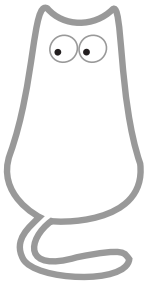
DNSSEC



РУКОВОДСТВО РЕГИСТРАТОРА ДОМЕНОВ

Что надо знать. Что требуется. Как сделать.

ВЕРСИЯ 1

Филипп Кулин, IV квартал 2016, UNLICENSE

Сложный DNSSEC

Технология DNSSEC сложная

Регистратору доменов не требуется поддерживать технологию полностью

Реализация необходимого минимума очень простая

DNSSEC в двух словах

Подпись записей зоны

Записи зоны подписаны с помощью системы электронной подписи.

Цепочка доверия

Родительская зона подтверждает достоверность ключа, которым подписана зона потомка.

От регистратора доменов требуется определить полномочия оператора домена на публикацию ключей и передать информацию о ключах этого домена в родительскую зону.

Оператор домена

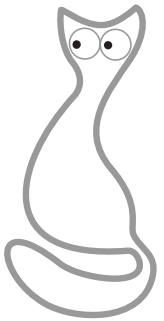
При выполнении действий связанных с делегированием домена исторически сложилась модель Администратор — Регистратор — Реестр. То, что обслуживание домена Администратор часто поручает сторонним исполнителям, не было большой проблемой, поскольку действия ограничивались редкими изменениями записей NS при делегировании.

Поддержка DNSSEC требует регулярных изменений в делегировании домена. В текущей модели это приводит к проблемам, поскольку требуется регулярное вовлечение в процесс технического обслуживания Администратора домена.

«Оператор домена» занимается техническим обслуживанием домена по поручению Администратора. В руководстве я буду использовать этот термин, поскольку он лучше отражает модель взаимодействия при поддержке DNSSEC.

Рассмотрим подробнее подпись зоны

Рассмотрение подписи записей зоны в рамках этого документа носит ознакомительный характер



Общий принцип подписи зоны

Для подписи записей применяется система электронной подписи. Записи зоны подписываются закрытым ключом. Подписи публикуются как записи RRSIG в зоне. Открытый ключ, соответствующий закрытому, публикуется как запись DNSKEY в зоне.

Далее в тексте всегда будет подразумеваться, что публикуется открытый ключ, а подпись делается соответствующим закрытым ключом, без уточнения.

