs_ctxt** сочетает в себе оба. Тактический обмен перестраивает инструкции в программах для создания наибольшего общего суффикса при сохранении эквивалентности наблюдений, т.е. $c_1' = \hat{c_1}; c$ и $c_2' = \hat{c_2}; c$ являются перестановками c_1 и c_2 (и $I' = I$ и $O' = O$). Тактика **deadcode** создает фрагменты исходных команд, используя переменные в O в качестве критериев среза. Кроме того, **CertiCrypt** автоматизирует другие распространенные программные преобразования: распространение выражений (**ep**), распределение переменных (**alloc**) и встраивание (**inline**). Показано, что эта тактика позволяет сохранить невмешательство. Тактика **sinline** сочетает в себе **inline**, **alloc**, **ep** и **deadcode** в одной мощной тактике.

Чтобы иметь возможность справляться с вызовами процедур, тактике необходима информация о процедурах в среде игр. Эта информация не может быть вычислена рекурсивно из-за присутствия злоумышленников, чей код неизвестен. Учитывая две среды E_1 и E_2 и процедуру f, тактика предполагает, что предоставляется следующая информация: Для каждой среды: набор W_i глобальных переменных, которые f может изменять, наборы I_i и O_i глобальных переменных и подмножество P_i его формальных параметров, так что для каждого выполнения тела процедуры конечные значения переменных в $O_i \cup fv(E_i(f).re)$ зависят только от начальных значений переменных во $I_i \cup P_i$. Формально,

$$\begin{aligned} W_i = globals(modifies(E_i(f).body,\ E_i)) \wedge \\ P_i \subseteq E_i(f).params\ \wedge \\ \models E_i(f).body,\ E_i \simeq_{O_i \cup fv(E_i(f).re)}^{I_i \cup P_i}\ E_i(f).body,\ E_i \end{aligned}$$

(modifies вычисляет чрезмерную аппроксимацию переменных, измененных фрагментом кода.) Эта информация используется тактиками *swap*, *deadcode*, *ep* и *inline*;
– Реляционная информация: наборы I и O глобальных переменных и набор формальных параметров, так что выполнение тела в каждой среде, начиная с памяти, равнозначно переменным $I \cup P$, приводит к мерам эквивалентными на $O \cup fv(E_i(f).re)$. Далее мы требуем, что $E_1 f.re = E_2(f).re$. Формально,

$$\begin{aligned} P_i \subseteq E_i(f).params\ \wedge P \subseteq E_2(f).params \wedge \\ \models E_i(f).body,\ E_i \simeq_{O \cup fv(E(f).re)}^{I \cup P}\ E_2(f).body,\ E_2 \end{aligned}$$

Эта информация используется тактиками *eqobs_in*, *eqobs_hd* и *eqobs_tl*. CertiCrypt предоставляет несколько механизмов для постепенного и автоматического построения вышеуказанной информации, когда тела процедур в E_1 и E_2 эквивалентны с точки зрения наблюдений распространению выражений по модулю и устранению мертвого кода. Также возможно получить информацию о противнике из информации о оракулах, которые он может вызвать. Это возможно при условии, что злоумышленник правильно сформирован, поскольку в этом случае мы знаем, что злоумышленник и любые подпроцедуры, которые он может вызвать, уважают политику управления доступом $(\mathcal{O}, \mathcal{RO}, \mathcal{RW})$: они могут вызывать оракулов только в $\mathcal{O}$, читать глобальные переменные в $\mathcal{RO}$ и читать или изменять глобальные переменные в $\mathcal{RW}$.

