**Федеральное государственное автономное образовательное учреждение**

**высшего образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

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

Направление подготовки/специальности

\_\_\_\_\_\_\_\_\_ «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Образовательная программа «Компьютерная безопасность»

**О Т Ч Е Т**

**о прохождении**

**учебной/производственной/научно-исследовательской/преддипломной практики**

*(нужное оставить)*

**Студент** Астраханцев Р.Г.  СКБ 171

(Фамилия И.О.) номер группы

**Руководитель практики студента:** Нестеренко Алексей Юрьевич, доцент МИЭМ \_ \_\_\_\_**\_\_\_\_**

(Фамилия И.О., должность и место работы, подпись)

**Руководитель практики от НИУ ВШЭ:** Лось Алексей Борисович ­\_\_\_\_\_\_\_\_\_

(Фамилия И.О., подпись)

**Практика пройдена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва, 2021**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc81321317)

[1 Определение пороговой схемы. 4](#_Toc81321318)

[2 Наивная схема 6](#_Toc81321319)

[3 Схемы с нулевым доверием 6](#_Toc81321320)

[3.1 Генерация ключа 7](#_Toc81321321)

[3.2 Генерация подписи 8](#_Toc81321322)

[ЗАКЛЮЧЕНИЕ 10](#_Toc81321323)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 11](#_Toc81321324)

[ПРИЛОЖЕНИЯ 12](#_Toc81321325)

[А. Схема Шамира 12](#_Toc81321326)

[Б. Схема электронной подписи Эль-Гамаля 13](#_Toc81321327)

[В. Умножение и инверсия секретов по их долям 13](#_Toc81321328)

# ВВЕДЕНИЕ

С появлением смарт-контрактов в блокчейн сетях появилась возможность принятия коллегиального решения разными сообществами. Наивной реализацией принятия такого решения было сочетание смарт-контрактов и мультиподписи. Процесс вынесения определённого решения заключался в создании смарт-контракта, который исполнялся тогда и только тогда, когда заданное количество собранных[[1]](#footnote-1) электронных подписей будет аккумулятивно подано на вход смарт-контракту (возможно за несколько транзакций).

Такой подход имеет ряд недостатков:

1. На каждое новое решение требуется создание нового *одноразового* смарт-контракта;
2. Проверка коллегиального решения требует проверки *каждой* поданной подписи;
3. Сбор электронных подписей требует взаимодействия с блокчейном, что в свою очередь может быть долго/ресурсозатратно.

Для решения этих проблем было предложено использование технологии пороговых подписей. Идея заключается в том, чтобы вместо нескольких уникальных подписей от разных участников вырабатывать коллективную подпись, которую затем можно будет проверить коллективным публичным ключом. Пороговой такая схема называется из-за того, что в самой схеме задаётся минимальное количество участников (*порог*), которое необходимо для выработки коллективной подписи произвольного сообщения.

Подобная схема избавлена от всех описанных выше недостатков, однако имеет сложности в своей реализации. Так, например, как вырабатывать секреты участников децентрализовано, как гарантировать, что именно минимальное количество участников сможет подписать сообщение. Эти и другие вопросы рассмотрены в этой работе.

Стоит отметить, что пороговые схемы находят своё применение не только в избавлении от генерации смарт-контрактов и увеличении производительности взаимодействия с блокчейном, но в любой задаче требующей выработки единого коллективного решения. К примеру, совет директоров той или иной компании может выработать единую подпись от лица самой компании без раскрытия информации о том, кто был за это решение, в то время как использование мультиподписи раскрывает эту информацию. Ещё одним применением этой технологии в современных реалиях будет возможность осуществления эффективного взаимодействия между блокчейнами или шардами одного блокчейна.

# Определение пороговой схемы.

В настоящее время существуют разные схемы создания электронной подписи (ЭП). Пороговая схема – это некоторое расширение ЭП. Определим пороговую схему полуформально как в [1], чтобы не привязываться к конкретной реализации ЭП.

