2019/2020

P.PORTO

Escola Superior de Tecnologia e Gestão

Licenciatura de Segurança Informática em Redes de Computadores

Criptografia Aplicada

Trabalho Prático #1

Soluções e aplicações que utilizem funcionalidades criptográficas: Mobile e PC

Docente:

Gonçalo Hermenegildo (gjh)

Trabalho realizado por:

Francisco Spínola (8180140)

Rúben Freitas (8180258)

Índice

Introdução **3**

Categorias **4**

C1 – Encriptação 4

AxCrypt 4

Telegram 6

C2 – Autenticação/Identificação 9

Google Authenticator 9

SSH 10

C3 – Outros 14

Microsoft Intune 14

Addigy 16

Algoritmos **18**

AES 18

PBKDF2 18

HMAC 18

RSA 19

SHA1 19

Blowfish 20

MTProto 2.0 20

CAST-128 21

Conclusão **22**

Referências **23**

Introdução

Atualmente, a evolução tecnológica com que estamos a lidar levanta diversas questões a nível de privacidade e segurança: dados confidenciais, informações de identificação pessoal, dados de saúde, informações pessoais, propriedade intelectual, dados e sistemas de informações governamentais, etc.

É importante estarmos a par dos riscos que tomamos ao aceder a determinadas tecnologias, sejam elas serviços ou aplicações.

Criptografia é o estudo dos princípios e técnicas pelas quais a informação pode ser transformada da sua forma original para outra ilegível, de forma que possa ser conhecida apenas por seu destinatário, o que a torna difícil de ser lida por alguém não autorizado. Assim sendo, só o recetor da mensagem pode ler a informação com facilidade.

Serão referidas 6 aplicações/serviços para *Mobile* e/ou *PC*, cujo foco será na criptografia e segurança, estando estas divididas por categorias: C1 – Encriptação (funcionalidades de cifra de dados genéricos ou de cifra de funcionalidades dos dispositivos), C2 – Autenticação/Identificação (aplicações ou serviços de autenticação/identificação) e C3 – Outros (aplicações de *Mobile Device Management* (MDM) ou outras funcionalidades de importância comprovada ou ainda de novas tendências com forte expressão na área da segurança).

Categorias

C1 – Encriptação

AxCrypt (PC)

Eleito o melhor *software* de encriptação do mundo, pela prestigiada *PC Magazine*, *AxCrypt* fornece diversas funcionalidades ligadas ao ramo da criptografia e segurança, desde uma forte encriptação de ficheiros, gestor de *passwords*, facilidade de utilização, multilinguismo…

Algoritmos/Implementação

As primitivas criptográficas são *AES-128* ou *AES-256* para criptografia em massa, *PBKDF2* com *HMAC-512* para derivação de chave, *RSA-4096* para a chave da conta e *HMAC-512* para verificação de integridade.

Os métodos de encriptação *AES* diferem da licença do *software*:

- Grátis (Mac): Não oferece qualquer método de encriptação
- Grátis (PC): *AES-128* (128 bits)
- *Premium* (Mac/PC) e Business (Mac/PC): *AES-256* (256 bits)

As especificações detalhadas das implementações dos algoritmos usados pelo *AxCrypt* podem ser encontradas aqui.

Criptografia

Este padrão utiliza chaves dos seguintes tamanhos: 128, 192 e 256 bits. O número de iterações é 10, 12 e 14 respetivamente. *Axcrypt* implementa ***AES*** com chaves de 128 e 256 bits como especificado anteriormente.

AES é um algoritmo de cifra simétrica por bloco.

Segurança

O algoritmo utilizado por *AES* é vulnerável a 2 tipos de ataques: **chaves relacionadas**, – quando se utiliza uma chave parecida/derivada da chave anterior – o que não é crítico desde que se utilizem sempre chaves aleatórias; ***side-channel***, – baseado em informações obtidas com a implementação de um sistema de computador, ao invés de fragilidades do próprio algoritmo – que podem revelar informações que permitam quebrar a chave. Este segundo ataque já não depende do algoritmo.

