**Описание и реализация алгоритмов шифрования Rabbit и хэширования Whirlpool**

**I)Rabbit**

**Rabbit** — высокоскоростной [поточный шифр](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) впервые представленный[[1]](https://ru.wikipedia.org/wiki/Rabbit#cite_note-1) в феврале 2003 года на 10-м симпозиуме FSE. В мае 2005, он был отправлен на конкурс [eStream](https://en.wikipedia.org/wiki/eSTREAM" \o "en:eSTREAM), целью которого было создание европейских стандартов для поточных систем шифрования.

Разработчиками Rabbit являются [Martin Boesgaard](https://ru.wikipedia.org/w/index.php?title=Martin_Boesgaard&action=edit&redlink=1), [Mette Vesterager](https://ru.wikipedia.org/w/index.php?title=Mette_Vesterager&action=edit&redlink=1), [Thomas Pedersen](https://ru.wikipedia.org/w/index.php?title=Thomas_Pedersen_(CS)&action=edit&redlink=1), [Jesper Christiansen](https://ru.wikipedia.org/w/index.php?title=Jesper_Christiansen_(CS)&action=edit&redlink=1" \o "Jesper Christiansen (CS) (страница отсутствует)) и [Ove Scavenius](https://ru.wikipedia.org/w/index.php?title=Ove_Scavenius&action=edit&redlink=1).

Rabbit используют 128-битный ключ и 64-битный инициализирующий вектор. Шифр был разработан с целью использования в программном обеспечении, как обладающий высокой скоростью шифрования. При этом скорость шифрования могла достигать 3.7 циклов в байт([CPB](https://en.wikipedia.org/wiki/Cycles_per_byte)) для процессора Pentium 3 и 10.5 циклов в байт для ARM7. Тем не менее, шифр также оказался быстрым и компактным при реализации в аппаратном обеспечении.

Основным компонентом шифра является генератор [битового потока](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), который шифрует 128 битов сообщения за итерацию. Достоинство шифра в тщательном перемешивании его внутренних состояний между двумя последовательными итерациями. Функция перемешивания полностью основана на арифметических операциях, доступных на современных процессорах, то есть [S-блоки подстановок](https://en.wikipedia.org/wiki/substitution_box) и поисковые таблицы не нужны для реализации шифра.

### Алгоритм шифрования: 1. Ключ и IV (инициализационный вектор)

**Ключ**: Это основная часть входных данных алгоритма. Rabbit использует 128-битный ключ, который разделяется на 8 16-битных частей. Этот ключ управляет состоянием генератора псевдослучайных чисел и влияет на генерацию потоков.

**IV (инициализационный вектор)**: Хотя IV не является обязательным, он добавляется для дополнительной безопасности, особенно если одно и то же сообщение шифруется разными ключами. IV — это 64-битная строка, которая используется для модификации начальных значений внутренних регистров.

**2. Инициализация состояния**

Rabbit управляет состоянием с помощью двух массивов:

* **8 внутренних регистров (x)**: Каждый регистр состоит из 32 бит. Они инициализируются на основе ключа и состояния генерации.
* **8 счетчиков (c)**: Также 32-битные регистры, которые обновляются на каждом шаге. Они управляют изменением состояния регистров x.

**Подготовка:**

* Ключ разбивается на восемь 16-битных частей (каждая по 2 байта). Они используются для начальной установки регистров и счетчиков:
  + Регистры x0, x2, x4, x6 инициализируются значениями первой половины ключа.
  + Регистры x1, x3, x5, x7 инициализируются значениями второй половины ключа, с обратной расстановкой байт.
  + Счетчики c0, c2, c4, c6 получают значения второй половины ключа.
  + Счетчики c1, c3, c5, c7 получают значения первой половины ключа.

Если используется IV, оно изменяет начальное состояние регистров и счетчиков. Каждый 16-битный фрагмент IV применяется для модификации счетчиков.

**3. Ключевое расширение**

Rabbit использует механизм расширения ключа для подготовки начального состояния. Это делается в несколько раундов:

1. **Выполняется 4 итерации обновления состояния**, чтобы ключ повлиял на начальные регистры и счетчики.
2. **Счетчики модифицируются дополнительно** с использованием нового состояния ключа.

На этом этапе регистры и счетчики будут полностью инициализированы и готовы для генерации потоков ключей.

**4. Генерация потоков ключей**

Алгоритм Rabbit генерирует поток ключей, который затем будет применен к исходному тексту.

