Chave

Uma chave é um pedaço de informação que controla a operação de um algoritmo de [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia). Na codificação, uma chave especifica a transformação do texto puro em [texto cifrado](https://pt.wikipedia.org/w/index.php?title=Texto_cifrado&action=edit&redlink=1" \o "Texto cifrado (página não existe)), ou vice-versa, durante a decodificação. Chaves são também usadas em outros algoritmos criptográficos, tais como esquemas de assinatura digital e funções hash (também conhecidas como [MAC](https://pt.wikipedia.org/w/index.php?title=Message_authentication_code&action=edit&redlink=1" \o "Message authentication code (página não existe))), algumas vezes para [autenticação](https://pt.wikipedia.org/wiki/Autentica%C3%A7%C3%A3o" \o "Autenticação).

Para um algoritmo bem projetado, cifrar o mesmo texto mas com uma chave diferente deverá produzir um texto cifrado totalmente diferente. Igualmente, decifrar o texto cifrado com a chave errada deverá produzir um texto aleatório ininteligível. Se a chave de decriptação for perdida, o dado cifrado praticamente não pode ser recuparado pelo mesmo algoritmo de criptografia.

As chaves usadas na criptografia de chave pública têm uma certa estrutura matemática. Por exemplo, as chaves públicas usadas no sistema [RSA](https://pt.wikipedia.org/wiki/RSA" \o "RSA) são o produto de dois [números primos](https://pt.wikipedia.org/wiki/N%C3%BAmero_primo" \o "Número primo). Por isso, sistemas de chave pública requerem chaves maiores do que os sistemas simétricos para um nível equivalente de segurança. 3072bits é o tamanho de chave sugerido para sistemas baseados em [fatoração](https://pt.wikipedia.org/wiki/Fatora%C3%A7%C3%A3o" \o "Fatoração) e [algoritmos discretos](https://pt.wikipedia.org/w/index.php?title=Algoritmo_discreto&action=edit&redlink=1" \o "Algoritmo discreto (página não existe)) inteiros que visam ter segurança equivalente a da cifra simétrica de 128bits. A [criptografia de curva elíptica](https://pt.wikipedia.org/wiki/Criptografia_de_curva_el%C3%ADptica" \o "Criptografia de curva elíptica) (CCE) pode permitir chaves de tamanhos menores para uma segurança equivalente, mas estes algoritmos são conhecidos há pouco tempo e, pelas estimativas atuais para a dificuldade de se encontrar suas chaves, eles não devem sobreviver. Recentemente, uma mensagem codificada usando uma chave de 109bits do algoritmo de curva elíptica foi quebrada por [força bruta](https://pt.wikipedia.org/wiki/For%C3%A7a_bruta" \o "Força bruta). A regra atual é usar uma chave de CCE com o dobro da segurança da chave simétrica para o nível desejado. Exceto para o *one time pad* aleatório, a segurança desses sistemas não foi provada matematicamente. Portanto, um ponto fraco teórico poderia fazer de tudo que você codificou um livro aberto. Esta é uma outra razão para se valorizar a escolha de chaves longas.

## A escolha da chave

Para evitar que uma chave seja adivinhada, as chaves precisam ser geradas aleatoriamente e conterem [entropia](https://pt.wikipedia.org/wiki/Teoria_da_informa%C3%A7%C3%A3o" \o "Teoria da informação) suficiente. O problema de como gerar seguramente chaves verdadeiramente aleatórias é difícil e tem sido encarado de várias formas por vários sistemas criptográficos. Existe um [RFC](https://pt.wikipedia.org/wiki/RFC" \o "RFC) sobre a geração de aleatoriedade ([RFC 1750](https://tools.ietf.org/html/rfc1750), *Randomness Recommendations for Security* [*Recomendações de Aleatoriedade para Segurança*]). Alguns sistemas operacionais incluem ferramentas para "coletar" entropia a partir da medição do tempo de certas operações imprevisíveis como os movimentos da cabeça do [drive](https://pt.wikipedia.org/wiki/Drive" \o "Drive) de disco. Para a produção de pequenas quantidades de material, um [dado](https://pt.wikipedia.org/wiki/Dado" \o "Dado) comum é uma boa fonte de aleatoriedade de alta qualidade.

Quando uma [senha](https://pt.wikipedia.org/wiki/Senha" \o "Senha) é usada como chave de codificação, os sistemas de criptografia bem projetados primeiro usam um algoritmo de derivação da chave, que adiciona um "[sal](https://pt.wikipedia.org/wiki/Sal_(criptografia)" \o "Sal (criptografia))" (salt) e o reduz ou o expande para o tamanho de chave desejado, por exemplo, através da redução de uma frase longa a um valor de 128bits apropriado para o uso em um [bloco cifrado](https://pt.wikipedia.org/w/index.php?title=Bloco_cifrado&action=edit&redlink=1" \o "Bloco cifrado (página não existe)).

Em [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia), **sal** (**salt** em inglês) é um dado [aleatório](https://pt.wikipedia.org/w/index.php?title=Gera%C3%A7%C3%A3o_de_n%C3%BAmero_aleat%C3%B3rio&action=edit&redlink=1" \o "Geração de número aleatório (página não existe)) que é usado como uma entrada adicional para uma [função unidirecional](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_de_m%C3%A3o_%C3%BAnica" \o "Função de mão única) que "[quebra](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica)" os [dados](https://pt.wikipedia.org/w/index.php?title=Dado_(computa%C3%A7%C3%A3o)&action=edit&redlink=1" \o "Dado (computação) (página não existe)), uma [senha](https://pt.wikipedia.org/wiki/Senha" \o "Senha) ou [frase-passe](https://pt.wikipedia.org/wiki/Frase-passe" \o "Frase-passe). Os sais são usados ​​para proteger as senhas no armazenamento. Historicamente, uma senha era armazenada em [texto simples](https://pt.wikipedia.org/wiki/Texto_simples" \o "Texto simples) em um sistema, mas com o tempo, salvaguardas adicionais foram desenvolvidas para proteger a senha de um usuário contra a leitura pelo sistema. O sal é um desses métodos.

Um novo sal é gerado aleatoriamente para cada senha. Em uma configuração típica, o sal e a senha (ou sua versão após o [alongamento de chave](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \o "Alongamento de chave)) são [concatenados](https://pt.wikipedia.org/wiki/Concatena%C3%A7%C3%A3o" \o "Concatenação) e processados ​​com uma [função hash criptográfica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica) e a saída resultante (mas não a senha original) é armazenada com o sal em um banco de dados. O hash permite a [autenticação](https://pt.wikipedia.org/wiki/Autentica%C3%A7%C3%A3o" \o "Autenticação) posterior sem manter, e, portanto, pôr em risco, a senha em [texto puro](https://pt.wikipedia.org/wiki/Texto_puro" \o "Texto puro), caso o armazenamento de dados de autenticação seja comprometido.

Sais permitem defender-se contra [ataques de dicionário](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio" \o "Ataque de dicionário) ou contra seu equivalente em hash, um ataque pré-computado de [tabela arco-íris](https://pt.wikipedia.org/wiki/Rainbow_table" \o "Rainbow table).[[1]](https://pt.wikipedia.org/wiki/Sal_(criptografia)" \l "cite_note-1) Como os sais não precisam ser memorizados por humanos, eles podem tornar o tamanho da tabela arco-íris necessário para um ataque bem-sucedido, proibitivamente grande sem sobrecarregar os usuários. Como os sais são diferentes em cada caso, eles também protegem as senhas comummente usadas ou os usuários que usam a mesma senha em vários sites, tornando todas as instâncias hash "salgadas" para a mesma senha, diferentes umas das outras.

Sais criptográficos são amplamente usados ​​em muitos sistemas de computadores modernos, desde credenciais do sistema [Unix](https://pt.wikipedia.org/wiki/Unix" \o "Unix) até [segurança na Internet](https://pt.wikipedia.org/w/index.php?title=Seguran%C3%A7a_na_Internet&action=edit&redlink=1" \o "Segurança na Internet (página não existe)).

Os sais estão intimamente relacionados ao conceito de um [nonce criptográfico](https://pt.wikipedia.org/wiki/Nonce" \o "Nonce).

## Implementações em Unix

Versões anteriores do Unix usavam um [gerenciador de senhas](https://pt.wikipedia.org/wiki/Passwd" \o "Passwd) (/etc/passwd) que guarda os hashes "salgados" das senhas (na prática, senhas com dois caracteres de sal pré-definidos randomicamente.) Nestas versões anteriores do Unix, o sal também era guardado no gerenciador (como um purotexto) junto com o hash da senha salgada. O gerenciador de senhas era legível para todos os usuários do sistema. Isto era necessário para que ferramentas de software privilegiados executados pelo usuário pudessem achar nomes de usuários e outras informações. A segurança das senhas é, portanto, garantida apenas pela função de mão-única (seja de [encriptação](https://pt.wikipedia.org/wiki/Encripta%C3%A7%C3%A3o" \o "Encriptação)ou de hash) usadas para este propósito.

Versões anteriores do Unix eram limitadas a senhas de 8 caracteres e usavam um sal com 12 bits, o que permitia alcançar 4.096 valores de sais possíveis. Enquanto 12 bits eram o suficiente para 1970s, mas em 2005 o armazenamento em disco tornou-se tão barato que um atacante poderia pré-computar os hashs de milhares de senhas comuns, incluindo todas as suas 4.096 possíveis variações quando concatenadas com o sal, e guardar estes valores em um único [disco rígido](https://pt.wikipedia.org/wiki/Unidade_de_disco_r%C3%ADgido" \o "Unidade de disco rígido). Um atacante com mais orçamento poderia guardar os valores pré-computados de todas as senhas com 6 caracteres e as senhas mais comuns com 7 e 8 caracteres, incluindo todas as suas 4.096 variantes com sal.

## Implementações em aplicações de Web

É comum uma aplicação de web armazenar em um [banco de dados](https://pt.wikipedia.org/wiki/Banco_de_dados" \o "Banco de dados)os valores de hash das senhas dos usuários. Sem um sal, um ataque bem sucedido de [Injeção de SQL](https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL" \o "Injeção de SQL) poderia acontecer facilmente, quebrando a segurança e integridade do banco de dados. Além disto, muitos usuários reutilizam suas senhas em múltiplos sites, então o uso do sal torna-se um importante componente geral de segurança de aplicações para Web. Afinal, cada aplicação terá gerado seu próprio sal, resultando em um hash diferente para a mesma senha.

## Benefícios

O sal, mesmo público, torna mais demorado quebrar uma lista de senhas. No entanto, isto não acontece quando um ataque de dicionário é executado em uma única senha. O atacante tem acesso a ambas as informações, senha e sal, então enquanto roda o ataque de dicionário, o atacante simplesmente pode usar o sal conhecido enquanto tenta quebrar a senha.

Para entender a diferença entre quebrar uma única senha e um conjunto delas, considere um único arquivo de senhas que contenha centenas de nomes de usuários e senhas. Sem o sal, um atacante pode computar hash(tentativa\_dicionario[0]) e então checar se este hash aparece em algum lugar do arquivo. A probabilidade de uma correspondência aumenta proporcionalmente ao número de senhas do arquivo. Se o sal está presente, então o atacante terá que computar hash(sal[a] . tentativa\_dicionario[0]), onde "." significa concatenação, comparar com a entrada A, e então computar hash(salt[b] . tentativa\_dicionario[0]), comparar com entrada B, e assim por diante. Isto derrota a reutilização de hash na tentativa de quebrar múltiplas senhas.

Sais também combatem o uso de [rainbow tables](https://pt.wikipedia.org/wiki/Rainbow_table" \o "Rainbow table) para quebrar senhas. Uma rainbow table é uma longa lista de hashes pré-computadas para as senhas mais comumente usadas. Para um gerenciador de senhas sem sal, um atacante pode ir através de cada entrada e verificar se ela corresponde com algum valor na tabela. Se a verificação for consideravelmente mais rápida que a função de hash (o que geralmente é), isto irá aumentar consideravelmente o tempo de quebra do gerenciador. No entanto, se o gerenciador de senhas utilizar sal, então a rainbow table terá de conter o hash pré-computado de "sal . senha". Se o sal for o suficientemente longo e o randômico, isto é muito improvável de ser feito. Senhas sem sal escolhidas por humanos tendem a ser vulneráveis a ataques de dicionário uma vez que eles precisam ser pequenas e claras o suficiente para serem memorizadas. Mesmo um pequeno dicionário (ou seus hashs equivalentes, uma rainbow table) tem chances grandes de quebrar as senhas mais comumente usadas. Devido aos sais não terem de ser memorizados por humanos, eles podem fazer com que o tamanho da rainbow table seja alto o suficiente para se tornar proibitivo sem ter que fazer o usuário da senha ter mais trabalho.

Mais tecnicamente, sais protegem contra rainbow tables porque eles, na prática, estendem exponencialmente a complexidade da senha. Se a rainbow table não tem senhas compatíveis com o tamanho (i.e. uma senha de 8-bytes com um sal de 2 bytes é, efetivamente, uma senha de 10 bytes) e com a complexidade (incluir caracteres não alfanuméricos aumentam a complexidade da senha) da senha com sal, então esta senha não será encontrada. Se for encontrada, então alguém terá que tirar o sal da senha antes que ele possa ser usado, até que o usuário entre novamente com sua senha e um novo sal possa ser gerado.

## Benefícios adicionais

O moderno sistema [shadow password](https://pt.wikipedia.org/wiki/Passwd" \o "Passwd), no qual o hash da senha e outros dados de segurança são guardados em um arquivo não público, de alguma forma atenua estas preocupações. No entanto, elas ainda continuam sendo relevantes em instalações multi-servidor onde usa-se uma senha central para ativar um gerenciador de senhas para ter acesso a senhas ou hashes de senhas para múltiplos sistemas. Nestas instalações, as contas raiz de cada sistema individual podem ser tratadas como menos confiáveis do que os administradores do sistema de senhas centralizada, então continua sendo útil garantir a segurança através de um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo) de hash de senhas, incluindo a geração de valores exclusivos de sais. [carece de fontes?]

Sais também fazem ataques de dicionário e [ataques de força bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta" \o "Ataque de força bruta) para quebrar um grande número de senhas vagarosos (mas não no caso de apenas uma senha). Sem sal, um atacante que está tentando quebrar muitas senhas ao mesmo tempo precisa apenas usar hash em um palpite de senha, e compará-lo com todos os outros. No entanto, com sal, cada senha provavelmente terá um sal diferente, o que torna consideravelmente mais devagar do que apenas comparar.

Outro benefício (menor) do sal segue: dois usuários podem escolher a mesma [string](https://pt.wikipedia.org/wiki/String" \o "String)para suas senhas, ou um mesmo usuário pode escolher usar a mesma senha em duas máquinas diferentes. Sem sal, a senha seria armazenada como a mesma hash de string no gerenciador de senhas. Se for divulgado que ambas as contas tem a mesma senha, isto permite que qualquer pessoa que conheça uma das senhas tenha acesso a outra conta. Salgando as senhas com caracteres aleatórios, mesmo se as duas contas usarem a mesma senha, ninguém poderá descobrir isto olhando apenas para o gerenciador de senhas, já que todas as entradas irão tender a ser diferentes devido ao fator de aleatoriedade.

## Exemplo prático

A listagem a seguir demonstra usuários e respectivas senhas que não são armazenadas no banco de dados, apenas sendo usadas para a autenticação.

A partir dessa técnica, mesmo que os dados do banco de dados estejam comprometidos quanto ao sigilo, o atacante não tem a senha revelada porque há apenas um hash guardado, uma vez que ele é apenas calculado pela aplicação em tempo de autenticação do usuário.

|  |  |  |  |
| --- | --- | --- | --- |
| **Usuario** | **Sal** | **Texto Codificado (Não armazenado)** | **Hash MD5Sum (Senha+Salt)** |
| usuario1 | E1F53135E559C253 | senha123+E1F53135E559C253 | 291d2ef346aa48a934a5aaa2616e1576 |
| usuario2 | 84B03D034B409D4E | senha321+84B03D034B409D4E | c6fba91002f0e8ad99e6dcd181e6f007 |

Em [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia), **alongamento de chave**(do inglês, *key stretching*), refere-se às técnicas usadas para fazer [chaves](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)) inseguras, tipicamente uma senha ou algo semelhante, mais seguras contra contra [ataque de força bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta" \o "Ataque de força bruta) ao aumentar o tempo que leva-se para testar cada possibilidade de chave. Senhas criadas por humanos costumam ser pequenas e previsíveis o suficiente para permitir a quebra de senha (do inglês, password cracking). Alongamento de chave torna este ataque mais difícil.

Alongamento de chave algumas vezes são referenciadas como "fortalecimento de chave (do inglês, key strengthening)", embora este último termo se refira à outra técnica com significantes diferenças nas propriedades de segurança e performance. (Veja sessão 6 do [[1]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-1) para uma comparação).

As técnicas de alongamento de chave geralmente funcionam da seguinte forma: a chave inicial serve de entrada para um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo) que tem como saída uma uma **chave reforçada**. A chave reforçada deverá ter tamanho o suficiente para tornar impossível de ser quebrada através da força bruta (pelo menos 128 bits por exemplo). Este algoritmo em sua forma geral deve ser capaz de ser seguro de maneira que não seja possível descobrir outra maneira de calcular a chave reforçada em menos tempo (menos trabalho de processador) que ele próprio.

O processo de alongamento de chave deixa o ataque com duas opções: ou ele tenta todas as possíveis combinações de chaves reforçadas (o que é impossível se a chave reforçada for grande o suficiente), ou então ele tenta todas as combinações da chave inicial. Neste último caso, se a chave inicial for uma senha, então o atacante poderá tentar usar primeiro o [ataque de dicionário](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio" \o "Ataque de dicionário), e caso não for bem sucedido, tentar todas as combinações de caracteres para a senha. O alongamento de senha não evita este caso, mas faz o atacante perder muito mais tempo em cada tentativa.

Se o atacante usa a mesma classe de hardware que o usuário, cada tentativa irá tomar o mesmo período de tempo de processo que o usuário ao entrar com sua senha (por exemplo, um segundo). Mesmo se um atacante tiver maiores recursos computacionais que o usuário, o alongamento de chave ainda irá atrasar o atacante, isto porque o usuário apenas computa a função de alongamento de chave uma vez, ao entrar com sua senha, enquanto que o atacante terá que fazer esta computação para cada palpite do seu ataque (de dicionário ou força bruta).

Existem muitas formas de fazer o alongamento de chave. Uma [função de hash criptográfica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica) ou um [bloco de cifra](https://pt.wikipedia.org/w/index.php?title=Bloco_de_cifra&action=edit&redlink=1" \o "Bloco de cifra (página não existe))podem ser repetidamente aplicados em um looping. Em aplicações onde a chave é usada para uma [cifragem](https://pt.wikipedia.org/wiki/Cifragem" \o "Cifragem), o esquema de chaves (do inglês, key schedule ou key set-up) em uma cifra de modo que leve um segundo para ser executada.

Uma técnica relacionada, a adição de [sal (criptografia)](https://pt.wikipedia.org/wiki/Sal_(criptografia)" \o "Sal (criptografia)), que protege contra ataques de balanceamento de tempo-memória (do inglês, time-memory tradoff), é usado frequentemente combinado com o alongamento de chave.

## Alongamento de chave baseado em hash

Muitas bibliotecas provém funções que executam o alongamento de chave como parte de sua funcionalidade. Veja [crypt(3)](https://pt.wikipedia.org/w/index.php?title=Crypt_(C)&action=edit&redlink=1" \o "Crypt (C) (página não existe)) como exemplo. Note que [PBKDF2](https://pt.wikipedia.org/w/index.php?title=PBKDF2&action=edit&redlink=1" \o "PBKDF2 (página não existe)) é para gerar uma chave de cifragem a partir de uma senha, e não necessariamente uma senha autenticada. PBKDF2 pode ser usada para ambos se o número de bits da saída for menor ou igual ao algoritmo interno de [hashing](https://pt.wikipedia.org/wiki/Tabela_de_dispers%C3%A3o" \o "Tabela de dispersão) no PBKDF2, que usualmente é o [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) (160 bits), ou é usado como uma chave de cifragem para cifrar dados estáticos.

## Resistência e tempo

Para estes exemplos assuma que o mais lento [computador pessoal](https://pt.wikipedia.org/wiki/Computador_pessoal" \o "Computador pessoal) usados atualmente (2011) pode fazer acerca de 65000 [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) hashes em um segundo usando um código [compilado](https://pt.wikipedia.org/wiki/Compilado" \o "Compilado). Então um programa que usa alongamento de chaves pode usar 65000 rodadas de hash e o tempo de espera do usuário não passará de muito mais que um segundo.

Testar uma senha tipicamente requer uma operação de hash. Mas se o alongamento de chave é usado, o atacante terá de computar uma chave reforçada para cada chave que que ele testar. Isto significa que são 65000 computações de hash por teste, aproximadamente 216, o que significa que uma chave reforçada vale um adicional de 16 bits de resistência na chave (veja [entropia da informação](https://pt.wikipedia.org/wiki/Entropia_da_informa%C3%A7%C3%A3o" \o "Entropia da informação)).

A comumente aceita [Lei de Moore](https://pt.wikipedia.org/wiki/Lei_de_Moore" \o "Lei de Moore) implica que a velocidade dos computadores dobram aproximadamente a cada um ano e meio. Assumindo isto, um bit a mais de resistência na chave é alcançado através da força bruta a cada 1.5 anos. Isto implica que 16 bits extras de resistência vale aproximadamente (16x1.5) 24 anos de atraso na quebra, mas isto também significa que o número de rodadas de alongamentos que um sistema usa deva ser dobrada aproximadamente a cada 1.5 anos para manter o mesmo nível de segurança. (Como a maioria das chaves são mais seguras que o necessário, sistemas que precisam de uma consistência determinística na geração de chaves provavelmente não atualizar o número de iterações usadas no alongamento de chaves. Nestes casos, o designer deve considerar o quão longo deseja-se que o sistema de derivação de chaves permaneça inalterado e escolher um número de hashes apropriado para o tempo de vida útil do sistema).

Funções de hash limitadas pela CPU podem ser vulneráveis a [ataques implementados em hardware](https://pt.wikipedia.org/w/index.php?title=Custom_hardware_attack&action=edit&redlink=1" \o "Custom hardware attack (página não existe)). Estas implementações de SHA-1 usam apenas 5000 gates e 400 ciclos de clock[[2]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-2). Com [FPGAs](https://pt.wikipedia.org/wiki/Field-programmable_gate_array" \o "Field-programmable gate array) com vários milhões de gates custando menos de $100[[3]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-3), um atacante pode construir um completo hardware de looping desenrolado (do inglês, [unrolled](https://pt.wikipedia.org/w/index.php?title=Loop_unwinding&action=edit&redlink=1" \o "Loop unwinding (página não existe))) por cerca de $5000. Este design, setado em 100 MHz de clock, pode testar cerca de 300.000 chaves por segundo. O atacante é livre para escolher uma boa relação de preço e custo. Por exemplo, um design de 150.000 chaves por segundo custará $2500. Isto não valerá muita coisa já que o alongamento de chave ainda retardará o atacante mesmo nesta situação; um design de $5000 atacando diretamente um hash SHA-1 a uma taxa de 300.000 chaves por segundo irá apenas produzir chaves reforçadas a uma taxa de (300.000/(216)) 4.6 chaves por segundo.