*Определение 1.1*. (t, n)-пороговой схемой для заданной схемы ЭП (т.е алгоритмов генерации ключа, подписи сообщений и верификации сообщений), включающей n > 1 участников *P1, . . . ,Pn*, способных осуществлять следующие протоколы:

* *Протокол распределённой генерации ключа (РГК)*, в результате работы которого вырабатывается некоторый публичный ключ *pk* и секретные доли *ski* приватного ключа *sk*. Каждый участник *Pi* получает соответствующую долю *ski* и только её. Важно, что все участники знают публичный ключ *pk,* и ни один из участников не получает никакой дополнительной информации о *sk*.
* *Протокол распределённой подписи (РП)*, в результате работы которого все участники получают сообщение и действуют совместно для выработки действующей подписи, проверяемой с помощью *pk*. Важно отметить, что для получения валидной подписи достаточно t уникальных участников схемы.

Кроме того, могут быть реализованы следующие опциональные протоколы:

* *Протокол выработки распределённой предподписи*, который является вспомогательным для выработки *распределённой подписи*. В результате его выполнения которого каждый участник получает т.н. *личную* *предподпись*, которая не зависит от подписываемого сообщения. Выработанная *предподпись* может быть использована на конкретном сообщении один раз во время выполенения протокола *распределённой подписи (РП)*. Этот протокол является вспомогательным из-за того, изолирует взаимодействие между участниками не требующее подписываемое сообщение.
* *Протокол перераспределения секрета,* который может быть совершён после *протокола распределённой генерации ключа*. В результате работы этого протокола текущая (t, n)-пороговая схема становится (t*’*, n)-пороговой схемой (t*’* < n), а секретные доли *ski* перераспределяются в секретные доли *ski’*.

Стоит отметить, что данное выше определение также охватывает пороговые схемы, в которых протокол РГК происходит с помощью третьей стороны, которая после выработки валидной ключевой вары забывает секрет или в которых протокол РП подразумевает восстановление исходного приватного ключа *sk*. В дальнейшем мы рассмотрим как подобные «наивные» схемы распределённой подписи, так и схемы без доверия третьим лицам и отсутствием необходимости восстановления приватного ключа для создания валидной подписи.

Прежде чем говорить о реализации пороговых схем, хочется понять, а возможно ли реализовать такую схему с максимальной степенью защиты, то есть с отсутствием доверия третьим сторонам. Для этого стоит обратиться к работам Эндрю Яо Цичжи, который доказал [2], что возможно создать протокол конфиденциального вычисления (англ. secure multi-party computation) и показал его реализацию. Сформулируем исходную задачу и дадим определения этого протокола.

Пусть в конфиденциальном вычислении участвуют *N* участников *p1, p2, …, pN*. У каждого из участников есть секретные входные данные *d1, d2, …, dN*. Участники хотят найти *F(d1, d2, …, dN)*  без раскрытия секретных данных друг другу, где *F* – известная всем участникам вычислимая функция от *N* аргументов.

*Определение 1.2.* Алгоритм действий, выполняемый всеми *N* участниками для решения задачи выше, будем называть *протоколом конфиденциального вычисления*.

Также допускается, что среди участников будут *получестные нарушители*, то есть те, которые верно следуют протоколу, но пытаются получить дополнительную информацию из любых промежуточных данных.

В своей работе Яо предложил идею построения искажённой схемы (англ. garbled circuit), который подразумевает, что выходная функция *F* представляется в виде логической схемы, а секретные данные каждого участника представляются как входные данные функции *F* в двоичном виде. Использование искажённой схемы подразумевает что участники шифруют исходную логическую схему функции *F*, и оперируют полученной искажённой схемой, подставляя в неё данные в зашифрованном виде.

Нетрудно понять, реализация пороговых схем с нулевым доверием опирается на уже решённую задачу конфиденциального вычисления. Таким образом, представленные в работе пороговые схемы с нулевым доверием являются более эффективными версиями решения этой задачи с точки зрения скорости вычисления.

# Наивная схема

Для начала попробуем создать простую пороговую *(t,n)*-схему, которая позволяет *t* участникам сгенерировать валидную подпись к сообщению. Пусть имеется схема ЭП и пусть имеется *(t, n)*-пороговая схема разделения секрета (например схема разделения секрета Шамира). Это означает, что любые *t* участников могут восстановить некоторый общий секрет.