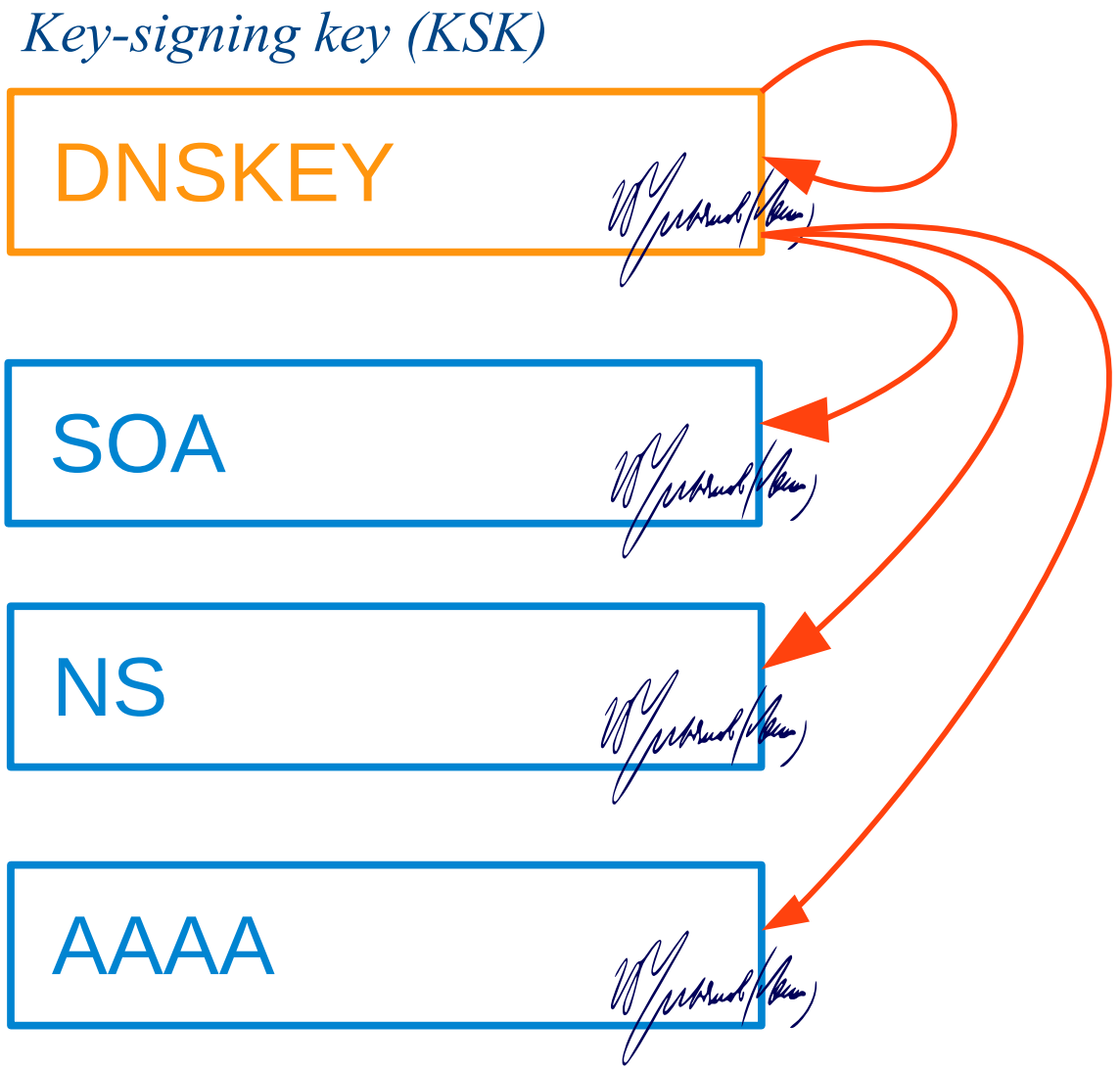
Подпись зоны одним ключом

Публикуется ключ. Записи зоны и сам ключ подписываются опубликованным ключом.

Записи зоны могут быть подписаны любым типом ключа. Но другой ключ (запись DNSKEY) может быть подписан только ключом типа KSK (Key-signing key).

Ключ KSK является «точкой входа» цепочки доверия в зону. Именно ключ KSK будет подтверждаться в родительской зоне.

Подпись зоны одним ключом



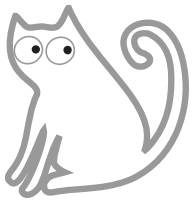
Ключом *KSK* подписан сам ключ *KSK* и все записи зоны

Подпись зоны двумя ключами

Публикуется ключ *KSK* и ключ *ZSK* (*Zone-signing key*).

Ключи *ZSK* и *KSK* подписываются опубликованным ключом *KSK*.

Записи зоны подписываются ключом *ZSK*.



Обновление ключа ZSK не требует подтверждения в родительской зоне. Следовательно ключ ZSK можно часто обновлять. Поэтому требования к безопасности ключа ZSK могут быть менее строгие.

Подпись зоны двумя ключами

Ключом *KSK* подписан сам ключ *KSK* и ключ *ZSK*

Ключом ZSK подписаны все записи зоны

Для поддержки цепочки доверия нас интересует только ключ KSK

Обобщим виды подписи зоны

Подпись с использованием одного ключа

Публикуется ключ KSK. Записи зоны и сам ключ KSK подписываются опубликованным ключом KSK.

Подпись с использованием двух ключей

Публикуется ключи KSK и ZSK. Ключи ZSK и KSK подписываются опубликованным ключом KSK._Записи зоны подписаны ключом ZSK.

Цепочка доверия

Ключи KSK *являются «точками входа» цепочки доверия в зону*.

Важные детали подписи зоны

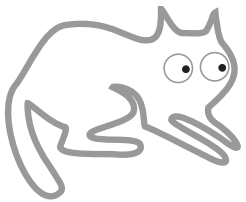
Ключей может быть несколько

Количество ключей и KSK и ZSK не оговаривается.
Ключи KSK должны подписывать каждый каждого.
Ключи ZSK должны быть подписаны ключами KSK.

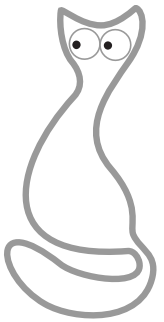
Зачем несколько ключей

Несколько ключей обязательно присутствует одновременно при их плановой ротации. Также несколько ключей используют для поддержки разных алгоритмов.

Рассмотрим подробнее организацию цепочки доверия



То, что нас интересует



Общий принцип цепочки доверия

Вспомним общий принцип подписи зоны

Записи зоны подписываются закрытым ключом. Подписи и открытый ключ, соответствующий закрытому публикуются в зоне.

> *Если мы доверяем данному ключу, то мы имеем возможность проверить любую подписанную запись.*

Принцип доверия к подписи

Родительский домен подтверждает открытый ключ KSK зоны потомка. Подтверждения выстраиваются в цепочку доверия.

Делегирование подписи

В родительской зоне публикуется отпечаток открытого ключа *KSK* домена потомка, которому надо оказать доверие – запись DS (*Delegation Signer).*

Запись DS для домена потомка публикуется только в родительской зоне.

Наличие хотя бы одной записи DS для домена рассматривается как обязательство подписи зоны этого домена.