A nível de privacidade do utilizador, algumas informações, sobre o mesmo, são guardadas no servidor do aplicativo.

Quando uma conta é criada, informações básicas são guardadas:

- Endereço *email* da conta.
- Estado do início de sessão, número de tentativas de *login* falhadas, último início de sessão…
- Estado de pagamento.
- Chave pública *RSA-4096* que está disponível para outros fazerem *download* com o fim de efetuar partilha de chaves.
- Uma ou mais chaves privadas *RSA-4096*, encriptadas usando o *AxCrypt* e a *password* do utilizador, para manter a sincronização entre dispositivos e como forma de *backup* em caso de perda ou destruição de um equipamento.
- Se o gestor de *passwords* for utilizado, é também guardado um ficheiro *XML-encrypted*, encriptado com *AES-256*, que contem a lista de *passwords*, encriptadas com a palavra-passe de início de sessão.
- *Logs*, que ajudam na monitorização da infraestrutura e a melhorar as capacidades de suporte. Os detalhes acerca do *logging* encontram-se neste *link*.

Telegram ***(Mobile/PC)***

Aplicativo de mensagens com foco na velocidade e segurança, simples e gratuito. Pode ser usado em todos os dispositivos ao mesmo tempo, incluindo *Linux*, *Windows*, *iOS*, *Android*, *macOS*,…

O *Telegram* é como *SMS* e *email* combinados – e pode cuidar de todas as necessidades de mensagens pessoais ou comerciais do utilizador. Além disso, suporta encriptação *end-to-end* (ponta-a-ponta) para chamadas de voz.

Foi construído com base em algoritmos testados pelo tempo para tornar a segurança compatível com entrega em alta velocidade e confiabilidade em conexões fracas.

Algoritmos/Implementação

Partes das apps do *Telegram* encontram-se no *GitHub* (projeto *open-source*), – *link* – onde poderá ser encontrada a implementação dos algoritmos de segurança.

O algoritmo usado por esta *App* é explicado na secção ***Algoritmos*** e nos parágrafos abaixo (Criptografia).

Criptografia

Baseado em criptografia *AES* simétrica de 256 bits, *RSA* de 2048 bits e troca segura de chaves *Diffie-Hellman*, *Telegram* utiliza o protocolo de encriptação: *MTProto* (tendo a equipa desenvolvedora da app contribuído para a constituição deste protocolo).

Antes de uma mensagem (ou uma mensagem com várias partes) ser transmitida por uma rede usando um protocolo de transporte, ela é encriptada de uma certa maneira e um cabeçalho externo é adicionado na parte superior da mensagem que consiste num identificador de chave de 64 bits: *auth_key_id* (que

identifica exclusivamente uma chave de autorização para o servidor e o usuário) e uma chave de mensagem de 128 bits: *msg_key*.

A chave de autorização *auth_key* combinada com a chave de mensagem *msg_key* define uma chave real de 256 bits – aes_key – e um vetor de inicialização de 256 bits, – *aes_iv* – que são usados para criptografar a mensagem usando a criptografia *AES-256* no modo *IGE* (*Infinite Garble Extension*). A parte inicial da mensagem contém dados variáveis (sessão, ID da mensagem, número de sequência, *salt* do servidor) que obviamente influenciam a chave da mensagem (e, portanto, a chave *AES* e *aes_iv*). No *MTProto 2.0*, a chave da mensagem é definida como os 128 bits do meio do *SHA-256* do corpo da mensagem (incluindo sessão, *ID* da mensagem, preenchimento etc.), precedidos por 32 bytes retirados da chave de autorização.

As mensagens com várias partes são encriptadas como uma mensagem singular (única).

Segurança

Telegram suporta duas camadas de encriptação segura. A **servidor-cliente** é usada em *chats* na nuvem (conversas particulares e em grupo), enquanto que os chats secretos usam uma camada adicional de encriptação **cliente-cliente**. Todos os dados, independentemente do tipo, são encriptados da mesma maneira – seja texto, *media* ou ficheiros.