Как было сказано ранее, некоторые преобразования, выполняемые во время доказательств, зависят от контекста. CertiCrypt позволяет подробно описать контекст, в котором допустимо преобразование, с использованием программных инвариантов. Таким образом, тактика распространяется на инварианты глобальных переменных; информация, которую они используют, вместо этого указывается в суждениях формы

$$| = c_1, E_1\ c_2, E_2 :=_I \wedge\phi \Rightarrow =_o \wedge\phi|$$

Глава 4

Семантическая безопасность хешированного шифрования Эль-Гамаля

Пусть G - циклическая группа простого порядка q и g - генератор, и пусть $(H_k)_{k\in K}$ - семейство хеш-функций с ключами, отображающих элементы из G в битовые строки определенной длины $\updownarrow$. Хешированный Эль-Гамаль - это схема шифрования с открытым ключом, безопасность которой, как считается, связана с проблемой дискретного логарифмирования в G. Ее алгоритмы генерации ключей, шифрования и дешифрования определены следующим образом:

$$
\begin{array}{ll}
KG() & \stackrel{\text{def}}{=} k \xleftarrow{\$} K; x \xleftarrow{\$} \mathbf{Z}_q; \text{return}((k,x),(k,g^x)) \\
Enc(km\alpha, m) & \stackrel{\text{def}}{=} y \xleftarrow{\$} \mathbb{Z}_q; h \leftarrow H_k(\alpha^y);\ \text{return}\ (g^y, h, \oplus m) \\
Dec(k,x,\beta,\zeta) & \stackrel{\text{def}}{=} h \leftarrow H_k(\beta^x);\ \text{return}\ h \oplus \zeta
\end{array}
$$

Пространство открытого текста в хешированном Эль-Гамале равно $0,1^{\updownarrow}$, в отличие от исходной схемы шифрования Эль-Гамале, пространство открытого текста которой просто G.

В оставшейся части этого раздела мы представляем основанные на играх доказательства семантической безопасности шифрования хешированной схемы Эль-Гамаля в двух разных условиях. Первое доказательство проводится в стандартной модели криптографии; он предполагает, что семейство $(H_k)_{k\in K}$ хеш-функций является сглаживающим по энтропии, и снижает семантическую безопасность до сложности проблемы DDH. Второе доказательство проводится в модели случайного оракула (ROM); она предполагает, что хеш-функции ведут себя как совершенно случайные функции, и снижает семантическую безопасность до уровня сложности (списка) проблемы CDH.

Чтобы формализовать доказательства в **CertiCrypt**, нам сначала необходимо расширить синтаксис и семантику игр, включив в них типы и операторы, используемые в описании схемы, которые еще не определены. Как объяснено в разделе 3, это делается модульным способом. Мы объявляем семейство циклических групп $(G_\eta)_{\eta\in\mathbb{N}}$, индексируемых параметром безопасности, и расширяем типы языка с помощью определяемых пользователем типов для элементов G_η и битовых строк длиной l. Мы расширяем $\mathcal{D}$ с помощью равномерного распределения по битовым строкам длины l. Мы, наконец, расширяем операторы языка с помощью обнуляющих операторов q и g для извлечения порядка и генератора, соответственно, двоичных операторов для произведения и мощности в группе и $\oplus$

для исключительных или для битовых строк длины l. Для доказательства безопасности в стандартной модели мы представляем хеш-функцию схемы в виде двоичного оператора, принимающего ключ K и значение G_η и возвращающего битовую строку длиной l, тогда как в доказательстве в модели случайного оракула мы непосредственно кодируем хеш-функцию как процедуру, и никаких дополнительных расширений не требуется.

4.1 Стойкость в стандартной модели

Доказательство, которые мы предъявляем опирается на два предположения: предположение, семейство хеш-функций $(H_K)_{k\in K}$ сглаживает энтропию, и DDH. Последнее предположение уже было формализовано ранее. Первое утверждение формализовано ниже:
Определение 2 (предположение о сглаживании энтропии). *Рассмотрим игры* и

Игра ES_0 :
$k \xleftarrow{\$} K$; $h \xleftarrow{\$} \{0,1\}^l$;
$d \leftarrow \mathcal{D}(h)$

Игра ES_1 :
$k \xleftarrow{\$} K$; $z \xleftarrow{\$} \mathcal{Z}_q$;
$d \leftarrow \mathcal{D}(H(k, g^z))$

определим

$$\epsilon_{ES(\eta)} \stackrel{\text{def}}{=} |Pr_{ES_0}[d = 1] - Pr_{ES_1}[d = 1]|$$

Тогда для каждого PPT злоумышленника $\mathcal{D}$, $\epsilon_{ES(\eta)}$ незначительная функция. Чтобы не загромождать описание игр, мы немного изменили представление алгоритма генерации ключей: вместо того, чтобы возвращать хэш-ключ как компонент секретного и открытого ключа, мы моделируем его как глобальную переменную k. Это позволит нам в то же время хорошо проиллюстрировать использование глобальных переменных в **CertiCrypt**.