**Основные шаги генерации потока ключей:**

1. **Обновление счетчиков (counter update)**: Сначала обновляются счетчики c. При каждом раунде они изменяются, используя простую формулу, которая учитывает их предыдущее значение и константу. Также добавляется так называемая "переход через ноль" (carry), который срабатывает при переполнении счетчика.
2. **Выработка новых значений регистров x (state update)**: Обновление регистров x происходит через нелинейную функцию, которая включает побитовые сдвиги, сложение по модулю и XOR. Здесь также используется предыдущие состояния регистров и счетчиков, так что новое состояние x зависит от обоих параметров.
3. **Выработка выходного потока (key stream)**: Каждый раз после обновления состояния и счетчиков вычисляется выходной поток из значений регистров x. Он разбивается на 128-битные блоки (4 32-битных слова), которые затем используются для шифрования.

Преобразование выполняется через дополнительное нелинейное преобразование и побитовые сдвиги значений регистров, чтобы сгенерировать псевдослучайный 128-битный поток.

**5. Шифрование/Расшифрование**

* **Шифрование**: Генерация потока ключей продолжается до тех пор, пока не будет покрыт весь исходный текст. Полученный поток используется для шифрования текста через побитовое XOR.

Для каждого блока текста, равного длине потока ключей (128 бит или 16 байт), происходит операция:

ciphertext=plaintext⊕keystreamciphertext = plaintext \oplus keystreamciphertext=plaintext⊕keystream

Весь открытый текст разбивается на блоки, и к каждому применяется операция XOR с соответствующим блоком потока ключей.

* **Расшифрование**: Для расшифрования используется тот же поток ключей, что и для шифрования. Так как операция XOR обратима (т.е. при повторном применении той же последовательности ключей получаем исходные данные), процесс расшифрования аналогичен процессу шифрования:

plaintext=ciphertext⊕keystreamplaintext = ciphertext \oplus keystreamplaintext=ciphertext⊕keystream

**6. Регенерация ключа (если нужно)**

Если требуется обработать больше данных, чем длина одного потока ключей (128 бит), генерируются новые блоки потока ключей. Этот процесс продолжается до тех пор, пока не будет зашифрована вся длина текста.

**Примерные шаги:**

1. Разбить исходный текст на блоки.
2. На каждом раунде обновить счетчики и регистры.
3. Вычислить поток ключей.
4. Выполнить XOR с блоками текста.
5. Продолжать процесс для всех блоков текста.

**7. Завершение**

Когда весь текст зашифрован, процесс завершается. При расшифровании данные поступают обратно через тот же алгоритм, с тем же ключом и IV, если используется, что гарантирует правильное восстановление исходного текста.

**Особенности Rabbit:**

* **Высокая скорость**: Rabbit изначально проектировался как быстрый потоковый шифр, и его архитектура оптимизирована для работы на платформах с низкими вычислительными ресурсами.
* **Безопасность**: Несмотря на свою простоту, Rabbit доказал свою стойкость на протяжении многих лет.
* **Малый размер**: Шифр занимает относительно мало места, как в памяти, так и по объему ключа.

Алгоритм Rabbit не является стандартом для потоковых шифров, таких как RC4, но все еще используется в некоторых приложениях благодаря его производительности и криптографической стойкости.

Пример реализации:

Enter text for encrypting:

i like crypography

Enter secret key:

secret-key data

x: 98530481 -755809617 46740207 604139324 972915202 93861803 448836655 183945266

UPDATE STATE

g: 41555190622081753466252958-570525878-755396238-958481423-589075794-192466487

newX[0]-730400506 newX[1]-179934281 newX[2]-961117236 newX[3]-605018337 newX[4]636832697 newX[5]298092309 newX[6]-194163567 newX[7]884020178 Reset c

c: 12953075975453926721431655764-15680039351977048436-14316557682113396603-1295307601

UPDATE STATE

g: 605334757-356944269-763857481824225402540476069-658299978673776319158975850

newX[0]-96192962 newX[1]-142917892 newX[2]-167200518 newX[3]966293216 newX[4]-860943029 newX[5]-712701728 newX[6]570883315 newX[7]882881406 Reset c

c: 12953075931090785340-886263100-115895943868174073-477218603886263075477218566

UPDATE STATE

g: 51300283409075233460098189396159007876406973363749692168787129-177420138

newX[0]885494005 newX[1]398540797 newX[2]-341314039 newX[3]563841629 newX[4]855429545 newX[5]-272906699 newX[6]-734442697 newX[7]325338670 Reset c

c: -1227133540-886263125-886263133-749914974-1431655826-1022611337-886263187477218471