Em termos de privacidade, alguns dados são guardados/usados pela *app*:

- Fotos de perfil, nome de utilizador e nome de ecrã.
- Endereço email (caso seja utilizado um email de recuperação de *password*).
- Número de telemóvel (por serem identificadores únicos)
- Contactos (se permitido pelo utilizador)
- *Cookies*
- Mensagens:
  - *Chats* em nuvem (mensagens, imagens, vídeos e documentos).
  - *Media* em *chats* secretos (ficheiros, imagens, vídeos…).

- *Chats* públicos.

Os *chats* secretos nunca são armazenados nos servidores do *Telegram*.

C2 – Autenticação/Identificação

Google Authenticator (Mobile)

Uma aplicação *mobile* disponível gratuitamente tanto para *Android* como para *iOS*. O ***Google Authenticator*** foi criado a pensar na autenticação de dois-passos para fortalecer a segurança das contas dos seus utilizadores. Assim, para além dos métodos de segurança comuns no dispositivo, é necessário introduzir uma *password* de 6 a 8 dígitos gerada através da aplicação *mobile* instalada no seu telemóvel.

Algoritmos/Implementação

Quando é realizada a sua ativação, o site em que a app for ativada fornece uma **chave secreta** ao *authenticator* que posteriormente servirá para realizar a **autenticação** do utilizador no respetivo site. Assim temos uma chave secreta para cada site e para cada utilizador, partilhada entre site e *authenticator* através de uma conexão segura.

Utiliza ***HCMA-SHA1*** para gerar um código através da chave secreta compartilhada anteriormente.

Criptografia

A chave secreta tem 80 bits, e é entregue na forma de uma *string* de 16, 26 ou 32 carateres de base 32 ou em forma de *QR code*.

O cliente cria um *HMAC-SHA1* através da chave secreta.

Uma porção do HMCA é extraído e convertido para um código de 6 a 8 dígitos.

Segurança

Pesquisadores descobriram um novo tipo de **malware** capaz de capturar os códigos de segurança produzidos pela app da Google. Através de um *trojan*,

conseguiria acesso ao dispositivo da vítima, podendo assim gerar os códigos a partir da app.

SSH – Secure Shell (PC)

Permite uma **administração segura** do sistema e transferências de ficheiros através de redes inseguras. É muito comum o seu uso em *datacenters* e em grandes empresas.

O serviço, que usa técnicas criptográficas para garantir que todas as comunicações do servidor remoto são encriptadas, foi criado com o intuito de substituir o ***Telnet***, que não era criptografado.

Fornece um mecanismo de autenticação de um utilizador remoto, transferindo entradas do cliente para o servidor de hospedagem e retransmitindo a saída para o cliente.

Algoritmos/Implementação

O ***SSH*** faz uso de vários algoritmos. ***AES, CAST128, Blowfish*** são alguns dos disponíveis. Os algoritmos que serão utilizados são gerados no processo de conexão aleatoriamente, tal como é explicado no próximo ponto. Estes são aprofundados na secção ***Algoritmos***.

Criptografia

A grande vantagem na utilização do *SSH*, comparativamente ao seu antecessor (*Telnet*), é o uso da criptografia para transmitir de forma segura, uma informação entre o cliente (computador do utilizador) e o *host* (servidor que pretende acessar).

As três tecnologias de criptografia usadas pelo SSH:

- Criptografia simétrica
- Criptografia assimétrica

- *Hashing*

Criptografia simétrica

Neste método criptográfico é utilizada uma chave secreta para codificar e descodificar uma mensagem do cliente e do servidor.

Chaves simétricas são utilizadas para codificar uma comunicação encriptada durante uma sessão de *SSH*. O cliente e o servidor derivam uma chave secreta usando um método agregador e que nunca é divulgado a terceiros.

O processo de criação de uma chave simétrica é formado pela chave de mudança de um algoritmo. A chave nunca é transmitida entre cliente e host. Assim sendo, mesmo que uma outra máquina calcule a chave o algoritmo não será reconhecido.