Para defender contra abordagens de hardware, [funções criptográficas limitantes pela memória](https://pt.wikipedia.org/w/index.php?title=Memory_bound_function&action=edit&redlink=1" \o "Memory bound function (página não existe))também têm sido propostas. Estes acessos a uma grande quantidade de memória ocorrem de maneira imprevisível, tornando os [caches](https://pt.wikipedia.org/wiki/Cache" \o "Cache)ineficazes. Uma vez que acessar grandes quantidades de memória com uma baixa latência é caro, isto pode dissuadir significantemente um atacante.

Existe uma fraqueza conhecida em algoritmos de alongamento de chave baseadas em hash que usam uma função de hash interativa. Este ataque é conhecido como ataque de estado transferível. O ataque envolve a transferência de estados anteriores dos hashes iterados diretamente para os métodos de transformação da próxima interação. Este ataque pode diminuir o tempo de alongar a chave em 80%-90% do tempo original de alongamento[[4]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-4). Ele foi implementado em SHA256[[5]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-5).

## História

O primeiro uso deliberado de funções de derivação de chaves lentas aconteceu em "[CRYPT](https://pt.wikipedia.org/w/index.php?title=CRYPT&action=edit&redlink=1" \o "CRYPT (página não existe))", descrito por [Robert Morris](https://pt.wikipedia.org/wiki/Robert_Morris_(criptografador)" \o "Robert Morris (criptografador)) em 1978 para encriptar senhas do [Unix](https://pt.wikipedia.org/wiki/Unix" \o "Unix)[[6]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-6). Este programa usava 25 iterações, um sal de 12 bits e uma variante de [DES](https://pt.wikipedia.org/wiki/DES" \o "DES) como uma sub-função (DES em si foi evitado como uma forma de evitar ataques ao hardware padrão do DES). Senhas foram limitadas ao máximo de oito caracteres [ASCII](https://pt.wikipedia.org/wiki/ASCII" \o "ASCII). Enquanto isto era visto como um grande avanço na época, CRYPTO agora é considerado inadequado. A quantidade de iterações, projetados para a era [PDP-11](https://pt.wikipedia.org/wiki/PDP-11" \o "PDP-11), é muito baixa, 12 bits de sal é um inconveniente, mas não para ataques de dicionário, e o limite de 8 caracteres evita o uso senhas mais resistentes.

Funções de derivação de chaves baseadas em senhas modernas, como o PBKDF2, usa hash criptográfico, como [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5)e SHA-1, um sal longo (exemplo: [64 bits](https://pt.wikipedia.org/wiki/64_bits" \o "64 bits)) e uma alta quantidade de iterações (geralmente 1000 ou mais).Em 2008, um algoritmo de fortalecimento de chave de memória intensiva [scrypt](https://pt.wikipedia.org/wiki/Scrypt" \o "Scrypt), foi introduzido com a intenção de limitar o uso de hardwares customizáveis altamente paralelos para aumentar a velocidade de teste da chave[[7]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-7)[[8]](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \l "cite_note-8).

Em [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia), um **nonce** é um número arbitrário que só pode ser usado uma vez. É basicamente uma palavra de uso único, daí o nome (N = Number (Número) e Once = Uma vez, em inglês), embora N seja de número, o processo também pode usar letras. Muitas vezes, é um [número aleatório](https://pt.wikipedia.org/wiki/N%C3%BAmero_aleat%C3%B3rio" \o "Número aleatório) ou [pseudoaleatório](https://pt.wikipedia.org/wiki/Pseudoaleat%C3%B3rio" \o "Pseudoaleatório) emitido em um protocolo de [autenticação](https://pt.wikipedia.org/wiki/Autentica%C3%A7%C3%A3o" \o "Autenticação) para garantir que as comunicações antigas não possam ser reutilizados em ataques de repetição. Eles podem também ser úteis como vectores de inicialização e em [função hash criptográfica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica).[[1]](https://pt.wikipedia.org/wiki/Nonce" \l "cite_note-1)

Em [criptoanálise](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \o "Criptoanálise) e [segurança de computadores](https://pt.wikipedia.org/wiki/Seguran%C3%A7a_de_computadores" \o "Segurança de computadores), um **ataque de dicionário** é um tipo de [ataque de força bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta" \o "Ataque de força bruta) destinado a burlar uma [cifra](https://pt.wikipedia.org/wiki/Cifra" \o "Cifra) ou mecanismo de autenticação com o objetivo de descobrir uma senha tentando centenas ou, algumas vezes, milhões de possibilidades, como por exemplo, palavras de um dicionário.[[1]](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio" \l "cite_note-1)

## Técnica

Um ataque de dicionário é caracterizado pela tentativa de todas as sequências de caracteres de uma lista predefinida normalmente derivada de uma lista de palavras como em um dicionário, daí o nome *ataque de dicionário*.

É diferente de um [ataque de força bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta" \o "Ataque de força bruta) onde uma grande parte do [campo de chaves](https://pt.wikipedia.org/wiki/Espa%C3%A7o_de_chave_(criptografia)" \o "Espaço de chave (criptografia)) é percorrido sistematicamente. Um ataque de dicionário somente tenta as possibilidades que se considera terem maior chance de sucesso.

Ataques de dicionário são bem sucedidos porque as pessoas tem a tendência de usar [senhas](https://pt.wikipedia.org/wiki/Senha" \o "Senha) curtas baseadas em palavras de uso comum ou senhas comuns, ou ainda variações simples dessas como por exemplo adicionar um dígito ou um caractér de pontuação.

Ataques de dicionário são relativamente fáceis de serem evitados: usando [frases-passe](https://pt.wikipedia.org/wiki/Frase-passe" \o "Frase-passe) ou escolhendo senhas que não sejam variações simples de palavras encontradas em dicionários comuns nem senhas comuns.[[2]](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio" \l "cite_note-2)

## Ataque precomputado / ataque rainbow table

É possível obter um bom [ganho de tempo por espaço](https://pt.wikipedia.org/wiki/Trade-off_Espa%C3%A7o-Tempo" \o "Trade-off Espaço-Tempo) [precomputando](https://pt.wikipedia.org/w/index.php?title=Precomputa%C3%A7%C3%A3o&action=edit&redlink=1" \o "Precomputação (página não existe)) uma lista de *[hashes](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica)* de palavras de dicionários e armazenando-a em um banco de dados usando o *hash* como [chave](https://pt.wikipedia.org/w/index.php?title=Chave_%C3%BAnica&action=edit&redlink=1" \o "Chave única (página não existe)). Iso exige um tempo considerável de preparação mas permite que o ataque em si seja executado com muito maior rapidez.[[3]](https://pt.wikipedia.org/wiki/Ataque_de_dicion%C3%A1rio" \l "cite_note-rainbow-3)

## **Injeção de SQL** (do [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa" \o "Língua inglesa) *SQL Injection*) é um tipo de ameaça de [segurança](https://pt.wikipedia.org/wiki/Seguran%C3%A7a" \o "Segurança) que se aproveita de [falhas em sistemas](https://pt.wikipedia.org/wiki/Vulnerabilidade_(computa%C3%A7%C3%A3o)" \o "Vulnerabilidade (computação)) que trabalham com [bases de dados](https://pt.wikipedia.org/wiki/Bases_de_dados" \o "Bases de dados) realizando ataques com [comandos SQL](https://pt.wikipedia.org/wiki/SQL" \o "SQL); onde o atacante consegue inserir uma [instrução SQL](https://pt.wikipedia.org/wiki/Instru%C3%A7%C3%B5es" \o "Instruções) personalizada e indevida através da entrada de dados de uma aplicação, como formulários ou [URL](https://pt.wikipedia.org/wiki/URL" \o "URL) de uma aplicação online.[[1]](https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL" \l "cite_note-:0-1)

Um usuário por meio de ataques com injeção SQL, é possível obter qualquer tipo de dado sigiloso mantido no banco de dados de um [computador servidor](https://pt.wikipedia.org/wiki/Servidor" \o "Servidor).[[1]](https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL" \l "cite_note-:0-1) Dependendo da versão do banco, também é possível inserir comandos maliciosos e conseguir permissão total ([acesso root](https://pt.wikipedia.org/wiki/Superusu%C3%A1rio" \o "Superusuário)) à máquina em que o banco está em execução.[[1]](https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL" \l "cite_note-:0-1)

***Backdoor*** (em português, "**porta dos fundos**") é um método, geralmente secreto, de escapar de uma [autenticação](https://pt.wikipedia.org/wiki/Autentica%C3%A7%C3%A3o" \o "Autenticação) ou [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia) normais em um sistema computacional, produto ou [dispositivo embarcado](https://pt.wikipedia.org/wiki/Dispositivos_embarcados" \o "Dispositivos embarcados) (por exemplo, um [roteador](https://pt.wikipedia.org/wiki/Roteador" \o "Roteador) doméstico), ou sua incorporação, por exemplo, como parte de um [sistema criptográfico](https://pt.wikipedia.org/wiki/Sistema_criptogr%C3%A1fico" \o "Sistema criptográfico), um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo), um *[chipset](https://pt.wikipedia.org/wiki/Chipset" \o "Chipset)* ou um "computador homúnculo" - um pequeno computador dentro de um maior (como o encontrado na [tecnologia AMT](https://pt.wikipedia.org/wiki/Tecnologia_de_Gerenciamento_Ativo" \o "Tecnologia de Gerenciamento Ativo) da [Intel](https://pt.wikipedia.org/wiki/Intel" \o "Intel)).[[1]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-1)[[2]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-2) Os *backdoors* costumam ser usados para proteger o acesso remoto a um computador ou obter acesso a [texto simples](https://pt.wikipedia.org/wiki/Texto_puro_(criptografia)" \o "Texto puro (criptografia)) em [sistemas criptográficos](https://pt.wikipedia.org/wiki/Sistemas_criptogr%C3%A1ficos" \o "Sistemas criptográficos).

Um *backdoor* pode assumir a forma de uma parte oculta de um programa,[[3]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-3) um programa separado (por exemplo, o [Back Orifice](https://pt.wikipedia.org/wiki/Back_Orifice" \o "Back Orifice) pode subverter o sistema através de um *[rootkit](https://pt.wikipedia.org/wiki/Rootkit" \o "Rootkit)*), um código no *[firmware](https://pt.wikipedia.org/wiki/Firmware" \o "Firmware)* do *[hardware](https://pt.wikipedia.org/wiki/Hardware" \o "Hardware)*[[4]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-wired2013-4) ou partes de um [sistema operacional](https://pt.wikipedia.org/wiki/Sistema_operacional" \o "Sistema operacional), como o [Microsoft Windows](https://pt.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows).[[5]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-5)[[6]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-6)[[7]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-7) [Cavalos de Troia](https://pt.wikipedia.org/wiki/Cavalo_de_troia_(computa%C3%A7%C3%A3o)" \o "Cavalo de troia (computação)) podem ser usados para criar vulnerabilidades em um dispositivo. Um destes pode parecer um programa inteiramente legítimo, mas quando executado, ele executa uma atividade que pode instalar um *backdoor*.[[8]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-8) Embora alguns sejam secretamente instalados, outros *backdoors* são deliberadamente e amplamente conhecidos. Esses tipos têm usos "legítimos", como fornecer ao fabricante uma maneira de restaurar as [senhas](https://pt.wikipedia.org/wiki/Senha" \o "Senha) dos usuários. O *backdoor* pode ser usado para obter acesso a senhas, excluir dados em [discos rígidos](https://pt.wikipedia.org/wiki/Discos_r%C3%ADgidos" \o "Discos rígidos) ou transferir informações dentro da [nuvem](https://pt.wikipedia.org/wiki/Nuvem_(computa%C3%A7%C3%A3o)" \o "Nuvem (computação)).

Muitos sistemas que armazenam informações dentro da nuvem não conseguem criar medidas de segurança precisas. Se muitos sistemas estiverem conectados na nuvem, os *[hackers](https://pt.wikipedia.org/wiki/Hacker" \o "Hacker)* podem obter acesso a todas as outras plataformas através do sistema mais vulnerável.[[9]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-9)

Senhas padrões (ou outras credenciais padrões) podem funcionar como *backdoors* se não forem alteradas pelo usuário. Alguns recursos de [depuração](https://pt.wikipedia.org/wiki/Depura%C3%A7%C3%A3o" \o "Depuração) também podem atuar dessa forma se não forem removidos na versão de lançamento.[[10]](https://pt.wikipedia.org/wiki/Backdoor" \l "cite_note-10)

Em [1993](https://pt.wikipedia.org/wiki/1993" \o "1993), o governo dos [Estados Unidos](https://pt.wikipedia.org/wiki/Estados_Unidos" \o "Estados Unidos) tentou implantar um sistema de criptografia, o *[chip Clipper](https://pt.wikipedia.org/wiki/Chip_Clipper" \o "Chip Clipper)*, com um *backdoor* explícito para a aplicação da lei e acesso à segurança nacional. Mas o *[chip](https://pt.wikipedia.org/wiki/Chip" \o "Chip)* não teve sucesso.

## Visão geral

A ameaça de *backdoors* contra multiusuário e [sistemas operacionais](https://pt.wikipedia.org/wiki/Sistemas_operacionais" \o "Sistemas operacionais) em rede tornou-se amplamente adotada. Petersen e Turn discutiram em um artigo publicado no processo da Conferência de 1967 AFIPS. Eles observaram uma classe de ataques de infiltração ativos que utilizam o chamado "alçapão" como pontos de entrada no sistema para ignorar as instalações de segurança e permitir o acesso direto aos dados. O uso da palavra "alçapão" aqui claramente coincide com as definições mais recentes de uma porta. De modo geral, essas falhas de segurança foram discutidas em profundidade em uma Corporação RAND com relatório da força tarefa publicado sob ARPA, patrocinado por JP Anderson e DJ Edwards, em [1970](https://pt.wikipedia.org/wiki/1970" \o "1970). Um *backdoor* em um sistema de *[login](https://pt.wikipedia.org/wiki/Login" \o "Login)* pode assumir a forma de um código com difícil combinação de usuário e senha que dá acesso ao sistema.

Embora o número de *backdoors* nos sistemas que utilizam *[software](https://pt.wikipedia.org/wiki/Software" \o "Software)* proprietário (*software* cujo código fonte não está disponível ao público) não é amplamente creditado, eles são, no entanto, frequentemente expostos. Os programadores têm mesmo conseguido discretamente instalar grandes quantidades de código benignos como *easter eggs* em programas, embora tais casos pudesse envolver a tolerância oficial, se não a permissão real.

## Exemplos

### **Worms**

Muitos *[worms](https://pt.wikipedia.org/wiki/Worm" \o "Worm)* de computador, tais como Sobig e Mydoom, instalam um *backdoor* no computador afetado (em geral, um computador com banda larga e com execução do sistema operacional Microsoft Windows, além do uso do [Microsoft Outlook](https://pt.wikipedia.org/wiki/Microsoft_Outlook" \o "Microsoft Outlook)). Tais *backdoors* parecem estar instalados para que os [spammers](https://pt.wikipedia.org/wiki/Spammer" \o "Spammer) possam enviar *[e-mails](https://pt.wikipedia.org/wiki/E-mail" \o "E-mail)* das máquinas infectadas.

A sofisticada tentativa de plantar uma *backdoor* no [kernel](https://pt.wikipedia.org/wiki/Kernel" \o "Kernel) do [Linux](https://pt.wikipedia.org/wiki/Linux" \o "Linux), exposta em [novembro](https://pt.wikipedia.org/wiki/Novembro" \o "Novembro) de [2003](https://pt.wikipedia.org/wiki/2003" \o "2003), acrescentou uma pequena e sutil alteração de código de subverter o sistema de controle de revisão. Neste caso, a mudança de duas linhas apareceu para verificar o acesso root, permissões de uma chamada para a função sys\_wait4, mas porque usou atribuição em vez de igualdade verificando, ele realmente tinha permissões concedidas ao sistema. Esta diferença é facilmente esquecida, e até poderia ser interpretado como um erro de digitação acidental, ao invés de um ataque intencional.

Em [janeiro](https://pt.wikipedia.org/wiki/Janeiro" \o "Janeiro) de [2014](https://pt.wikipedia.org/wiki/2014" \o "2014), um *backdoor* foi descoberto em determinados [dispositivos móveis](https://pt.wikipedia.org/wiki/Dispositivos_m%C3%B3veis" \o "Dispositivos móveis), como o Galaxy, ao qual mantém o sistema operacional [Samsung](https://pt.wikipedia.org/wiki/Samsung" \o "Samsung) Android, permitindo o acesso remoto aos dados armazenados no dispositivo. Em particular, o *software* Samsung Android que está encarregado de lidar com as comunicações com o modem, utilizando o protocolo Samsung IPC, implementa uma classe de pedidos conhecidos como servidores de arquivos remotos (RFS) comandos, que permite ao operador *backdoor* para executar via *[modem](https://pt.wikipedia.org/wiki/Modem" \o "Modem)* remoto I / O, operações no disco rígido ou outro dispositivo de armazenamento. Como o modem está sobre responsabilidade da Samsung, assim como o *software* [Android](https://pt.wikipedia.org/wiki/Android" \o "Android), é provável que ele ofereça *over-the-air*, controle remoto que pode ser usado para emitir os comandos RFS e, portanto, para acessar o sistema de arquivos no dispositivo.

### **Utilização de backdoors por cavalos de Troia**

Os [cavalos de Troia](https://pt.wikipedia.org/wiki/Cavalo_de_Troia" \o "Cavalo de Troia) batizados como *backdoors* (por usar [portas](https://pt.wikipedia.org/wiki/Portas" \o "Portas) para entrar em computadores alheios) são muito comuns na [Internet](https://pt.wikipedia.org/wiki/Internet" \o "Internet). Normalmente eles utilizam as portas 666, 888, 3333, entre outras, para que seu usuário tenha acesso ao computador da vítima. O atacante procura garantir uma forma de retornar a um computador comprometido, sem precisar recorrer aos métodos utilizados na invasão, assim o atacante pode retornar ao computador comprometido sem ser notado.

Os cavalos de Troia do tipo *backdoor* ainda são divididos por suas funções. Por exemplo, um *backdoor* com **função de espião** faz uma troca de informações através da porta que ele abre para seu usuário poder visualizar arquivos da vítima, ver imagens pela *[webcam](https://pt.wikipedia.org/wiki/Webcam" \o "Webcam)* (ativando-a sem o conhecimento da vítima), ligar o [microfone](https://pt.wikipedia.org/wiki/Microfone" \o "Microfone), abrir e fechar o *drive* de CD/DVD-ROM, *[blu-ray](https://pt.wikipedia.org/wiki/Blu-ray" \o "Blu-ray)*, podendo corromper o Windows de forma irrecuperável, etc.

São conhecidos como RAT, da sigla inglesa *Remote Administrator Tool*, que por sua vez, em português, significa "ferramenta de administração remota".

### **Backdoors - código de objetos**

Mais difíceis de detectar são os *backdoors* que envolvem a modificação de código-objeto, ao invés de código-fonte — [código objeto](https://pt.wikipedia.org/wiki/C%C3%B3digo_objeto" \o "Código objeto) é muito mais difícil de fiscalizar. Estes podem ser inseridos diretamente no código objeto do disco rígido, ou inserido em algum momento durante a compilação, ligando a montagem, ou de carga — neste último caso, a porta dos fundos nunca aparece no disco, apenas na memória. *Backdoors* código-objeto são difíceis de detectar por inspeção do código objeto, mas são facilmente detectados pela simples verificação de alterações (diferenças), nomeadamente no comprimento ou na soma de verificação, e em alguns casos podem ser detectados ou analisados por desmontar o código objeto. Além disso, objetos *backdoors* referentes ao código podem ser removidos, basta recompilar a partir da fonte.

Para evitar a detecção, todas as cópias existentes de um binário devem ser subvertidas, e quaisquer somas de verificação de validação também devem ser comprometidas e a fonte deve estar disponível, para evitar a recompilação. Alternativamente, essas outras ferramentas (cheques comprimento, *[diff](https://pt.wikipedia.org/wiki/Diff" \o "Diff)*, *[checksum](https://pt.wikipedia.org/wiki/Checksum" \o "Checksum)*, *[disassemblers](https://pt.wikipedia.org/wiki/Disassembler" \o "Disassembler)*) pode-se estar comprometido para esconder o backdoor, por exemplo, detectar que o binário está sendo subvertido ao *checksummed* e retornando o valor esperado, e não o valor real. Para esconder essas novas subversões, as ferramentas também devem esconder as mudanças em si - por exemplo, um *checksummer* subvertido também deve detectar se ele é em si *checksum* (ou outras ferramentas subverteram) e retornam valores falsos. Isto leva a grandes mudanças no sistema e ferramentas sendo necessário para esconder uma única mudança.

O código objeto pode ser regenerado, recompilando do código-fonte original, fazendo uma persistência ao código objeto *backdoor* (sem modificar o código-fonte), ao qual é necessário subverter o compilador em si - para que, quando ele detecte quando está sendo realizado o compilador do programa sobre ataque, inserindo o *backdoor* - ou, alternativamente, o *[assembler](https://pt.wikipedia.org/wiki/Assembler" \o "Assembler)*, *[linker](https://pt.wikipedia.org/wiki/Linker" \o "Linker)* ou carregador.

Como isso exige subverter o compilador, este por sua vez, pode ser corrigido por recompilar o compilador, removendo o código de inserção *backdoor*. Essa defesa pode por sua vez ser subvertida, colocando uma fonte de *meta-backdoor* no compilador, para que, quando ele detecta que está compilando somente a si, inserindo em seguida, insere este um gerador de *meta-backdoor*, em conjunto com o gerador de *backdoor* original para o programa original sobre ataque. Depois que isso for feito, a fonte de *meta-backdoor* pode ser removido, e o compilador recompilado da fonte original com o [executável](https://pt.wikipedia.org/wiki/Execut%C3%A1vel" \o "Executável) do compilador comprometido.

### **Backdoors assimétricos**

Um *backdoor* assimétrico só pode ser usado pelo atacante que plantá-lo, mesmo que a plena implementação do *backdoor* se torne pública (por exemplo, por meio de publicação, sendo descoberta e divulgada pela [engenharia reversa](https://pt.wikipedia.org/wiki/Engenharia_reversa" \o "Engenharia reversa), etc.). Além disso, é computacionalmente intratável para detectar a presença de um *backdoor* assimétrico sob consultas de "caixa preta". Esta classe de ataques foi denominada como *kleptography*; eles podem ser realizados em *software*, hardware (por exemplo, cartões inteligentes), ou uma combinação dos dois. A teoria de backdoors assimétricas é parte de um campo maior agora chamado *cryptovirology*.

## Lista de backdoors conhecidos em normas

O [hash MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5) mostrou ter várias deficiências, em [1996](https://pt.wikipedia.org/wiki/1996" \o "1996), por Hans Dobbertin. Estas debilidades permitem ao invasor substituir o seu próprio artigo por um original assinado-MD5. O código malicioso é assim introduzido em um sistema. Turner e Chen, em [RFC 6149](https://datatracker.ietf.org/doc/html/rfc6149), escreveram que "[MD2](https://pt.wikipedia.org/wiki/MD2" \o "MD2) não deve ser utilizado para assinaturas digitais", pois poderia ser falsificada. Escreveu também a [RFC 6150](https://tools.ietf.org/html/rfc6150), que [MD4](https://pt.wikipedia.org/wiki/MD4" \o "MD4) "não devem ser usadas para criptografar a chave de criptografia de 80 *[bits](https://pt.wikipedia.org/wiki/Bit" \o "Bit)* ou mais."; e ataques MD4 são praticáveis. SHA-0 (*aka* FIPS-180) foi retirado após CRYPTO '98. [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) (aka FIPS 180-1-) mostrou ser atacável em [2005](https://pt.wikipedia.org/wiki/2005" \o "2005), por Eli Biham e co-autores, bem como Vincent Rijmen e Elisabeth Oswald.