Теорема 1 (Стойкость хешированной схемы Эль-Гамаля в стандартной модели). Для каждого PPT злоумышленника $(\mathcal{A}, \mathcal{A}')$,

$$|Pr_{IND-CPA}[d] - \frac{1}{2}| \leq \epsilon_{DDH}(\eta) + \epsilon_{ES}(\eta)$$

Из этого следует, что при предположениях DDH и ES, $Pr_{IND-CPA}[d]$ незначительно стремится к $\frac{1}{2}$.

$$\begin{aligned}
|Pr_{IND-CPA}[b = b'] - \frac{1}{2}| &= |Pr_{G_4}[b = b'] - \frac{1}{2}| \\
&= |Pr_{G_4}[b = b'] - Pr_{G_6}[b = b']| \\
&= |Pr_{G_4}[b = b'] - Pr_{G_5}[b = b']| \\
&\leq Pr_{G_5}[\mathbf{bad}] \\
&\leq Pr_{G_6}[\,\in \mathrm{dom}(\boldsymbol{L})] \\
&= Pr_{LCDH}[g^{xy} \in L'] \\
&= \epsilon_{LCDH}(\eta)
\end{aligned}$$

Из приведенного выше уравнения и согласно допущению списка CDH (или, что эквивалентно, при простом предположении CDH), преимущество IND-CPA противника (A, A') результаты пренебрежимо близки к $\frac{1}{2}$. Чтобы убедиться в этом, достаточно проверить, что противник C выполняется за вероятностное полиномиальное время. Это так, потому что противник (A, A') выполняет, а C не выполняет дорогостоящих вычислений.

Глава 5

Похожие работы

Эль-Гамаль - это стандартный пример криптографического доказательства, основанного на игре, который обеспечивает эталонный тест, с которым можно сравнивать другие работы. Кратко прокомментируем три доказательства, тесно связанных с нашим. Для более общего описания связанных работ мы отсылаем к [**?**]. Самым последним и тесно связанным с этим является формализация в Coq основанного на игре доказательства семантической безопасности Эль-Гамаля, выполненная Новаком [**?**]. Хотя мы выбираем глубокое встраивание, Новак использует неглубокое встраивание и моделирует противников напрямую как функции Coq. Как следствие, получившаяся структура обеспечивает лишь ограниченную поддержку автоматизации проверки. По той же причине формализация Новака не может иметь дело со случайными оракулами, поэтому он представляет только доказательство Хешированного Эль-Гамаля в стандартной модели криптографии. Наконец, неясно, как формализовать сложность в контексте неглубокого внедрения, а формализация Новака полностью игнорирует сложность; в результате предположения безопасности, такие как DDH, не могут быть точно смоделированы. Более ранняя работа Барта, Седерквиста и Таренто [**?**] предоставляет основы формального доказательства безопасности подписанного шифрования Эль-Гамаля в Coq. В отличие от нашей работы, они рассматривают идеализированную модель криптографии, которая абстрагирует многие детали системы и определения безопасности. Таким образом, связь между формализацией и заявлением о безопасности не так сильна, как хотелось бы. Корин и ден Хартог [**?**] разработали (нереляционную) логику Хоара для рассуждений о вероятностных алгоритмах. Они использовали его для построения доказательства семантической безопасности шифрования Эль-Гамаля, но нам не известно о какой-либо другой системе, проверяемой с использованием этой логики. Поскольку они основаны на простом вероятностном расширении логики Хоара, их формализм недостаточно выразителен, чтобы моделировать понятие сложности PPT, и поэтому цели и гипотезы безопасности не могут быть точно выражены. В более общем смысле, логика сама по себе не предоставляет средств для рассуждений о контекстно-зависимых преобразованиях программ или преобразованиях, выполненных в оракулах.