Делегирование подписи

.tld

example DS

Запись DS — это отпечаток ключа *KSK* потомка

Запись DS для example.tld размещена у «родителя»

example.tld

DNSKEY *(KSK)*

Родительский домен подтверждает ключ KSK потомка

Цепочка доверия

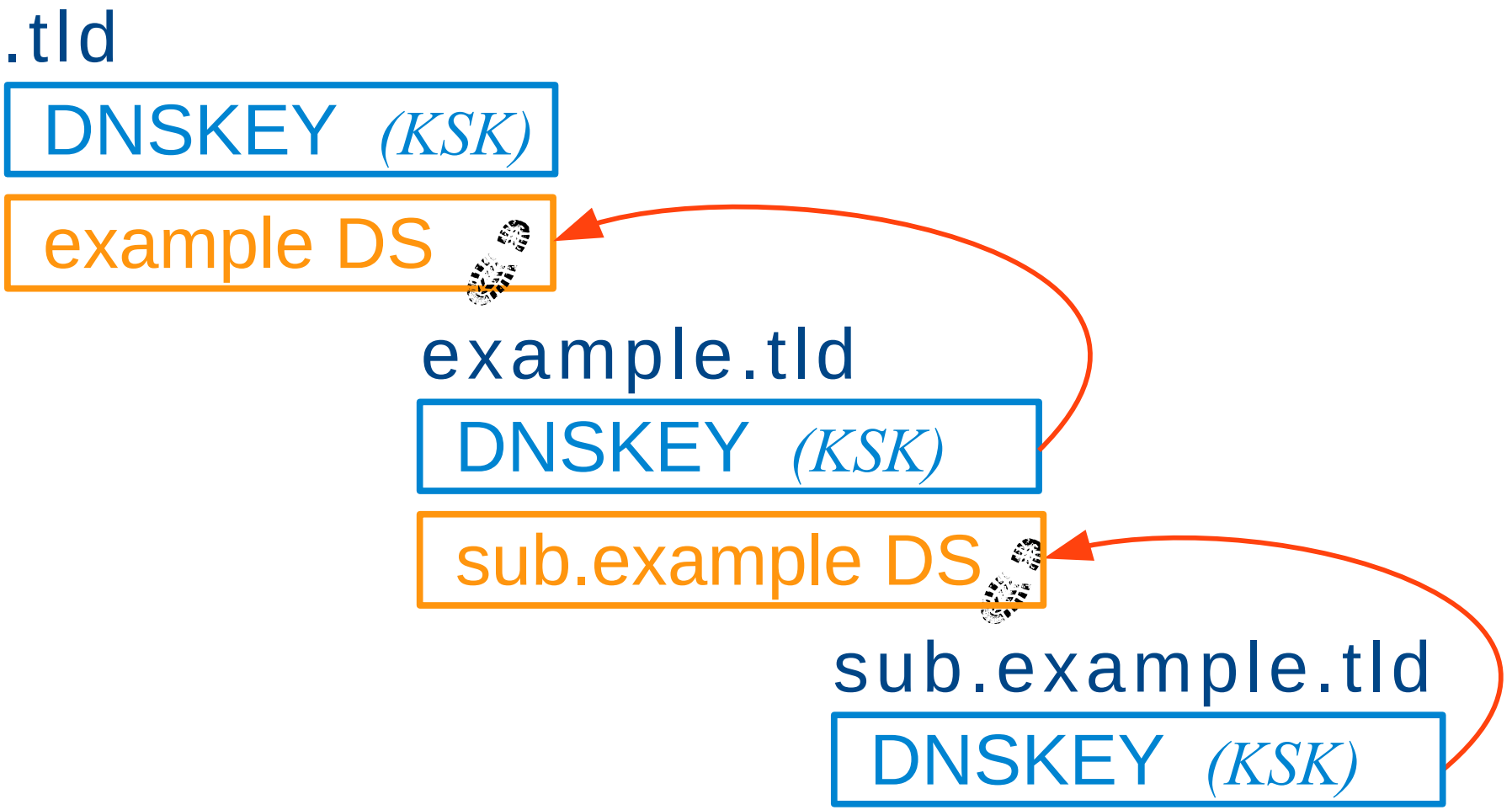
В резолверы «прошиты» корневые ключи.

