QApp

Гиперикум — проект квантово-устойчивой цифровой подписи для стандартизации в России

Сергей Гребнев
Руководитель направления прикладных исследований
структурного подразделения «КуАпп»
Российского квантового центра

При поддержке ГАЗПРОМБАНК

QApp

ВВЕДЕНИЕ

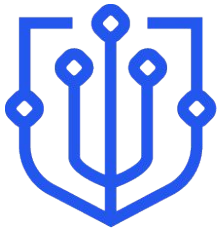
Общая схема

в целом следует SPHINCS+ (Andreas Hülsing et al, 2017), но использует отечественную хэш-функцию «Стрибог-256» и ряд усовершенствований, не вошедших в вариант для NIST

Статус работ

- готово общее описание схемы
- получено обоснование стойкости мета-схемы
- завершена работа по модификации схемы
- завершается окончательный выбор параметров
- разработан прототип программной реализации
- рассматривается реализация аппаратного ускорения совместно с отечественными производителями микропроцессоров

QApp



гиперикум

Метафоры и этимология названия: новые криптографические алгоритмы сейчас принято называть в честь объектов флоры и фауны, например, «Крыжовник» или «Кузнечик»

По этой же логике и было выбрано название «**Гиперикум**»

- Зверобой (лат. **Hypericum**) — род цветковых растений семейства «Зверобойные»
- Также название является отсылкой к криптографическому термину «гипер-дерево»

QApp гиперикум

СТРОИТЕЛЬНЫЕ БЛОКИ ГИПЕРИКУМА

- WOTS-TW — одноразовая схема подписи
- схема подписи XMSS (eXtended Merkle Signature Scheme)
- гипердеревья (HT)
- лес случайных подмножеств (FORS)

QApp гиперикум

НАБОРЫ ПАРАМЕТРОВ ГИПЕРИКУМА

n: параметр стойкости в битах

w: параметр Винтерница

h: высота гипердерева

d: число уровней в гипердереве

k: число деревьев в схеме FORS

t: число листьев в деревьях FORS

m: длина хэша сообщения M в байтах: $m = \lfloor (k \log t + 7)/8 \rfloor + \lfloor (h - h/d + 7)/8 \rfloor + \lfloor (h/d + 7)/8 \rfloor$.

QApp гиперикум

ВЫРАБОТКА КЛЮЧА

Входные значения : Секретное начальное значение **SK**.seed, адрес $\mathbf{ADRS}$, индекс секретного ключа idx = $i \cdot t + j$

Выходные значения: Секретное значение FORS sk

1 **SK**.seed= sec_rand(n);
2 **SK**.prf= sec_rand(n);
3 **PK**.seed= sec_rand(n);
4 PK.root = ht_PKgen (**SK**.seed, **PK**.seed);
5 Вернуть((**SK**.seed, **SK**.prf, **PK**.seed, PK.root), (**PK**.seed, PK.root));

QApp гиперикум

ВЫРАБОТКА ПОДПИСИ

Входные значения : Сообщение M, секретный ключ SK = (**SK**.seed, **SK**.prf, **PK**.seed, PK.root)
Выходные значения: Подпись SIG