Desta forma, cada sessão de SSH é específica para cada *token* e de forma secreta, o que requer uma autenticação prioritária do cliente. Uma vez gerada a chave, todos os pacotes serão movidos a duas máquinas de forma encriptada, pelas suas chaves privativas. Isso inclui senha digitada pelo utilizador, assim como as credenciais protegidas pelo pacote de rede.

Antes de estabelecer uma ligação segura o cliente e o host decidem que algoritmos utilizar, dependendo das máquinas que se vão comunicar.

Criptografia assimétrica

Contrariamente á criptografia simétrica, a assimétrica utiliza duas chaves separadas para encriptar e desencriptar. Essas chaves são conhecidas por **chave pública** e **chave privada**.

A **chave pública**, como o próprio nome sugere, é distribuída e compartilhada com todas as partes. A relação entre duas chaves é extremamente complexa:

"Uma mensagem é encriptada por uma chave pública e pode apenas ser desencriptada pela mesma chave privada."

A **Segurança** reside na capacidade de manter secreta a chave privada.

Ao contrário da perceção geral, a criptografia assimétrica não é utilizada para criptografar todas as sessões ***SSH***. É apenas utilizada para o algoritmo de troca de chaves de **criptografia simétrica**.

Antes de iniciar uma conexão segura, ambas as partes geram pares de **chaves público-privadas** temporárias e compartilham entre si suas respetivas chaves privadas para produzir a chave secreta compartilhada.

Uma vez que uma comunicação simétrica segura for estabelecida, o servidor usa a chave pública dos clientes para gerar, testar e transmitir ao cliente para ***autenticação***. Se o cliente conseguir descodificar com sucesso a mensagem, isso significa que contém a **chave privada** necessária para a conexão. Então, a sessão ***SSH*** começa.

Hashing

One-way hashing é mais uma das formas de criptografia utilizada em conexões seguras de *Shell*. As funções do *one-way-hash* diferem de ambas as formas de criptografia mencionadas anteriormente, pois nunca devem ser descodificadas.

Geram um valor exclusivo, um sentido único de comprimento fixo para cada entrada que não mostra nenhuma tendência clara que possa eventualmente ser explorada. De um ponto de vista de ***Segurança*** é algo muito interessante, pois torna praticamente impossível de realizar uma reversão. É fácil gerar uma *hash* vindo de uma porta de entrada, mas é impossível gerar uma entrada a partir de uma *hash*. Isso significa que, quando um cliente determina uma entrada correta, pode gerar uma *hash*, comparar o valor e verificar onde fica a entrada correta.

SSH usa *hashes* para verificar a autenticidade das mensagens. Isso é feito usando os códigos de autenticação de mensagens baseados em *hash*, um

comando específico não utilizado por mais ninguém. Enquanto o algoritmo é selecionado, uma mensagem de autenticação é também selecionada. Após estar selecionado tanto o algoritmo como a respetiva mensagem, é enviada a mensagem que irá conter um ***MAC***, que é calculado através de uma *chave simétrica*. É enviado para fora de forma ***simétrica***, com dados encriptados numa sessão.

Segurança

Chegamos então ao passo final, antes do utilizador conseguir as credenciais de acesso ao servidor de forma autenticada, a ***Autenticação***.

A maioria dos utilizadores de *SSH* utiliza uma senha, que é necessária introduzir junto com o nome de utilizador.

Essas credenciais são transmitidas através de um **túnel criptografado simétrico**, logo será praticamente impossível a sua interceção. Assim as *passwords* são encriptadas, mas mesmo assim não é aconselhado a utilização de conexões não seguras.

Muitos robôs podem simplesmente utilizar senhas padrão até conseguirem um acesso. Isso apresenta um problema do ponto de vista de Segurança. Por isso mesmo, é recomendada a utilização de pares de chave *SSH*. É utilizado um padrão de chave simétricas para autenticar o utilizador sem que ele precise de criar uma senha.

C3 – Outros

Microsoft Intune (PC/Mobile)

Serviço baseado em nuvem que se foca na gestão de dispositivos e aplicações móveis. Está incluído na suite *Enterprise Mobility + Security* da *Microsoft*, e permite que os utilizadores sejam produtivos, mantendo os dados da sua organização protegidos.