Цепочка доверия идёт от корневой зоны по записям DS

> *Иерархия доверия всегда соответствует иерархии делегирования зон*

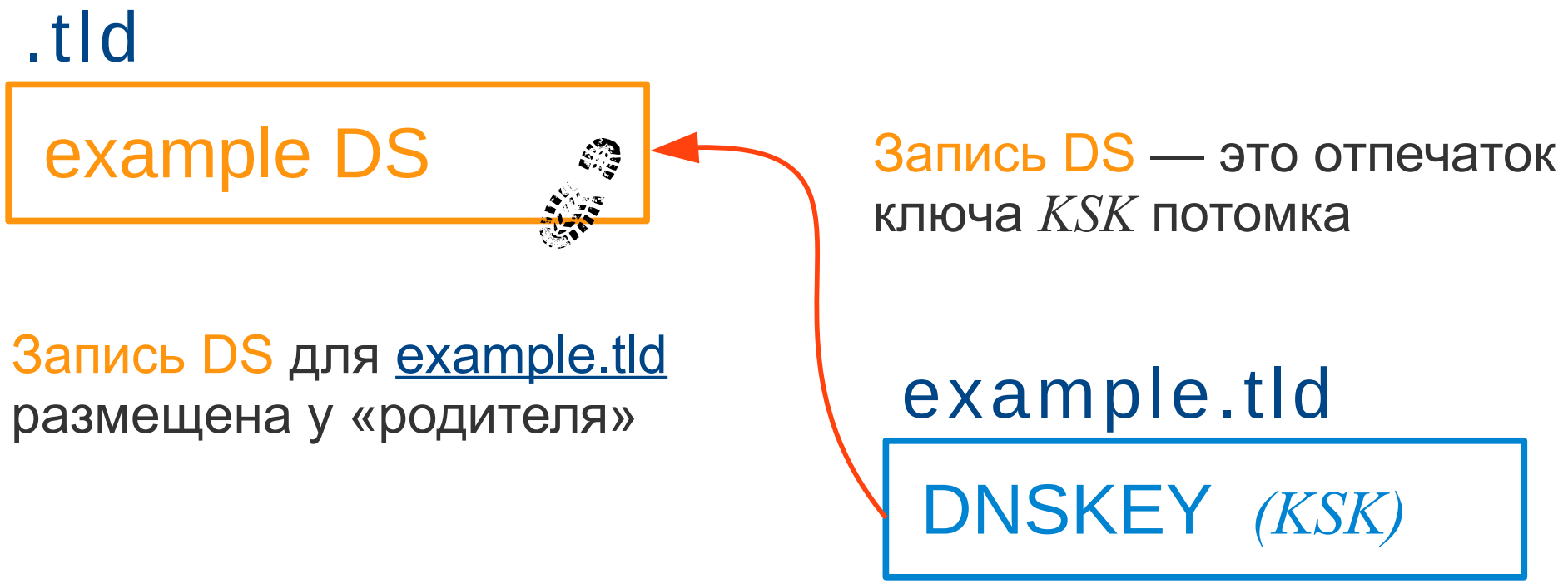
Нет разницы, откуда получена информация о записях, если она сопровождается корректными подписями, ведущими по цепочке доверия к корневому ключу, которому доверяет резолвер.

Цепочка доверия

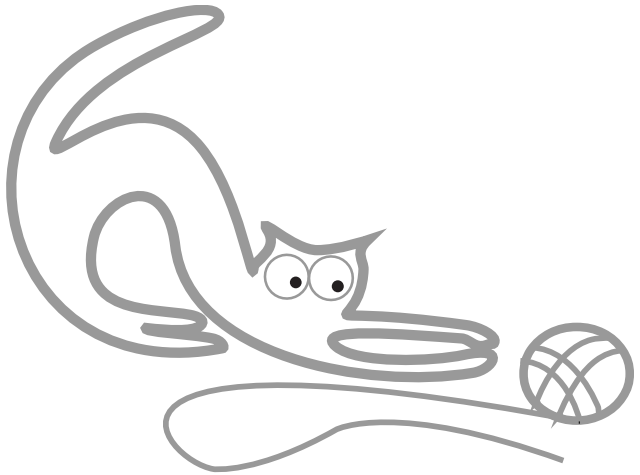


Что требуется от регистратора

Регистратор доменов определяет полномочия оператора домена на публикацию ключей *KSK* и обеспечивает передачу информации об актуальных ключах *KSK* данного домена в реестр. В виде данных DNSKEY или DS.



Техническое взаимодействие с реестрами



Размещение информации о подписи в реестре

Вне зависимости от реализации API оператора реестра передача информации о ключах в реестр происходит одним из двух способов

Передача ключа KSK (DNSKEY-интерфейс)

Реестр сам формирует записи DS из предоставленного ключа KSK.

Передача готовой записи DS (DS-интерфейс)

Реестр размещает предоставленную запись DS. Возможен вариант, когда реестр требует и ключ KSK для проверки переданной записи DS.

Расширение протокола EPP

Наиболее распространенным протоколом взаимодействия с реестрами является EPP. Для него существует расширение DNSSEC — RFC 5910 https://tools.ietf.org/html/rfc5910

Расширение предоставляет два возможных интерфейса взаимодействия — DNSKEY и DS.

RFC 5910 подробно описывает протокол и содержит исчерпывающие примеры.

Примеры интерфейсов реестров

Технический Центр Интернет (.RU, .РФ, .SU, .ДЕТИ, .TATAR)

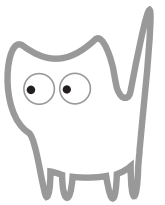
Реестр предоставляет DS интерфейс DNSSEC расширения протокола EPP с обязательной передачей соответствующего DNSKEY для проверки.

http://tcinet.ru/documents/

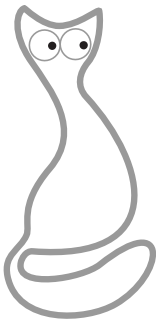
Целевое учреждение Eesti Interneti SA (.EE)

Реестр предоставляет DNSKEY интерфейс DNSSEC расширения протокола EPP. Запись DS формируется реестром самостоятельно.

https://www.internet.ee/dnssec-ru



Рассмотрим подробнее записи DNSKEY и DS



Именно с ними имеет дело регистратор

Запись DNSKEY

```
example.com.    IN    DNSKEY  257 3 13 (6a81escFb5QysOzJop
                                        VCPslEyldHjxOLNIq3
                                        ol0xZPeLn6HBLwdRIa
                                        xzlaYpefJHPaj+seBt
                                        i4j5gLWYetY3vA==)
```