1 **ADRS** = toByte(0, 32);
2 **Начать** Генерация рандомизации
3 opt = rand(n);
4 R = $\mathbf{PRF_{msg}}$ (**PK**.seed, **SK**.prf, opt, *M*); SIG = SIG || R;
5 **Начать** Вычисление хэша сообщения
6 digest = $\mathbf{H_{msg}}$ (R, **PK**.seed, PK.root, *M*);
7 tmp_md = первые floor((ka +7)) бит от digest;
8 tmp_idx_tree = следующие floor$((h - h/d + 7))$ бит от digest;
9 tmp_idx_leaf = следующие floor$((h/d + 7))$ бит от digest;
10 md = первые *ka* бит от tmp_md;
11 **if** последние *a* бит от md $\neq 0$ **then**
12 GO TO 2
13 idx_tree = первые $h - h/d$ бит от tmp_idx_tree;
14 idx_leaf = первые h/d бит от tmp_idx_leaf;
15 **Начать** Вычисление подписи FORS
16 **ADRS**.setLayerAddress(0);
17 **ADRS**.setTreeAddress(idx_tree);
18 **ADRS**.setType(FORS_TREE);
19 **ADRS**.setKeyPairAddress(idx_leaf);
20 SIG_FORS = fors_sign (md, **SK**.seed, **PK**.seed, **ADRS**);
21 SIG = SIG || SIG_FORS;
22 **Начать** Вычисление открытого ключа FORS
23 PK_FORS = fors_pkFromSig (SIG_FORS, md, **PK**.seed, **ADRS**);
24 **Начать** Вычисление подписи открытого ключа FORS
25 **ADRS**.setType(TREE);
26 SIG_HT = ht_sign (PK_FORS, **SK**.seed, **PK**.seed, idx_tree, idx_leaf);
27 SIG = SIG || SIG_HT;
28 Вернуть(SIG);

QApp гиперикум

ПРОВЕРКА ПОДПИСИ

Входные значения : Сообщение M, подпись SIG, открытый ключ PK
Выходные значения: Булево значение

```
1  ADRS = toByte(0, 32);
2  Начать Инициализация
3      ADRS = toByte(0, 32);
4      R = SIG.getR();
5      SIG_FORS = SIG.getSIG_FORS();
6      SIG_HT = SIG.getSIG_HT();
7  Начать Вычисление хэша сообщения
8      digest = H_msg (R, PK.seed, PK.root, M);
9      tmp_md = первые floor((ka +7)) бит от digest;
10     tmp_idx_tree = следующие floor((h − h/d + 7)) бит от digest;
11     tmp_idx_leaf = следующие floor((h/d + 7)) бит от digest;
12     md = первые ka бит от tmp_md;
13     if последние a бит от md ≠ 0 then
14         Вернуть(0)
15     idx_tree = первые h − h/d бит от tmp_idx_tree;
16     idx_leaf = первые h/d бит от tmp_idx_leaf;
17 Начать Вычисление открытого ключа FORS
18     ADRS.setLayerAddress(0);
19     ADRS.setTreeAddress(idx_tree);
20     ADRS.setType(FORS_TREE);
21     ADRS.setKeyPairAddress(idx_leaf);
22     PK_FORS = fors_pkFromSig (SIG_FORS, md, PK.seed, ADRS);
23 Начать Верификация подписи открытого ключа FORS
24     ADRS.setType(TREE);
25     Вернуть(ht_verify (PK_FORS, SIG_HT, PK.seed, idx_tree, idx_leaf, PK.root));
```

QApp гиперикум

ПАРАМЕТРЫ

Выбор параметров для различных уровней стойкости

	n	h	d	$\log(t)$	k	w	бит, кл/кв	уровень	Sig, байтов
Гиперикум-128s	128	63	7	12	14	16	133/56	**1**	7856
Гиперикум-128f	128	66	22	6	33	16	128/56	**1**	17088
Гиперикум-256s	256	64	8	14	22	16	255/120	**2**	29792
Гиперикум-256f	256	68	17	9	35	16	255/120	**2**	49856

Зависимость размера ключей и подписи от параметров

	SK	PK	Sig
Size	$4n$	$2n$	$(h + (k-1)(\log t + 1) + d \cdot (\text{len} + 1) + 1)n$

QApp гиперикум

ГИПЕРИКУМ И СТРИБОГ

Используется «Стрибог-256» (ГОСТ 34.11-2018), псевдослучайные функции и функция HMAC по Рекомендациям Р 50.1.113–2016.

$$\mathbf{H_{msg}}(R, \mathbf{PK}.\text{seed}, PK.root, M) =$$
$$= \text{PRF\_TLS\_GOSTR3411\_2012\_256}(R||\mathbf{PK}.\text{seed}||\text{Streebog\_256}(R||\mathbf{PK}.\text{seed}||PK.root||M))$$