O *Dual\_EC\_DRBG* gerador de números pseudo-aleatórios criptograficamente seguro foi revelado em [2013](https://pt.wikipedia.org/wiki/2013" \o "2013) para, eventualmente, ter um (assimétrico) *backdoor kleptographic* inserido deliberadamente pelo NSA, que também tinha a chave privada para o *backdoor*.

## Backdoors em compiladores

Uma forma sofisticada de caixa preta é um *backdoor* em um [compilador](https://pt.wikipedia.org/wiki/Compilador" \o "Compilador), onde não é só um compilador subvertido (para inserir um backdoor em algum outro programa, como um programa de login), é modificado para detectar quando for compilar a si mesmo e, em seguida, inserir tanto o código de inserção *backdoor* (visando o outro programa) e modificar o código de auto-compilação. Isto pode ser feito através da modificação do código-fonte, e o compilador comprometido (código objeto) resultante pode compilar o código original.

Este ataque foi originalmente apresentado em Karger & Schell (1974, p 52, seção 3.4.5:. "*Trap Door Insertion*"), que foi uma análise do Multics, onde eles descreveram como um ataque a um PL de segurança *United States Air Force / compiler*, e chamá-lo de "alçapão compilador"; eles também mencionam uma variante em que o código de inicialização do sistema é modificada para inserir um backdoor durante a inicialização, como este é complexo e mal compreendido, e chamá-lo de um "alçapão de inicialização", este que é agora conhecido como um vírus do setor de inicialização.

Este ataque foi então efetivamente implementado e popularizado por [Ken Thompson](https://pt.wikipedia.org/wiki/Ken_Thompson" \o "Ken Thompson), em Thompson ([1984](https://pt.wikipedia.org/wiki/1984" \o "1984)), em seu discurso de aceitação do [Prêmio Turing](https://pt.wikipedia.org/wiki/Pr%C3%AAmio_Turing" \o "Prêmio Turing) em [1983](https://pt.wikipedia.org/wiki/1983" \o "1983) (publicado em 1984), "Reflexões sobre a Confiança", onde aponta que a confiança é relativa, o único software que se pode realmente confiar é um código onde cada passo do *[bootstrapping](https://pt.wikipedia.org/wiki/Bootstrapping" \o "Bootstrapping)* seja inspecionado. Este mecanismo *backdoor* é baseado no fato de as pessoas só o reverem código fonte (*human-written*), e código de máquina não compilado (código objeto). O programa chamado compilador é utilizado para criar o segundo a partir do primeiro, e o compilador é geralmente confiável para fazer um trabalho honesto.

O trabalho de Thompson descreve uma versão modificada do compilador C Unix que faria: colocar um *backdoor* invisível no comando de login do Unix quando se notou que o programa de login está sendo compilado, e como uma torção.  Também pode adicionar esse recurso indetectável para versões futuras do compilador em cima de sua compilação também. Por o próprio compilador ser um programa compilado, seria extremamente improvável que os usuários notassem as instruções de código de máquina que realizaram essas tarefas. O pior, na prova da execução, conceito de Thompson, o compilador subvertido também subverteu o programa de análise (o desmontador), de modo que qualquer um que examinou os binários na forma habitual, não seria realmente ver o código real que estava correndo, mas sim outra coisa em seu lugar.

Uma análise atualizada do *exploit* original foi dado em Karger & Schell (2002, item 3.2.4: *Doors Compiler Trap*), e um panorama histórico e pesquisa da literatura foi dado em Wheeler (2009, Seção 2: Fundo e trabalhos relacionados).

## Ocorrências

A versão de Thompson, oficialmente, nunca foi apresentada. Acredita-se, porém, que a versão foi distribuída à BBN e pelo menos um uso do *backdoor* foi gravado. Não houve relatos de tais *backdoors* nos anos subsequentes.

Este ataque foi recentemente (em agosto de 2009) descoberto por laboratórios Sophos: A / Induc-A do vírus W32 infectando o compilador do programa para [Delphi](https://pt.wikipedia.org/wiki/Delphi_(software)" \o "Delphi (software)), uma [linguagem de programação](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o" \o "Linguagem de programação) do Windows. O vírus introduziu o seu próprio código para a elaboração de novos programas Delphi, permitindo-lhe infectar e propagar para muitos sistemas, sem o conhecimento do programador de software. Um ataque que se propaga através da construção de seu próprio cavalo de Troia pode ser especialmente difícil de descobrir. Acredita-se que *a-Induc* (um vírus) tinha sido propagado durante pelo menos um ano antes de ser descoberto.

## Contramedidas

Uma vez que um sistema foi comprometido com um *backdoor* ou cavalo de Troia, como o compilador Confiando Trust, é muito difícil para o usuário "legítimo" recuperar o controle do sistema - tipicamente deve-se reconstruir um sistema limpo e transferir dados. No entanto, foram sugeridas várias deficiências práticas no esquema Confiando Trust. Por exemplo, um usuário suficientemente motivado poderia meticulosamente rever o código de máquina do compilador não confiável antes de usá-lo. Como mencionado acima, existem várias maneiras de esconder o cavalo de Tróia, como subverter o desmontador, mas existem maneiras de combater essa defesa, também, como escrever o seu próprio desmontador a partir do zero.

Um método genérico para combater os ataques de confiança é chamado Diverse Double-compilação (DDC). O método requer um compilador diferente e o código-fonte do-under-test compilador. Essa fonte, compilada com ambos os compiladores, resulta em duas fases, compiladores diferentes, que, no entanto devem ter o mesmo comportamento. Assim, a mesma fonte compilada com ambos (estágio 1) compiladores devem resultar em seguida, dois idênticos (estagio 2) compiladores. Uma prova formal só é dada quando esta última comparação garanta que o código fonte suposto é executável do-under-test compilador correspondam, em algumas hipóteses. Este método foi aplicado pelo seu autor para verificar se o compilador C da suíte GCC (v. 3.0.4) não continha “*trojan*” usando ICC (v. 11.0), como o compilador diferente.

Na prática, essas verificações não são feitas por usuários finais, exceto em circunstâncias extremas de detecção e análise de intrusão, devido à raridade de tais ataques sofisticados, pois os programas são normalmente distribuídos em formato binário. A remoção de *backdoors* (inclindo compiladores) normalmente é feita para reconstruir um sistema limpo. No entanto, essas verificações sofisticadas são de interesse de fornecedores de sistemas operacionais, para garantir que eles não estejam distribuindo um sistema comprometido, e em ambientes de alta segurança, onde tais ataques são uma preocupação realista.

Uma **função de dispersão criptográfica** ou **função hash criptográfica** é uma [função hash](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash" \o "Função hash) considerada [praticamente impossível de inverter](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_de_m%C3%A3o_%C3%BAnica" \o "Função de mão única), isto é, de recriar o valor de entrada utilizando somente o valor de dispersão. Essas funções hash unidirecionais têm sido chamadas de "os operários da criptografia moderna".[[1]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-1) Os dados de entrada costumam ser chamados de *mensagem*, e o valor de dispersão *mensagem resumida* ou simplesmente *resumo*.

Uma função de dispersão criptográfica deve possuir quatro propriedades principais:

* deve ser fácil computar o valor de dispersão para qualquer mensagem
* deve ser [difícil](https://pt.wikipedia.org/wiki/Complexidade_computacional" \l "Intratabilidade" \o "Complexidade computacional) gerar uma mensagem a partir de seu resumo
* deve ser [difícil](https://pt.wikipedia.org/wiki/Complexidade_computacional" \l "Intratabilidade" \o "Complexidade computacional) modificar a mensagem sem modificar o seu resumo
* deve ser [difícil](https://pt.wikipedia.org/wiki/Complexidade_computacional" \l "Intratabilidade" \o "Complexidade computacional) encontrar duas mensagens diferentes com o mesmo resumo.

Funções hash criptográficas possuem várias aplicações em [segurança da informação](https://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o" \o "Segurança da informação), principalmente em [assinatura digital](https://pt.wikipedia.org/wiki/Assinatura_digital" \o "Assinatura digital), [código de autenticação de mensagem](https://pt.wikipedia.org/w/index.php?title=C%C3%B3digo_de_autentica%C3%A7%C3%A3o_de_mensagem&action=edit&redlink=1" \o "Código de autenticação de mensagem (página não existe)) (MACs), e outras formas de [autenticação](https://pt.wikipedia.org/wiki/Autentica%C3%A7%C3%A3o" \o "Autenticação). Elas também podem ser utilizadas como funções hash, para indexar dados em [tabelas hash](https://pt.wikipedia.org/wiki/Tabelas_hash" \o "Tabelas hash), para [impressão digital](https://pt.wikipedia.org/wiki/Impress%C3%A3o_digital_(computa%C3%A7%C3%A3o)" \o "Impressão digital (computação)), para detectar dados duplicados ou identificar arquivos únicos, e como [checksum](https://pt.wikipedia.org/wiki/Checksum" \o "Checksum) para detectar corrupção de dados acidental. De fato, no contexto da segurança da informação, valores de dispersão criptográficos são às vezes conhecidos como *impressão digital*, *checksums*, ou apenas *valores de dispersão*, apesar de todos esses termos se referirem a funções mais gerais com propriedades e propósitos diferentes.

## Propriedades

Muitas funções hash criptográficas são projetadas para receber uma [cadeia de caracteres](https://pt.wikipedia.org/wiki/Cadeia_de_caracteres" \o "Cadeia de caracteres) de qualquer tamanho como entrada e produzir um valor hash de tamanho fixo.

Uma função hash criptográfica deve ser capaz de aguentar todo [tipo de ataque cripto-analítico](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \l "Ataques_caracter%C3%ADsticos" \o "Criptoanálise) conhecido. No mínimo, ela deve possuir as seguintes propriedades:

* *Resistência à pré-imagem*

Dado um valor hash *h* deve ser difícil encontrar qualquer mensagem *m* tal que *h = hash(m)*. Este conceito está relacionado ao da [função de mão única](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_de_m%C3%A3o_%C3%BAnica" \o "Função de mão única) (ou função unidirecional). Funções que não possuem essa propriedade estão vulneráveis a [ataques de pré-imagem](https://pt.wikipedia.org/w/index.php?title=Ataque_de_pr%C3%A9-imagem&action=edit&redlink=1" \o "Ataque de pré-imagem (página não existe)).

* *Resistência à segunda pré-imagem*

Dada uma entrada *m*1 deve ser difícil encontrar outra entrada *m*2 tal que *hash*(*m*1) = *hash*(*m*2). Funções que não possuem essa propriedade estão vulneráveis a [ataques de segunda pré-imagem](https://pt.wikipedia.org/w/index.php?title=Ataque_de_pr%C3%A9-imagem&action=edit&redlink=1" \o "Ataque de pré-imagem (página não existe)).

* *[Resistência à colisão](https://pt.wikipedia.org/w/index.php?title=Resist%C3%AAncia_%C3%A0_colis%C3%A3o&action=edit&redlink=1" \o "Resistência à colisão (página não existe))*

Deve ser difícil encontrar duas mensagens diferentes *m*1 e *m*2 tal que *hash*(*m*1) = *hash*(*m*2). Tal par é chamado de [colisão hash](https://pt.wikipedia.org/wiki/Tabela_de_dispers%C3%A3o" \l "Colis%C3%B5es" \o "Tabela de dispersão) criptográfica. Essa propriedade também é conhecida como *forte resistência à colisão.* Ela requer um valor hash com pelo menos o dobro do comprimento necessário para resistência à pré-imagem; caso contrário, colisões podem ser encontradas através de um [ataque do aniversário](https://pt.wikipedia.org/wiki/Ataque_do_anivers%C3%A1rio" \o "Ataque do aniversário).

Uma função com todos esses critérios ainda pode possuir propriedades indesejadas. Atualmente, funções hash criptográficas populares estão vulneráveis a [ataques de extensão de comprimento](https://pt.wikipedia.org/wiki/HMAC" \l "Princ%C3%ADpios_de_projeto" \o "HMAC): dado *hash*(*m*) e *len*(*m*) mas não *m*, escolhendo um *m*’ adequado um atacante pode calcular *hash*(*m* || *m*’) onde || denota [concatenação](https://pt.wikipedia.org/wiki/Concatena%C3%A7%C3%A3o" \o "Concatenação).[[2]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-2) Essa propriedade pode ser usada para quebrar esquemas de autenticação ingênuos baseados em funções hash. A construção [HMAC](https://pt.wikipedia.org/wiki/HMAC" \o "HMAC) contorna esses problemas.

Idealmente, o projetista da função deve desejar por condições mais fortes. Deve ser impossível para um adversário encontrar duas mensagens com resumos substancialmente similares; ou inferir qualquer informação útil sobre os dados a partir de seu resumo. Portanto, uma função hash criptográfica deve se comportar tão semelhante quanto possível a uma [função aleatória](https://pt.wikipedia.org/w/index.php?title=Fun%C3%A7%C3%A3o_aleat%C3%B3ria&action=edit&redlink=1" \o "Função aleatória (página não existe)) ainda sendo determinística e eficientemente computável.

Algoritmos de checksum, tais como [CRC32](https://pt.wikipedia.org/wiki/CRC32" \o "CRC32) e outras [verificação de redundância cíclicas](https://pt.wikipedia.org/w/index.php?title=Verifica%C3%A7%C3%A3o_de_redund%C3%A2ncia_c%C3%ADclica&action=edit&redlink=1" \o "Verificação de redundância cíclica (página não existe)), são projetadas para atender requisitos mais fracos, e são geralmente inapropriados para atuar como funções hash criptográficas. Por exemplo, um CRC foi utilizado para a integridade de mensagens na encriptação [WEP](https://pt.wikipedia.org/wiki/Wired_Equivalent_Privacy" \o "Wired Equivalent Privacy) padrão, mas um ataque que explorava a linearidade do checksum foi logo descoberto.

### **Grau de dificuldade**

Na prática criptográfica, “dificuldade” geralmente significa “quase certamente longe do alcance de qualquer adversário que deve ser impedido de quebrar o sistema enquanto a segurança de tal sistema for importante”. O significado do termo, portanto, depende da aplicação, já que o esforço que o agente malicioso deve colocar na tarefa é normalmente proporcional ao que ele espera conseguir. Entretanto, como o esforço necessário geralmente cresce rapidamente com o comprimento do resumo, até um atacante com um grande poder de processamento pode ser neutralizado adicionando algumas dúzias de bits ao resumo.

Em algumas [análises teóricas](https://pt.wikipedia.org/wiki/Complexidade_computacional" \o "Complexidade computacional) “dificuldade” tem um significado matemático específico, como "não solucionável em [tempo polinomial](https://pt.wikipedia.org/wiki/Complexidade_de_tempo" \l "Tempo_Polinomial" \o "Complexidade de tempo) [assintótico](https://pt.wikipedia.org/wiki/An%C3%A1lise_assint%C3%B3tica" \o "Análise assintótica)". Essas interpretações de *dificuldade* são importantes no estudo de funções hash criptográficas provadas seguras mas não costumam ter uma forte relação com segurança na prática. Por exemplo, um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo) de [tempo exponencial](https://pt.wikipedia.org/wiki/Complexidade_de_tempo" \l "Complexidade_Exponencial" \o "Complexidade de tempo) pode ser rápido o bastante para realizar um ataque factível. Reciprocamente, um algoritmo de tempo polinomial (e.g., um que requer *n*20 passos por chaves de *n*-dígitos) pode ser lento demais para usos práticos.

## Ilustração

Uma ilustração do uso potencial de uma dispersão criptográfica é a seguinte: [Alice](https://pt.wikipedia.org/wiki/Alice_e_Bob" \o "Alice e Bob) propõe um problema matemático complicado para [Bob](https://pt.wikipedia.org/wiki/Alice_e_Bob" \o "Alice e Bob) e afirma ser capaz de resolvê-lo. Bob quer tentar resolvê-lo sozinho, mas primeiro precisa ter certeza de que Alice não está blefando. Portanto, Alice escreve a solução, computa o valor hash e diz a Bob tal valor (mantendo a solução em segredo). Então, quando Bob vier com sua solução alguns dias depois, Alice pode provar que ela já tinha a solução, tirando o valor hash da solução de Bob e comparando-o ao valor que ela havia calculado. (Este é um exemplo simples de [esquema de comprometimento](https://pt.wikipedia.org/w/index.php?title=Esquema_de_comprometimento&action=edit&redlink=1" \o "Esquema de comprometimento (página não existe)); numa prática real, Alice e Bob seriam programas de computador, e o segredo algo menos facilmente falsificado como uma afirmação de solução a um enigma).

## Aplicações

### **Verificação da integridade de arquivos ou mensagens**

Uma importante aplicação desses hashes de segurança é na verificação da integridade de mensagens. Determinar se qualquer alteração foi feita a uma mensagem (ou um [arquivo](https://pt.wikipedia.org/wiki/Arquivo_de_computador" \o "Arquivo de computador)), por exemplo, pode ser realizado comparando mensagens resumidas, calculadas antes e depois da transmissão (ou qualquer outro evento).

Por essa razão, muitos algoritmos de [assinatura digitai](https://pt.wikipedia.org/w/index.php?title=Assinatura_digitai&action=edit&redlink=1" \o "Assinatura digitai (página não existe)) apenas confirmam a autenticidade do resumo de uma mensagem para ser "autenticado". Verificar a autenticidade do resumo de uma mensagem é considerado prova de que a mensagem em si é autêntica.

Hashes [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5), [SHA1](https://pt.wikipedia.org/wiki/SHA1" \o "SHA1), ou [SHA2](https://pt.wikipedia.org/w/index.php?title=SHA2&action=edit&redlink=1" \o "SHA2 (página não existe)) são ocasionalmente informados juntos de seus arquivos em websites e fóruns o que permite a verificação de integridade.[[3]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-3) Essa prática estabelece uma [cadeia de confiança](https://pt.wikipedia.org/w/index.php?title=Cadeia_de_confian%C3%A7a&action=edit&redlink=1" \o "Cadeia de confiança (página não existe)), contanto que as hashes sejam postadas em sites autenticados por [HTTPS](https://pt.wikipedia.org/wiki/HTTPS" \o "HTTPS).

### **Verificação de senha**

Uma aplicação relacionada é a verificação de [senha](https://pt.wikipedia.org/wiki/Senha" \o "Senha) (inventada por [Roger Needham](https://pt.wikipedia.org/w/index.php?title=Roger_Needham&action=edit&redlink=1" \o "Roger Needham (página não existe))). Armazenar todas as senhas de um usuário como [purotexto](https://pt.wikipedia.org/wiki/Purotexto" \o "Purotexto) pode resultar em uma quebra massiva de segurança caso o arquivo de senha seja comprometido. Uma maneira de reduzir esse perigo é de apenas armazenar o resumo de cada senha. Para autenticar um usuário, calcula-se o resumo da senha fornecida pelo usuário e o compara-se ao resumo armazenado. (Note que essa abordagem impede que a senha original seja recuperada se esquecida ou perdida, e terá que ser substituída por uma nova.) A senha, geralmente, é concatenada com um valor aleatório, não-secreto conhecido como sal, antes de ser aplicada a função hash. O sal é armazenado junto ao hash da senha. Como usuários devem possuir sais diferentes, é impraticável armazenar tabelas de valores hash pré-computados para senhas comuns. Funções de [alongamento de chave](https://pt.wikipedia.org/wiki/Alongamento_de_chave" \o "Alongamento de chave), como [PBKDF2](https://pt.wikipedia.org/w/index.php?title=PBKDF2&action=edit&redlink=1" \o "PBKDF2 (página não existe)), [Bcrypt](https://pt.wikipedia.org/wiki/Bcrypt" \o "Bcrypt) ou [Scrypt](https://pt.wikipedia.org/wiki/Scrypt" \o "Scrypt), costumam utilizar chamadas repetidas de hash criptográficos para aumentar o tempo necessário de um [ataque de força bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta" \o "Ataque de força bruta) sobre o resumo das senhas armazenadas.

Em 2013, uma [Password Hashing Competition](https://pt.wikipedia.org/w/index.php?title=Password_Hashing_Competition&action=edit&redlink=1" \o "Password Hashing Competition (página não existe)) foi anunciada para escolher um novo algoritmo padrão para computar hash de senhas.[[4]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-4)

### **Prova-de-trabalho**

Um sistema a prova-de-trabalho (ou protocolo, ou função) é uma medida econômica para deter [ataques de negação de serviço](https://pt.wikipedia.org/wiki/Ataque_de_nega%C3%A7%C3%A3o_de_servi%C3%A7o" \o "Ataque de negação de serviço) e outros abusos de serviço como spam. O sistema solicita ao solicitante do serviço que faça algum trabalho, o que normalmente significa tempo de processamento para o computador. Uma característica chave desses esquemas é sua assimetria: esse trabalho deve ser moderadamente difícil (mas factível) para o solicitante e fácil de verificar para o fornecedor do serviço. Um sistema muito popular - utilizado em [mineração de bitcoin](https://pt.wikipedia.org/w/index.php?title=Minera%C3%A7%C3%A3o_de_bitcoin&action=edit&redlink=1" \o "Mineração de bitcoin (página não existe)) e [hashcash](https://pt.wikipedia.org/w/index.php?title=Hashcash&action=edit&redlink=1" \o "Hashcash (página não existe)) - utiliza inversões parciais de hash para provar que o trabalho foi realizado, como um símbolo de boa-vontade para mandar um e-mail. O remetente é solicitado a encontrar a mensagem cujo valor hash começa com um número de bits zero. O trabalho médio que o remetente precisa realizar para encontrar a mensagem válida é exponencial no número de bits zero necessários no valor hash, enquanto o beneficiário pode verificar a validação da mensagem executando uma única função hash. Por exemplo, em hashcash, o remetente é solicitado a gerar um cabeçalho cujo valor hash SHA1 de 160 bits tenha seus primeiros 20 bits como 0. O remetente terá que tentar, em média, 220 vezes até encontrar um cabeçalho válido.

### **Identificador de arquivo ou dados**

Um resumo de mensagem também pode servir como um identificador confiável de arquivo; diversos sistemas de gerenciadores de [código fonte](https://pt.wikipedia.org/wiki/C%C3%B3digo-fonte" \o "Código-fonte), como [Git](https://pt.wikipedia.org/wiki/Git" \o "Git), [Mercurial](https://pt.wikipedia.org/wiki/Mercurial" \o "Mercurial) e [Monotone](https://pt.wikipedia.org/w/index.php?title=Monotone&action=edit&redlink=1" \o "Monotone (página não existe)), usam o [sha1sum](https://pt.wikipedia.org/w/index.php?title=Sha1sum&action=edit&redlink=1" \o "Sha1sum (página não existe)) de vários tipos de conteúdo (conteúdo de arquivo, árvores de diretório, informação ancestral, etc.) para identificá-los univocamente. Hashes são usados para identificar arquivos em redes de [compartilhamento de arquivos](https://pt.wikipedia.org/wiki/Peer-to-peer" \o "Peer-to-peer). Por exemplo, em um link [ed2k](https://pt.wikipedia.org/wiki/Ed2k" \o "Ed2k), uma variação do hash [MD4](https://pt.wikipedia.org/wiki/MD4" \o "MD4) é combinado com o tamanho do arquivo, fornecendo informação suficiente para a localização das fontes do arquivo, baixando o arquivo e verificando seus conteúdos. [Magnet-links](https://pt.wikipedia.org/w/index.php?title=Magnet-links&action=edit&redlink=1" \o "Magnet-links (página não existe)) são outros exemplos. Tais hashes de arquivo costumam aparecer no começo de [listas de hash](https://pt.wikipedia.org/w/index.php?title=Listas_de_hash&action=edit&redlink=1" \o "Listas de hash (página não existe)) ou em [árvores de hash](https://pt.wikipedia.org/wiki/%C3%81rvores_de_Merkle" \o "Árvores de Merkle) o que propicia ainda mais benefícios.