Флаги	Протокол	Алгоритм	Ключ

Флаги — тип ключа. Для *KSK* значение всегда 257

Протокол — номер протокола DNSSEC. Всегда 3

Алгоритм — алгоритм ключа. Значения от 1 до 14

Ключ — публичный ключ в base64 кодировке

Алгоритмы DNSKEY

Алгоритмы ключей интересуют регистратора только как источник информации

Устаревшие алгоритмы

5 RSA/SHA-1
7 RSASHA1-NSEC3-SHA1

Современные алгоритмы

8 RSA/SHA-256
10 RSA/SHA-512
12 GOST R 34.10-2001

Новейшие алгоритмы

13 ECDSA Curve P-256 with SHA-256
14 ECDSA Curve P-384 with SHA-384

Синтаксическая проверка DNSKEY

Я настоятельно рекомендую считать допустимыми ключи от алгоритма 8 и выше, а длину ключа алгоритмов 8 и 10 считать допустимой от 2048 бит и больше

Допустимые алгоритмы

8 (RSA/SHA-256), 10 (RSA/SHA-512), 12 (GOST R 34.10-2001), 13 (ECDSA Curve P-256 with SHA-256), 14 (ECDSA Curve P-384 with SHA-384)

Допустимая длина ключа

Декодированная бинарная часть ключа

8 и 10 алгоритмы — от 256 (2048 бит) байт до 512 (4096 бит) байт
12 и 13 алгоритмы — 64 байта
14 алгоритм — 96 байт

Запись DS

Запись DS формируется из записи DNSKEY ключа, которому требуется оказать доверие

```
example.com.     IN     DS     20545 13 1 (40bd7cf025eeb433f9
                                           e74127009bd0af8c16
                                           f449)
```

Тег ключа | Алгоритм | Тип | Отпечаток

Тег ключа — контрольная сумма DNSKEY

Алгоритм — алгоритм ключа. Значения от 1 до 14

Тип — алгоритм отпечатка. Значения от 1 до 4

Отпечаток — отпечаток DNSKEY

Типы отпечатка DS

Тип отпечатка — это алгоритм, который используется для создания отпечатка

Устаревшие типы отпечатка

1 SHA-1

Современные типы отпечатка

2 SHA-256

3 GOST R 34.10-2001

Самые «сильные» типы отпечатка

4 SHA-384

Синтаксическая проверка DS

Я настоятельно рекомендую считать допустимым тип отпечатка от 2 и выше, несмотря на то, что тип 1 часто используется на практике

Тег ключа

Тег ключа — целое длиной 2 байта без знака

Допустимые типы отпечатка

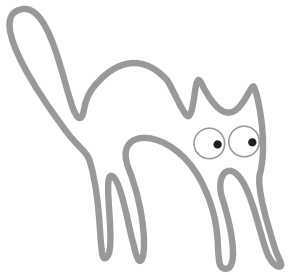
2 (SHA-256), 3 (GOST R 34.11-94), 4 (SHA-384)

Длина отпечатка

Посимвольная длина текстового представления отпечатка

Отпечаток типа 2 имеет длину 64 символа, отпечаток типа 3 имеет длину 64 символа, отпечаток типа 4 имеет длину 96 символов

Формирование записи DS



Формирование DS может пригодиться для проверки пользовательских данных

Формирование DS

Тег ключа | Алгоритм | Тип | Отпечаток

Тег ключа

Тег ключа — технически это контрольная сумма поля данных записи DNSKEY, значение которой используется как идентификатор ключа

Алгоритм

Значение алгоритма соответствует значению поля алгоритма ключа DNSKEY

Отпечаток

Отпечаток желаемого типа формируется из данных, полученных путем конкатенации доменного имени и поля данных записи DNSKEY

Особенности формирования DS

Описание формирования записи DS в RFC 4034, которое затем копируют производные документы, крайне плохое. Поэтому необходимо дать комментарии.

Тег ключа

Не существует описания алгоритма вычисления тега ключа. Алгоритм представлен в RFC как программный код на языке Си.

Исходные данные

Все исходные данные для вычислений отпечатка берутся в бинарном виде — так, как они находятся в протоколе DNS. Доменное имя для отпечатка должно быть полным и в том виде, в котором оно находится в протоколе DNS в поле имени.

Формирование DS на языках программирования

Примеры

Я написал несколько примеров формирования записи DS. Код не является оптимальным. Он служит только для понимания и примера методов формирования тега ключа и отпечатка.

Исходные тексты программ

Исходные тексты программ примеров вычислений отпечатка и тега ключа на языках программирования PHP, Perl, Python версии 2 и 3, Go и Java с комментариями и тестовыми данными выложены на GitHub:
https://github.com/diphost/ds-calc