5.1 Заключение

CertiCrypt - это полностью формализованный фреймворк, который помогает создавать криптографические игровые доказательства. Доказательства в CertiCrypt полагаются на минимальную надежную базу, и их правильность может быть автоматически проверена третьими сторонами. В этой статье мы проиллюстрировали некоторые ключевые аспекты CertiCrypt посредством формализации семантических доказательств безопасности схемы шифрования с открытым ключом хешированная схема Эль-Гамаля в стандартной и слу-

чайной модели оракула, и мы выделили некоторые существенные различия между нашими доказательствами и теми, которые появляются. в литературе. *Благодарности* Мы хотели бы поблагодарить Дэниела Хедина за его полезные комментарии к более раннему черновику этой работы.

Литература

[1] P. Audebaud and C. Paulin-Mohring. Proofs of randomized algorithms in coq. *Science of Computer Programming*, 2008.

[2] G. Barthe, J. Cederquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In *2nd International Joint Conference on Automated Reasoning*, pages 385–399. Springer-Verlag, 2004.

[3] G. Barthe, B. Gr´egoire, and S Zanella B´eguelin. Formal certification of code-based cryptographic proofs, 2009.

[4] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs in: Advances in cryptology – eurocrypt'06. *Lecture Notes in Computer Science*, 4004:409–426, 2006.

[5] N. Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31th ACM Symposium on Principles of Programming Languages*, pages 14–25. ACM Press, 2004.

[6] R. Corin and J. den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. in: Proceedings of the 33rd international colloquium on automata, languages and programming. *Lecture Notes in Computer Science*, 4052:252–263, 2006.

[7] The Coq development team. The coq proof assistant reference manual, 2008. Available at: http://coq.inria.fr.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci*, 28:270–299, 1984.

[9] S. Halevi. A plausible approach to computer-aided cryptographic proofs. *Cryptology ePrint Archive*, 2005/181, 2005.

[10] B. Jonsson, K.G. Larsen, and W. Yi. Probabilistic extensions of process algebras. *Handbook of Process Algebra*, pages 685–711, 2001.

[11] D. Nowak. A framework for game-based security proofs. *Information and Communications Security*, 4861:319–333, 2007.

[12] A. Sabelfeld and D. Sands. A per model of secure information flow in sequential programs. *Higher-Order and Symbolic Computation*, 14(1):29–91, 2001.

[13] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *Cryptology ePrint Archive*, 2004/332, 2004.

[14] J. Stern. Why provable security matters? *Lecture Notes in Computer Science*, 2656, 2003.

Глава 6

Информация о курсовой

6.1 Ссылка на репозиторий

`https://github.com/bulgvkov/latexProject/`

6.2 Компилятор

• LaTex

6.3 Поля и шрифты

- *Размеры полей:* top = 20 mm, bottom = 20 mm, left = 25 mm, right = 15 mm.
- *Размер шрифта:* 12pt.

6.4 Стилевые пакеты

1. russian - устанавливает кодировку шрифтов. Он является частью системы многоязыковой поддержки babel. Пакет не только устанавливает кодировку текстовых шрифтов T2A, но и включает правила переноса русских слов. Кроме того, пакет russian переопределяет команды с предопределённым текстом типа Contents или Figure, и вводит новые команды для печати ряда символов согласно российской традиции.
2. amsmath - набор предоставляет дополнительные математические символы, множество удобных возможностей для оформления математических формул.
3. amssymb -
4. multicol - для создания колонок текста.
5. color - позволяет задавать цвет текста и фона, как отдельного блока, так и всего документа.
6. geometry - настройки полей документа.
7. hyperref - гиперссылки внутри документа.
8. graphicx - для вставки изображения в документ.

6.5 Кастомная команда и сниппеты