Для *распределённой генерации ключа* пригласим *дилера*, то есть некоторое третье лицо, которое будет выполнять алгоритм создания валидной ключевой пары. После генерации открытого и закрытого ключа дилер должен разделить секрет в соответствии с правилами *(t, n)*-пороговой схемы разделения секрета. Раздав полученные доли секрета участникам, дилер должен забыть ранее сгенерированный закрытый ключ.

Для *выработки валидной подписи* *t* участников должны будут восстановить закрытый ключ, дальше любой из этих участников должен будет сгенерировать валидную подпись к сообщению и забыть закрытый ключ.

Отметим недостатки этой схемы:

1. для генерации ключевой пары привлекается третья сторона (*дилер*);
2. закрытый ключ восстанавливается входе вычисления подписи;
3. во время генерации подписи один или несколько участников имеют полный доступ к закрытому ключу;

Несмотря на наивность этой схемы можно заметить, что пороговые схемы опираются на пороговые схемы разделения секрета. Существуют разные схемы разделения секрета, однако в дальнейшем мы будем использовать схему разделения секрета Шамира. В некоторых источниках используется схема VSS (verifiable secret sharing) или проверяемая схема разделения секрета, которая является улученной версией разделения секрета Шамира, и позволяет проверить, является ли та или иная доля секрета валидной (то есть позволяет бороться со злоумышленниками, которые пытаются выдать себя за владельца части подписи). Принцип схемы Шамира описан в приложении А.

# Схемы с нулевым доверием

Для более детального понимания принципов работы схем с нулевым доверием начнём рассматривать схему Эль-Гамаля в качестве схемы электронной подписи. Её описание может быть найдено в приложении Б. Основная проблема в преобразовании DSA в пороговый вариант – это вычисление значений *R* и *s* (смотри приложение Б), поскольку они требуют умножения и модульных обратных операций [3].

## Генерация ключа

Прежде чем переходить к построению алгоритма распределённой генерации ключа стоит отметить аддитивную особенность схемы разделения секрета Шамира. Это означает, что имея доли *(x1, y1), …, (xn, yn)* и *(x1, z1), …, (xn, zn)* соответствующие секретам *y* и *z*, доли *(x1, y1+z1), …, (xn, yn+zn)* будут соответствовать секрету *y* + *z*. С учётом аддитивного свойства открытого и закрытого ключа схемы Эль-Гамаля, данный факт легко описывает схему генерации ключа.

Опишем алгоритм с позиции *i*-го участника:

1) вычислить валидную пару *ski* и *pki* – закрытый и открытый ключи схемы Эль-Гамаля (стоит пояснить, что общий закрытый ключ схемы будет иметь вид *sk=sk1+…+skn*)

2) раскрыть публичный ключ *pki* остальным участникам и дождаться публичных ключей остальных участников

3) разделить секретный ключ *ski* на доли *ski1, … skin* по *(t, n)*-схеме разделения Шамира (стоит отметить, что обозначение *skim* означает долю *(xm, y­im)*, то есть участники заранее договариваются о точках, в которых будут вычисляться значения многочленов в ходе вычисления долей секрета Шамира)

3) раздать долю *skim* участнику под номером *m* и поделать этот пункт для каждого m=1…n, после чего дождаться долей от остальных участников (к концу этого шага у *i*-го участника будут доли *sk1i,…, skni*)

4) сформировать свою долю секрета *ski*=*sk1i+…+skni* (то есть итоговая доля секрета будет иметь вид *(xi, y1i+…+yni)*)

5) сформировать общий публичный как *pk=pk1+…+pkn*

Из-за описанных выше свойств аддитивности по окончании работы алгоритма произвольные *t* участников смогут совместно восстановить общий закрытый ключ *sk* (соответствующий общему публичному ключу *pk*) и сделать валидную подпись.

Стоит отметить, что в описанном выше алгоритме закрытый ключ *sk* не раскрывается ни на одном этапе. Таким образом, для построения пороговой схемы с нулевым доверием необходимо описать алгоритм конфиденциального вычисления подписи без раскрытия долей секрета выше.