с длиной выхода m

$$\mathbf{PRF}(\mathbf{PK}.\text{seed}, \mathbf{SK}.\text{seed}, \mathbf{ADRS}) =$$
$$= \text{PRF\_TLS\_GOSTR3411\_2012\_256}(\mathbf{SK}.\text{seed}, \mathbf{ADRS}, \mathbf{PK}.\text{seed});$$
$$\mathbf{PRF_{msg}}(\mathbf{PK}.\text{seed}, \mathbf{SK}.\text{prf}, OptRand, M) =$$
$$= \text{HMAC\_GOSTR3411\_2012\_256}(\mathbf{SK}.\text{prf}, \mathbf{PK}.\text{seed}||OptRand||M)$$

$$F(\mathbf{PK}.\text{seed}, \mathbf{ADRS}, M1) = \text{Streebog\_256}(BlockPad(\mathbf{PK}.\text{seed})||\mathbf{ADRS}||M1)$$
$$H(\mathbf{PK}.\text{seed}, \mathbf{ADRS}, M1||M2) = \text{Streebog\_256}(BlockPad(\mathbf{PK}.\text{seed})||\mathbf{ADRS}||M1||M2)$$
$$\mathbf{Th}_l(\mathbf{PK}.\text{seed}, \mathbf{ADRS}, M) = \text{Streebog\_256}(BlockPad(\mathbf{PK}.\text{seed})||\mathbf{ADRS}||M)$$

$BlockPad(\mathbf{PK}.\text{seed}) = \mathbf{PK}.\text{seed}||toByte(0, bl - n)$, где $bl = 64$ для Стрибог-256.

QApp гиперикум

РЕДУКЦИОНИСТСКАЯ СТОЙКОСТЬ

Вероятность успеха общих атак:

Property	Success probability	Status
SM-TCR	$\Theta((q+1)^2/2^n)$	доказано
SM-DSPR	$\Theta((q+1)^2/2^n)$	предположение
SM-PRE	$\Theta((q+1)^2/2^n)$	основано на предположении
PRF	$\Theta(12q/\sqrt{2^n})$	доказано
SM-UD	$\Theta(12q/\sqrt{2^n})$	доказано
ITSR	$\Theta((q+1)^2 \cdot X)$	предположение
S-TCR(+C)	$\Theta((q+1)^2/2^n)$	доказано

Andreas Hülsing, Mikhail Kudinov. Recovering the tight security proof of SPHINCS+.
Daniel J. Bernstein и др. The SPHINCS+ Signature Framework.

QApp гиперикум

ПРОИЗВОДИТЕЛЬНОСТЬ

Прототип без оптимизаций:

Схема	Стойкость	KeyGen	Sig	SVerify
RSA-2048	1 (NIST)	66	0.66	0.02
Hypericum-128s	1	509	7188	8.68
Hypericum-128f	1	15.98	463	20.57
SPHINCS+(s), SHA-256	1 (NIST)	275	3938	4.6
SPHINCS+(f), SHA-256	1 (NIST)	8.68	252.55	11.19
Hypericum-256s	2	1145	13894	21.23
Hypericum-256f	2	71.64	1616	40.78
SPHINCS+(s), SHA-256	5 (NIST)	693	8567	12.66
SPHINCS+(f), SHA-256	5 (NIST)	43.51	995	24.84
Крыжовник	1	0.24	20.92	0.27

Мс/операция, Intel Core i7-7500U CPU 2.7GHz, google benchmark, realtime ядро linux версии 4.18, отключены turbo boost и hyper threading

QApp гиперикум

ОБЛАСТИ ПРИМЕНЕНИЯ

Как и другие схемы без сохранения состояния:
подпись образов прошивок ПЗУ

Схемы с сохранением состояния (e.g. Iggdrasil (В. Цыпышев)):
системы цепного реестра

QApp

гиперикум

Сергей Гребнев
Руководитель направления прикладных исследований
структурного подразделения «КуАпп»
Российского квантового центра
sg@qapp.tech
+7 910 469-91-26

Благодарим за консультации М.Кудинова

QApp.tech

Sk Киберкластер

При поддержке ГАЗПРОМБАНК