

ЭКЗОТИКА, ЭЗОТЕРИКА, ПЕРСПЕКТИВНЫЕ НАПРАВЛЕНИЯ

Дисциплина: Криптографические методы защиты информации

Преподаватель: Миронов Константин Валерьевич

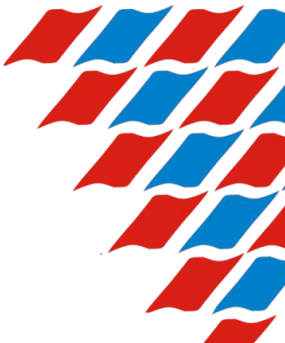
Поток: БПС-3, ИКТ-5

Учебный год: 2020/21



Содержание лекции

- **Подписи на основе многократного хеширования**
 - **Подпись Лэмпорта**
 - **Подпись Винтерница**
 - **Дерево Меркла**
- Распределенные реестры
- Прочее

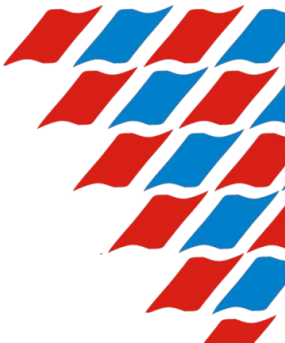


Одноразовые подписи

One-Time Signature, OTS

Отличия от обычных ЭП:

- Математический аппарат асимметричной криптографии не используется
- Используется хеш-функция, при этом, криптостойкость подписи обеспечивается исключительно криптостойкостью используемой хеш-функции
- Каждую пару ОК-ЗК можно использовать один раз
- Длины ключей больше, вычислительные затраты меньше
- Считаются устойчивыми к гипотетическим квантовым атакам



Подпись Лэмпорта

Вариант на основе 256-битной хеш-функции

ЗК: 512 случайных блоков по 256 бит, объединенные в 256 пар

$$\text{ЗК} = \begin{matrix} \text{ЗК}_{1,1} \text{ЗК}_{1,2} \dots \text{ЗК}_{1,256} \\ \text{ЗК}_{2,1} \text{ЗК}_{2,2} \dots \text{ЗК}_{2,256} \end{matrix}$$

- Если блоки генерируются на ГПСП, в памяти можно хранить только IV и ключ генератора

ОК: 512 256-битных хешей каждого кода ОК[1,1]...ОК[1,256]ОК[2,1]...ОК[2,256]

$$\text{ОК} = \begin{matrix} \text{ОК}_{1,1} \text{ОК}_{1,2} \dots \text{ОК}_{1,256} \\ \text{ОК}_{2,1} \text{ОК}_{2,2} \dots \text{ОК}_{2,256} \end{matrix}$$

$$\text{ОК}_{i,j} = \text{hash}(\text{ЗК}_{i,j})$$

- Можно публиковать не сам ОК, а только хеш-функцию от него; в таком случае вместе с подписью к документу нужно будет прикладывать ОК

ЭП: 256 кодов по 256 бит

$$\text{ЭП} = \text{ЭП}_1 \dots \text{ЭП}_{256}$$



Подпись Лэмпорта

Вариант на основе 256-битной хеш-функции

Постановка подписи:

шаг 1. Вычисляется $h = hash(OT)$

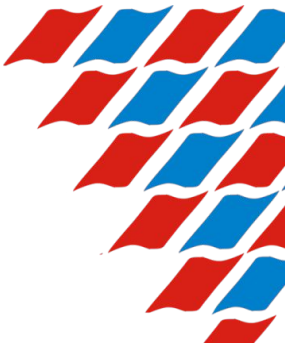
шаг 2. Для каждого бита h

если i -й бит = 0 то $ЭП_i = ЗК_{1,i}$

иначе $ЭП_i = ЗК_{2,i}$

Проверка подписи:

Вычислить хеши от ЭП и сравнить с соответствующими значениями из ОК



Подпись Винтерница

Вариант на основе 256-битной хеш-функции

Требует меньше памяти, но больше операций хеширования

- 256-битный $h = hash(OT)$ разбивается на блоки $h_1 h_2 \dots h_m$ по w бит (w – степень двойки); каждый блок трактуется как целое число
- ЗК: m псевдослучайных 256-битных случайных кодов

$$ЗК = ЗК_1 ЗК_2 \dots ЗК_m$$

- ОК: m 256-битных блоков

$$ОК = ОК_1 ОК_2 \dots ОК_m$$

каждый j -й блок ОК вычисляется на основе соответствующего блока ЗК по алгоритму:

$$a = ЗК_j$$

для i от 1 до $2^w - 1$

$$a = hash(a)$$

$$ОК_j = a$$



Подпись Винтерница

Вариант на основе 256-битной хеш-функции

ЭП: m 256-битных блоков

$$\text{ЭП} = \text{ЭП}_1 \text{ЭП}_2 \dots \text{ЭП}_m$$

Процедура постановки:

$$h_1 h_2 \dots h_m = \text{hash}(\text{OT})$$

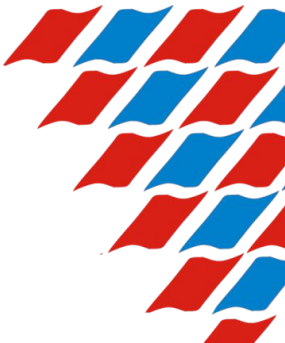
для j от 1 до m

$$a = \text{ЗК}_j$$

для i от 1 до h_j

$$a = \text{hash}(a)$$

$$\text{ЭП}_j = a$$



Подпись Винтерница

Вариант на основе 256-битной хеш-функции

Процедура проверки:

$$h_1 h_2 \dots h_m = \text{hash}(\text{OT})$$

для j от 1 до m

$$a = \text{ЭП}_j$$

для i от 1 до $(w - h_j - 1)$ выполнить $a = \text{hash}(a)$

если $a \neq \text{OK}_j$ то прервать проверку (ПОДПИСЬ НЕВЕРНА)

конец цикла

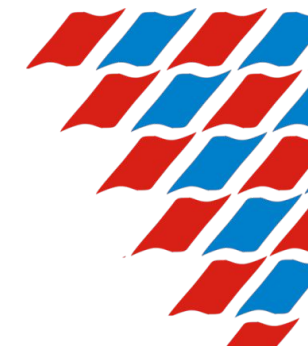
завершить проверку (ПОДПИСЬ ВЕРНА)



Одноразовые подписи

Уменьшение длины ключей

- Если ЗК является ПСП, то в памяти можно хранить только синхропосылку генератора
- Вместо ОК в «цифровой телефонной книге» можно публиковать его хеш-функцию
 - Тогда к самой ЭП надо прикладывать значение ОК
 - т.е. объем телефонной книги уменьшается за счет увеличения объема каждой конкретной подписи



Одноразовые подписи

Постановка подписи при уменьшенной длине ключей

$\backslash \backslash$ $ZK_{\text{внут}}$, $OK_{\text{внут}}$ – ключи заданные исходным алгоритмом *OTS*

$\backslash \backslash$ $ZK_{\text{внеш}}$, $OK_{\text{внеш}}$ – ZK , хранящийся в памяти шифратора, и OK , публикуемый в телефонной книге

$ZK_{\text{внут}} = PRF(ZK_{\text{внеш}}) \backslash \backslash PRF()$ – функция вычисления ПСП на основе синхропосылки

$ЭП_{\text{внут}} = SIGN(OT, ZK_{\text{внут}}) \backslash \backslash SIGN()$ – функция постановки ЭП исходным алгоритмом *OTS*

$OK_{\text{внут}} = getOK(ZK_{\text{внут}}) \backslash \backslash getOK()$ – функция генерации OK в исходном алгоритме *OTS*

$ЭП_{\text{внеш}} = \{ЭП_{\text{внут}}, OK_{\text{внут}}\}$

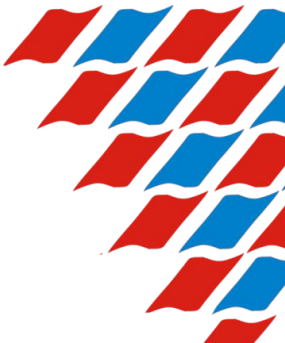
Проверка подписи при уменьшенной длине ключей