Uma das principais aplicações de uma função hash é permitir uma rápida visualização dos dados em uma tabela hash. Por serem funções hash com propriedades específicas, funções hash criptográficas também podem ser utilizadas para essas aplicações.

Entretanto, comparadas a funções hash padrão, funções hash criptográficas tendem a ser computacionalmente mais caras. Por essa razão, elas costumam ser usadas em contextos onde é necessário que usuários protejam a si mesmo contra a possibilidade de falsificação (criação de dados com o mesmo resumo que os dados esperados) por agentes maliciosos em potencial.

### **Geradores pseudo-aleatórios e derivação de chave**

Funções hash também podem ser utilizadas para gerar bits [pseudo-aleatórios](https://pt.wikipedia.org/wiki/Pseudoaleatoriedade" \o "Pseudoaleatoriedade), ou para derivar novas chaves ou senhas de uma única e segura chave ou senha.

## Funções hash baseadas em cifras de bloco

Há diversos métodos de utilizar uma [cifra de bloco](https://pt.wikipedia.org/wiki/Cifra_de_bloco" \o "Cifra de bloco) para construir funções hash criptográficas, especificamente uma [função de compressão unidirecional](https://pt.wikipedia.org/w/index.php?title=Fun%C3%A7%C3%A3o_de_compress%C3%A3o_unidirecional&action=edit&redlink=1" \o "Função de compressão unidirecional (página não existe)).

Esse método é semelhante ao [modo de operação](https://pt.wikipedia.org/wiki/Modo_de_opera%C3%A7%C3%A3o_(criptografia)" \o "Modo de operação (criptografia)) de uma cifra de bloco geralmente utilizado em encriptação. Muitas funções hash conhecidas, como [MD4](https://pt.wikipedia.org/wiki/MD4" \o "MD4), [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5), [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) e [SHA-2](https://pt.wikipedia.org/wiki/SHA-2" \o "SHA-2) são construídas de componentes semelhantes a cifras de bloco, projetados para esse propósito, com feedback para garantir que a função resultante não seja inversível. Os finalistas da competição de função hash da [NIST](https://pt.wikipedia.org/wiki/NIST" \o "NIST) projetistas do [SHA-3](https://pt.wikipedia.org/wiki/SHA-3" \o "SHA-3), incluíram funções semelhantes a bloco de cifras como componentes (e.g., [Skein](https://pt.wikipedia.org/w/index.php?title=Skein&action=edit&redlink=1" \o "Skein (página não existe)), [BLAKE](https://pt.wikipedia.org/w/index.php?title=BLAKE&action=edit&redlink=1" \o "BLAKE (página não existe))) apesar da função selecionada, [Keccak](https://pt.wikipedia.org/w/index.php?title=Keccak&action=edit&redlink=1" \o "Keccak (página não existe)), ao contrário, ter sido construída sobre uma [esponja criptográfica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_esponja" \o "Função esponja).

Uma cifra de bloco padrão como a [AES](https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard" \o "Advanced Encryption Standard) pode ser utilizada no lugar dessas cifras de bloco personalizadas; isso pode ser útil quando um [sistema embarcado](https://pt.wikipedia.org/wiki/Sistema_embarcado" \o "Sistema embarcado) precisa implementar funções de encriptação e hash utilizando um tamanho mínimo de código ou área de [hardware](https://pt.wikipedia.org/wiki/Hardware" \o "Hardware). Entretanto, essa abordagem pode ter um custo de eficiência e segurança. As cifras em função hash são construídas para calcular hash: elas utilizam chaves e bloco de grande comprimento, podem mudar eficientemente de chaves em cada bloco, e foram projetadas e corrigidas para resistir a [ataques de chave relacionadas](https://pt.wikipedia.org/w/index.php?title=Ataques_de_chave_relacionadas&action=edit&redlink=1" \o "Ataques de chave relacionadas (página não existe)). Cifras de propósito geral costumam possuir objetivos de projeto diferentes. Em particular, AES possui tamanhos de chave e blocos que torna não-trivial a geração de valores hash longos; encriptação AES torna-se menos eficiente quando a chave muda a cada bloco; e ataques de chave relacionadas a tornam potencialmente menos segura para uso em função hash do que em encriptação.

## Uso na construção de outras primitivas criptográficas

Funções hash podem ser utilizadas na construção de outras primitivas criptográficas. Para que essas outras primitivas sejam criptograficamente seguras, alguns cuidados devem ser tomados para construí-las corretamente.

[Autenticadores de mensagem](https://pt.wikipedia.org/wiki/Autenticador_de_mensagem" \o "Autenticador de mensagem) (MACs) (também chamados de função hash chaveadas) costumam ser construídos a partir de função hash. [HMAC](https://pt.wikipedia.org/wiki/HMAC" \o "HMAC) é um MAC.

Assim como cifras de bloco podem ser utilizadas para construir funções hash, funções hash também podem ser utilizadas para construir cifras de bloco. A [cifra de Feistel](https://pt.wikipedia.org/w/index.php?title=Cifra_de_Feistel&action=edit&redlink=1" \o "Cifra de Feistel (página não existe)) utilizando funções hash podem ser provadas seguras caso a função hash utilizada seja segura. Além disso, muitas funções hash (como [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) 2 [SHA-2](https://pt.wikipedia.org/wiki/SHA-2" \o "SHA-2)) são construídas utilizando uma cifra de bloco de propósito especial em uma [Davies-Meyer](https://pt.wikipedia.org/w/index.php?title=Davies-Meyer&action=edit&redlink=1" \o "Davies-Meyer (página não existe)) ou outra construção. Essa cifra também pode ser utilizada em modo de operação convencional, sem as mesmas garantias de segurança. Por exemplo, [SHACAL](https://pt.wikipedia.org/w/index.php?title=SHACAL&action=edit&redlink=1" \o "SHACAL (página não existe)), [BEAR](https://pt.wikipedia.org/w/index.php?title=BEAR&action=edit&redlink=1" \o "BEAR (página não existe)) e [LION](https://pt.wikipedia.org/w/index.php?title=LION&action=edit&redlink=1" \o "LION (página não existe)).

[Geradores de números pseudoaleatórios](https://pt.wikipedia.org/wiki/Gerador_de_n%C3%BAmeros_pseudoaleat%C3%B3rios" \o "Gerador de números pseudoaleatórios) (GNPAs) podem ser construídos a partir de funções hash. Isso é feito combinando uma semente aleatória e secreta a um contador e fornecendo essa combinação a função hash.

Algumas função hash, tais como [Skein](https://pt.wikipedia.org/w/index.php?title=Skein&action=edit&redlink=1" \o "Skein (página não existe)), [Keccak](https://pt.wikipedia.org/w/index.php?title=Keccak&action=edit&redlink=1" \o "Keccak (página não existe)) e [RadioGatún](https://pt.wikipedia.org/w/index.php?title=RadioGat%C3%BAn&action=edit&redlink=1" \o "RadioGatún (página não existe)) fornecem como saída uma cadeia arbitrariamente longa e podem ser utilizadas como [cifra de fluxo](https://pt.wikipedia.org/wiki/Cifra_de_fluxo" \o "Cifra de fluxo), e cifras de fluxo podem ser construídas de funções hash com resumo de comprimento fixo. Frequentemente, isso é feito construindo, primeiro, um [gerador de números pseudoaleatórios criptograficamente seguro](https://pt.wikipedia.org/w/index.php?title=Gerador_de_n%C3%BAmeros_pseudoaleat%C3%B3rios_criptograficamente_seguro&action=edit&redlink=1" \o "Gerador de números pseudoaleatórios criptograficamente seguro (página não existe)) e usando seu fluxo de bytes aleatórios como [fluxo de chave](https://pt.wikipedia.org/w/index.php?title=Fluxo_de_chave&action=edit&redlink=1" \o "Fluxo de chave (página não existe)) (*keystream*). [SEAL](https://pt.wikipedia.org/wiki/SEAL" \o "SEAL) é uma cifra de fluxo que utiliza [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) para gerar tabelas internas, que são então utilizadas em um gerador de fluxo de chave com certa relação ao algoritmo de hash. SEAL não possui garantia de ser tão forte (ou fraca) quanto SHA-1. Similarmente, a expansão de chave das cifras de fluxo [HC-128](https://pt.wikipedia.org/w/index.php?title=HC-128&action=edit&redlink=1" \o "HC-128 (página não existe)) e [HC-256](https://pt.wikipedia.org/w/index.php?title=HC-256&action=edit&redlink=1" \o "HC-256 (página não existe)) fazem alto uso da função hash [SHA256](https://pt.wikipedia.org/wiki/SHA-2" \o "SHA-2).

## Concatenação de funções hash criptográficas

Concatenar saídas de múltiplas funções hash provê resistência a colisões tão fortes quanto os algoritmos inclusos no resultado concatenado. Por exemplo, uma versão antiga da [TSL/SSL](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \o "Transport Layer Security) usa somas da [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5) e [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) concatenadas - isso garante que o método de detectar colisões em uma das funções não permite falsificação do tráfego protegido por ambas as funções.

Para funções hash Merkle-Damgård, a função concatenada é tão resistente a colisão quanto seu componente mais forte, mas nunca mais resistente a colisão. Joux[[5]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-5) nota que 2 colisões levam a n colisões: se encontrar duas mensagens com a mesma hash MD5 é factível, então não é mais difícil encontrar quantas mensagens quanto o atacante desejar com hashes MD5 idênticas. Entre as n mensagens com a mesma hash MD5, é provável que haja uma colisão em SHA-1. O trabalho adicional necessário para encontrar a colisão em SHA-1 (além da busca de aniversário exponencial) é [polinomial](https://pt.wikipedia.org/wiki/Complexidade_de_tempo" \l "Tempo_Polinomial" \o "Complexidade de tempo). Esse argumento é sumarizado por [Finney](http://article.gmane.org/gmane.comp.encryption.general/5154).

## Algoritmos hash criptográficos

Há uma lista longa de algoritmos hash criptográficos, apesar de muitos terem sido mostrados vulneráveis e não devem mais ser utilizados. Mesmo que uma função hash nunca tenha sido quebrada, um ataque bem sucedido quanto uma variante mais fraca, pode enfraquecer a confiança de profissionais da área e levar a seu abandono. Por exemplo, em Agosto de 2004, foram descobertas fraquezas em um número de funções hash bastante populares na época, incluindo SHA-0, RIPEMD, e MD5. Isso colocou em questão a segurança a longo prazo de algoritmos derivados dessas funções hash - em particular, SHA-1 (uma versão mais forte de SHA-0), RIPEMD-128, e RIPEMD-160 (ambas versões mais fortes de RIPEMD). Nem SHA-0 nem RIPEMD costumam ser utilizadas desde que foram substituídas por suas versões mais fortes.

Desde 2009, as funções hash criptográficas mais utilizadas são [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5) e [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1). Entretanto, MD5 foi quebrada; um ataque contra essa função foi utilizada para quebrar [SSL](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \o "Transport Layer Security) em 2008.[[6]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-6)

As funções hash [SHA-0](https://pt.wikipedia.org/w/index.php?title=SHA-0&action=edit&redlink=1" \o "SHA-0 (página não existe)) e [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) foram desenvolvidas pela [NSA](https://pt.wikipedia.org/wiki/NSA" \o "NSA).

Em 12 de Agosto de 2004, uma colisão para o algoritmo SHA-0 foi anunciada por Joux, Carribault, Lemuet, e Jalby. Essa colisão foi encontrada a partir de uma generealização do ataque de Chabaud e Joux. Achar tal colisão tem complexidade 251 e levou cerca 80,000 horas de CPU em um [supercomputador](https://pt.wikipedia.org/wiki/Supercomputador" \o "Supercomputador) com 256 processadores [Itanium 2](https://pt.wikipedia.org/w/index.php?title=Itanium_2&action=edit&redlink=1" \o "Itanium 2 (página não existe)). (O equivalente a 13 dias de uso total do computador.)

Em Fevereiro de 2005, foi anunciado um ataque sobre SHA-1, que encontraria uma colisão em cerca de 269 operações hash, do que as esperadas 280 para uma função hash de 160 bits. Em Agosto de 2005, foi anunciado outro ataque a SHA-1, dessa vez, encontrando uma colisão em 263 operações. Apesar do SHA-1 possuir fraquezas teóricas,[[7]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-7)[[8]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-8) nenhuma colisão foi encontrada ainda. Não obstante, sugere-se que a quebra de SHA-1 torne-se prática em alguns anos, e que novas aplicações utilizem membros mais novos da família SHA, como [SHA-2](https://pt.wikipedia.org/wiki/SHA-2" \o "SHA-2), ou usar técnicas como aleatoriedade de hash[[9]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-9)[[10]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-10) que não exigem resistência a colisão.

Entretanto, para garantir a robustez a longo prazo de aplicações que utilizem função hash, uma competição de função hash da NIST foi realizada para projetar um substituto a SHA-2. Em 2 de Outubro de 2012, Keccak foi eleita a vencedora dessa competição. É esperado que uma versão desse algoritmo torne-se um padrão [FIPS](https://pt.wikipedia.org/wiki/FIPS" \o "FIPS) em 2014 sob o nome [SHA-3](https://pt.wikipedia.org/wiki/SHA-3" \o "SHA-3).[[11]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \l "cite_note-11)

um **gerador de número pseudo-aleatório criptograficamente seguro** (**CSPRNG**, na sigla em inglês) ou **gerador de números pseudoaleatórios criptográfico** (**CPRNG**, na sigla em inglês)[[1]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-1) é um [gerador de números pseudoaleatórios](https://pt.wikipedia.org/wiki/Gerador_de_n%C3%BAmeros_pseudoaleat%C3%B3rios" \o "Gerador de números pseudoaleatórios) (PRNG) com propriedades que o torna adequado para o uso na [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia).

Muitos aspectos da criptografia requerem números [aleatórios](https://pt.wikipedia.org/wiki/Aleatoriedade" \o "Aleatoriedade), como por exemplo:

* [geração de chave](https://pt.wikipedia.org/w/index.php?title=Gera%C3%A7%C3%A3o_de_chaves&action=edit&redlink=1" \o "Geração de chaves (página não existe))
* [nonces](https://pt.wikipedia.org/w/index.php?title=Nonce_criptogr%C3%A1fica&action=edit&redlink=1" \o "Nonce criptográfica (página não existe))
* [cifras de uso único](https://pt.wikipedia.org/wiki/One-time_pad" \o "One-time pad)
* [sais](https://pt.wikipedia.org/wiki/Sal_(criptografia)" \o "Sal (criptografia)) em certos esquemas de assinatura, como [ECDSA](https://pt.wikipedia.org/wiki/ECDSA" \o "ECDSA), [RSASSA-PSS](https://pt.wikipedia.org/w/index.php?title=RSASSA-PSS&action=edit&redlink=1" \o "RSASSA-PSS (página não existe))

A "qualidade" da aleatoriedade necessária para essas aplicações varia. Por exemplo, criar uma [nonce](https://pt.wikipedia.org/w/index.php?title=Nonce_criptogr%C3%A1fica&action=edit&redlink=1" \o "Nonce criptográfica (página não existe)) para algum [protocolo](https://pt.wikipedia.org/wiki/Protocolo_de_seguran%C3%A7a" \o "Protocolo de segurança) requer apenas singularidade. Por outro lado, a geração de uma [chave](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)) mestre requer mais qualidade, tal como [entropia](https://pt.wikipedia.org/wiki/Entropia_da_informa%C3%A7%C3%A3o" \o "Entropia da informação). E no caso de [cifras de uso único](https://pt.wikipedia.org/wiki/One-time_pad" \o "One-time pad), a garantia da [teoria da informação](https://pt.wikipedia.org/wiki/Teoria_da_informa%C3%A7%C3%A3o" \o "Teoria da informação) de sigilo perfeito apenas se mantém caso a geração da chave utilizar uma fonte verdadeiramente aleatória com alta entropia.

Idealmente, a geração de números aleatórios em CSPRNGs usa entropia para obter fontes de alta qualidade, que podem ser um [hardware](https://pt.wikipedia.org/wiki/Gerador_de_n%C3%BAmero_pseudoaleat%C3%B3rio_(hardware)" \o "Gerador de número pseudoaleatório (hardware)) ou um processo de sistema imprevisível; apesar de correlações inesperadas terem sido encontradas em tais diversos processos ostensivamente independentes. Do ponto de vista da informação teórica, a quantidade de aleatoriedade, a entropia que pode ser gerada, é igual a entropia fornecida pelo sistema. Porém, às vezes, em situações práticas, é necessário mais números aleatórios do que há entropia disponível. Além disso, o processo de extrair aleatoriedade de um sistema em execução pode ser lento na prática. Em tais casos, um CSPRNG pode ser utilizado. Um CSPRNG pode "ampliar" a entropia disponível sobre mais bits.

## Requisitos

Os requisitos de um PRNG padrão também são satisfeitos por um PRNG criptograficamente seguro, porém, não se pode dizer o mesmo do inverso. Os requisitos do CSPRNG podem ser divididos em dois grupos: primeiramente, eles devem passar por testes de aleatoriedade estatísticos; e segundo, eles devem ser resistentes a ataques, mesmo quando parte de seu estado inicial ou em execução chegue ao conhecimento do atacante.

* Todo CSPRNG deve satisfazer o [teste do próximo bit](https://pt.wikipedia.org/w/index.php?title=Teste_do_pr%C3%B3ximo_bit&action=edit&redlink=1" \o "Teste do próximo bit (página não existe)). Isto é, dados os primeiros k bits de uma sequência aleatória, não há algoritmo de tempo polinomial capaz de predizer o (k+1)-ésimo bit com probabilidade de sucesso maior que 50%. [Andrew Yao](https://pt.wikipedia.org/w/index.php?title=Andrew_Yao&action=edit&redlink=1" \o "Andrew Yao (página não existe)) provou em 1982 que um gerador que passe nesse teste, passará em qualquer outro teste estatístico de tempo polinomial para aleatoriedade.[[2]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-yao82-2)
* Todo CSPRNG deve resistir a "extensões de compromisso de estado". Caso parte ou o todo de seus estados seja revelado (ou predito corretamente), deve ser impossível de reconstruir o fluxo de números aleatórios antes dessa revelação. Além disso, caso haja uma entrada de entropia em execução, deve ser inviável utilizar-se de qualquer conhecimento sobre o estado da entrada para predizer condições futuras sobre o estado do CSPRNG.

Exemplo: Se o CSPRNG sob consideração produzir uma saída computando bits de [π](https://pt.wikipedia.org/wiki/Pi" \o "Pi) em sequência, começando de alguém ponto desconhecido dessa expansão binária, ele pode satisfazer o teste do próximo e ser considerado estatisticamente aleatório, já que π aparenta ser uma sequência aleatória. (Isso poderia ser garantido caso π seja um [número normal](https://pt.wikipedia.org/wiki/N%C3%BAmero_normal" \o "Número normal), por exemplo.) Entretanto, esse algoritmo não é criptograficamente seguro; um atacante pode determinar em qual bit de π (i.e. o estado do algoritmo) ele se encontra e será capaz de calcular todos os bits anteriores também.

A maioria dos PRNGs não são adequados para uso como CSPRNGs e não irão satisfazer ambos requisitos. Primeiro, enquanto muitos PRNGs dão saídas aparentemente aleatórias para alguns testes estatísticos, eles não são capazes de resistir a determinados métodos de [engenharia reversa](https://pt.wikipedia.org/wiki/Engenharia_reversa" \o "Engenharia reversa). Testes estatísticos especializados podem ser especialmente melhorados para mostrar que esses números aleatórios não são verdadeiramente aleatórios. Segundo, para a maioria dos PRNGs, quando o seu estado atual é revelado, todos os números aleatórios passados podem ser calculados, permitindo a um atacante ler todas as mensagens passadas, bem como as futuras.

CSPRNGs são projetados intencionalmente para resistir a esse tipo de [criptoanálise](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \o "Criptoanálise).

## Algumas informações básicas

Santha e Vazirani provaram que vários fluxos de bit com baixa aleatoriedade podem ser combinados para produzir um fluxo de bit quase-aleatório de melhor qualidade.[[3]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-santha-vazirani-3) Antes ainda, [John von Neumann](https://pt.wikipedia.org/wiki/John_von_Neumann" \o "John von Neumann) provou que um [simples algoritmo](https://pt.wikipedia.org/w/index.php?title=Extrator_de_Von_Neumann&action=edit&redlink=1" \o "Extrator de Von Neumann (página não existe)) é capaz de remover quantidade considerável de viés em qualquer fluxo de bits,[[4]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-neumann-random-4) que pode ser aplicado em cada fluxo de bit antes de usar qualquer variante do projeto de Santha-Vazirani. Essa área de atuação é chamada de *[extração de entropia](https://pt.wikipedia.org/w/index.php?title=Extra%C3%A7%C3%A3o_de_entropia&action=edit&redlink=1" \o "Extração de entropia (página não existe))* e é objeto de pesquisas em atividade (e.g., [N Nisan](https://pt.wikipedia.org/wiki/Noam_Nisan" \o "Noam Nisan), [S Safra](https://pt.wikipedia.org/w/index.php?title=Shmuel_Safra&action=edit&redlink=1" \o "Shmuel Safra (página não existe)), R Shaltiel, A Ta-Shma, C Umans, D Zuckerman).

## Projetos

Na discussão abaixo, os projetos de CSPRNG são divididos em três classes: 1) as baseadas em primitivas criptográficas como [cifras](https://pt.wikipedia.org/wiki/Cifra" \o "Cifra) e [hashes criptográficos](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica), 2) as baseadas em problemas matemáticos considerados difíceis, e 3) projetos de propósito-especial. Esse último, por muitas vezes introduz entropia adicional quando disponível e , a rigor, não são geradores de números aleatórios "puro", pois suas saídas não são determinadas pelo seus estados iniciais. Essa adição pode prevenir ataques mesmo que o estado inicial seja comprometido.