UPDATE STATE

g:624866051-92527227016075367091908904626820170910894245359771710228872921

newX[0]36207015 newX[1]-879050703 newX[2]-8233067 newX[3]-919312319 newX[4]-301484591 newX[5]787541647 newX[6]-406580569 newX[7]859383233 Reset c

c: 545392528-1090785522-1090785563-54539294015680036051431655360-2045227521158958811

UPDATE STATE

g: 441121866-58812327-543018472281407821-150840612628297873712542507910845114

newX[0]591260415 newX[1]857190606 newX[2]53991172 newX[3]-915823739 newX[4]-693329911 newX[5]-631338230 newX[6]104514567 newX[7]-249377535 Reset c

c: -129530836011589584899544360181704350675-1772527941545390511-4090471661636174737

UPDATE STATE

g: -985173869-488767952-4213860228499153-22458522288864098-682550497310204407

newX[0]-813499624 newX[1]-598321279 newX[2]765227336 newX[3]885953 newX[4]-863022462 newX[5]-530574371 newX[6]793222413 newX[7]-545771542 Reset c

c: 16361739742045217538-409050648-1704359672681647581499818361340856270-1431673197

UPDATE STATE

g: 999996368559016438-442888267-163820284-577881208-340171775784768241-503041669

newX[0]632972159 newX[1]612528677 newX[2]-160241682 newX[3]698558054 newX[4]555154498 newX[5]766156541 newX[6]-258782728 newX[7]411631049 Reset c

c: 1499808374-1499856309-1022643865-1431695940-2113446107272635346-136423309-681833414

UPDATE STATE

g: -390046487724717990529982386514547260-738648134-307736280-674224859-408488058

newX[0]346164703 newX[1]910312832 newX[2]316475512 newX[3]536289381 newX[4]-942170552 newX[5]-555366568 newX[6]938694238 newX[7]460503478 Reset c

c: 2113282557-136488677-272869450-4092577931022348471-211372038712949086751499338353

UPDATE STATE

g: 768824102-555463433-289577220-245109824899287419216006851-448203566225273186

newX[0]324467626 newX[1]-487591559 newX[2]-320774086 newX[3]324839679 newX[4]371952402 newX[5]202207656 newX[6]829342938 newX[7]346022198 Reset c

c: 612961211-27344239133995125112260010551498434601270977306815971056-1093394795

UPDATE STATE

g: -558528719-732821594-495862217-889188861-690749013-902328968263496511449443107

newX[0]-654555846 newX[1]-26999058 newX[2]700467199 newX[3]589664284 newX[4]380006421 newX[5]747675346 newX[6]464729614 newX[7]145402172 Reset c

c: 814874013-2084833031017731040-755927604-7407928114983247012158886011008756898

UPDATE STATE

g: 377328562635928518-759722521592247732790675577-503378486415944240-47537024

newX[0]979570883 newX[1]-104074715 newX[2]-13020843 newX[3]285951840 newX[4]635850901 newX[5]952883366 newX[6]-104695986 newX[7]630071933 Reset c

c: -1176028788-2134427016-230432884308947109-44837574415877198182053693494-346253812

UDATE STATE

g: 214771458973318691792875753-691404212310430522534615995-210960592-281781821

newX[0]291963150 newX[1]184388846 newX[2]297209319 newX[3]154305885 newX[4]-162139116 newX[5]499019053 newX[6]-911037483 newX[7]932615080 Reset c

c: -22697500311836503521839480560-851232030-20495226994244602111728238780-2026719236

UPDATE STATE

g: 810164525275235641-514422599-242033846-135691909261851862715168987-462993519

newX[0]577411434 newX[1]-447495230 newX[2]-204976506 newX[3]237393235 newX[4]216946599 newX[5]-66345372 newX[6]902950193 newX[7]87810747 Reset c

c: -958386642-524651215-2093874859-1649799292-15426962011564536832134777538994321394

UPDATE STATE

g: 299620676449656578836042625203201028-562096989240425477815948560-560243777

newX[0]-873777250 newX[1]-731765608 newX[2]690056365 newX[3]-837326270 newX[4]286373465 newX[5]882663957 newX[6]-995852586 newX[7]-700842256 Reset c

c: 133124234956676209-1150935558-15054272531885355498-3668950231017967590-1396415220