ЕСЛИ $CHECK(ЭП_{\text{внут}}, OK_{\text{внут}}) = FALSE$ ТО ПОДПИСЬ_НЕВЕРНА

$\backslash \backslash CHECK()$ – функция проверки ЭП исходным алгоритмом *OTS*

ИНАЧЕ ЕСЛИ $HASH(OK_{\text{внут}}) \neq OK_{\text{внеш}}$ ТО ПОДПИСЬ_НЕВЕРНА

ИНАЧЕ ПОДПИСЬ_ВЕРНА



Дерево Меркла

- Режим использования алгоритма OTS, позволяющий публиковать один короткий ОК и затем использовать его многократно
- С помощью ГПСП генерируется $m = 2^n$ закрытых ключей исходного алгоритма OTS $ЗК_1 \dots ЗК_m$
 - Желательно, чтобы ГПСП позволял вычислять отдельные ЗК без знания остальных (например, можно использовать шифр в режиме CTR)
- На их основе вычисляется m открытых ключей $ОК_1 \dots ОК_m$ и **хеши нижнего уровня** $h_{0,1} \dots h_{0,m}$

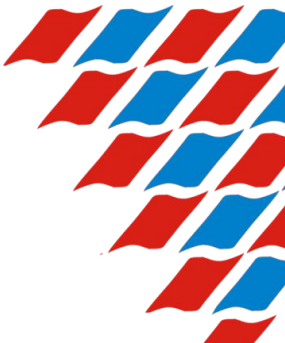
$$h_{0,j} = \text{hash}(ОК_i)$$

- Хеши нижнего уровня объединяются попарно, и от каждой пары вычисляется хеш следующего уровня; затем хеши каждого нового уровня объединяются и хешируются аналогичным образом:

$$h_{i,j} = \text{hash}(h_{i-1,2j-1} || h_{i-1,2j})$$

- На верхнем уровне остается только один хеш, он и публикуется в качестве ОК

$$ОК_{\text{внеш}} = h_{n,1}$$



Дерево Меркла

Процедура постановки подписи

$\backslash \backslash alt_1 \dots alt_{n-1}$ – массив альтернативных хешей, которые понадобятся для проверки подписи

$\backslash \backslash i$ – хранимый в памяти шифратора порядковый номер подключа

$\backslash \backslash \{c, d\} = IntDiv(a, b)$ – функция целочисленного деления a на b , возвращаеь частное c и остаток d

$ЭП_{внут} = SIGN(OK, ЗК_i) \backslash \backslash SIGN()$ – функция постановки ЭП исходным алгоритмом *OTS*