## Генерация подписи

Как уже отмечалось ранее основная трудность в создании пороговой схемы из схемы Эль-Гамаля заключается умножении и модульных обратных операций, поскольку они не тривиальны для схемы разделения секрета Шамира. Чтобы преодолеть эту проблему, Дональд Бивер продемонстрировал [4] [5] протоколы умножения и обратной инверсии, которые опираются на мультипликативные тройки Бивера (англ. Beaver multiplicative triples). Это такие мультипликативные тройки чисел *(a, b, c)*, что *c=a\*b*, что они были получены с позиции участника без вычисления непосредственных значений *a, b, c*. Другими словами каждый участник генерирует у себя случайные доли *s­a, sb, sc* секретов a, b, c такие, что *c=a\*b* и сами секреты *a, b, c* не раскрываются ни на одном этапе. Варианты решение задачи генерации троек Бивера можно найти в [6]

Помимо описанных выше протоколов существуют другие, использующие свойства гомоморфной криптографии. Так, например, Дженнаро и Гольдфедер предложили [7] протокол, который преобразует мультипликативные доли секрета в его аддитивные доли.

Подробнее протоколы Бивера описаны в приложении В, дальнейшее их описание будет происходить в виде вызовов функций *Mul(x, y)* и *Invert(z)*. С точки зрения одного участника фраза «вычислить *m=Mul(x, y)*» будет обозначать, что у участника есть доли *x* и *y* секретов *X* и *Y* соответственно и этот участник (совместно с другими участниками) инициирует протокол Бивера для вычисления произведения *M* секретов *X* и *Y* (*M=X\*Y*) после чего получает долю m секрета *M*. Аналогично с функцией *Invert*. Важной особенностью протоколов Бивера является тот факт, что протоколы *Mul* и *Invert* не требуют в своей реализации раскрытия секретов, используемых в вычислениях.

Продемонстрируем алгоритм пороговой подписи, описав алгоритм с позиции *i*-го участника. Будем считать у него имеется доля *ski* секрета *sk*. Пусть задана циклическая группа с порядком *q* (простое число)и генератором группы . Пусть так же задана хеш-функции *H: {0,1}\* ⟶ Zq* и *H’: G ⟶ Zq*

Алгоритм пороговой подписи:

*Входные данные* – произвольное сообщение *M* (представленное в виде бит).

*Выходные данные* – подпись входного сообщения.

1. Совместно с другими участниками вычислить долю *ki*, случайного секрета *K*. Важно, чтобы участники сгенерировали секрет *K* не раскрывая его значения (без дилера).Это может быть сделано аналогично протоколу распределённой генерации ключа (пункт *3.1*).
2. Аналогично пункту 1 вычислить долю *wi* секрета *W*.
3. Вычислить долю *d­i­ = Mul(wi, ki)* секрета *D=W\*K*
4. Вычислить долю *u­i­ = Mul(wi, ski)* секрета *U=W\*sk*
5. Совместно с другими участниками восстановить значение D по принципу *(t,n)*-пороговой схемы
6. Вычислить , отправить *R­i* другим участникам и дождаться сообщений других.
7. Совместно с другими участниками восстановить значение

Это может быть сделано по тому же принципу восстановления секрета, как и в исходной *(t,n)*-пороговой схеме. На примере разделения секрета Шамира можно заметить, что вид полинома не меняется при переходе к элементам циклической группы. Соответственно точки *Ri* могут быть так же использованы в полиноме для восстановления секрета *R*.

1. Вычислить *r = H’(R)*.
2. Вычислить *s­i = H(M) \* k­i + r \* u­i* – долю подписи *S*.
3. Совместно с другими участниками восстановить значение *S* по принципу *(t,n)*-пороговой схемы
4. Вернуть пару *(r, S)*

Представленный алгоритм можно разделить на две части. В пунктах 1-7 никак не используется подписываемое сообщение. Соответственно можно вынести эти пункты в отдельный протокол *выработки предподписи*.

В итоге можно видеть, что была получена подпись сообщения без раскрытия закрытого ключа ни одной стороной. Проверка же полученной подписи происходит по обычным правилам проверки схемы Эль-Гамаля.

# ЗАКЛЮЧЕНИЕ

Рассмотренные вопросы построения пороговых схем на примере схемы Эль-Гамаля показали, что возможно достичь эффективного вычисления пороговой подписи с нулевым доверием.

Стоит сказать, что также возможно построить пороговые схемы и для других криптографических схем. Так, например в работе [8] представлена RSA-пороговая схема без доверенного дилера.