UPDATE STATE

g: 506138772-375671216116983809778912589544937783-89059021876524447-470371272

newX[0]150884893 newX[1]-935558565 newX[2]-185211738 newX[3]689326764 newX[4]896710748 newX[5]249909276 newX[6]-646519853 newX[7]-580870432 Reset c

c: 1230134726536896010272223544621038881197544461171691253019849651951474813067

UPDATE STATE

g: -918686931596861124-371774898-427685457-381925236-131665922673759579-40144543

newX[0]-37913119 newX[1]-789207126 newX[2]87287242 newX[3]-787234230 newX[4]-13855069 newX[5]737705572 newX[6]-872396216 newX[7]787033062 Reset c

c: -294711906-5077308216507558152008167300-1839170460764005162199905543265164295

UPDATE STATE

g: 38469646801806921281916303-812674628-28903743-297182035310041397-435004149

newX[0]267756968 newX[1]-952061966 newX[2]352202162 newX[3]125044696 newX[4]465829626 newX[5]777077814 newX[6]-287988981 newX[7]786846189 Reset c

c: 1065759986-19188576013451331473536407482059522659-5851763836639641241615391511

UPDAT STATE

g: -886944784479648235-392106343-442014216-31962361-260541263155918065376360772

newX[0]39657940 newX[1]-154125283 newX[2]875267556 newX[3]-526202204 newX[4]326071554 newX[5]-14898585 newX[6]-747014829 newX[7]557911116 Reset c

c: -318508202-1260308887971087352-1674931599-365323865-64237156-150187957-1943500650

UPDATE STAT

g: -931548651-940330633313723193-383336438192720794-715755893-469723349656987949

newX[0]520193177 newX[1]-273189555 newX[2]-793508691 newX[3]-665724323 newX[4]-339896546 newX[5]919084919 newX[6]-744124782 newX[7]-152754639 Reset c

c: -9667012551318042229-1119574623-14991986251680529881-17924114791602452935545215377

UPDTE STATE

g: -876819668315069664375919983-319270730-126025068270090530-304264554585781117

newX[0]-552835183 newX[1]281918302 newX[2]203718625 newX[3]616132080 newX[4]-832531433 newX[5]773067876 newX[6]819350097 newX[7]559071156 Reset c

c: 873821719144194902312086374921004746464193536142010292130331881751043-981737784

UPDATE STATE

g: -909170566-890135077592539252671829805-595017799-615195520-588624843734556815

newX[0]-906231194 newX[1]44409658 newX[2]-91688765 newX[3]-122632589 newX[4]-316788500 newX[5]-386561571 newX[6]245372794 newX[7]434919780 Reset c

c: 11873915321879425630-3206410821979412979-113010782278536830319172044211821729729

UPDATE STATE

g: -466135602-717514376-97906198355417969173019473520392526-659175319737283073

newX[0]-583157663 newX[1]203312915 newX[2]-549255054 newX[3]-270968003 newX[4]-16205402 newX[5]474045707 newX[6]42036941 newX[7]217302729 Reset c

c: 946156211389722671704594277684347557-1195675190475956205164324570156271226

Encrypted: [105, 32, 108, 105, 107, 101, 32, 99, 114, 121, 112, 111, 103, 114, 97, 112, 104, 121]

UPDATE STATE

g: -432014424300128189180511384-857227414-799449889-843585631324466793318387162

newX[0]779991999 newX[1]-139074544 newX[2]-33604970 newX[3]784890153 newX[4]140696277 newX[5]235943470 newX[6]-327982846 newX[7]-236665703 Reset c

c: 13610403851750097727459878002025642954800528391442272136-1959364384-1016830066

UPDATE STATE

g: 527465025106739880936112350-801890613764061627319723474790672482534834495

newX[0]-167691331 newX[1]-697248346 newX[2]-324135383 newX[3]-367887090 newX[4]24791867 newX[5]693843130 newX[6]-491235550 newX[7]-397181786 Reset c

c: 1639517916-1655266578-723015686-16970324311928072779-3835928915473286981416761724

UPDATE STATE

g: 337970457759431483-193508627495082126-696085041-76715482518387675756064974

newX[0]913201635 newX[1]861562690 newX[2]368383459 newX[3]632027640 newX[4]144174353 newX[5]227256659 newX[6]971718600 newX[7]-192751921 Reset c

c: 56619941194640573421096531401707928306-1408881136-560977333236436440-1755506040