$k = i$

для j от 1 до n

$\{k, rem\} = IntDiv(k, 2)$

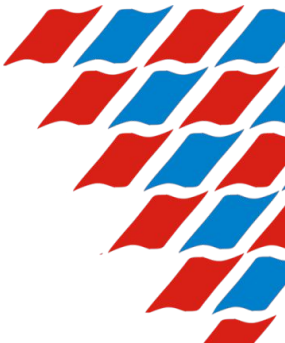
если $rem = 0$ то

$alt_j = h_{j, k-1}$

иначе

$alt_j = h_{j, k+1}$

$ЭП_{внеш} = \{ЭП_{внут}, OK_i, i, alt_1 \dots alt_{n-1}\}$



Дерево Меркла

Процедура проверки подписи

если $CHECK(\text{ЭП}_{\text{внут}}, \text{ОК}_i) = FALSE$ то ПОДПИСЬ_НЕВЕРНА

$\setminus \setminus CHECK()$ – функция проверки ЭП исходным алгоритмом *OTS*

иначе

$$a = HASH(\text{ОК}_i)$$

$$k = i$$

для j от 1 до $n - 1$ выполнить

$$\{k, rem\} = IntDiv(k, 2)$$

если $rem = 0$ то

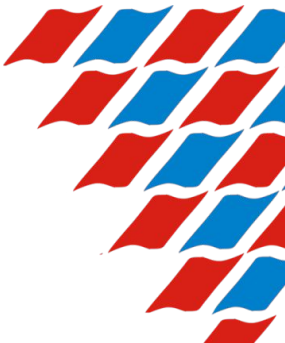
$$a = HASH(\text{ОК}_i || alt_j)$$

иначе

$$a = HASH(alt_j || \text{ОК}_i)$$

если $a \neq \text{ОК}_{\text{внеш}}$ то ПОДПИСЬ_НЕВЕРНА

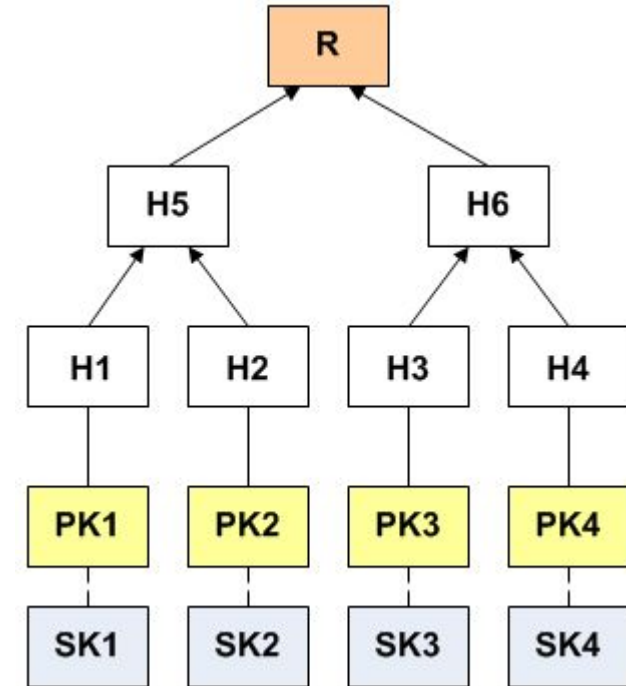
иначе ПОДПИСЬ_ВЕРНА



Дерево Меркла

Особенности

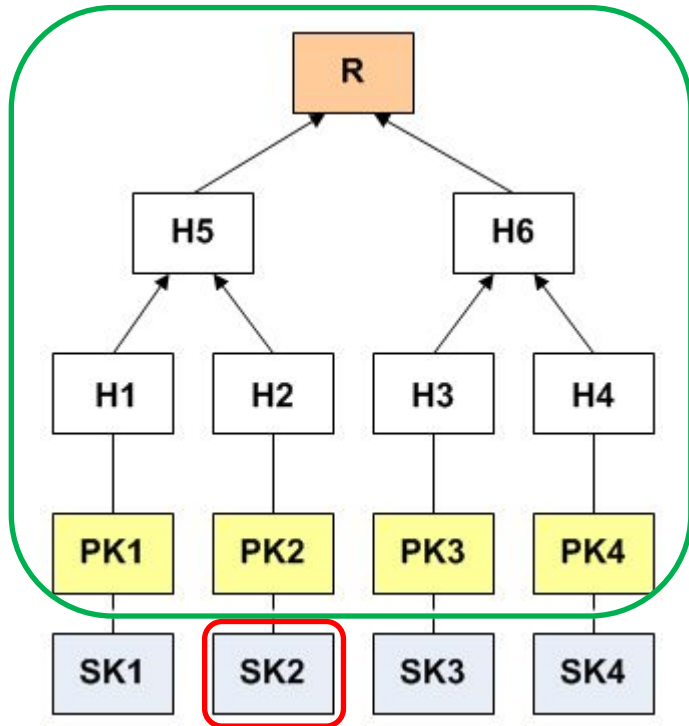
- На практике $n \leq 24$
- Одним ключом можно подписать m документов
 - m -ю подпись можно поставить на ключ нового дерева
- В памяти шифратора хранятся следующие данные
 - Синхросылка генератора закрытых ключей (т.е. общий ЗК)
 - Порядковый номер текущего подключа
 - Дерево хешей, либо его верхняя часть