Integra-se com outros serviços, incluindo o *Microsoft 365* e o *Azure Ative Directory* para controlar quem tem acesso e aquilo a que têm acesso, e a *Proteção de Informação Azure* para proteção de dados.

Algoritmos/Implementação

Intune utiliza aplicações externas para encriptar os discos/drives do dispositivo:

- *macOS*: ***FileVault***
- *Windows 10* ou mais recente: ***BitLocker***

BitLocker utiliza diversos algoritmos de encriptação, sendo o seu método padrão *XTS-AES 128 bit*. Este software permite encriptar discos de sistemas operativos, *data-drives* (discos de informação) e unidades removíveis, com opção do utilizador, entre os seguintes algoritmos:

- *AES-CBC 128-bit*
- *AES-CBC 256-bit*
- *XTS-AES 128-bit*
- *XTS-AES 256-bit*

Já o *FileVault* permite a encriptação completa de um disco, utilizando o algoritmo *XTS-AES 128 bit*.

Criptografia

Quanto aos ficheiros e documentos, todos são guardados e gerenciados em nuvem, protegidos com encriptação num dos *data centers*.

Para além do mencionado em cima, é um pouco escassa a informação sobre os algoritmos utilizados para guardar a informação em nuvem.

Segurança

A Microsoft usa uma variedade de tecnologias e procedimentos de segurança para ajudar a proteger informações contra acesso, uso ou divulgação não autorizados. Por exemplo, armazenam as informações fornecidas em sistemas de computador com acesso limitado, localizados em instalações controladas. Podem ser encontradas mais informações no *link*.

A empresa usa dados estatísticos agregados, tendências e informações de uso, derivadas do uso que o utilizador faz dos serviços, com a finalidade de fornecer, operar, manter ou melhorar os mesmos, bem como quaisquer produtos e serviços da Microsoft usados para fornecer os serviços.

Microsoft usará os dados do cliente (conforme definido no seu contrato com a Microsoft) apenas para fornecer os serviços ao próprio utilizador. Isso pode incluir a solução de problemas que visa prevenir, detetar ou reparar problemas que afetam a operação desses serviços e o melhoramento de recursos que envolvem a deteção e a proteção contra ameaças emergentes e em evolução para os utilizadores (como malware ou spam).