Пример на языке Python 3

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import struct, hashlib, base64
# $domain — каноническое имя домена с точкой на конце
# $algorithm и $publickey — алгоритм ключа и сам ключ в base64 (как в DNSKEY)
# возвращает keytag и отпечаток SHA256
def calc_ds_sha256(owner, algorithm, publickey):
        flags, protocol = 257, 3
        # сформировать бинарную DNSKEY RDATA
        dnskey_rdata = struct.pack('!HBB', int(flags), int(protocol), int(algorithm))
        dnskey_rdata += base64.b64decode(publickey)
        # вычислить контрольную сумму keytag
        crc = 0
        for i in range(len(dnskey_rdata)):
                b = struct.unpack('B', dnskey_rdata[i:i+1])[0]
                crc += b if i & 1 else b << 8
        keytag = ((crc & 0xFFFF) + ((crc >> 16) & 0xFFFF)) & 0xFFFF
        # сформировать бинарный вид доменного имени
        domain_wire_format = b''
        for part in bytes(owner,'ascii').split(b'.'):
                domain_wire_format += struct.pack('B', len(part)) + part
        # создать отпечаток
        digest = hashlib.sha256(domain_wire_format + dnskey_rdata)
        return (keytag, digest.hexdigest().upper())
```

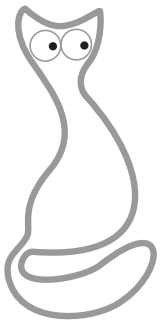
Пример DS на языке Perl

```perl
use MIME::Base64;
use Digest::SHA;
# $domain — каноническое имя домена с точкой на конце
# $algorithm и $publickey — алгоритм ключа и сам ключ в base64 (как в DNSKEY)
# возвращает keytag и отпечаток SHA256
sub calc_ds_sha256($$$) {
    my ($domain, $algorithm, $publickey) = @_;
    my ($flags, $protocol) = (257,3);
    # сформировать бинарную DNSKEY RDATA
    my $dnskey_rdata = pack('nCC', $flags, $protocol, $algorithm);
    $dnskey_rdata .= decode_base64($publickey);
    # вычислить контрольную сумму keytag
    my $crc = 0;
    for(my $i = 0; $i < length($dnskey_rdata); $i++) {
        my $b = ord(substr $dnskey_rdata, $i, 1);
        $crc += ($i & 1) ? $b : $b << 8;
    };
    my $keytag =  (($crc & 0xffff) + (($crc >> 16) & 0xffff )) & 0xffff;
    # сформировать бинарный вид доменного имени
    my @parts = split(/\./, $domain, -1);
    my $domain_wire_format = '';
    foreach my $part (@parts) {
        $domain_wire_format .= pack('C', length $part ) . $part;
    };
    # создать отпечаток
    my $digest = uc Digest::SHA::sha256_hex($domain_wire_format . $dnskey_rdata);
    return $keytag, $digest;
};
```

Пример DS на языке Go

```
package main
import "encoding/base64"
import "strings"
import "crypto/sha256"
import "encoding/hex"
func calc_ds_sha256(owner string, algorithm uint8, publickey string) (uint16,[]byte) {
        var flags uint16 = 257
        var protocol uint8 = 3
        var keytag uint32
        dnskey_rdata := []byte{byte(flags >> 8), byte(flags), byte(protocol), byte(algorithm)}
        decoded_publickey, _ := base64.StdEncoding.DecodeString(publickey)
        dnskey_rdata = append(dnskey_rdata, decoded_publickey...)
        for i, b := range dnskey_rdata {
                if i&1 != 0 {
                        keytag += int(b)
                } else {
                        keytag += int(b) << 8
                }
        }
        keytag = ((keytag & 0xFFFF) + ((keytag >> 16) & 0xFFFF)) & 0xFFFF
        domain_wire_format := make([]byte, 0, 256)
        for _, part := range strings.Split(owner, ".") {
                domain_wire_format = append(domain_wire_format, byte(len(part)))
                domain_wire_format = append(domain_wire_format, part[:]...)
        }
        hasher := sha256.New()
        hasher.Write(append(domain_wire_format, dnskey_rdata[:]...))
        digest := make([]byte, 2 * hasher.Size())
        hex.Encode(digest, hasher.Sum(nil))
        return uint16(keytag), digest
}
```

Инструменты работы с записями DS и DNSKEY



Инструменты командной строки пригодятся для отладки

DNSSEC Tools

https://www.dnssec-tools.org/

Один из самых популярных инструментов командной строки. Входит в полную поставку ISC BIND. В большинстве UNIX-дистрибутивах имеет название *dnssec-tools*

Создание ключа KSK

```
$ dnssec-keygen  -a ECDSAP256SHA256 -f KSK example.com.
Generating key pair.
Kexample.com.+013+34607
```

Создание записи DS типа 2

```
$ dnssec-dsfromkey -a SHA-256  Kexample.com.+013+34607
example.com. IN DS 34607 13 2 3F6ED17BBCAAD567619C0D43F05DD9BD0...
```

LDNS Utils

https://www.nlnetlabs.nl/projects/ldns/

Набор инструментов командной строки на основе библиотеки Ldns. Разработан при поддержке RIPE. В большинстве UNIX-дистрибутивах имеет название *ldns-utils*

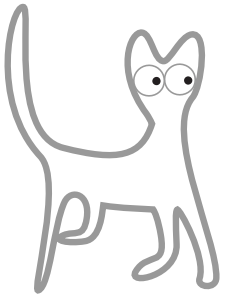
Создание ключа KSK

```
$ ldns-keygen  -a ECDSAP256SHA256 -k -r /dev/urandom  example.com
Kexample.com.+013+09534
```

Создание записи DS типа 2