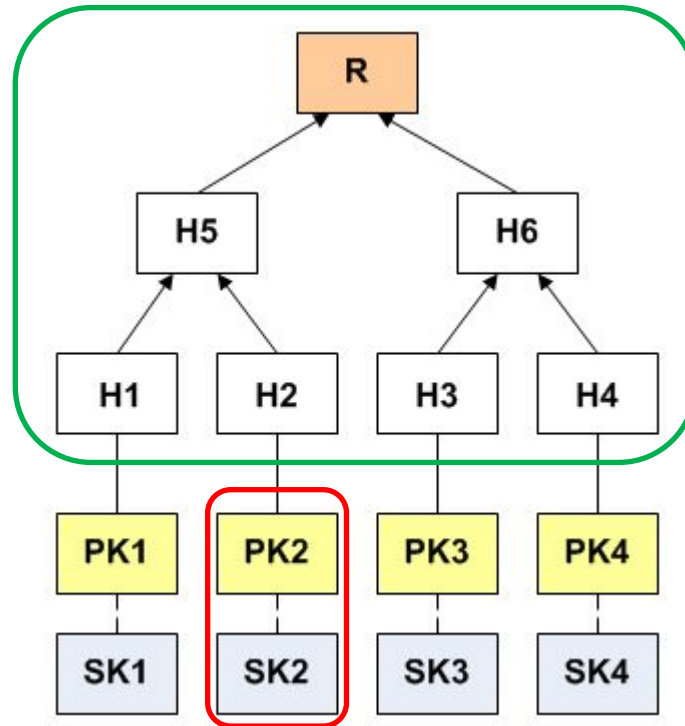
Дерево Меркла

Особенности

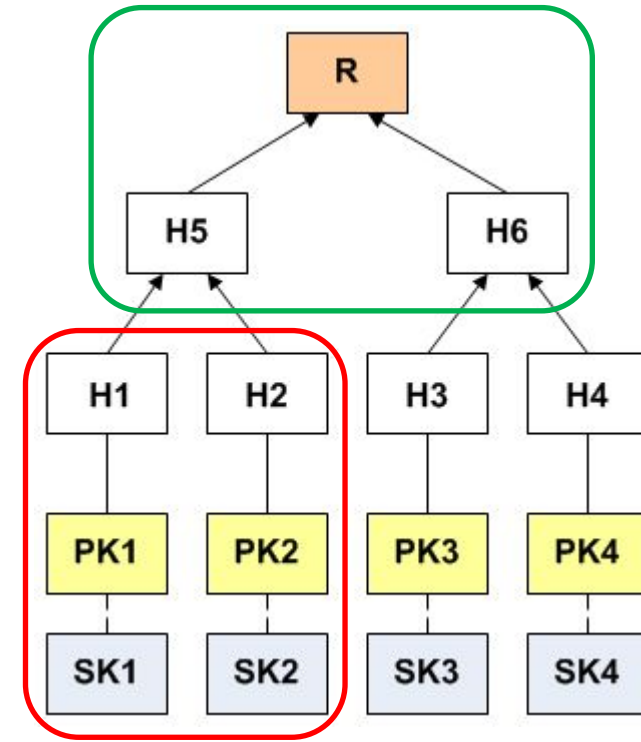
*Много памяти, мало
вычислений*



*Немного памяти, немного
вычислений*

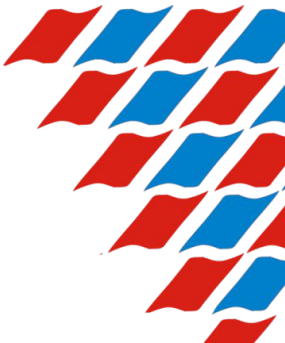


*Мало памяти, много
вычислений*



Содержание лекции

- Подписи на основе многократного хеширования
- **Распределенные реестры**
- Прочее