UPDATE STATE

g: 599809085-553781515-564473181306790119170329485479408443187111011-644280336

newX[0]410980645 newX[1]802704529 newX[2]-772424833 newX[3]610808284 newX[4]71750169 newX[5]236035727 newX[6]-823968802 newX[7]-362999905 Reset c

c: -403578502792912307-506138757-1797870150338301085663586844150108359-719134589

UPDATE STATE

g: 5862456771406122784912929027010977-917208776938969339-474817395-449143332

newX[0]-776203885 newX[1]477842677 newX[2]-944584707 newX[3]224131621 newX[4]814398761 newX[5]-780469649 newX[6]924665811 newX[7]-649399212 Reset c

c: 17259450621559188859571622393153670-3184601701231389766631583200798711703

UPDATE STATE

g: 435266412466545015-944127251-694928724-739159652-316630167345364106485353495

newX[0]246794664 newX[1]26822985 newX[2]-250390801 newX[3]-43639494 newX[4]-222637448 newX[5]-148664186 newX[6]-233616206 newX[7]218026972 Reset c

c: -20283534901732290769-1080697212307764055-76061104013570418181238710093-1371282408

UPDATE STATE

g: -221043100-954872111-646626372-754706503-178350820-754453457-165149242-397062958

newX[0]-536108963 newX[1]-59482320 newX[2]296079041 newX[3]-320206909 newX[4]289977806 newX[5]391855909 newX[6]348920135 newX[7]-68085143 Reset c

c: -2104328301-1121952457-1316386577286685075-1223840890101946402015082591881023239872

UPDATE STATE

g: -5719192358655537299759330-472368358-779034371-299605038284614553781229958

newX[0]465069586 newX[1]163902613 newX[2]578577448 newX[3]-97703725 newX[4]201566859 newX[5]-26945279 newX[6]749074 newX[7]426395647 Reset c

c: 214219168-1657648214-2087771699-5057790271815432454-5738077301845365332094039497

UPDTE STATE

g: 138469160-848359173-9588002114596783-658047036-236954091936849847853742717

newX[0]349495851 newX[1]-615834765 newX[2]-767140416 newX[3]-376330869 newX[4]922801812 newX[5]979220350 newX[6]36738905 newX[7]-338030533 Reset c

c: -6914010341196003123-55054847840230861849540615-21329713191596617585281952878

UPDATE STATE

g: -283546420426920175533234315685312672841158901-338962874-446991407-754229076

newX[0]-855694629 newX[1]987339492 newX[2]-125369737 newX[3]763183175 newX[4]16772374 newX[5]850094214 newX[6]616442191 newX[7]-607238598 Reset c

c: 8858594411331947639-2082262049-29313661096694324-1500139036966887571864904727

UPDATE STATE

g: 534183647-640855113-758420844209105383-198402469-107751111-990282075-350213500

newX[0]908126097 newX[1]161677399 newX[2]698593578 newX[3]-771032935 newX[4]-494532614 newX[5]-650642164 newX[6]866074541 newX[7]717887478 Reset c

c: -248895531333137183-86286177442951445777629385615125430451459316877-84482600

UPDATE STATE

g: -884339818-39043532-348595977-812199394-879694637569164607182180652-597564072

newX[0]714651564 newX[1]-982879198 newX[2]-538498946 newX[3]264699909 newX[4]887792284 newX[5]-875044248 newX[6]-830082112 newX[7]-402734993 Reset c

c: 961929466545151724568553042-2001592200-197521326942359286811329948201934775312

UPDATE STATE

g: -299237498-250328381651300943-512205767-514369406-913720537693025435-266717305

newX[0]310473340 newX[1]-474037546 newX[2]4054280 newX[3]-253120039 newX[4]-6512589 newX[5]-388722087 newX[6]131194468 newX[7]-22686886 Reset c

c: -102954921-30771812211470980124408134092010652511-974458825-591378930-2065307822

UPDATE STATE

g: -517486400-809540330759208008918346436-858181979-597378529-311855004-923583933

newX[0]514454045 newX[1]-859208739 newX[2]-926960214 newX[3]596136846 newX[4]699656104 newX[5]273691853 newX[6]-376783561 newX[7]-794385662 Reset c

c: -872955146-19305881931027729111838893917-1195335793-12835315261670141915491097185

UPDATE STATE

g: -976517276-213526072-213730810-943896653-48066622800461017605033230267177219

newX[0]-234998315 newX[1]-511681232 newX[2]488886638 newX[3]651541509 newX[4]643584707 newX[5]-473197124 newX[6]732574354 newX[7]-954643527 Reset c

c: 913449636-1767053482-778017479-193878326141093331713664883933891873-1983715146

UPDATE STATE

g: -142714981610357329399210555317542145-860894527-904036949-796324662-607474878

newX[0]654901940 newX[1]-375744023 newX[2]-33684661 newX[3]-57564349 newX[4]579891709 newX[5]-686004356 newX[6]-775296086 newX[7]-307320858 Reset c

c: 22504208720030409762111286589146781092511288293172028757292-20822330561115282186