```
$ ldns-key2ds -n -2 Kexample.com.+013+09534.key
example.com.   3600    IN  DS  9534 13 2 1a7f795c11c3f0e3b74299e1e...
```

Подходы к реализации



Что поддерживать. Что не поддерживать. Как поддерживать.

Поддержка нескольких ключей

Обязательно поддерживать

Несколько разных KSK одновременно используются при ротации ключей и при переносе подписанной зоны с одного сервера имен на другой. Обеспечение возможности иметь несколько KSK является обязательной.

Совместимость

Несколько ключей KSK с разными алгоритмами иногда делают для поддержки совместимости с резолверами. Однако, уровень распространения DNSSEC сегодня так низок, что тянуть бремя совместимости бессмысленно. По возможности следует избегать такой практики.

Поддержка нескольких типов отпечатков одного и того же ключа

Размер ответа

Для делегирования подписи достаточно одной записи DS для каждого ключа KSK. Несколько дублирующих записей увеличивают размер DNS-ответа.

Совместимость

Несколько записей DS с разными алгоритмами отпечатка, ссылающимися на один ключ, делают для поддержки совместимости с резолверами. Однако, уровень распространения DNSSEC сегодня так низок, что тянуть бремя совместимости бессмысленно.

Поддержка нескольких DS с разными алгоритмами отпечатка уместна, если вы решили поддерживать маргинальные алгоритмы. Например ГОСТ.

Поддержка алгоритмов ГОСТ

Маргинальность

Набор алгоритмов ГОСТ поддерживается не всеми криптографическими библиотеками. Для программных реализаций часто требуются нестандартные решения. Ключи и отпечатки набора алгоритмов ГОСТ будут распознаваться не всеми резолверами, поэтому их необходимо дублировать более традиционными ключами и отпечатками.

Быть или не быть

Набор алгоритмов ГОСТ вне зависимости от нашего желания используется и будет использоваться на территории России. Поддержка этих алгоритмов в DNSSEC будет небольшим шажком по выходу алгоритмов ГОСТ из маргинального поля.

Программная поддержка набора алгоритмов ГОСТ

Для тех, кто хочет поддерживать набор алгоритмов ГОСТ (алгоритм 12 ключа и тип 3 отпечатка) в DNSSEC

В своих примерах я реализовал поддержку алгоритмов ГОСТ для создания отпечатков. В PHP и Perl я использовал интерфейсы к OpenSSL. Для остальных языков я использовал сторонние библиотеки.

PHP	с версии *5.6* поддерживает нужный алгоритм
Perl	модуль *Digest::GOST::CryptoPro*
Python	пакет https://pypi.python.org/pypi/pygost/
Go	пакет http://www.cypherpunks.ru/gogost/
Java	криптопровайдер http://www.bouncycastle.org/

Рекомендуемые алгоритмы

Регистратор не должен ограничивать пользователя в выборе алгоритмов ключа или отпечатка, если это не подразумевает политика конкретной зоны. Но регистратор может рекомендовать использовать те или иные алгоритмы.

Эффективным алгоритмом для ключей DNSSEC является ECDSA (алгоритм ключа 13). По защищенности он не хуже RSA. Создание подписи ECDSA в разы быстрее аналогичной RSA. Ключ ECDSA по размеру меньше ключа RSA при аналогичном уровне защиты. Размер ключа ECDSA позволяет умещать большинство ответов DNS в один UDP пакет.

Для отпечатка наиболее практично использовать алгоритм SHA-256 (тип отпечатка 2).

Наличие API

Несколько доменов

Один пользователь может иметь несколько доменов. Можно предположить, что у них будут разные KSK.

Ротация ключей

Существует несколько практик ротации ключей KSK. Но все они подразумевают разнесенный по времени процесс добавления, удаления записей DNSKEY и DS.

Компрометация ключей KSK

Если ключ KSK скомпрометирован, следует незамедлительно убрать или сменить запись DS.

Отсутствие API с DNSSEC у регистратора равно отсутствию внедрения DNSSEC

DNSSEC API регистратора

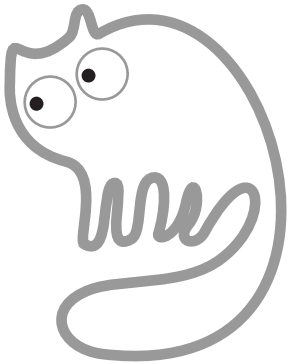
API для партнеров и администраторов доменов

Рекомендую сделать API для работы с DNSSEC как для партнеров, так и для конечных Администраторов доменов. Даже если в API для Администраторов не будет других функций.

API для операторов доменов?

Реестр доменов .CA (CIRA) предлагает специальный протокол для операторов доменов: https://github.com/CIRALabs/DSAP/

Нельзя не сделать DNSSEC API, когда на слайде есть котик



Мысли вслух

У разных реестров могут быть реализованы разные интерфейсы для работы с DNSSEC. Регистратор может для своих пользователей в API реализовать только передачу DNSKEY, а записи DS создавать самостоятельно в зависимости от требования конкретного реестра.

Что ещё может регистратор

Регистратор может предоставить инструмент для проверки делегирования домена.
Например: http://dnsviz.net/

Регистратор может разработать мониторинг подписанных доменов на ошибки.