### **Projetos baseados em primitivas criptográficas**

* Uma [cifra de bloco](https://pt.wikipedia.org/wiki/Cifra_de_bloco" \o "Cifra de bloco) segura pode ser transformado em um CSPRNG executando-o em [modo contador](https://pt.wikipedia.org/wiki/Modo_de_opera%C3%A7%C3%A3o_(criptografia)" \l "Modo_CTR_(Counter)" \o "Modo de operação (criptografia)). Isso pode ser feito, escolhendo uma chave aleatória e encriptando-a junto com 0, depois com 1, depois com 2, etc. O contador também pode começar em um número arbitrário diferente de 0. Assumindo que uma cifra de bloco de *n*-bit, a saída pode ser distinguida de um dado aleatório após, aproximadamente, 2*n/2* blocos desde, seguindo o [paradoxo do aniversário](https://pt.wikipedia.org/wiki/Paradoxo_do_anivers%C3%A1rio" \o "Paradoxo do aniversário), blocos em colisão devem se tornar mais prováveis a partir desse ponto, onde uma cifra de blocos em modo CTR nunca dará como saída blocos idênticos. Para cifras de bloco de [64 bits](https://pt.wikipedia.org/wiki/64_bits" \o "64 bits), isso limita o tamanho da saída segura para alguns gigabytes, com blocos de 128 bits, a limitação é grande o suficiente para não afetar aplicações típicas.
* Um [hash](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica) criptograficamente seguro de um contador também pode atuar como um bom CSPRNG em alguns casos. Nesse caso, é necessário que o valor inicial do contador seja aleatório e secreto. Entretanto, há poucos estudos desses algoritmos utilizados dessa maneira, e pelo menos alguns autores alertam contra este uso.[[5]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-Malicious_Cryptography-5)
* A maioria das [cifras de fluxo](https://pt.wikipedia.org/wiki/Cifra_de_fluxo" \o "Cifra de fluxo) funcionam através da geração de fluxo de bits pseudoaleatórios que são combinados (quase sempre com uma operação [XOR](https://pt.wikipedia.org/wiki/Disjun%C3%A7%C3%A3o_exclusiva" \o "Disjunção exclusiva)) com o [purotexto](https://pt.wikipedia.org/wiki/Purotexto" \o "Purotexto); executando a cifra em modo contador irá retornar um novo fluxo pseudoaleatório, possivelmente com um período mais longo. A cifra só pode ser segura se o fluxo original for um bom CSPRNG, apesar desse nem sempre ser o caso (veja a cifra [RC4](https://pt.wikipedia.org/wiki/RC4" \o "RC4)). Novamente, o estado inicial deve ser mantido em segredo.

### **Projetos em teoria dos números**

* O algoritmo [Blum Blum Shub](https://pt.wikipedia.org/wiki/Blum_Blum_Shub" \o "Blum Blum Shub) possui uma prova de segurança baseado na dificuldade do problema dos [resíduos quadráticos](https://pt.wikipedia.org/wiki/Res%C3%ADduo_quadr%C3%A1tico" \o "Resíduo quadrático) (veja [lei da reciprocidade quadrática](https://pt.wikipedia.org/wiki/Lei_da_reciprocidade_quadr%C3%A1tica" \o "Lei da reciprocidade quadrática)). Como a única maneira conhecida de resolver esse problema é através da fatoração dos módulos, e, geralmente, é considerado que a dificuldade da [fatoração de inteiros](https://pt.wikipedia.org/wiki/Fatora%C3%A7%C3%A3o_de_inteiros" \o "Fatoração de inteiros) fornece uma prova segura condicional para o algoritmo Blum Blum Shub. Entretanto, esse algoritmo é muito ineficiente e por isso, impraticável, exceto quando medidas de extrema segurança são necessárias.
* O [algoritmo Blum-Micali](https://pt.wikipedia.org/w/index.php?title=Algoritmo_Blum-Micali&action=edit&redlink=1" \o "Algoritmo Blum-Micali (página não existe)) possui uma prova de segurança incondicional baseada na dificuldade do problema do [logaritmo discreto](https://pt.wikipedia.org/wiki/Logaritmo_discreto" \o "Logaritmo discreto), que também é muito ineficiente.
* Daniel Brown da [Certicom](https://pt.wikipedia.org/wiki/BlackBerry_(empresa)" \o "BlackBerry (empresa)) escreveu em 2006 uma prova de segurança para [Dual\_EC\_DRBG](https://pt.wikipedia.org/w/index.php?title=Dual_EC_DRBG&action=edit&redlink=1" \o "Dual EC DRBG (página não existe)), baseada na suposta dificuldade do *[problema decisional Diffie-Hellman](https://pt.wikipedia.org/w/index.php?title=Problema_Diffie-Hellman&action=edit&redlink=1" \o "Problema Diffie-Hellman (página não existe))*, o *problema do logaritmo-x*, e o *problema do ponto truncado*. A prova de 2006 assume, explicitamente, um *outlen* menor que o do Dual\_EC\_DRBG padrão, e que o P e o Q no Dual\_EC\_DRBG padrão (que, em 2013, foi revelado que a NSA teria introduzido um [backdoor](https://pt.wikipedia.org/wiki/Backdoor" \o "Backdoor) nesse sistema) são substituídos por valores que não possuem essa falha de backdoor.

### **Projetos especiais**

Há diversos PRNGs práticos que foram projetados para ser criptograficamente seguros, como:

* o [algoritmo Yarrow](https://pt.wikipedia.org/wiki/Algoritmo_Yarrow" \o "Algoritmo Yarrow) que tenta avaliar a qualidade entrópica de suas entradas. Yarrow é utilizado em [FreeBSD](https://pt.wikipedia.org/wiki/FreeBSD" \o "FreeBSD), [OpenBSD](https://pt.wikipedia.org/wiki/OpenBSD" \o "OpenBSD) e [Mac OS X](https://pt.wikipedia.org/wiki/Mac_OS_X" \o "Mac OS X) (e também em [/dev/random](https://pt.wikipedia.org/wiki//dev/random" \o "/dev/random)).
* o [algoritmo Fortuna](https://pt.wikipedia.org/wiki/Algoritmo_Yarrow" \o "Algoritmo Yarrow), o sucessor do Yarrow, que não tenta avaliar a qualidade entrópica de suas entradas.
* a função [CryptGenRandom](https://pt.wikipedia.org/w/index.php?title=CryptGenRandom&action=edit&redlink=1" \o "CryptGenRandom (página não existe)) fornecida pela [CryptoAPI](https://pt.wikipedia.org/w/index.php?title=CryptoAPI&action=edit&redlink=1" \o "CryptoAPI (página não existe)) da [Microsoft](https://pt.wikipedia.org/wiki/Microsoft" \o "Microsoft).
* [ISAAC](https://pt.wikipedia.org/w/index.php?title=ISAAC_(cifra)&action=edit&redlink=1" \o "ISAAC (cifra) (página não existe)) baseado em uma variante da cifra [RC4](https://pt.wikipedia.org/wiki/RC4" \o "RC4).
* [Algoritmo evolutivo](https://pt.wikipedia.org/wiki/Algoritmo_evolutivo" \o "Algoritmo evolutivo) baseado no Statistical Test Suite da [NIST](https://pt.wikipedia.org/wiki/NIST" \o "NIST).[[6]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-6)[[7]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-7)
* arc4random, uma [API](https://pt.wikipedia.org/wiki/Application_programming_interface" \o "Application programming interface), originária do [OpenBSD](https://pt.wikipedia.org/wiki/OpenBSD" \o "OpenBSD), que fornece acesso a diversos geradores de números aleatórios baseados em [RC4](https://pt.wikipedia.org/wiki/RC4" \o "RC4).
* o DRBG da [AES-CTR](https://pt.wikipedia.org/w/index.php?title=AES-CTR&action=edit&redlink=1" \o "AES-CTR (página não existe)) é frequentemente utilizado como um gerador de números aleatórios em sistemas que utilizam a encriptação [AES](https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard" \o "Advanced Encryption Standard).[[8]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-8)[[9]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-9)
* o padrão [ANSI](https://pt.wikipedia.org/wiki/American_National_Standards_Institute" \o "American National Standards Institute) X9.17 (*Financial Institution Key Management (atacado)*), que foi adotado como um padrão [FIPS](https://pt.wikipedia.org/wiki/Federal_Information_Processing_Standard" \o "Federal Information Processing Standard) também. Ele recebe como entrada um pacote de chaves *k* numa variação do [3DES](https://pt.wikipedia.org/wiki/3DES" \o "3DES) (ao invés de utilizar três chaves distintas, a terceira é igual a primeira) e (o valor inicial) de uma [semente aleatória](https://pt.wikipedia.org/wiki/Semente_aleat%C3%B3ria" \o "Semente aleatória) *s* de 64 bits.[[10]](https://pt.wikipedia.org/wiki/CSPRNG" \l "cite_note-10) Toda vez que um número aleatório é requisitado, ele:
  + Obtém a data/hora atual *D* com a precisão máxima possível.
  + Computa um valor temporário *t* = TDEA*k*(*D*)
  + Computa um valor aleatório *x* = TDEA*k*(*s* ⊕ *t*), onde ⊕ denota [ou exclusivo](https://pt.wikipedia.org/wiki/Disjun%C3%A7%C3%A3o_exclusiva" \o "Disjunção exclusiva) bit a bit.
  + Atualiza a semente *s* = TDEA*k*(*x* ⊕ *t*)

Obviamente, essa técnica é facilmente generalizada por qualquer bloco de cifra; [AES](https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard" \o "Advanced Encryption Standard) tem sido sugerido (Young and Yung, op cit, sect 3.5.1).

**One-time pad** (OTP), em português **cifra de uso único** ou **chave de uso único**, é uma técnica de [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia) que não pode ser quebrada se utilizada corretamente.

## Visão geral

Consiste num [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo) em que o *[purotexto](https://pt.wikipedia.org/wiki/Purotexto" \o "Purotexto)* é combinado, caractere por caractere, a uma chave secreta aleatória que para isso deve ter, no mínimo, o mesmo número de caracteres do purotexto. Para garantir que a criptografia seja imperscrutável, a chave só deve ser usada uma única vez, sendo imediatamente destruída após o uso. Foi primeiramente descrita pelo banqueiro e criptografista Frank Miller em [1882](https://pt.wikipedia.org/wiki/1882" \o "1882).[[1]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-1) Em [1917](https://pt.wikipedia.org/wiki/1917" \o "1917) foi re-inventada e, poucos anos depois, registrada.

A chave de uso único deriva da [cifra de Vernam](https://pt.wikipedia.org/w/index.php?title=Cifra_de_Vernam&action=edit&redlink=1" \o "Cifra de Vernam (página não existe)), cujo nome se deve a Gilbert Vernam, um de seus inventores. O sistema de Vernam consistia numa cifra que combinava uma mensagem com uma chave lida numa [fita perfurada](https://pt.wikipedia.org/wiki/Paper_tape" \o "Paper tape). Em sua forma original, o sistema de Vernam era vulnerável porque a fita com as chaves era reutilizada ao chegar ao fim. O uso único veio apenas posteriormente, quando Hoseph Maugorgne percebeu que se a fita fosse totalmente aleatória, a [criptoanálise](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \o "Criptoanálise) poderia ser impossível.[[2]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-2)

Se a chave for verdadeiramente aleatória, nunca reutilizada, e mantida em segredo, a cifra de uso único é imperscrutável. Também provou-se que toda a cifra teórica inquebrável deve usar chaves com as mesmas exigências que chaves de OTP.

A expressão "pad" do nome em inglês deve-se ao fato de que cifras de uso único comumente são utilizadas sob a forma de cadernetas, de modo que a folha superior possa ser arrancada e destruída após a utilização. Para facilitar sua ocultação, o bloco era às vezes muito pequeno, sendo quase impossível de ser utilizado sem uma [lupa](https://pt.wikipedia.org/wiki/Lupa" \o "Lupa). Imagens acessíveis na Internet mostram cadernetas da KGB, capturadas por agentes oponentes, que cabem na palma da mão ou em uma casca de [noz](https://pt.wikipedia.org/wiki/Noz" \o "Noz).

Existe muita ambiguidade na terminologia referente a esse tipo de cifra, já que alguns autores usam indistintamente os termos “cifra de Vernam” e “cifra de uso único”, enquanto outros utilizam a expressão "cifra de Vernam" para qualquer [cifra de fluxo](https://pt.wikipedia.org/wiki/Cifra_de_fluxo" \o "Cifra de fluxo), inclusive as baseadas em [CSPRNG](https://pt.wikipedia.org/wiki/CSPRNG" \o "CSPRNG).[[3]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-kahn-3)

## História

O banqueiro Frank Miller, em 1882, foi o primeiro a descrever o funcionamento de chaves de uso único para uso em mensagens telegráficas.[[4]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-bio-4)

Já no [século XX](https://pt.wikipedia.org/wiki/S%C3%A9culo_XX" \o "Século XX), em 1917, foi criado para a [AT&T](https://pt.wikipedia.org/wiki/AT&T" \o "AT&T) o sistema eléctrico de Gilbert Vernam, patenteado em [1919](https://pt.wikipedia.org/wiki/1919" \o "1919), que se baseava na tecnologia utilizada no [teletipo](https://pt.wikipedia.org/wiki/Teletipo" \o "Teletipo). Cada caractere na mensagem era electricamente combinado com uma chave contida numa [fita perfurada](https://pt.wikipedia.org/wiki/Paper_tape" \o "Paper tape). Joseph Mauborgne, então capitão das [forças armadas dos Estados Unidos](https://pt.wikipedia.org/wiki/For%C3%A7as_Armadas_dos_Estados_Unidos" \o "Forças Armadas dos Estados Unidos), percebeu que caso a sequência de caracteres na fita fosse totalmente aleatória, a criptoanálise seria mais difícil. Foi então introduzido o primeiro sistema de fita de uso único.

O próximo passo foi o desenvolvimento do sistema que utilizava a caderneta de papel. Durante muito tempo, os diplomatas usaram códigos e cifras para receber informações confidenciais e também reduzir os custos com [telégrafo](https://pt.wikipedia.org/wiki/Tel%C3%A9grafo" \o "Telégrafo). Utilizando os códigos, as palavras e as frases eram convertidas em grupos de números (geralmente 4 ou 5 dígitos) usando um livro de códigos que funcionava como uma espécie de dicionário. Para aumentar a segurança, números secretos podiam ser combinados (geralmente por meio de [adição modular](https://pt.wikipedia.org/wiki/Aritm%C3%A9tica_modular" \o "Aritmética modular)) com cada grupo de números antes da transmissão, com os números secretos sendo trocados periodicamente. Esse sistema era chamado de *superencryption*. No início dos anos 1920, três criptografistas alemães — Werner Kunze, Rudolf Schauffler e Erich Langlotz -— que estavam envolvidos em descodificar esses sistemas, perceberam que eles poderiam ser indecifráveis se um número escolhido aleatoriamente fosse usado para codificar cada grupo. Eles possuíam cadernetas de papel idênticas, com grupos de números aleatórios. Cada página tinha um número de série e oito linhas; por conseguinte cada linha tinha seis números de cinco dígitos cada um. Uma página seria usada para codificar uma mensagem e imediatamente destruída. O número serial da página seria enviado com a mensagem codificada, para que o receptor soubesse qual a página a usar. Para descodificar, o destinatário deveria apenas realizar o processo inverso ao de codificação, destruindo a página após terminar. O ministério das relações exteriores alemão pôs o sistema em operação em [1923](https://pt.wikipedia.org/wiki/1923" \o "1923).[[3]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-kahn-3)

A última descoberta relacionada à cifra de uso único deveu-se a [Claude Shannon](https://pt.wikipedia.org/wiki/Claude_Shannon" \o "Claude Shannon) que, nos [anos 1940](https://pt.wikipedia.org/wiki/Anos_1940" \o "Anos 1940), comprovou cientificamente a segurança desse sistema. Shannon relatou seus resultados em um relatório confidencial em [1945](https://pt.wikipedia.org/wiki/1945" \o "1945) e publicou-os abertamente em [1949](https://pt.wikipedia.org/wiki/1949" \o "1949).[[5]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-shannon-5) Na mesma época, [Vladimir Kotelnikov](https://pt.wikipedia.org/wiki/Vladimir_Kotelnikov" \o "Vladimir Kotelnikov) havia provado independentemente a segurança das chaves de uso único. Seus resultados foram registrados em um relatório que aparentemente permanece confidencial.[[6]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-kotelnikov-6)

## Segurança

As cifras de uso único são “informações teoricamente seguras” já que as mensagens cifradas não fornecem nenhuma informação sobre a mensagem original a um criptanalista, exceto seu tamanho máximo possível.[[7]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-7) Esta é uma noção muito forte de segurança primeiramente desenvolvida e provada matematicamente durante a 2ª Guerra Mundial por Claude Shannon, cujos resultados foram publicados no Jornal Técnico dos Laboratórios Bell em [1949](https://pt.wikipedia.org/wiki/1949" \o "1949).[[8]](https://pt.wikipedia.org/wiki/One-time_pad" \l "cite_note-Shannon-8) As cifras de uso único, se usadas corretamente são seguras mesmo se o adversário tiver capacidade computacional infinita.

A maioria de algoritmos convencionais de criptografia, seja simétrica ou assimétrica, usa padrões de teste complexos de substituição e transposição. Para o uso melhor destes atualmente, não se sabe se pode haver um procedimento criptanalítico que possa inverter (ou inverter parcialmente) estas transformações sem saber a chave usada durante a criptografia. Em termos práticos nenhum procedimento como esse é conhecido, embora já possam existir algoritmos de computador que poderiam fazê-lo em tempo “razoável”. Um dos problemas proeminentes centrais não-solucionados da informática carrega neste problema; se [P=NP](https://pt.wikipedia.org/wiki/P=NP" \o "P=NP) então seria possível que tais algoritmos pudessem ser encontrados, e se ele fossem procurados mais vigorosamente do que são hoje. Mesmo se não, os cripto sistemas atuais do indivíduo poderiam ainda ser quebrados.

Entretanto, a cifra de uso único não seria feita menos segura por uma prova que P=NP. A maioria dos observadores informados acreditam que P≠NP e, em qualquer o caso, muitos duvidam que esta pergunta tenha qualquer relevância prática ao projeto do algoritmo de criptanalise e criptografia.

## Aplicabilidade da cifra de uso único

A segurança perfeita teórica da cifra de uso único aplica-se somente em um ajuste teoricamente perfeito; nenhuma execução do mundo-real de todo o cripto-sistema pode fornecer a segurança perfeita porque as considerações práticas introduzem vulnerabilidades potenciais. Estas considerações práticas da segurança e da conveniência significam que cifra de uso único é, na prática, pouco usada.

* As cifras de uso único resolvem poucos problemas práticos atuais na criptografia. As cifras da alta qualidade que se submeteram à revisão pública rigorosa estão extensamente disponíveis e sua segurança não é considerada uma preocupação principal no presente. Tais cifras são quase sempre mais fáceis de empregar do que as cifras de uso único; a quantidade do material chave que deve ser gerada corretamente e distribuído firmemente é bem menor, e a chave pública criptográfica faz com que o problema seja mais fácil.
* Os números aleatórios da alta qualidade podem ser difíceis de gerar. As funções da geração do número aleatório na maioria das bibliotecas da língua de programação não são apropriadas para o uso criptográfico. Mesmo aqueles geradores que são apropriados para o uso criptográfico normal, incluindo /dev/random e muitos geradores do número aleatório , fazem algum uso das funções criptográficas cuja segurança não é provada.
* Distribuir as chaves cifra de uso único é inconveniente. Os meios de armazenamento tais como “thumb drivers”, “DVD-Rs” ou “digital audio players” pessoais podem ser usados para carregar uma cifra de uso único muito grande de um lugar para outro de uma maneira não-suspeita, mas nivelar assim que a necessidade de transportar a caderneta fisicamente é um muito peso comparando aos protocolos chaves da negociação de um cripto-sistema moderno de chave-pública, e estes meios não podem confiantemente serem apagados por nenhum meio brevemente de incineração. Uns 4,7 GB DVD-R completamente cheios de dados de cifra de uso único, se reduzi-los em partículas de 1 mm2 no tamanho, saem com mais de 100 kilobits dos dados em cada partícula. Além disso o risco do compromisso durante o trânsito é provavelmente muito maior na prática do que a probabilidade do compromisso para uma cifra tal como AES. Finalmente, o esforço necessitou controlar as escalas cifra de uso único ruins do material chave das grandes redes (networks). O número das cifras requeridas são maiores que o quadrado do número de usuários que trocam mensagens livremente entre si. Para uma comunicação entre somente duas pessoas, ou uma topologia da “rede estrela”, este é um problema menor.
* O material chave deve ser depois de usado firmemente disposto, assegurar que o material chave nunca seja reutilizado e proteger as mensagens emitidas. Porque o material chave deve ser transportado de um ponto final a outro, e persiste até que a mensagem esteja emitida ou recebida, pode ser mais vulnerável para a recuperação forense do que o plaintext que transiente ele protege.
* Como as usadas tradicionalmente, cifras de uso único não fornecem nenhuma autenticação da mensagem, a falta de uma frequente fonte de segurança faz com que ocorra falhas em sistemas de mundo-real. A natureza aditiva direta do “keystream” faz esta vulnerabilidade especialmente simples para explorar - por exemplo, um “atacante” que saiba que a mensagem contém “Meet Jane and me tomorrow at 3:30 pm (Encontre Jane e eu amanhã às 15h30)” pode substituir esse índice por todo o outro índice do mesmo comprimento, tal como “ 3:30 meeting is cancelled, stay home (O encontro de 15h30 esta cancelado casa, fique em casa”, sem ter o acesso à cifra de uso único. O hashing universal fornece uma maneira de autenticar as mensagens até um limite arbitrário de segurança mas este usa dados aleatórios adicionais da cifra, e remove a possibilidade de executar o sistema sem um computador.

Contudo, um cifra de uso único retêm algum interesse prático limitado:

* A cifra de uso único é o único criptosistema provido de segurança. Embora a maioria dos peritos tenham confiança em criptosistemas padrões para finalidades práticas, um pode não estar certo de que uma descoberta criptanalítica futura, ou uma descoberta no hardware de um computador tal como o [computador quântico](https://pt.wikipedia.org/wiki/Computador_qu%C3%A2ntico" \o "Computador quântico), não vão quebrá-los. cifra de uso único é um dos métodos mais práticos de criptografia onde uma ou ambas as obrigações dos partidos faz todo o trabalho à mão, sem o DAE (Dispositivo Automático de Entrada) de um computador; isto foi importante na era “pré-computador”, e poderia concebivelmente ainda ser útil nas situações onde há um computador é ilegal ou em incriminação ou onde os sistemas de computador operativos de confiança, não estão disponíveis.
* Fazer e usar uma cifra de uso único tem valor educacional. Nenhum equipamento especial é requerido e serve como uma introdução boa para as diversas ideias criptográficas.
* OTP pode ser usado, junto com um criptosistema mais padrão, em um esquema de super-criptografia. Adicionar uma camada de OTP é um exemplo especial de super-criptografia em que se pode provar que, desde que você usa as chaves que são estatística independente para cada camada (por exemplo RNGs independente), a combinação seria pelo menos tão forte quanto a camada a mais forte.

## Usos

Em algumas situações de espionagem, a chave de uso único pode ser útil porque pode ser computada à mão, com lápis e papel. Quase todas as outras cifras de igual qualidade são pouco práticas se não se utilizar um computador. Espiões podem receber suas cadernetas pessoalmente por meio de portadores. Entretanto, no mundo moderno os computadores e outros dispositivos capazes de realizar cálculos complexos são tão ubíquos que possuir um dispositivo capaz de executar criptografia convencional (por exemplo, um telefone que possa funcionar um software criptográfico escondido) não atrairá nenhuma suspeita.