UPDATE STATEg: 94505124898312298649882376-346319754314030418714946823-937844527-464624681

newX[0]-440768516 newX[1]-682657968 newX[2]-526253190 newX[3]-924278710 newX[4]957928566 newX[5]542434822 newX[6]733985 newX[7]331691237 Reset c

c: -1659335426-406209375-170362699010594915321438405924584590121521278327-772143691

UPDATE STATEg: 45455082-994129064-935925268-527997749-374979341641584104-676324347900123350

newX[0]693081965 newX[1]215622470 newX[2]-890049374 newX[3]996529770 newX[4]206002587 newX[5]439867621 newX[6]946421242 newX[7]-131708809 Reset c

c: -113617152020026714761185307578-754860589-6636959087835251416497159161763835317

Decrypted: i like crypography Test passed: true

**II)Whirlpool**

**Whirlpool** — [криптографическая хеш-функция](https://xn--h1ajim.xn--p1ai/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8), разработанная [Винсентом Рэйменом](https://xn--h1ajim.xn--p1ai/%D0%92%D0%B8%D0%BD%D1%81%D0%B5%D0%BD%D1%82_%D0%A0%D1%8D%D0%B9%D0%BC%D0%B5%D0%BD) и [Пауло Баррето](https://xn--h1ajim.xn--p1ai/index.php?title=%D0%91%D0%B0%D1%80%D1%80%D0%B5%D1%82%D0%BE,_%D0%9F%D0%B0%D1%83%D0%BB%D0%BE&action=edit&redlink=1)[[англ.]](https://en.wikipedia.org/wiki/Paulo_S._L._M._Barreto). Опубликована в ноябре [2000 года](https://xn--h1ajim.xn--p1ai/2000_%D0%B3%D0%BE%D0%B4). [Хеширует](https://xn--h1ajim.xn--p1ai/%D0%A5%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Хеширование) входное [сообщение](https://xn--h1ajim.xn--p1ai/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5) с длиной до 2^256 битов. Выходное значение [хеш-функции](https://xn--h1ajim.xn--p1ai/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) Whirlpool, называемое [хешем](https://xn--h1ajim.xn--p1ai/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0), составляет 512 битов.

## **Описание**

Основная версия — хеш-функции — третья; в отличие от первой версии, [S-box](https://xn--h1ajim.xn--p1ai/S-box) определён, а диффузная матрица заменена на новую после доклада Сирая и Сибутани[[1]](https://xn--h1ajim.xn--p1ai/Whirlpool_(%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F)" \l "cite_note-Shirai&Shibutani-1).

Whirlpool состоит из повторного применения [функции сжатия](https://xn--h1ajim.xn--p1ai/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D1%8F%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F), основой которой является специальный 512-битный [блочный шифр](https://xn--h1ajim.xn--p1ai/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) 𝑊 с 512-битным ключом.

лгоритм хеширования **Whirlpool** — это криптографическая хеш-функция, которая преобразует входные данные в фиксированный 512-битный (64 байта) хеш. Теперь опишем процесс хеширования пошагово, как это происходит в Whirlpool.

**Шаг 1. Инициализация**

1. **Начальное состояние (State)**: Перед началом хеширования Whirlpool инициализирует состояние хеш-функции — массив 512 бит (64 байта), заполненный нулями:

makefile

Копировать код

State = 0x0000000000000000000000000000000000000000000000000000000000000000

**Шаг 2. Подготовка данных (Padding)**

1. **Выравнивание данных (Padding)**: Если длина входного сообщения не кратна 512 битам (64 байтам), то данные дополняются так, чтобы их длина стала кратной 512 битам. Дополнение добавляется следующим образом:
   * К исходным данным добавляется 1 бит, затем несколько 0, и последние 64 бита хранят длину исходного сообщения.

Например, для короткого сообщения Hello, World! добавляются нули и информация о длине.

**Шаг 3. Разбиение на блоки**

1. **Разбиение сообщения на блоки**: После подготовки данные разбиваются на блоки по 512 бит (64 байта). Каждый блок будет обрабатываться отдельно. Например, если сообщение слишком длинное, то оно разбивается на несколько блоков по 512 бит.

**Шаг 4. Инициализация раундовых ключей**

1. **Генерация раундовых ключей**: Whirlpool использует **10 раундов**, каждый из которых включает генерацию ключей на основе входного блока данных. В каждом раунде используются уникальные ключи, которые создаются на основе текущего состояния и констант.