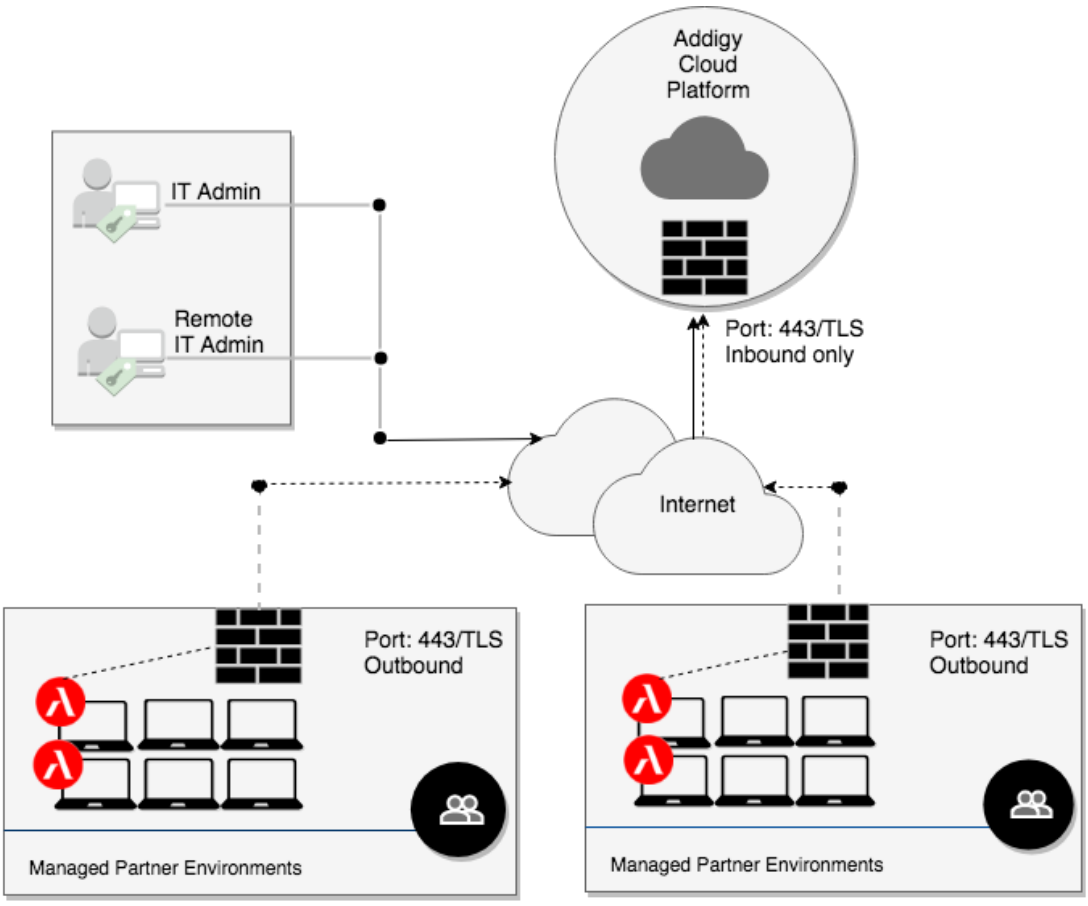
Os funcionários/colaboradores da empresa não processam os dados do cliente sem autorização. Estes são obrigados a manter a confidencialidade de quaisquer informações relativas ao utilizador e essa obrigação continua mesmo após o término do seu compromisso.

Addigy ***(PC/Mobile)***

Aplicativo de *device management*. Feito exclusivamente para dispositivos *Apple*, podendo ser utilizado em ambas as plataformas, *PC* e *mobile*.

A sua principal função é a de manter o dispositivo atualizado, executar tarefas simples facilmente programáveis através da app e poder controlar outros dispositivos remotamente de maneira segura. Para além disso tem também a funcionalidade de encriptar e guardar ficheiros em *cloud*.

Algoritmo



Criptografia

Como podemos observar pelo esquema representado acima, a porta utilizada nas conexões é a 443, que faz uso do protocolo ***TLS***.

Com o protocolo *TLS*, a conexão é segura, pois é utilizada **criptografia simétrica** para encriptar os dados transmitidos. As chaves para essa encriptação são geradas exclusivamente para cada conexão e são baseadas em segredo compartilhado, que foi negociado no início da sessão.

O servidor e o cliente negociam os detalhes de qual algoritmo de criptografia e chaves criptográficas utilizar antes que o primeiro *byte* de dados seja transmitido.

O *TLS* suporta muitos métodos diferentes para trocar chaves, encriptar dados e autenticar a integridade da mensagem.

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).

Segurança

Como na maioria das aplicações, os dados que são armazenados sobre o utilizador são uma preocupação.

Dados como nome, idade, sexo, email, etc., são armazenados. Quanto à compra de um serviço, afirmam que toda a informação é processada sobre os *PCI Compliant Standards*. Informação sensível como por exemplo um cartão de crédito, não é guardada, mas sim uma versão *hashed*.

Quanto á utilização da *app* e do próprio *website*, todas as ações são observadas e guardadas num *log*. Toda a informação como links que foram abertos e ficheiros que fizermos *upload* para um serviço da *app* (nome, tamanho, e tipo de ficheiro) são armazenados com referência ao utilizador ou um *email*.

Após abordarmos o que é armazenado, vamos abordar o que é feito com tal informação. A informação básica pessoal, como o nome, é utilizada para *marketing* da própria empresa e para divulgação de serviços. Quanto á informação de cartões de crédito, apenas são utilizadas para realizar os pagamentos.