Регистратор может внедрить DNSSEC на своих серверах имен и предложить услугу «полного цикла».

Регистратор может рассказать о своём опыте на конференциях.

Итог в трех слайдах

Слайд «вопросы» близко

Итог. Слайд 1

.tld

example DS

Запись DS — это отпечаток ключа *KSK* потомка

Запись DS для example.tld размещена у «родителя»

example.tld

DNSKEY *(KSK)*

Регистратор определяет полномочия оператора домена на публикацию ключей и обеспечивает передачу информации об актуальных ключах *KSK* поддерживаемого домена в родительскую зону согласно политике реестра.

Итог. Слайд 2

Взаимодействие с реестром

Оператор реестра родительской зоны предоставляет один из двух типов интерфейсов — передача записи DNSKEY или передача записи DS. Во втором случае реестр может требовать соответствующую запись DNSKEY для проверки.

Расширение к протоколу EPP

Наиболее распространенным протоколом взаимодействия с реестрами является EPP. Для него существует расширение DNSSEC — RFC 5910

Итог. Слайд 3

Множественность записей

Необходимо обеспечить возможность размещения нескольких разных ключей KSK для одного домена одновременно. Записей DS на один ключ тоже может быть несколько, но имеет смысл только для совместимости при использовании алгоритма ГОСТ.

API

Регистратор реализует API и иные интерфейсы для определения полномочий оператора домена на публикацию ключей и передачу в реестр информации о ключах в виде записей DS и/или DNSKEY. Регистратор также может делать самостоятельную предварительную проверку.

Пишите мне

Если возникли вопросы, предложения или требуется помощь, да и в любом случае — пишите мне:

phil@diphost.ru

Пожертвования на кофе

Я хочу, чтобы мои статьи были в свободном доступе и бесплатными. Я потратил больше сотни часов на создание этого документа. Если вы хотите, чтобы я продолжал работу по созданию новых руководств, пожалуйста, рассмотрите возможность пожертвования.

- **PayPal https://www.paypal.me/schors**
- **Яндекс.Деньги http://yasobe.ru/na/schors**
- **Bitcoin** 17V94QS4vaBwec1Qwqp2ow5b3tbrRGGcne

Благодарности

Особая благодарность Вартану Хачатурову за поддержку и заразительный энтузиазм. Без него бы не было этого руководства. Большое спасибо Александру Венедюхину и Дмитрию Белявскому за консультации, техническую помощь и рецензирование. Спасибо Олегу Цареву за регулярную техническую помощь. Без редактирования текста горячо любимой Натальи Кулиной мысли жили бы на произвольных страницах руководства своей жизнью в окружении грамматических ошибок. Сергей Мясоедов подкинул несколько интересных ссылок, которые я с удовольствием использовал. Несомненно это руководство не появилось бы без моих друзей и знакомых, которые активно сопротивлялись моему увлечению DNSSEC, за что я им тоже благодарен.

Примеры из этого руководства

Исходные тексты программ примеров вычислений отпечатка и тега ключа на языках программирования PHP, Perl, Python версии 2 и 3, Go и Java с комментариями и тестовыми данными выложены на GitHub:

https://github.com/diphost/ds-calc

Полезные ресурсы

Самое популярное программное обеспечение для работы с DNSSEC. Проект OpenDNSSEC
https://www.opendnssec.org/

Утилиты командной строки dnssec-tools
https://www.dnssec-tools.org/

Библиотека Ldns
https://www.nlnetlabs.nl/projects/ldns/

Визуализация DNS и DNSSEC
http://dnsviz.net/

Предложение CIRA по протоколу обновления DS
https://github.com/CIRALabs/DSAP/

Библиотеки для языков программирования

Библиотека для работы с алгоритмами ГОСТ на Go
http://www.cypherpunks.ru/gogost/

Библиотека для работы с алгоритмами ГОСТ на Python
https://pypi.python.org/pypi/pygost/

Внешний криптопровайдер для Java BouncyCastle
http://www.bouncycastle.org/

Библиотека работы с DNS для Java
http://www.dnsjava.org/

Библиотека работы с DNS для Perl
http://search.cpan.org/~nlnetlabs/Net-DNS/

Подборка RFC по DNSSEC

RFC 4033 Введение в DNSSEC
RFC 4034 Ресурсные записи для DNSSEC
RFC 4035 Модификации протокола DNS для DNSSEC

RFC 3110 RSA/SHA-1 подписи в DNS
RFC 4509 Использование SHA-256 для записей DS
RFC 5702 Использование SHA-2 в DNSKEY и RRSIG
RFC 5933 Использование алгоритмов ГОСТ в DNSSEC
RFC 6605 Использование ECDSA и SHA-384 в DNSSEC

RFC 5910 Расширение протокола EPP для DNSSEC

RFC 6781 Эксплуатация DNSSEC
RFC 7583 Соображения по ротации ключей DNSSEC

Общественное достояние

Это бесплатный документ, переданный в общественное достояние.

Любой человек может свободно копировать, изменять, публиковать, цитировать, использовать, продавать или распространять этот документ на любых носителях целиком или по частям для любых коммерческих или некоммерческих целей во всех смыслах.