На данный момент самой главной проблемой пороговых схем является эффективное вычисление произведения двух секретных значений на основе их долей без раскрытия самих секретов. Именно поэтому большей популярностью пользуются схемы с наименьшим числом мультипликативных операций, как например схема Эль-Гамаля.

Аддитивное свойство алгоритмов разделения секретов, которые используются внутри протоколов выработки пороговой подписи, являются ещё одной причинной эффективного вычисления некоторых значений входе выработки подписи по схеме Эль-Гамаля. Вычисления получаются не только интуитивно понятными, но и не требуют дополнительных манипуляций.

С практической точки зрения пороговые схемы позволяют выразить общее мнение некоторой группы лиц по тому или иному вопросу без непосредственного раскрытия лиц того большинства, которое проголосовало за это мнение. Наблюдая за стремлением мира криптовалют к межблокченовому взаимодействию, можно сказать, что всё больше систем будут использовать пороговые подписи в своей работе.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

[1] Jean-Philippe Aumasson, Adrian Hamelink, Omer Shlomovits: *A Survey of ECDSA Threshold Signing*. IACR Cryptol. ePrint Arch. 2020: 1390 (2020)

[2] A. Yao. How to generate and exchange secrets. In 27th FOCS, 162—167, 1986.

[3] G Tillem, O Burundukov*.*: *Threshold Signatures using Secure Multiparty Computation*

[4] Donald Beaver: *Efficient multiparty protocols using circuit randomization*. In CRYPTO, 1991.

[5] Judit Bar-Ilan and Donald Beaver: *Non-cryptographic faulttolerant computing in constant number of rounds of interaction*. In ACM Symposium on Principles of Distributed Computing. Association for Computing Machinery, 1989.

[6] Springer, Heidelberg (1999) Pullonen, P.: *Actively secure two-party computation: efficient beaver triple generation*

[7] Rosario Gennaro and Steven Goldfeder: *Fast multiparty threshold ECDSA with fast trustless setup*. IACR Cryptol. ePrint Arch., 2019:114, 2019.

[8] Damgård, I. B., & Koprowski, M. (2000): *Practical Threshold RSA Signatures Without a Trusted Dealer*. BRICS Report Series, *7*(30).

# ПРИЛОЖЕНИЯ

## А. Схема Шамира

Опишем принцип разделения секрета Шамира. Он основан на том факте, что для интерполяции многочлена степени *k-1* требуется *k* точек. Каждую точку будем принимать за часть доли секрета. Таким образом никакие *k-1* точек и меньше не смогут восстановить многочлен степени *k*, а следовательно, не смогут восстановить секрет. Более формально ниже.

Пусть нужно разделить секрет *M* между *n* сторонами таким образом, чтобы любые *k* участников могли бы восстановить секрет. Будем описывать алгоритм разделения секрета со стороны дилера.

Выберем некоторое простое число *p>M*, которое будет задавать размер конечного поля. Его можно открыто сообщать всем сторонам. Над этим полем построим многочлен степени *k-1* вида

*F(x) = M + a1 x + a2 x2 + ... + ak-1 xk-1* mod *p*,

где *a1, a1, ..., ak-1* – случайно выбранные коэффициенты, *M* – разделяемый секрет. Дилер не должен разглашать коэффициенты многочлена. Особенность этого многочлена в том, что его значение в нуле равно заданному секрету.

Для генерации долей секрета дилер должен вычислить значения построенного многочлена в *n* различных точках отличных от нуля. Он может это сделать, например, следующим образом.

*x1=1, y1 = F(1)*

*x2=2, y2 = F(2)*

*...*

*xn=n, yn = F(n)*

После чего дилер должен раздать секреты участникам, то есть *i*-ый участник получит пару *(x­­i, yi)* как свою долю секрета. Коэффициенты многочлена и сам секрет дилером забываются.

Для восстановления секрета *k*-участников должны обменяться своими секретами и вычислить интерполяционный многочлен Лагранжа и его значение в нуле. Построение этого много члена выглядит как:

Особенностью схемы является то, что вероятность раскрытия секрета в случае известных *k-1* долей оценивается как *p­­-1*. То есть в результате интерполяции по *k-1* точке секретом может быть любой элемент поля с равной вероятностью. При этом попытка полного перебора всех возможных теней не позволит злоумышленникам получить дополнительную информацию о секрете.