As cifras de uso único foram usadas em circunstâncias especiais desde os anos 1900. O Serviço Diplomático da [República de Weimar](https://pt.wikipedia.org/wiki/Rep%C3%BAblica_de_Weimar" \o "República de Weimar) começou a usar o método por volta de [1920](https://pt.wikipedia.org/wiki/1920" \o "1920). Em [1923](https://pt.wikipedia.org/wiki/1923" \o "1923), era empregado para comunicações diplomáticas pelo serviço diplomático alemão. A quebra da frágil criptografia soviética pelos Ingleses, com publicação das mensagens por razões políticas em dois momentos nos anos 1920, parece ter levado a URSS a adotar chaves de uso único por volta de [1930](https://pt.wikipedia.org/wiki/1930" \o "1930). Os espiões da KGB são conhecidos também por terem continuado a usar lápis e papel nas cifras de uso único mais recentemente. Um exemplo é o do coronel Rudolf Abel, que foi preso e condenado em Nova York nos anos 1950, e os “Krogers” (Morris e Lona Cohen), que foram presos e condenados por espionagem no [Reino Unido](https://pt.wikipedia.org/wiki/Reino_Unido" \o "Reino Unido) nos anos 60. Ambos foram encontrados em posse cifras de uso único físicas.

Um grande número de nações usou sistemas de cifras de uso único para tráfego de informação confidencial. Leo Marks relatou que as operações executivas especiais britânicas usaram cifras de uso único para codificar a troca de informações entre seus escritórios. As cifras de uso único para uso com agentes ultramarinos foram introduzidas mais tarde, na guerra. Outras máquinas one-time da cifra de fita incluem as máquinas inglesas Rockex e Noreen.

A [NSA](https://pt.wikipedia.org/wiki/Ag%C3%AAncia_de_Seguran%C3%A7a_Nacional" \o "Agência de Segurança Nacional) descreve os sistemas *one-time* de fita como SIGTOT e 5-UCO como tendo sido usados para o tráfego da inteligência até a introdução da cifra eletrônica baseada em KW-26.

O “scrambler” de voz SIGSALY, da segunda guerra mundial, era uma forma de sistema de uso único. Ele adicionava ruído ao sinal no transmissor e removia-o no receptor. O padrão de ruído era distribuído aos dois canais sob a forma de gravações em [goma-laca](https://pt.wikipedia.org/wiki/Goma-laca" \o "Goma-laca) idênticas, das quais apenas duas cópias haviam sido feitas. Aconteceram problemas de sincronização que tiveram que ser resolvidos antes que o sistema pudesse ser usado.

O conhecido [Telefone vermelho](https://pt.wikipedia.org/wiki/Telefone_vermelho" \o "Telefone vermelho) entre Moscou e [Washington D.C.](https://pt.wikipedia.org/wiki/Washington,_D.C." \o "Washington, D.C.), estabelecido em [1963](https://pt.wikipedia.org/wiki/1963" \o "1963) após a [Crise dos Mísseis de Cuba](https://pt.wikipedia.org/wiki/Crise_dos_M%C3%ADsseis_de_Cuba" \o "Crise dos Mísseis de Cuba), utilizava teletipos protegidos por um sistema de fita de uso único comercial. Cada país preparou as fitas com as chaves usadas para codificar suas mensagens e entregou-as através de sua Embaixada no outro país. Uma vantagem original das cifras de uso único neste caso era que nenhum país teve que revelar métodos mais elaborados de criptografia ao outro.

Durante a [Invasão de Granada](https://pt.wikipedia.org/wiki/Invas%C3%A3o_de_Granada" \o "Invasão de Granada), em [1983](https://pt.wikipedia.org/wiki/1983" \o "1983), as forças dos Estados Unidos encontraram um acervo de pares de cadernetas de cifras de uso único em um armazém Cubano.

O código de comunicação tática BATCO do exército britânico é um sistema de lápis-e-papel do sistema cifra de uso único .

O material chave é fornecido nas folhas de papel que são mantidas em uma carteira plástica especial com um ponteiro de deslizamento que indica a última chave usada. As folhas novas são fornecidas diariamente e as velhas são destruídas. BATCO é usado em redes da voz do campo de batalha; as parcelas mais sensíveis de uma mensagem são codificadas e a mensagem cifrada são lidas para fora, letra por letra.

Uma noção relacionada é o código one-time, um sinal, usado somente uma vez, por exemplo o “alfa” para a “missão terminada” e o “Bravo” para a “missão falhada” não pode ser decifrado em nenhum sentido razoável da palavra. Compreender a mensagem requererá a informação adicional, frequentemente “profundidade” da repetição, ou alguma análise de tráfego. Entretanto, tais estratégias .Embora tais estratégias não sejam um “cifra de uso único” criptográficos em qualquer senso significativo.

## Façanhas

Cifras de uso único são seguras se geradas e usadas corretamente, mas erros pequenos podem fazer com que a criptoanálise seja bem sucedida.

Em [1944](https://pt.wikipedia.org/wiki/1944" \o "1944)-[1945](https://pt.wikipedia.org/wiki/1945" \o "1945), a agência da segurança do exército dos Estados Unidos conseguiu decifrar o sistema de cifras de uso único usado alemão GEE (Erskine, 2001). O GEE era inseguro porque as chaves não eram completamente aleatórias - a máquina usada gerar as cifras produziu saídas previsíveis.

Em 1945, os Estados Unidos descobriram que mensagens enviadas de Camberra a [Moscou](https://pt.wikipedia.org/wiki/Moscou" \o "Moscou) eram cifradas primeiramente usando um livro de códigos e depois uma cifra de uso único. Entretanto, a cifra de uso único usada era mesma usada por Moscou para mensagens de Washington, D.C.. Isso, somado ao facto de que algumas das mensagens de Camberra a Moscou incluíram documentos originais britânicos conhecidos pelo governo [estadunidense](https://pt.wikipedia.org/wiki/Estadunidense" \o "Estadunidense), permitiu que algumas das mensagens cifradas fossem quebradas.

Agências de espionagem [Soviéticas](https://pt.wikipedia.org/wiki/Uni%C3%A3o_Sovi%C3%A9tica" \o "União Soviética) empregaram cifras de uso único para comunicações secretas com seus agentes. A análise mostrou que as cifras usadas foram geradas por datilógrafos usando máquinas de escrever reais, o que não é um método realmente aleatório, já que determinadas sequências de teclas mais confortáveis acabam por ser mais prováveis do que outras. Mesmo assim, esse sistema provou-se eficaz na prática. Sem cópias do material chave usado, somente algum defeito no método da geração ou reutilização das chaves davam alguma esperança aos criptoanalistas. A partir dos anos 1940, as agências de inteligência dos Estados Unidos e do [Reino Unido](https://pt.wikipedia.org/wiki/Reino_Unido" \o "Reino Unido) conseguiram decodificar mensagens cifradas soviéticas para Moscou durante a segunda guerra graças a erros cometidos na geração e na distribuição do material chave. Uma hipótese é que o centro pessoal de Moscou, agindo às pressas devido à presença de tropas alemãs nos arredores da cidade entre o fim de 1941 e o início de 1942, acabou por produzir mais de uma cópia do mesmo material chave durante esse período. Essa operação, que durou mais de uma década e recebeu o nome de [VENONA](https://pt.wikipedia.org/w/index.php?title=VENONA&action=edit&redlink=1" \o "VENONA (página não existe)), produziu uma considerável quantidade de informação, incluindo informações sobre espiões atômicos soviéticos. Mesmo assim, somente uma pequena percentagem das mensagens interceptadas foi decifrada inteira ou parcialmente (cerca de mil de um total de cem mil).

A **criptografia de ponta-a-ponta** (*end-to-end encryption* ou *E2EE*) é um recurso de segurança que protege os dados durante uma troca de mensagens, de forma a que o conteúdo só possa ser acessado pelos dois extremos da [comunicação](https://pt.wikipedia.org/wiki/Comunica%C3%A7%C3%A3o" \o "Comunicação): o remetente e o destinatário. Usada atualmente em aplicativos como o [Telegram](https://pt.wikipedia.org/wiki/Telegram_(aplicativo)" \o "Telegram (aplicativo)) e o [WhatsApp](https://pt.wikipedia.org/wiki/WhatsApp" \o "WhatsApp), a ferramenta é uma implementação da [criptografia assimétrica](https://pt.wikipedia.org/wiki/Criptografia_assim%C3%A9trica" \o "Criptografia assimétrica) e garante que as informações não sejam interceptadas. Ninguém mais além dos envolvidos na conversa deve ter acesso ao conteúdo transmitido por meio da criptografia de ponta-a-ponta, nem mesmo as gestoras dos aplicativos.[[1]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-N%C3%A3o_nomeado-xdod-1-1)

A criptografia de ponta-a-ponta, como o nome sugere, protege os dados para que só possam ser lidos nas duas extremidades do processo, pelo remetente e pelo destinatário. Em teoria, ninguém mais pode ler os dados criptografados, incluindo hackers, governos e até mesmo o [servidor](https://pt.wikipedia.org/wiki/Servidor" \o "Servidor) por meio do qual os dados são enviados.

Dessa forma, a [tecnologia de segurança](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia) protege não só os usuários, mas também as empresas, que não podem ser pressionadas por autoridades a entregar informações privadas porque não as possuem em seus servidores. A criptografia de ponta-a-ponta não garante, no entanto, que informações não sejam acessadas caso um [hacker](https://pt.wikipedia.org/wiki/Hacker" \o "Hacker) tenha acesso ao dispositivo do usuário ou invada sua conta. O recurso protege apenas o processo de envio do conteúdo e garante que ele não seja lido quando interceptado no envio.

A criptografia de ponta-a-ponta não depende de uma ação do usuário. Ela é usada como recurso de aplicativos e softwares e tudo é automatizado pelos serviços. Além do Telegram e do WhatsApp, o [iMessage](https://pt.wikipedia.org/wiki/IMessage" \o "IMessage) usa esse tipo de criptografia, assim como o [Skype](https://pt.wikipedia.org/wiki/Skype" \o "Skype) e serviços de e-mail como [ProtonMail](https://pt.wikipedia.org/wiki/ProtonMail" \o "ProtonMail) e [Microsoft Outlook](https://pt.wikipedia.org/wiki/Microsoft_Outlook" \o "Microsoft Outlook). Uma desvantagem dos algoritmos de E2EE é que eles comumente são mais lentos que os métodos simétricos – mas a cada dia essa diferença fica menos relevante.[[1]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-N%C3%A3o_nomeado-xdod-1-1)

## Definição

A criptografia de ponta a ponta (E2EE) é um método de comunicação segura que impede que terceiros acessem dados enquanto são transferidos de um sistema ou dispositivo final para outro. No E2EE, os dados são criptografados no sistema ou no dispositivo do remetente e somente o destinatário pode descriptografá-lo. Ninguém no meio, seja ele um provedor de serviços de Internet, provedor de serviços de aplicativos ou hacker, pode lê-lo ou adulterá-lo.[[2]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-2)

Existem várias vantagens do E2EE sobre a criptografia padrão que a maioria dos serviços utiliza:

* Ele mantém seus dados seguros contra hacks. E2EE significa que menos pessoas têm acesso aos seus dados não criptografados. Mesmo que os hackers comprometam os servidores onde seus dados estão armazenados (por exemplo, hack de email do Yahoo), eles não podem descriptografar seus dados porque eles não possuem as chaves de descriptografia.
* Mantém seus dados privados. Se você usa o Gmail, o [Google](https://pt.wikipedia.org/wiki/Google" \o "Google) pode conhecer todos os detalhes íntimos que você colocou em seus e-mails e salvar seus e-mails, mesmo que você os exclua. O E2EE permite que você controle quem lê suas mensagens.
* É bom para a democracia. Todo mundo tem direito à privacidade. O E2EE protege a liberdade de expressão e protege ativistas, dissidentes e jornalistas perseguidos da intimidação.[[3]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-N%C3%A3o_nomeado-xdod-2-3)

Quando você usa o E2EE para enviar um [e-mail](https://pt.wikipedia.org/wiki/E-mail" \o "E-mail) ou uma mensagem para alguém, ninguém que monitora a rede pode ver o conteúdo de sua mensagem - não hackers, nem o governo, nem mesmo a empresa que facilite sua comunicação. Isso difere da criptografia que a maioria das empresas já usa, que protege apenas os dados em trânsito entre seu dispositivo e os servidores da empresa. Por exemplo, quando você envia e recebe um e-mail usando um serviço que não fornece E2EE, como o [Gmail](https://pt.wikipedia.org/wiki/Gmail" \o "Gmail) ou o [Hotmail](https://pt.wikipedia.org/wiki/Hotmail" \o "Hotmail), a empresa pode acessar o conteúdo de suas mensagens porque elas também contêm as chaves de criptografia. O E2EE elimina essa possibilidade porque o [provedor de serviços](https://pt.wikipedia.org/wiki/Provedor_de_servi%C3%A7os" \o "Provedor de serviços) não possui a chave de decodificação. Por causa disso, o E2EE é muito mais forte que a criptografia padrão.[[3]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-N%C3%A3o_nomeado-xdod-2-3)

## Criptografia Simétrica x Assimétrica

Os serviços de criptografia ponta-a-ponta se utilizam de um tipo de criptografia chamada assimétrica, que funciona de modo diferente à mais convencional, chamada simétrica. Esse modo de criptografia é tido como mais seguro e é o que garante que nem os provedores do serviço tenham acesso às chaves de encriptação e decriptação.

Na [criptografia simétrica](https://pt.wikipedia.org/wiki/Criptografia_sim%C3%A9trica" \o "Criptografia simétrica), a mais comum, apenas uma chave precisa ser aplicada para criptografar a mensagem em uma ponta e para recuperá-la na outra. Esse tipo de criptografia era usado em mensagens durante a Segunda Guerra Mundial, por exemplo. A chave é um tipo de código que permite reconstruir a mensagem original.

Quando se deseja ter mais segurança nas trocas, a solução é a criptografia assimétrica. Nela, dois tipos de chaves são usados para cada ponta da comunicação, uma chave pública e uma chave privada. As chaves públicas estão disponíveis para as ambas as partes e para qualquer outra pessoa, na verdade, porque todos compartilham suas chaves públicas antes da comunicação. Cada pessoa possui um par de chaves, que são complementares. Então, se, por exemplo, Pedro deseja enviar mensagens para Joana em um aplicativo com criptografia de ponta-a-ponta, ele vai usar a chave pública de Joana para criptografar a mensagem. O conteúdo só poderá ser descriptografado usando essa chave pública (de Joana) junto à chave privada dela, a que nem Pedro tem acesso. Essa chave privada é o único elemento que torna impossível para qualquer outro agente descriptografar a mensagem, já que ela não precisa ser compartilhada.[[1]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-N%C3%A3o_nomeado-xdod-1-1)

## Utilização em produtos populares

### **WhatsApp**

Após uma serie de ataques da justiça [brasileira](https://pt.wikipedia.org/wiki/Brasil" \o "Brasil) ao aplicativo e aos seus detentores, incluindo um pedido de prisão do Vice-Presidente do [Facebook](https://pt.wikipedia.org/wiki/Facebook" \o "Facebook) no país, foi inserido esse serviço no WhatsApp, com a explicação de que seria para garantir que não haveriam interceptações de mensagens, para salvaguardar a segurança de seus clientes. Antes do caso, existia uma implementação desde o final de 2014, do serviço de criptografia ponta a ponta: isso foi ativado por padrão em mensagens de texto enviadas entre dispositivos [Android](https://pt.wikipedia.org/wiki/Android" \o "Android), exceto nos chats em grupo. Após o caso, todas as mensagens em todas as plataformas serão protegidas dessa maneira.[[4]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-4) O caso quando o juiz Marcel Montalvão, da comarca de Lagarto (SE) determinou que as cinco principais operadoras do país bloqueassem o WhatsApp em todo o território nacional por 72 horas, devido a uma grande investigação sobre [tráfico de drogas](https://pt.wikipedia.org/wiki/Narcotr%C3%A1fico" \o "Narcotráfico) em que a justiça solicitou ao Facebook, dono do WhatsApp a quebra de sigilo do conteúdo das mensagens e a empresa vem recusou todos os pedidos da justiça brasileira a cooperar com as investigações, e inclusive não pagando as multas que lhe foram aplicadas.[[5]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-5)

### **Telegram**

Ao contrário do concorrente, o Telegram não utiliza criptografia de ponta a ponta por padrão. Isso permite que mensagens enviadas pelo app possam ser interpretadas corretamente ao serem interceptadas quando circulam pela infraestrutura da internet. O aplicativo [russo](https://pt.wikipedia.org/wiki/R%C3%BAssia" \o "Rússia) até tem uma função que aplica o uso de criptografia, o Chat Secreto, mas esta deve ser ativada pelos participantes da conversa para ser utilizada. Quando a criptografia de ponta a ponta está ativada, como acontece no WhatsApp, apenas o emissor e o receptor da mensagem têm acesso ao seu conteúdo. Isso porque só as duas pontas da comunicação (daí o nome da tecnologia) têm acesso a uma chave que decifra as informações, que trafegam "embaralhadas" pela internet. Assim, mesmo se forem interceptadas, as mensagens não podem ser decodificadas por hackers.[[6]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-6)

### **Signal**

O [Signal](https://pt.wikipedia.org/wiki/Signal" \o "Signal) é um aplicativo de mensagens instantâneas que funciona nos moldes do WhatsApp e do Telegram. A ONG que o desenvolve, a Signal Foundation, é dona do Signal Protocol, um protocolo que garante a criptografia de ponta-a-ponta e que é usado também pelo WhatsApp. Entre os fatores que levam o Signal a ser considerado mais seguro, estão os mecanismos que permitem a autodestruição das mensagens e os que verificam a identidade do interlocutor, através da geração de um [código QR](https://pt.wikipedia.org/wiki/C%C3%B3digo_QR" \o "Código QR) ou uma combinação numérica, então, esse mecanismo de identificação deve ser enviado ao interlocutor por outro meio de contato (e-mail ou telefone por exemplo), que já tenha sido comprovado pertencer realmente à pessoa com quem se deseja conversar no Signal. Outro aspecto que pode ser diferencial na hora de garantir a segurança é o mecanismo capaz de evitar a engenharia social – tipo de ataque que obtém informações por meio de persuasão ou ingenuidade da vítima.[[7]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-7)

### **Serviços de E-mail**

O ProtonMail, serviço de email favorito entre defensores da privacidade, devido ao uso automático de criptografia de ponta-a-ponta, lançou um novo produto com o objetivo de trazer o mesmo nível de privacidade para os clientes de email de desktop mais populares.[[8]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-8)O serviço fornece acesso via web ou [smartphone](https://pt.wikipedia.org/wiki/Smartphone" \o "Smartphone), e funciona de maneira bem simples. Se você enviar uma mensagem para outro usuário do ProtonMail, ela é criptografada direto no navegador. A empresa diz que tem um milhão de usuários cadastrados.Cada usuário tem que usar duas senhas para proteger suas informações: uma para [login](https://pt.wikipedia.org/wiki/Login" \o "Login), e outra para descriptografar seus e-mails. [[9]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-9)

## Desafios a Combater

### **Ataque man-in-middle**

O [man-in-the-middle](https://pt.wikipedia.org/wiki/Ataque_man-in-the-middle" \o "Ataque man-in-the-middle) (pt: Homem no meio, em referência ao atacante que intercepta os dados) é uma forma de ataque em que os dados trocados entre duas partes (por exemplo, você e o seu banco), são de alguma forma interceptados, registrados e possivelmente alterados pelo atacante sem que as vitimas se apercebam. Durante o ataque man-in-the-middle, a comunicação é interceptada pelo atacante e retransmitida por este de uma forma discricionária. O atacante pode decidir retransmitir entre os legítimos participantes os dados inalterados, com alterações ou bloquear partes da informação. Como os participantes legítimos da comunicação não se apercebem que os dados estão a ser adulterados tomam-nos como válidos, fornecendo informações e executando instruções por ordem do atacante.[[10]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-10)

### **Segurança na Ponta**

Dado que as chaves que garantem a segurança da encriptação estão em aparelhos eletrônicos que usam o serviço, o nível de segurança desses aparelhos é um risco a segurança geral do serviço, pois se forem hackeados ou violados, tais chaves podem ser roubadas ou pode-se apenas ler as mensagens decriptadas recebidas em tempo real.[[11]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-11)

## Controvérsias

Existem controversas no uso de tal serviço como o fato de impossibilitar a órgãos governamentais de investigar crimes cometidos por meio de programas que o utilizam. Já houve discussões diversas a respeito de responsabilizar empresas de tecnologia por não cooperarem com a lei e obstruírem investigações, junto com varias ações para quebra do sigilo do WhatsApp no Brasil e também alguns órgãos dos EUA estudam tornar a criptografia ponta a ponta ilegal, o que forçaria Apple, Google, Facebook e etc. a abandonarem a tecnologia. Uma outra possibilidade levantada seria a de implementar backdoors, que são meios de acesso criados pelas empresas detentoras de aplicativos e sistemas, coisa que já asseguraria as finalidades desses orgãos, mas não é nem cogitada pelas empresas de tecnologia que se utilizam desse tipo de criptografia.[[12]](https://pt.wikipedia.org/wiki/Criptografia_de_ponta-a-ponta" \l "cite_note-12)

A **criptoanálise** é a arte de tentar descobrir o texto cifrado e/ou a lógica utilizada em sua encriptação (chave). As pessoas que participam desse esforço são denominadas criptoanalistas. Da fusão da criptografia com a criptoanálise, forma-se a criptologia.

A criptoanálise representa o esforço de decodificar ou decifrar mensagens sem que se tenha o conhecimento prévio da chave secreta que as gerou. As diversas técnicas de criptoanálise são os caminhos que os analistas podem seguir para conseguir descobrir ou quebrar a codificação das mensagens que estiverem cifradas, e não apenas a simples decifração de uma mensagem.

Atualmente, a criptoanálise é utilizada a fim de se consultar as tentativas de quebra de segurança de outros tipos de algoritmos e de protocolos criptográficos. Em geral, a quebra ou então, a publicação de uma criptografia é um ato ilegal, previsto em lei. Entretanto, a criptoanálise exclui geralmente os ataques que não alvejam primeiramente fraquezas na criptografia real. Os métodos, tais como o suborno, coerção física, furto de dados, keylogger, embora esses tipos de ataque advém pela necessidade da segurança computacional, estão gradativamente tornando-se menos eficazes em relação a criptoanálise tradicional.

## História

A criptoanálise tem se desenvolvido junto com a [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia), e essa competição pode ser seguida com a história da criptografia, cifras novas que estão sendo projetadas para substituir projetos velhos, e as novas técnicas de criptoanálise inventadas para quebrar os esquemas melhorados. Na prática, são vistos como os dois lados da mesma moeda: a fim de criar uma criptografia segura.

### **Criptoanálise clássica**

[](https://pt.wikipedia.org/wiki/Ficheiro:Al-kindi-cryptanalysis.png)

Primeira página do manuscrito - Decifrando mensagens criptográficas de Al-Kindi's

Embora o termo criptoanálise seja relativamente recente (foi inventada por William Friedman, em 1920), métodos para quebrar códigos e cifras são muito mais velhos. A primeira explicação gravada, conhecida da criptoanálise, foi realizada pelo árabe Abu Yusuf Yaqub ibn Ishaq al-Sabbah Al-Kindi em Um Guia em Decifração de Mensagens Criptografadas.

A análise de frequência é a ferramenta básica para quebrar cifras clássicas. Em línguas naturais, determinadas letras do alfabeto aparecem mais frequentemente do que outras; em inglês, “e” é a letra mais comum em toda a amostra dada do texto. Similarmente, o dígrafo “th” é o par mais provável de letras, e assim por diante. A análise de frequência confia numa cifra que não esconde esses padrões de frequência. Por exemplo, numa cifra simples de substituição (em que cada letra é substituída simplesmente por outra), a letra mais frequente numa mensagem cifrada de um texto em português seria a que representa a letra “a”.