**Шаг 5. Основные шаги каждого раунда**

Каждый раунд состоит из четырех операций: **SubBytes**, **ShiftColumns**, **MixRows** и **AddRoundKey**. Рассмотрим их по очереди.

**1. SubBytes**

* **Подстановка байтов**: Каждый байт данных (из 512-битного блока) проходит через **S-бокс** — таблицу замены, которая заменяет каждый байт на другой байт из заранее заданной таблицы. Это операция добавляет нелинейность к хешированию.

**2. ShiftColumns**

* **Циклический сдвиг колонок**: Каждая строка блока данных сдвигается циклически на определённое количество байтов (аналогично операции в AES):
  + Первая строка не сдвигается.
  + Вторая строка сдвигается на 1 байт влево.
  + Третья строка — на 2 байта и так далее.

**3. MixRows**

* **Линейное перемешивание строк**: В этой стадии каждая строка данных умножается на заранее определённую матрицу (с использованием арифметики конечных полей), чтобы перемешать байты внутри строки. Это обеспечивает распространение изменения по всей строке и затрудняет восстановление исходных данных.

**4. AddRoundKey**

* **Добавление раундового ключа**: На этом этапе к каждому байту блока данных прибавляется (по модулю 2) соответствующий байт раундового ключа, который был сгенерирован в начале этого раунда. Это позволяет добавить элемент, зависящий от текущего состояния.

**Шаг 6. Повторение шагов**

1. **Повторение раундов**: Операции SubBytes, ShiftColumns, MixRows и AddRoundKey повторяются **10 раз** (10 раундов) для каждого блока данных. Это гарантирует, что даже небольшие изменения в сообщении или ключе приводят к значительным изменениям в финальном хеше.

**Шаг 7. Хеширование всех блоков**

1. **Обновление состояния (State)**: После завершения раундов для каждого блока данных, состояние обновляется путём добавления результата раунда к текущему состоянию (обычно с помощью операции XOR).
2. **Переход к следующему блоку**: Если входное сообщение состоит из нескольких блоков (например, сообщение длиннее 512 бит), алгоритм переходит к следующему блоку и повторяет раунды.

**Шаг 8. Финальная обработка**

1. **Получение финального хеша**: После обработки всех блоков, текущее состояние является хешем исходного сообщения. Он имеет длину 512 бит (64 байта).

**Резюме процесса хеширования Whirlpool:**

1. Подготовка и выравнивание данных (padding).
2. Разбиение сообщения на блоки по 512 бит.
3. Инициализация состояния и раундовых ключей.
4. Прохождение через 10 раундов для каждого блока данных.
   * Подстановка байтов (SubBytes).
   * Циклический сдвиг колонок (ShiftColumns).
   * Линейное перемешивание строк (MixRows).
   * Добавление раундового ключа (AddRoundKey).
5. Обновление состояния после каждого блока.
6. Финальная обработка и получение 512-битного хеша.

Алгоритм Whirlpool разработан так, чтобы быть устойчивым к криптографическим атакам и обеспечивать равномерное распределение хешей даже при небольших изменениях входных данных.

### Пример работы:

### Enter data to be hashed: i like cryptography!

### Mix Columns - Column 0: New State: [[60, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0], [115, 0, 0, 0, 0, 0, 0, 0], [121, 0, 0, 0, 0, 0, 0, 0], [64, 0, 0, 0, 0, 0, 0, 0], [184, 0, 0, 0, 0, 0, 0, 0], [156, 0, 0, 0, 0, 0, 0, 0], [54, 0, 0, 0, 0, 0, 0, 0]]

### Mix Columns - Column 1: New State: [[60, 49, 0, 0, 0, 0, 0, 0], [0, 125, 0, 0, 0, 0, 0, 0], [115, 124, 0, 0, 0, 0, 0, 0], [121, 14, 0, 0, 0, 0, 0, 0], [64, 61, 0, 0, 0, 0, 0, 0], [184, 45, 0, 0, 0, 0, 0, 0], [156, 111, 0, 0, 0, 0, 0, 0], [54, 50, 0, 0, 0, 0, 0, 0]]

### Mix Columns - Column 2: New State: [[60, 49, 62, 0, 0, 0, 0, 0], [0, 125, 102, 0, 0, 0, 0, 0], [115, 124, 105, 0, 0, 0, 0, 0], [121, 14, 63, 0, 0, 0, 0, 0], [64, 61, 30, 0, 0, 0, 0, 0], [184, 45, 38, 0, 0, 0, 0, 0], [156, 111, 114, 0, 0, 0, 0, 0], [54, 50, 42, 0, 0, 0, 0, 0]]