## Б. Схема электронной подписи Эль-Гамаля

Пусть задана циклическая группа *G* с порядком *q* (простое число)и генератором группы *g*. Пусть так же задана хеш-функции *H: {0,1}\* ⟶ Zq* и *H’: G ⟶ Zq*.

* *Закрытым ключом* является произвольное выбранное значение *x ∈ Zq*
* *Открытым ключом* будет *y=gx ∈ G*

Алгоритм подписи:

*Входные данные* – произвольное сообщение *M* (представленное в виде бит).

*Выходные данные* – подпись входного сообщения.

1. Вычислить *m=H(M)∈ Zq*
2. Выбрать случайное *k* *∈ Zq*
3. Вычислить *R=g1/k ∈ G* и *r=H’(R) ∈ Zq*
4. Вычислить *s=k(m + x r)* mod *q*
5. Вернуть пару *(r, s)*

Алгоритм верификации:

*Входные данные* – проверяемое сообщение *M*, подпись *(r, s)*, и открытый ключ *y*

*Выходные данные* – результат проверки.

1. Вычислить *R’= y r/s g m/s ∈ G*
2. Если *H’(R’)=r*, то вернуть, что подпись верна, иначе отклонить её.

## В. Умножение и инверсия секретов по их долям

Пусть у нас есть доли *x1, …, xn* секрета *X* и *y1, …, yn* секрета *Y*, полученные в результате работы некоторой аддитивной *(t,n)*-пороговой схемы разделения секрета. Перед участниками стоит задача получения долей секретов *Z=X\*Y* и *V=X-1* без раскрытия изначальных секретов *X* и *Y*. Описанные ниже алгоритмы будут проводиться с позиции i-го участника, который владеет долями *xi* и *yi*.

Алгоритм умножения *(Mul)*:

*Входные данные* – доли *xi* и *yi* секретов *X* и *Y*.

*Выходные данные* – доля *zi* секрета *Z=X\*Y*.

1. Совместно с другими участниками сгенерировать доли *ai, bi, ci* соответствующие тройке Бивера *A, B, C (C=A\*B)*
2. Посчитать *di = x­i + ai*, *ei = yi + bi*.

По свойству аддитивности полученные значения будут долями секретов *D=X+A* и *E=Y+B* соответственно.

1. Совместно с другими участниками восстановить значения *D* и *E* по принципу *(t,n)*-пороговой схемы.
2. Посчитать *zi = c­i + E\*xi + D\*yi – E\*D*.

По свойству аддитивности полученное значение будет долей секрета *C + E X + D Y – E D = A B + (Y+B) X + (X+A) Y – (Y+B) (X+A) =*

*= A B + X Y + X B + X Y + Y A – X Y – X B – Y A – A B = X Y = Z.*

1. Вернуть *zi*.

Стоит отметить, что на 2-м шаге происходит реконструирование значений *D* и *E*, что может показаться небезопасным. Однако поскольку ни один из секретов *X, Y, A* и *B* не был реконструирован, то и раскрытие *D* и *E* не даёт никакой информации об исходных секретах.

Алгоритм инверсии *(Invert)*:

*Входные данные* – доля *xi* секрета *X*.

*Выходные данные* – доля *vi* секрета *V=X-1*.

1. Совместно с другими участниками вычислить долю *ai*, случайного секрета *A*. Важно, чтобы участники сгенерировали секрет *A* не раскрывая его значения (без дилера).Это может быть сделано аналогично протоколу распределённой генерации ключа (пункт *3.1*).
2. Вычислить долю *w­i = Mul(a­i, xi)*, секрета *W=A\*X*.
3. Совместно с другими участниками восстановить значение *W* по принципу *(t,n)*-пороговой схемы
4. Вычислить *vi = W-1 a­i*

По свойству аддитивности полученное значение будет долей секрета

*W-1 A = (A X)-1A = A-1 X-1 A = X-1 = V*

1. Вернуть *vi*.

1. *собранной* подпись считается тогда, когда член коллегии подпишет своей личной подписью определённое сообщение (например «Согласен с решением») и подаст полученный результат на вход смарт-контракту. Кроме того, смарт-контракт может реализовать механизм отзыва согласия, работающий по аналогичному признаку. [↑](#footnote-ref-1)