A análise de frequência confia tanto no conhecimento linguístico, como o faz nas estatísticas, mas, como as cifras se tornaram mais complexas, a matemática aproximou-se gradualmente da criptoanálise. Essa mudança era particularmente evidente durante a Segunda Guerra Mundial, quando os esforços para quebrar cifras de linha central requereram níveis novos de sofisticação matemática. Além disso, a automatização era aplicada pela primeira vez na criptoanálise com a máquina [Colossus](https://pt.wikipedia.org/wiki/Colossus_(computador)" \o "Colossus (computador)).

### **Criptoanálise moderna**

[](https://pt.wikipedia.org/wiki/Ficheiro:Bombe-wh.700px.jpg)

Replica de um dispositivo de Bomba

Com a computação usada para beneficiar a criptoanálise na Segunda Guerra Mundial, criaram-se novos possíveis métodos de criptografia de ordem de magnitude elevada e mais complexos do que eram antes.

Avaliando tudo, a criptografia moderna tornou muito mais intensa a criptoanálise do que os sistemas “papel-e-caneta” do passado, e parece agora ser superior à criptoanálise pura. As notas do historiador David Kahn: “Muitos são os criptossistemas oferecidos pelas centenas dos vendedores comerciais hoje que não podem ser quebrados por qualquer método conhecido da criptoanálise. Certamente que, em tais sistemas, mesmo um ataque de texto escolhido, em que um texto selecionado é combinado contra sua mensagem cifrada, não pode entregar a chave que destrava outras mensagens. Por isso, então, a criptoanálise está inoperante. Mas esse não é o fim da história. A criptoanálise pode estar inoperante, mas há – para misturar minhas metáforas – mais do que um único caminho para tirar a pele de um gato”.

Kahn pode ter sido prematuro em sua análise. As cifras fracas não foram extintas, e os métodos criptanalíticos empregados por agências de inteligência remanescem não publicados. Na academia, os projetos novos são apresentados regularmente, e também quebrados frequentemente: a cifra de bloco Madryga de 1984 foi tida como obsoleta devido aos ataques de mensagem cifrada somente em 1998. FEAL-4, proposta como um substituto para o DES, foi demolido por uma enxurrada de ataques vindos da comunidade acadêmica, muito deles completamente praticáveis. Na indústria, também, cifras não são livres de falhas: por exemplo, os algoritmos A5/1, A5/2 e CMEA, usados na tecnologia de telefone celular, podem ser quebrados em horas, minutos ou até em tempo real usando equipamentos computacionais amplamente disponíveis ao público. Em 2001, Wired Equivalent Privacy (WEP), um protocolo usado para a segurança de redes Wi-Fi, foi revelada ser suscetível de um ataque prático de chave relacionada. Essa fraqueza não era realmente do algoritmo em si, mas principalmente devido ao seu uso impróprio dentro do protocolo, de modo a comprometer sua força.

## Formas

* Criptoanálise linear é uma forma genérica de criptoanálise. A criptoanálise linear é uma das formas de ataque mais largamente utilizada em cifragem em blocos. A outra maneira é a criptoanálise diferencial.[[1]](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \l "cite_note-1)
* Inventado por [Martin Hellman](https://pt.wikipedia.org/wiki/Martin_Hellman" \o "Martin Hellman) e [Susan K. Langford](https://pt.wikipedia.org/w/index.php?title=Susan_K._Langford&action=edit&redlink=1" \o "Susan K. Langford (página não existe)) em 1994, o ataque por criptoanálise diferencial-linear é uma mistura da criptoanálise linear com a criptoanálise diferencial.
* Criptoanálise diferencial é uma forma genérica de criptoanálise aplicável primeiramente à [cifra em bloco](https://pt.wikipedia.org/w/index.php?title=Cifra_em_bloco&action=edit&redlink=1" \o "Cifra em bloco (página não existe)), mas também a [cifra em fluxo](https://pt.wikipedia.org/w/index.php?title=Cifra_em_fluxo&action=edit&redlink=1" \o "Cifra em fluxo (página não existe)) e a *cryptographic hash function*. Em geral, é o estudo de como as diferenças em uma entrada podem condicionar a diferença resultante na saída. No caso de uma cifra em bloco, isto se refere a um conjunto de técnicas para traçar diferenças através da rede de transformações, descobrindo onde a [cifra](https://pt.wikipedia.org/wiki/Cifra" \o "Cifra) exibe riscos não aleatórios e explorando tal propriedade para recuperar a [chave secreta](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)).

## Resultado

A criptoanálise bem sucedida influenciou sem dúvida a história; a habilidade de ler os segredos presumidos e as plantas de outros, pode ser uma vantagem decisiva. Por exemplo, na 1º Guerra Mundial, quebrar o [telegrama de Zimmermann](https://pt.wikipedia.org/wiki/Telegrama_Zimmermann" \o "Telegrama Zimmermann) era fundamental para trazer os Estados Unidos para guerra. Na segunda guerra mundial, a criptoanálise das cifras alemães - incluindo a [máquina Enigma](https://pt.wikipedia.org/wiki/M%C3%A1quina_Enigma" \o "Máquina Enigma) e a [cifra de Lorenz](https://pt.wikipedia.org/w/index.php?title=Cifra_de_Lorenz&action=edit&redlink=1" \o "Cifra de Lorenz (página não existe)) – foi conhecido como o ponto decisivo para chegar ao fim da guerra européia por alguns meses e determinar o resultado eventual. Os Estados Unidos beneficiaram-se também da criptoanálise do código ROXO japonês.

Os governos reconheceram por muito tempo os benefícios potenciais da criptoanálise para a inteligência militar e diplomática, e têm estabelecido as organizações dedicadas a quebrar os códigos e as cifras de outras nações, por exemplo, GCHQ e o NSA, as organizações que são ainda muito ativas hoje. Até 2004, relatou-se que os Estados Unidos tinham quebrado cifras Iranianas. Não se sabe, entretanto, se este era criptoanálise pura, ou se outros métodos foram utilizados.

## Ataques característicos

Os ataques criptanalíticos variam de potência e o quanto eles podem ameaçar os criptosistemas de mundo-real (real-world). Uma fraqueza de certificação é um ataque teórico que seja improvável de ser aplicado em qualquer situação de mundo-real (real-world); a maioria dos resultados encontrados nas pesquisas criptanalíticas modernas são destes tipos. Essencialmente, a importância de um ataque é dependente das respostas das três perguntas seguintes:

* Que conhecimento e potencial são necessários como pré-requisito?
* Quanta informação secreta adicional é deduzida?
* O que é a complexidade computacional?

### **Classificação do sucesso da criptoanálise**

Os resultados da criptoanálise podem também variar na utilidade. Por exemplo, o criptógrafo Lars Knudsen (1998) classificou vários tipos de ataque em cifras do bloco de acordo com a quantidade e a qualidade da informação secreta que foi descoberta:

* Ruptura total - o atacante deduz a chave secreta.
* Dedução global - o atacante descobre um algoritmo funcionalmente equivalente para a criptografia e descriptografia, mas sem aprender a chave.
* Dedução (local) do exemplo - o atacante descobre os plaintexts adicionais (ou as mensagens cifradas) não conhecidos previamente.
* Dedução da informação - o atacante ganha alguma informação de Shannon sobre os plaintexts (ou as mensagens cifradas) não conhecidos previamente.
* Algoritmo distinguindo - o atacante pode distinguir a cifra de uma permutação aleatória.

As considerações similares aplicam-se aos ataques em outros tipos de algoritmo criptográficos.

### **Complexibilidade**

Os ataques podem também ser caracterizados pela quantidade de recursos que requerem. Isto pode estar no formulário de:

* Tempo - o número “das operações primitivas” que devem ser executadas. Isto é completamente frouxamente; as operações primitivas podiam ser instruções de computador básicas, tais como a adição, XOR, deslocamento, e assim por diante, ou métodos inteiros de criptografia.
* Memória - a quantidade de armazenamento requerida para executar o ataque.
* Dados - a quantidade dos plaintexts e das mensagens cifradas requeridas.

## Criptoanálise de criptografia assimétrica

A criptografia assimétrica (ou a criptografia de chave pública) é a criptografia que usa duas chaves; uma confidencial, e uma pública. Tais cifras invariáveis confiam em problemas matemáticos “difíceis” como a base de sua segurança, assim um ponto óbvio do ataque é desenvolver métodos para resolver esses problemas.

A segurança da criptografia das duas chaves depende dos problemas matemáticos.

Os esquemas assimétricos são projetados em torno da dificuldade de resolver vários problemas matemáticos. Se um algoritmo melhorado puder resolver o problema, o sistema está enfraquecido então. Por exemplo, a segurança do esquema da troca da chave de Diffie-Hellman depende da dificuldade de calcular o logaritmo discreto. Em 1983, Don Coppersmith encontrou uma maneira mais rápida de encontrar logaritmos discretos (em determinados grupos), desse modo usam-se grupos criptográficos maiores (ou tipos diferentes de grupos). A segurança de [RSA](https://pt.wikipedia.org/wiki/RSA" \o "RSA) depende (em parte) da dificuldade da fatoração do inteiro - uma descoberta da fatoração acabaria com a segurança de RSA.

Em 1980, era possível fatorar um número de 50 dígitos em 10^12 operações de computador elementares. Em 1984 o último modelo de fatorar algoritmos tinha avançado a um ponto onde um número de 75 dígitos poderia ser fatorado em 10^12 operações. O avanço da tecnologia computacional significou também que as operações poderiam ser executadas com uma velocidade muito maior. A partir da lei de Moore, pode-se dizer que a velocidade dos computadores continuará a aumentar.

Um dígito de 150 números do tipo, usado uma vez em RSA foram fatorados. O esforço era maior do que acima, mas não era utilizado em computadores modernos rápidos. Pelo começo do século XXI, 150 números do dígito foram considerados já não um tamanho chave grande bastante para RSA. Os números com cem dígitos diversos eram considerados ainda difíceis de fatorar em 2005, embora os métodos continuarão provavelmente a melhorar ao longo do tempo, requerendo com que o tamanho chave mantenha o ritmo ou os algoritmos novos serão usados.

Uma outra característica que destinge os esquemas assimétricos é que, ao contrário dos ataques em criptosistemas simétricos, qualquer criptanálise tem a oportunidade de empregar o conhecimento ganho da chave pública.

## Aplicações computacionais *quantum*

Os computadores quantum têm potencial para serem usados na criptanálise. Porque os estados quantum podem existir na superposição, um paradigma novo para a computação é possível. Peter Shor dos laboratórios Bell provou a possibilidade, e as várias equipes têm demonstrado um outro aspecto da engenharia de computadores quantum há anos. Assim, somente a prova de projetos da possibilidade foi demonstrada.

Um **ataque de negação de serviço** (também conhecido como **DoS Attack**, um [acrônimo](https://pt.wikipedia.org/wiki/Acr%C3%B4nimo" \o "Acrônimo) em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa" \o "Língua inglesa) para *Denial of Service*), é uma tentativa de tornar os recursos de um sistema indisponíveis para os seus utilizadores. Alvos típicos são [servidores web](https://pt.wikipedia.org/wiki/Servidor_web" \o "Servidor web), e o ataque procura tornar as páginas hospedadas indisponíveis na [rede](https://pt.wikipedia.org/wiki/WWW" \o "WWW). Não se trata de uma invasão do sistema, mas sim da sua invalidação por sobrecarga. Os ataques de negação de serviço são feitos geralmente de duas formas:

* forçar o sistema vítima a reinicializar ou consumir todos os recursos (como [memória](https://pt.wikipedia.org/wiki/Mem%C3%B3ria_(computador)" \o "Memória (computador)) ou [processamento](https://pt.wikipedia.org/wiki/Processamento" \o "Processamento), por exemplo) de forma que ele não possa mais fornecer seu serviço;
* obstruir a mídia de comunicação entre os utilizadores e o sistema vítima de forma a não se comunicarem adequadamente.

## Ataque distribuído

Num ataque distribuído de negação de serviço (também conhecido como **DDoS**, um [acrônimo](https://pt.wikipedia.org/wiki/Acr%C3%B4nimo" \o "Acrônimo) em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa" \o "Língua inglesa) para *Distributed Denial of Service*), um [computador](https://pt.wikipedia.org/wiki/Computador" \o "Computador) mestre denominado *master* pode ter sob seu comando até milhares de computadores *zombies*, literalmente zumbis. Nesse caso, as tarefas de ataque de negação de serviço são distribuídas a um "exército" de máquinas escravizadas.

O ataque consiste em fazer com que os *zombies* (máquinas infectadas e sob comando do Mestre) se preparem para aceder a um determinado recurso num determinado servidor numa mesma hora de uma mesma data. Passada essa fase, na determinada hora, todos os *zombies* (ligados e conectados à rede) acedem ao mesmo recurso do mesmo servidor. Como servidores web possuem um número limitado de utilizadores que pode atender simultaneamente (*slots*), o grande e repentino número de requisições de acesso esgota esse número, fazendo com que o servidor não seja capaz de atender a mais nenhum pedido.

Dependendo do recurso atacado, o servidor pode chegar a reiniciar ou até mesmo ficar travado.

Vírus conhecidos criados para a distribuição de rotinas de ataque de negação de serviço incluem "Codered", "Slammer", "MyDoom", que escravizam o infectado. Ferramentas conhecidas de ataques DDos incluem "Fabi" (1998), "Blitznet", "Trin00" (jun/1999), "TFN" (ago/1999), "Stacheldraht" (set/1999), "Shaft", "TFN2K" (dez/1999), "Trank".

Uma outra estratégia de ataque seria por meio dos *[botnets](https://pt.wikipedia.org/wiki/Botnet" \o "Botnet)* praticando ataques de negação de serviço contra alvos remotos.

## Ataques de negação de serviços

O principal objetivo de um ataque de negação de serviço é deixar um recurso computacional inacessível aos seus utilizadores legítimos. As duas classes principais de métodos de ataque são diminuição de largura de banda e esgotamento de recursos. Ataques de diminuição de largura de banda são caracterizados pelos ataques por inundação e amplificação. Ataques de esgotamento de recursos são ataques que fazem uso indevido dos protocolos de comunicação de rede ou enviam pacotes de rede malformados.[[1]](https://pt.wikipedia.org/wiki/Ataque_de_nega%C3%A7%C3%A3o_de_servi%C3%A7o" \l "cite_note-ROCHARodrigoCO-1)

### **Ataques por inundação**

Ataques por inundação se caracterizam por enviarem um grande volume de tráfego ao sistema da vítima primária de modo a congestionar a sua banda. O impacto deste ataque pode variar entre deixar o sistema lento, derrubá-lo ou sobrecarregar a banda da rede da vítima. Ataques por inundação podem usar pacotes UDP ([User Datagram Protocol](https://pt.wikipedia.org/wiki/User_Datagram_Protocol" \o "User Datagram Protocol)) ou ICMP ([Internet Control Message Protocol](https://pt.wikipedia.org/wiki/Internet_Control_Message_Protocol" \o "Internet Control Message Protocol)).[[1]](https://pt.wikipedia.org/wiki/Ataque_de_nega%C3%A7%C3%A3o_de_servi%C3%A7o" \l "cite_note-ROCHARodrigoCO-1) Um dos grandes problemas desse ataque é que qualquer tipo de pacote pode ser usado, bastando ele ter permissão de andar pelos enlaces até o sistema visado, e dessa forma, consumindo toda a capacidade do servidor.

### **Ataques por amplificação**

Ataques por amplificação se caracterizam por enviarem requisições forjadas para uma grande quantidade de computadores ou para um [endereço IP](https://pt.wikipedia.org/wiki/Endere%C3%A7o_IP" \o "Endereço IP) de *broadcast*, que por sua vez responderão às requisições. Forjando o endereço IP de origem das requisições para o endereço IP da vítima primária fará com que todas as respostas sejam direcionadas para o alvo do ataque. O endereço IP de *broadcast* é um recurso encontrado em roteadores. Quando uma requisição possui um endereço IP de *broadcast* como endereço de destino, o [roteador](https://pt.wikipedia.org/wiki/Roteador" \o "Roteador) replica o pacote e o envia para todos os endereços IP dentro do intervalo de *broadcast*. Em ataques por amplificação, endereços de *broadcast* são usados para amplificar e refletir o tráfego de ataque, reduzindo então a banda da vítima primária.[[1]](https://pt.wikipedia.org/wiki/Ataque_de_nega%C3%A7%C3%A3o_de_servi%C3%A7o" \l "cite_note-ROCHARodrigoCO-1)

### **Ataques por exploração de protocolos**

Ataques por exploração de protocolos se caracterizam por consumir excessivamente os recursos da vítima primária explorando alguma característica específica ou falha de implementação de algum protocolo instalado no sistema da vítima.

Os principais ataques por exploração de protocolos são por uso indevido de pacotes TCP SYN (Transfer Control Protocol Synchronize) ou de pacotes TCP PUSHACK.[[1]](https://pt.wikipedia.org/wiki/Ataque_de_nega%C3%A7%C3%A3o_de_servi%C3%A7o" \l "cite_note-ROCHARodrigoCO-1)

Uma **função hash** é um [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo) que mapeia [dados](https://pt.wikipedia.org/wiki/Dados_(computa%C3%A7%C3%A3o)" \o "Dados (computação)) de comprimento variável para dados de comprimento fixo. Os valores retornados por uma função hash são chamados **valores hash**, **códigos hash**, **somas hash** (**hash sums**), **checksums** ou simplesmente **hashes**. Um uso é uma [estrutura de dados](https://pt.wikipedia.org/wiki/Estrutura_de_dados" \o "Estrutura de dados) chamada de [tabela hash](https://pt.wikipedia.org/wiki/Tabela_hash" \o "Tabela hash), amplamente usada em software de [computador](https://pt.wikipedia.org/wiki/Computador" \o "Computador) para consulta de dados rápida. Funções hash aceleram consultas a tabelas ou [bancos de dados](https://pt.wikipedia.org/wiki/Banco_de_dados" \o "Banco de dados) por meio da detecção de registros duplicados em um arquivo grande. Um exemplo é encontrar trechos similares em sequências de DNA. Eles também são úteis em [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia). Uma [função hash criptográfica](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica) permite verificar facilmente alguns mapeamentos de dados de entrada para um valor hash fornecido, mas se os dados de entrada são desconhecidos, é deliberadamente difícil reconstruí-lo (ou alternativas equivalentes) conhecendo o valor do hash armazenado. Isto é usado para assegurar a [integridade](https://pt.wikipedia.org/wiki/Integridade_de_dados" \o "Integridade de dados) de dados transmitidos e é o bloco de construção para [HMACs](https://pt.wikipedia.org/wiki/HMAC" \o "HMAC), que fornecem [autenticação de mensagem](https://pt.wikipedia.org/w/index.php?title=Autentica%C3%A7%C3%A3o_de_mensagem&action=edit&redlink=1" \o "Autenticação de mensagem (página não existe)).

Funções hash estão relacionadas a (e frequentemente confundidas com) [somas de verificação](https://pt.wikipedia.org/wiki/Soma_de_verifica%C3%A7%C3%A3o" \o "Soma de verificação) (checksums), [dígitos verificadores](https://pt.wikipedia.org/wiki/D%C3%ADgito_verificador" \o "Dígito verificador), [impressões digitais](https://pt.wikipedia.org/wiki/Impress%C3%A3o_digital_(computa%C3%A7%C3%A3o)" \o "Impressão digital (computação)), [compressão com perdas](https://pt.wikipedia.org/wiki/Compress%C3%A3o_com_perda_de_dados" \o "Compressão com perda de dados), [funções de aleatorização](https://pt.wikipedia.org/w/index.php?title=Fun%C3%A7%C3%A3o_de_aleatoriza%C3%A7%C3%A3o&action=edit&redlink=1" \o "Função de aleatorização (página não existe)), códigos de correção de erros e cifras. Apesar desses conceitos se sobreporem até certo ponto, cada um tem seus próprios usos e requisitos e são projetados e otimizados de maneiras diferentes. O banco de dados [HashKeeper](https://pt.wikipedia.org/w/index.php?title=HashKeeper&action=edit&redlink=1" \o "HashKeeper (página não existe)), mantido pelo American National Drug Intelligence Center, por exemplo, é mais apropriadamente descrito como um catálogo de arquivos de impressões digitais do que de valores de hash.

## Descrição

Um **hash** (ou escrutínio) é uma sequência de [bits](https://pt.wikipedia.org/wiki/Bit" \o "Bit) geradas por um [algoritmo de dispersão](https://pt.wikipedia.org/wiki/Algoritmo_de_dispers%C3%A3o" \o "Algoritmo de dispersão), em geral representada em base [hexadecimal](https://pt.wikipedia.org/wiki/Hexadecimal" \o "Hexadecimal), que permite a visualização em letras e [números](https://pt.wikipedia.org/wiki/N%C3%BAmero" \o "Número) (0 a 9 e A a F), representando um [nibble](https://pt.wikipedia.org/wiki/Nibble" \o "Nibble) cada. O conceito teórico diz que "hash é a transformação de uma grande quantidade de dados em uma pequena quantidade de informações".

Essa sequência busca identificar um arquivo ou informação unicamente. Por exemplo, uma mensagem de [correio eletrônico](https://pt.wikipedia.org/wiki/E-mail" \o "E-mail), uma senha, uma [chave criptográfica](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)) ou mesmo um [arquivo](https://pt.wikipedia.org/wiki/Arquivo" \o "Arquivo). É um método para transformar dados de tal forma que o resultado seja (quase) exclusivo. Além disso, funções usadas em criptografia garantem que não é possível a partir de um valor de hash retornar à informação original.

Como a sequência do hash é limitada, muitas vezes não passando de 512 bits, existem colisões (sequências iguais para [dados](https://pt.wikipedia.org/wiki/Dados" \o "Dados) diferentes). Quanto maior for a dificuldade de se criar colisões intencionais, melhor é o [algoritmo](https://pt.wikipedia.org/wiki/Algoritmo" \o "Algoritmo).

Uma [função](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_(matem%C3%A1tica)" \o "Função (matemática)) de hash recebe um valor de um determinado tipo e retorna um código para ele. Enquanto o ideal seria gerar identificadores únicos para os valores de entrada, isso normalmente não é possível: na maioria dos casos, o [contradomínio](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_(matem%C3%A1tica)" \l "Dom%C3%ADnio,_contradom%C3%ADnio_e_imagem" \o "Função (matemática)) de nossa função é muito menor do que o seu domínio, ou seja, � (o tipo de entrada) pode assumir uma gama muito maior de valores do que hash⁡(�) (o resultado da função de hash).