### Mix Columns - Column 3: New State: [[60, 49, 62, 59, 0, 0, 0, 0], [0, 125, 102, 99, 0, 0, 0, 0], [115, 124, 105, 106, 0, 0, 0, 0], [121, 14, 63, 20, 0, 0, 0, 0], [64, 61, 30, 35, 0, 0, 0, 0], [184, 45, 38, 51, 0, 0, 0, 0], [156, 111, 114, 97, 0, 0, 0, 0], [54, 50, 42, 54, 0, 0, 0, 0]]

### Mix Columns - Column 4: New State: [[60, 49, 62, 59, 56, 0, 0, 0], [0, 125, 102, 99, 52, 0, 0, 0], [115, 124, 105, 106, 7, 0, 0, 0], [121, 14, 63, 20, 5, 0, 0, 0], [64, 61, 30, 35, 4, 0, 0, 0], [184, 45, 38, 51, 12, 0, 0, 0], [156, 111, 114, 97, 88, 0, 0, 0], [54, 50, 42, 54, 54, 0, 0, 0]]

### Mix Columns - Column 5: New State: [[60, 49, 62, 59, 56, 61, 0, 0], [0, 125, 102, 99, 52, 121, 0, 0], [115, 124, 105, 106, 7, 120, 0, 0], [121, 14, 63, 20, 5, 10, 0, 0], [64, 61, 30, 35, 4, 57, 0, 0], [184, 45, 38, 51, 12, 57, 0, 0], [156, 111, 114, 97, 88, 99, 0, 0], [54, 50, 42, 54, 54, 58, 0, 0]]

### Mix Columns - Column 6: New State: [[60, 49, 62, 59, 56, 61, 34, 0], [0, 125, 102, 99, 52, 121, 122, 0], [115, 124, 105, 106, 7, 120, 109, 0], [121, 14, 63, 20, 5, 10, 19, 0], [64, 61, 30, 35, 4, 57, 50, 0], [184, 45, 38, 51, 12, 57, 58, 0], [156, 111, 114, 97, 88, 99, 126, 0], [54, 50, 42, 54, 54, 58, 42, 0]]

### Mix Columns - Column 7: New State: [[60, 49, 62, 59, 56, 61, 34, 55], [0, 125, 102, 99, 52, 121, 122, 119], [115, 124, 105, 106, 7, 120, 109, 118], [121, 14, 63, 20, 5, 10, 19, 8], [64, 61, 30, 35, 4, 57, 50, 55], [184, 45, 38, 51, 12, 57, 58, 55], [156, 111, 114, 97, 88, 99, 126, 109], [54, 50, 42, 54, 54, 58, 42, 62]]

### Updated State: [[60, 49, 62, 59, 56, 61, 34, 55], [0, 125, 102, 99, 52, 121, 122, 119], [115, 124, 105, 106, 7, 120, 109, 118], [121, 14, 63, 20, 5, 10, 19, 8], [64, 61, 30, 35, 4, 57, 50, 55], [184, 45, 38, 51, 12, 57, 58, 55], [156, 111, 114, 97, 88, 99, 126, 109], [54, 50, 42, 54, 54, 58, 42, 62]]

### State: [[85, 67, 78, 59, 56, 61, 34, 55], [32, 4, 14, 99, 52, 121, 122, 119], [31, 12, 16, 106, 7, 120, 109, 118], [16, 122, 30, 20, 5, 10, 19, 8], [43, 82, 31, 35, 4, 57, 50, 55], [221, 74, 38, 51, 12, 57, 58, 55], [188, 29, 114, 97, 88, 99, 126, 109], [85, 83, 42, 54, 54, 58, 42, 158]]

### Flattened:[85, 67, 78, 59, 56, 61, 34, 55, 32, 4, 14, 99, 52, 121, 122, 119, 31, 12, 16, 106, 7, 120, 109, 118, 16, 122, 30, 20, 5, 10, 19, 8, 43, 82, 31, 35, 4, 57, 50, 55, 221, 74, 38, 51, 12, 57, 58, 55, 188, 29, 114, 97, 88, 99, 126, 109, 85, 83, 42, 54, 54, 58, 42, 158]

### Whirlpool Hash: 55434e3b383d223720040e6334797a771f0c106a07786d76107a1e14050a13082b521f2304393237dd4a26330c393a37bc1d726158637e6d55532a36363a2a9e