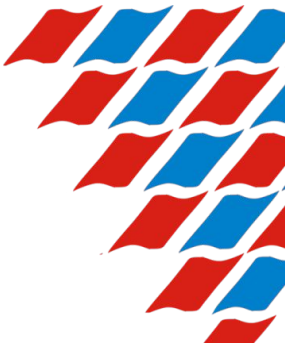
Распределенный реестр

Структура данных, характеризующаяся особенностями:

- Хранимые данные представляют собой **множество записей**, каждая из которых содержит **электронную подпись** ее автора
- Записи могут быть объединены в **блоки**
- Каждая запись (блок) содержит **криптографическую хеш-функцию** одной или нескольких предыдущих записей (блоков)
- Копии структуры хранятся на **множестве узлов сети**

Следствия:

- Добавляя запись в реестр, автор косвенным образом подписывает множество уже содержащихся в нем записей
- Хранение данных в распределенном реестре обеспечивает:
 - криптографическую защиту от подмены данных, в том числе самим автором
 - фиксацию времени их опубликования
- Распределенное хранение защищает от попытки подменить весь реестр целиком
 - Злоумышленник должен контролировать >50% узлов сети



Распределенный реестр

Поколения распределенных реестров

1-е поколение (DLT 1.0)

- Bitcoin, 2008-2009
- «Классические» криптовалюты
- Реестр используется для хранения данных, а практическое применение этих данных осуществляется другими приложениями

2-е поколение (DLT 2.0)

- Ethereum, 2013
- **Смарт-контракт** – алгоритм, задающий взаимодействие пользователей реестра в зависимости от того, какая информация записывается в реестр
- Реестр становится распределенным приложением для выполнения программ

3-е поколение (DLT 3.0)

???

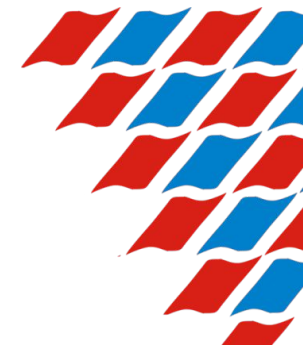
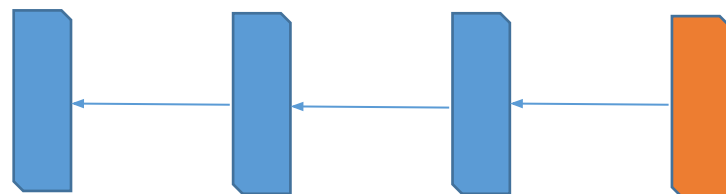


Виды распределенных реестров

Цепной реестр

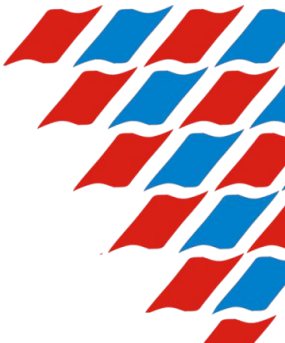
Blockchain

- Все блоки упорядочены хронологически
- Каждая следующая запись содержит хеш-функцию предыдущей
- Проблема: временной промежуток между добавлениями новых записей должен быть больше, чем время синхронизации копий на разных узлах
 - Записи объединяют в блоки
 - Применяются механизмы искусственного замедления генерации блоков
 - Proof of Authority – записывать блоки в реестр могут только определенные узлы
 - Proof of Work – вычислительная сложность добавления записей искусственно завышается, чтобы делать это могли только наиболее мощные узлы (майнеры)



Содержание лекции

- Подписи на основе многократного хеширования
- Распределенные реестры
- **Прочее**
 - **Низкоресурсная криптография**
 - **Квантовые технологии**
 - **Гомоморфные шифры**
 - **Доказательство с нулевым разглашением**
 - **Протоколы тайного голосования**



Содержание лекции

- Распределенные реестры
- Подписи на основе многократного хеширования
- **Прочее**
 - **Квантовые технологии**
 - **Гомоморфные шифры**
 - **Доказательство с нулевым разглашением**
 - **Протоколы тайного голосования**