Os algoritmos de hash mais usados são os de 16 [bytes](https://pt.wikipedia.org/wiki/Byte" \o "Byte) (ou 128 bits, tamanho do *message digest*) [MD4](https://pt.wikipedia.org/wiki/MD4" \o "MD4) e [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5) ou o [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1), de 20 bytes (160 bits). Características de alguns algoritmos:

1. [MD4](https://pt.wikipedia.org/wiki/MD4" \o "MD4): Desenvolvido em 1990/91 por [Ron Rivest](https://pt.wikipedia.org/wiki/Ronald_Rivest" \o "Ronald Rivest), vários ataques foram detectados, o que fez com que o algoritmo fosse considerado frágil. Descrito na RFC 1320[[1]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash" \l "cite_note-1)[*[carece de fontes](https://pt.wikipedia.org/wiki/Wikip%C3%A9dia:Livro_de_estilo/Cite_as_fontes" \o "Wikipédia:Livro de estilo/Cite as fontes)*]
2. [MD5](https://pt.wikipedia.org/wiki/MD5" \o "MD5): O MD5 (Message-Digest algorithm 5) é um algoritmo de hash de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na [RFC 1321](https://tools.ietf.org/html/rfc1321), e muito utilizado por softwares com protocolo par-a-par ([P2P](https://pt.wikipedia.org/wiki/P2P" \o "P2P), ou Peer-to-Peer, em inglês), verificação de integridade e logins. Existem alguns métodos de ataque divulgados para o MD5.[[2]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash" \l "cite_note-2)[[3]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash" \l "cite_note-3)
3. [SHA-1](https://pt.wikipedia.org/wiki/SHA-1" \o "SHA-1) (*Secure Hash Algorithm*): Desenvolvido pelo [NIST](https://pt.wikipedia.org/wiki/NIST" \o "NIST) e [NSA](https://pt.wikipedia.org/wiki/NSA" \o "NSA). Já foram exploradas falhas no SHA.[[4]](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash" \l "cite_note-4)
4. [WHIRLPOOL](https://pt.wikipedia.org/wiki/Whirlpool_(criptografia)" \o "Whirlpool (criptografia)): função criptográfica de hash desenvolvida por [Paulo S. L. M. Barreto](https://pt.wikipedia.org/wiki/Paulo_S._L._M._Barreto" \o "Paulo S. L. M. Barreto) e por [Vincent Rijmen](https://pt.wikipedia.org/wiki/Vincent_Rijmen" \o "Vincent Rijmen) (co-autor do [AES](https://pt.wikipedia.org/wiki/Criptografia_de_chave_p%C3%BAblica" \o "Criptografia de chave pública)). A função foi recomendada pelo projeto [NESSIE](https://pt.wikipedia.org/w/index.php?title=NESSIE&action=edit&redlink=1" \o "NESSIE (página não existe)) (Europeu). Foi também adotado pelo [ISO](https://pt.wikipedia.org/wiki/Normas_ISO" \o "Normas ISO) e [IEC](https://pt.wikipedia.org/wiki/IEC" \o "IEC) como parte do padrão internacional [ISO 10118-3](https://pt.wikipedia.org/w/index.php?title=ISO_10118-3&action=edit&redlink=1" \o "ISO 10118-3 (página não existe)).

O processo é unidirecional e impossibilita descobrir o conteúdo original a partir do Hash. O valor de conferência ("[Soma de verificação](https://pt.wikipedia.org/wiki/Soma_de_verifica%C3%A7%C3%A3o" \o "Soma de verificação)") muda se um único bit for alterado, acrescentado ou retirado da mensagem.

As **Cifras de fluxo** são um tipo de [Algoritmos de chave simétrica](https://pt.wikipedia.org/wiki/Algoritmo_de_chave_sim%C3%A9trica" \o "Algoritmo de chave simétrica). Neste tipo de cifras, os bits originais são combinados com uma corrente de bits de [cifragem](https://pt.wikipedia.org/wiki/Encripta%C3%A7%C3%A3o" \o "Encriptação) (chamados de fluxo de chave) vindos de um gerador de dígitos pseudo-aleatório. Normalmente esta combinação é feita através de uma operação de [disjunção exclusiva](https://pt.wikipedia.org/wiki/Disjun%C3%A7%C3%A3o_exclusiva" \o "Disjunção exclusiva) ou XOR.[[1]](https://pt.wikipedia.org/wiki/Cifra_de_fluxo" \l "cite_note-1)

## Segurança

*Artigo principal:* [Ataques a Cifras de Fluxo](https://pt.wikipedia.org/wiki/Ataques_a_Cifras_de_Fluxo" \o "Ataques a Cifras de Fluxo)

Para que uma cifra de fluxo seja segura, o fluxo de dígitos pseudo-aleatórios deve ter um [período](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_peri%C3%B3dica" \o "Função periódica) grande. De preferência, este período deve ser maior que o tamanho da mensagem. Caso contrário, os ciclos deste fluxo de dígitos pseudo-aleatório vão facilitar a tarefa de [criptoanálise](https://pt.wikipedia.org/wiki/Criptoan%C3%A1lise" \o "Criptoanálise) se forem conhecidas amostras do conteúdo original.

Deve ainda ser impossível recuperar a [chave](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)) ou o estado interno a partir do fluxo de dígitos.

Os criptógrafos exigem que estes fluxos de dígitos pseudo-aleatórios sejam livres de padrões que permitam aos atacantes distinguir o fluxo de simples ruído aleatório. Não deve ser possível detectar relações entre fluxos que sejam gerados a partir de chaves relacionadas. Isto deve acontecer para todas as chaves (não deve haver chaves fracas) mesmo que o atacante consiga saber ou escolher alguma parte da mensagem original ou cifrada.

Para que seja mantida a segurança, quando se usa uma cifra de fluxo sincronizada, exige-se que nunca se reutilize o mesmo fluxo de dígitos. Isso significa, geralmente, que a cifra deve ser invocada com uma chave diferente.

Os programadores de aplicações devem reconhecer que a maioria das cifras de fluxo não garantem autenticidade, apenas privacidade. As mensagens cifradas podem ter sido modificadas durante a transmissão.

**Ou exclusivo** ou **disjunção exclusiva** é uma [operação lógica](https://pt.wikipedia.org/wiki/L%C3%B3gica" \o "Lógica) entre dois operandos que resulta em um [valor lógico verdadeiro](https://pt.wikipedia.org/wiki/Valor_de_verdade" \o "Valor de verdade) [se e somente se](https://pt.wikipedia.org/wiki/Se_e_somente_se" \o "Se e somente se) os dois operandos forem diferentes, ou seja, se um for verdadeiro e o outro for falso. É conhecido também pelas abreviações ***XOR*** ou ***EXOR*** (em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa" \o "Língua inglesa): *exclusive or*) e ainda por **XOU** ou **EXOU**.

### **Operação Bitwise**

Bitwise XOR é simplesmente uma operação XOR "bit a bit" sobre dois valores binários que indica "1", se e somente se um dos bits possuír valor lógico "1", e "0", se dois ou mais bits possuírem "1" como valor lógico, ou ainda se todos os bits possuírem "0" como valor lógico.

Na ciência da computação, a disjunção exclusiva tem vários usos, tais quais:

* dizer quando dois [bits](https://pt.wikipedia.org/wiki/Bits" \o "Bits) são diferentes;
* ele é um [circuito negador](https://pt.wikipedia.org/wiki/Nega%C3%A7%C3%A3o" \o "Negação). ex: 1 0 0 1 xor 1 1 1 1 equivale a 0 1 1 0;
* dizer se existe um número ímpar de bits 1s (�⊕�⊕�⊕�⊕�) é verdadeiro se um número ímpar de variáveis são verdadeiras).

Em circuitos lógicos, um somador pode ser implementado usando uma porta XOR para somar os números, e uma série de AND's, OR's e NOT's para os carry's ("vai um") de saída.

Em algumas arquiteturas de computadores, é mais eficiente armazenar um zero em um [registrador](https://pt.wikipedia.org/wiki/Registrador" \o "Registrador) realizando a operação XOR com ele mesmo, ao invés de carregar e armazenar o valor zero.

O XOR também é usado para misturar funções na [criptografia](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia), como por exemplo no [One-time Pad](https://pt.wikipedia.org/wiki/One-time_pad" \o "One-time pad).

O XOR também tem sua utilidade na segurança da informação armazenada em discos rígidos. A técnica [RAID](https://pt.wikipedia.org/wiki/RAID" \o "RAID) 3-6 usa o conceito lógico do operador XOR para em caso de falha em um dos discos, os dados sejam reconstituídos aplicando XOR ao dado armazenado no disco de backup. Pode-se dizer que o XOR realiza uma operação reversível, pois se aplicarmos �⊕� e reaplicarmos o XOR no resultado com o mesmo B, teremos A, como vemos a seguir:

(�⊕�)⊕�≡�

E é baseado na reversibilidade da operação XOR que a técnica recupera os dados armazenados no disco de backup.

O [algoritmo XOR Swap](https://pt.wikipedia.org/wiki/Algoritmo_XOR_Swap" \o "Algoritmo XOR Swap) usa a lógica do conectivo XOR a fim de trocar os valores numéricos de 2 variáveis.

O **Transport Layer Security** (**TLS**),[[nota 1]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-1) assim como o seu antecessor *Secure Sockets Layer* (**SSL**),[[nota 2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-2)[[1]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-3) é um [protocolo de segurança](https://pt.wikipedia.org/wiki/Protocolo_de_seguran%C3%A7a" \o "Protocolo de segurança) projetado para fornecer [segurança nas comunicações](https://pt.wikipedia.org/w/index.php?title=Seguran%C3%A7a_nas_comunica%C3%A7%C3%B5es&action=edit&redlink=1" \o "Segurança nas comunicações (página não existe)) sobre uma [rede de computadores](https://pt.wikipedia.org/wiki/Rede_de_computadores" \o "Rede de computadores).[[2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:02-4) Várias versões do protocolo encontram amplo uso em aplicativos como [navegação na web](https://pt.wikipedia.org/w/index.php?title=Navega%C3%A7%C3%A3o_na_web&action=edit&redlink=1" \o "Navegação na web (página não existe)), [email](https://pt.wikipedia.org/wiki/Email" \o "Email), [mensagens instantâneas](https://pt.wikipedia.org/wiki/Mensagens_instant%C3%A2neas" \o "Mensagens instantâneas) e [voz sobre IP](https://pt.wikipedia.org/wiki/Voz_sobre_IP" \o "Voz sobre IP) (VoIP). Os sites podem usar o TLS para proteger todas as comunicações entre seus servidores e [navegadores web](https://pt.wikipedia.org/wiki/Navegador_web" \o "Navegador web).

O protocolo TLS visa principalmente fornecer [privacidade](https://pt.wikipedia.org/wiki/Privacidade" \o "Privacidade) e [integridade de dados](https://pt.wikipedia.org/wiki/Integridade_de_dados" \o "Integridade de dados) entre dois ou mais aplicativos de computador que se comunicam.[[2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:02-4) Quando protegidos por TLS, conexões entre um cliente (por exemplo, um navegador da Web) e um servidor (por exemplo, wikipedia.org) devem ter uma ou mais das seguintes propriedades:

* A conexão é *privada* (ou *segura*) porque a [criptografia simétrica](https://pt.wikipedia.org/wiki/Algoritmo_de_chave_sim%C3%A9trica" \o "Algoritmo de chave simétrica) é usada para [criptografar](https://pt.wikipedia.org/wiki/Encripta%C3%A7%C3%A3o" \o "Encriptação) os dados transmitidos. As [chaves](https://pt.wikipedia.org/wiki/Chave_(criptografia)" \o "Chave (criptografia)) para essa criptografia simétrica são geradas exclusivamente para cada conexão e são baseadas em um [segredo compartilhado](https://pt.wikipedia.org/w/index.php?title=Segredo_compartilhado&action=edit&redlink=1" \o "Segredo compartilhado (página não existe)) que foi negociado no início da [sessão](https://pt.wikipedia.org/wiki/Sess%C3%A3o_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)" \o "Sessão (ciência da computação)) (veja [§ Handshake TLS](https://pt.wikipedia.org/wiki/Usu%C3%A1rio:Vin%C3%ADcius_Kury/Transport_Layer_Security" \l "Handshake_TLS" \o "Usuário:Vinícius Kury/Transport Layer Security)). O servidor e o cliente negociam os detalhes de qual algoritmo de criptografia e chaves criptográficas usar antes que o primeiro byte de dados seja transmitido (ver § Algoritmo abaixo). A negociação de um segredo compartilhado é segura (o segredo negociado não está disponível para bisbilhoteiros e não pode ser obtido, mesmo por um invasor que se coloque no meio da conexão) e confiável (nenhum invasor pode modificar as comunicações durante a negociação sem ser detectado).
* A identidade das partes em comunicação pode ser autenticada usando criptografia de chave pública. Essa autenticação pode ser opcional, mas geralmente é necessária para pelo menos uma das partes (geralmente o servidor).
* A conexão é confiável porque cada mensagem transmitida inclui uma verificação de integridade de mensagem usando um código de autenticação de mensagem para evitar perda não detectada ou alteração dos dados durante a transmissão.[[2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:02-4)

Além das propriedades acima, a configuração cuidadosa do TLS pode fornecer propriedades adicionais relacionadas à privacidade, como sigilo de encaminhamento, garantindo que qualquer divulgação futura de chaves de criptografia não possa ser usada para descriptografar as comunicações TLS registradas no passado.[[3]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-5)

O TLS suporta muitos métodos diferentes para trocar chaves, criptografar dados e autenticar a integridade da mensagem (consulte § Algoritmo abaixo). Como resultado, a configuração segura do TLS envolve muitos parâmetros configuráveis ​​e nem todas as opções fornecem todas as propriedades relacionadas à privacidade descritas na lista acima (consulte § Troca de chave (autenticação), § Segurança de codificação e § Tabelas de integridade de dados).

Tentativas foram feitas para subverter aspectos da segurança das comunicações que o TLS procura fornecer, e o protocolo foi revisado várias vezes para lidar com essas ameaças de segurança (ver § Segurança). Os desenvolvedores de navegadores da Web também revisaram seus produtos para se defenderem de potenciais pontos fracos de segurança depois que eles foram descobertos (veja o histórico de suporte a TLS / SSL dos navegadores da Web).[[2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:02-4)

O protocolo TLS compreende duas camadas: o registro TLS e os protocolos de handshake TLS.

O TLS é um padrão proposto pela IETF (Internet Engineering Task Force), definido pela primeira vez em 1999, e a versão atual é o TLS 1.3 definido no [RFC 8446](https://tools.ietf.org/html/rfc8446) (agosto de 2018). O TLS baseia-se nas especificações SSL anteriores (1994, 1995, 1996) desenvolvidas pela Netscape Communications[[4]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-RFC61012-6) para adicionar o protocolo HTTPS ao navegador da Web Navigator.

## Descrição

Aplicações [cliente-servidor](https://pt.wikipedia.org/wiki/Modelo_cliente%E2%80%93servidor" \o "Modelo cliente–servidor) fazem uso do [protocolo](https://pt.wikipedia.org/wiki/Protocolo_de_seguran%C3%A7a" \o "Protocolo de segurança) TLS para se comunicar através de uma rede de forma a prevenir a interceptação e adulteração da informação.

Uma vez que aplicações podem se comunicar tanto através de TLS (ou SSL) como sem ele, é necessário que o [cliente](https://pt.wikipedia.org/wiki/Cliente_(computa%C3%A7%C3%A3o)" \o "Cliente (computação)) sinalize ao [servidor](https://pt.wikipedia.org/wiki/Servidor_(computa%C3%A7%C3%A3o)" \o "Servidor (computação)) para a configuração de uma conexão TLS[[5]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-7). Uma das maneiras de se obter isso é utilizar números de [porta](https://pt.wikipedia.org/wiki/Porta_(redes_de_computadores)" \o "Porta (redes de computadores)) diferentes, por exemplo a porta 443 para [HTTPS](https://pt.wikipedia.org/wiki/Hyper_Text_Transfer_Protocol_Secure" \o "Hyper Text Transfer Protocol Secure). Outro mecanismo é uma requisição específica por parte do cliente ao servidor para uma transição para a conexão TLS; por exemplo, ao fazer uma requisição STARTTLS ao utilizar protocolos de email.

Uma vez que o cliente e o servidor concordaram quanto ao uso do TLS, eles negociam uma conexão de [estado](https://pt.wikipedia.org/wiki/Estado_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)" \o "Estado (ciência da computação)) por meio de um procedimento de [handshake](https://pt.wikipedia.org/wiki/Handshake" \o "Handshake)[[6]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-8). Os protocolos utilizam um handshake com uma [chave pública](https://pt.wikipedia.org/wiki/Criptografia_de_chave_p%C3%BAblica" \o "Criptografia de chave pública) para estabelecer as configurações de criptografia e uma chave de sessão única compartilhada através da qual toda a comunicação é criptografada utilizando uma [chave simétrica](https://pt.wikipedia.org/wiki/Algoritmo_de_chave_sim%C3%A9trica" \o "Algoritmo de chave simétrica). Durante esse handshake, o cliente e o servidor concordam a respeito dos vários parâmetros necessários para estabelecer a segurança da conexão:

* O handshake é iniciado quando o cliente se conecta a um servidor habilitado para TLS requisitando uma conexão segura e apresentando uma lista de algoritmos suportados ([cifras](https://pt.wikipedia.org/wiki/Cifra" \o "Cifra) e [funções hash](https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_hash_criptogr%C3%A1fica" \o "Função hash criptográfica)).
* A partir dessa lista, o servidor seleciona uma cifra e uma função hash para as quais também tenha suporte e notifica ao cliente a decisão.
* O servidor então geralmente apresenta informações de identificação na forma de um [certificado digital](https://pt.wikipedia.org/wiki/Certificado_digital" \o "Certificado digital). O certificado contém o nome do servidor, a [autoridade de certificação](https://pt.wikipedia.org/wiki/Autoridade_de_certifica%C3%A7%C3%A3o" \o "Autoridade de certificação) (CA) que concedeu o certificado, e a chave pública do servidor.
* O cliente confirma a validade do certificado antes de continuar.
* Para gerar as chaves de sessão utilizadas na conexão segura, o cliente:
  + criptografa um [número aleatório](https://pt.wikipedia.org/wiki/N%C3%BAmero_aleat%C3%B3rio" \o "Número aleatório) com a chave pública recebida e envia o resultado ao servidor (o único capaz de descriptografar a mensagem com sua chave privada); ambas as partes então fazem uso do número aleatório para gerar uma chave de sessão única para a criptografia subsequente dos dados durante a sessão, ou
  + inicia uma [troca de chaves de Diffie-Hellman](https://pt.wikipedia.org/wiki/Troca_de_chaves_de_Diffie%E2%80%93Hellman" \o "Troca de chaves de Diffie–Hellman) para gerar seguramente uma chave de sessão aleatória e única, utilizada para criptografar e descriptografar os dados e que ainda possui a propriedade de *[foward secrecy](https://pt.wikipedia.org/wiki/Perfect_Foward_Secrecy" \o "Perfect Foward Secrecy)*: caso a chave privada do servidor seja vazada no futuro, ela é incapaz de descriptografar a sessão atual, mesmo que esta tenha sido interceptada e gravada por um terceiro.

Isso conclui o handshake e inicia a conexão segura, que é criptografada e descriptografada com a chave de sessão até o fim da conexão. Se qualquer um dos passos acima falhar, o handshake TLS também falha e a conexão não é criada.

Os protocolos TLS e SSL não se encaixam perfeitamente em nenhuma camada dos modelos [OSI](https://pt.wikipedia.org/wiki/Modelo_OSI" \o "Modelo OSI) ou [TCP/IP](https://pt.wikipedia.org/wiki/TCP/IP" \o "TCP/IP).[[7]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:1-9)[[8]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:2-10) O TLS é implementado "sobre um protocolo de comunicação confiável (por exemplo, o TCP)", o que implica que ele está acima da camada de transporte. Ele serve para criptografar as camadas superiores, o que normalmente seria função da camada de apresentação. Contudo, as aplicações geralmente fazem uso do TLS como se fosse uma camada de transporte,[[7]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:1-9)[[8]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:2-10) mesmo que essas aplicações devam controlar ativamente o início dos procedimentos de handshake e o gerenciamento dos certificados de autenticação compartilhados.[[2]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-:02-4)

[](https://pt.wikipedia.org/wiki/Ficheiro:TLS_protocol_stack.svg)

Protocolo TLS

## Funcionamento

O [servidor](https://pt.wikipedia.org/wiki/Servidor" \o "Servidor) do [site](https://pt.wikipedia.org/wiki/Site" \o "Site) que está sendo acessado envia uma [chave pública](https://pt.wikipedia.org/wiki/Criptografia_de_chave_p%C3%BAblica" \o "Criptografia de chave pública) ao [browser](https://pt.wikipedia.org/wiki/Navegador_(inform%C3%A1tica)" \o "Navegador (informática)), usada por este para enviar uma chave secreta simetrica, criada aleatoriamente. Desta forma, fica estabelecida a troca de dados [criptografados](https://pt.wikipedia.org/wiki/Criptografia" \o "Criptografia) entre dois [computadores](https://pt.wikipedia.org/wiki/Computador" \o "Computador).Baseia-se no [protocolo](https://pt.wikipedia.org/wiki/Protocolo_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)" \o "Protocolo (ciência da computação)) [TCP](https://pt.wikipedia.org/wiki/Transmission_Control_Protocol" \o "Transmission Control Protocol) da suíte [TCP/IP](https://pt.wikipedia.org/wiki/TCP/IP" \o "TCP/IP) e utiliza-se do conceito introduzido por [Diffie-Hellman](https://pt.wikipedia.org/wiki/Diffie-Hellman" \o "Diffie-Hellman) nos anos 70 (criptografia de chave pública) e [Phil Zimmermann](https://pt.wikipedia.org/wiki/Phil_Zimmermann" \o "Phil Zimmermann) (criador do conceito [PGP](https://pt.wikipedia.org/wiki/PGP" \o "PGP)).

## História e desenvolvimento

A primeira versão foi desenvolvida pela [Netscape](https://pt.wikipedia.org/wiki/Netscape" \o "Netscape) em 1994. O SSL versão 3.0 foi lançado em [1996](https://pt.wikipedia.org/wiki/1996" \o "1996), e serviu posteriormente de base para o desenvolvimento do TLS versão 1.0, um protocolo padronizado da [IETF](https://pt.wikipedia.org/wiki/Internet_Engineering_Task_Force" \o "Internet Engineering Task Force) originalmente definido pelo [RFC 2246](https://tools.ietf.org/html/rfc2246). Grandes instituições financeiras como [Visa](https://pt.wikipedia.org/wiki/Visa" \o "Visa), [MasterCard](https://pt.wikipedia.org/wiki/MasterCard" \o "MasterCard), [American Express](https://pt.wikipedia.org/wiki/American_Express" \o "American Express), dentre outras, aprovaram o SSL para [comércio eletrônico](https://pt.wikipedia.org/wiki/Com%C3%A9rcio_eletr%C3%B4nico" \o "Comércio eletrônico) [seguro](https://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o" \o "Segurança da informação) na Internet.O SSL opera de forma modular, possui *design* extensível e apresenta compatibilidade entre pares com versões diferentes do mesmo.O SSL executa a autenticação das 2 partes envolvidas nas comunicações (cliente e servidor) baseando-se em [certificados digitais](https://pt.wikipedia.org/wiki/Certificado_digital" \o "Certificado digital).

## Segurança

Ataques significativos contra TLS/SSL estão listados abaixo.

Em fevereiro de 2015, a IETF emitiu um informativo RFC[[9]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-11) resumindo os vários ataques conhecidos contra TLS/SSL.

A [Apple](https://pt.wikipedia.org/wiki/Apple" \o "Apple) corrigiu a vulnerabilidade BEAST implementando a divisão 1/n-1 e ativando-a por padrão no OS X Mavericks, lançado em 22 de outubro de 2013.[[10]](https://pt.wikipedia.org/wiki/Transport_Layer_Security" \l "cite_note-12)