Algoritmos

AES

Advanced Encryption Standard, também conhecido por *Rijndael*, é uma especificação para a criptografia de dados eletrónicos, estabelecida pelo Instituto Nacional de Padrões e Tecnologia dos EUA (NIST) em 2001. Foi adotado pelo governo dos EUA e agora é usado em todo o mundo.

É baseado no princípio conhecido por *rede de substituição-permutação* e eficiente em ambos *software* e *hardware*, com um bom desempenho numa ampla variedade de *hardware*, desde *smart cards* de 8 bits a computadores de alto desempenho.

PBKDF2

Password-Based Key Derivation Function 2 é uma função de derivação de chave que faz parte da série *PKCS* (*Public-Key Cryptography Standards*) da *RSA Laboratories*.

PBKDF2 aplica uma função pseudoaleatória, como uma hash criptográfica, cifra ou *HMAC*, à palavra-passe ou *passphrase* de entrada, junto com um determinado valor de sal, e repete o processo várias vezes para produzir uma chave derivada, que pode ser usada como chave criptográfica nas subsequentes operações. O trabalho computacional adicionado torna as tentativas de quebra de *password* (*cracking*) muito mais difíceis. Em 2000, o número mínimo recomendado de iterações era 1000. O parâmetro deve aumentar ao longo do tempo à medida que a velocidade da *CPU* aumenta. A adição de um sal à senha reduz a capacidade de usar *hashes* pré-computadas (*rainbow tables*) para ataques e significa que várias senhas precisam ser testadas individualmente – não todas de uma vez.

HMAC

Keyed-Hash Message Authentication Code ou *Hash-based Message Authentication Code*, é um tipo específico de código de autenticação de mensagens (*MAC* – *Message Authentication Code*) que envolve uma função *hash* criptográfica e uma chave criptográfica secreta. Como em qualquer *MAC*,

pode ser usado para verificar simultaneamente a integridade dos dados e a autenticidade de uma mensagem.

HMAC possui dois passos no procedimento da computação de *hash*. A chave secreta é usada primeiro para derivar duas chaves - interna e externa. O primeiro passo do algoritmo produz uma *hash* interna derivada da mensagem e da chave interna. O segundo passo produz o código *HMAC* final derivado do resultado da *hash* interna e da chave externa. Assim, o algoritmo fornece uma melhor imunidade contra ataques de extensos.

RSA

Rivest–Shamir–Adleman é um dos primeiros sistemas de criptografia de chave pública e é amplamente usado para transmissão segura de dados. Nesse sistema de criptografia, a chave encriptada é pública e distinta da chave de desencriptação que é mantida em segredo (privada).

É criada e publica uma chave pública com base em dois números primos grandes, juntamente com um valor auxiliar. Os números primos devem ser mantidos em segredo. Qualquer pessoa pode usar a chave pública para criptografar uma mensagem, mas apenas alguém com conhecimento dos números primos pode descodificar a mensagem. Até hoje, não há métodos publicados para derrotar o sistema se for usada uma chave suficientemente grande.

RSA é um algoritmo relativamente lento e é, por isso, menos usado para encriptar diretamente os dados do usuário. *RSA* passa chaves compartilhadas encriptadas para criptografia de chave simétrica com mais frequência que, por sua vez, pode executar operações de encriptação e desencriptação em massa a uma velocidade muito maior.

SHA1

Função *hash* criptográfica desenvolvida pela Agência de Segurança Nacional (*NSA*) dos Estados Unidos. Produz um valor de dispersão de 160 bits (20 bytes) conhecido como resumo da mensagem. *SHA-1* é a mais amplamente utilizada das funções de dispersão *SHA* existentes, sendo empregada em vários protocolos e aplicações amplamente utilizadas, incluindo *TLS* e *SSL*, *PGP*, *S/MIME* e *IPsec*.

Blowfish

É um algoritmo de bloco simétrico. Apresenta uma rede de *Feistel* de 16 iterações com tamanho de bloco de 64-bits, uma chave que pode variar entre 32 e 448-bits, e *S-boxes* altamente dependentes de chaves, tornando-o ideal para aplicações tanto domésticas, quanto comerciais.

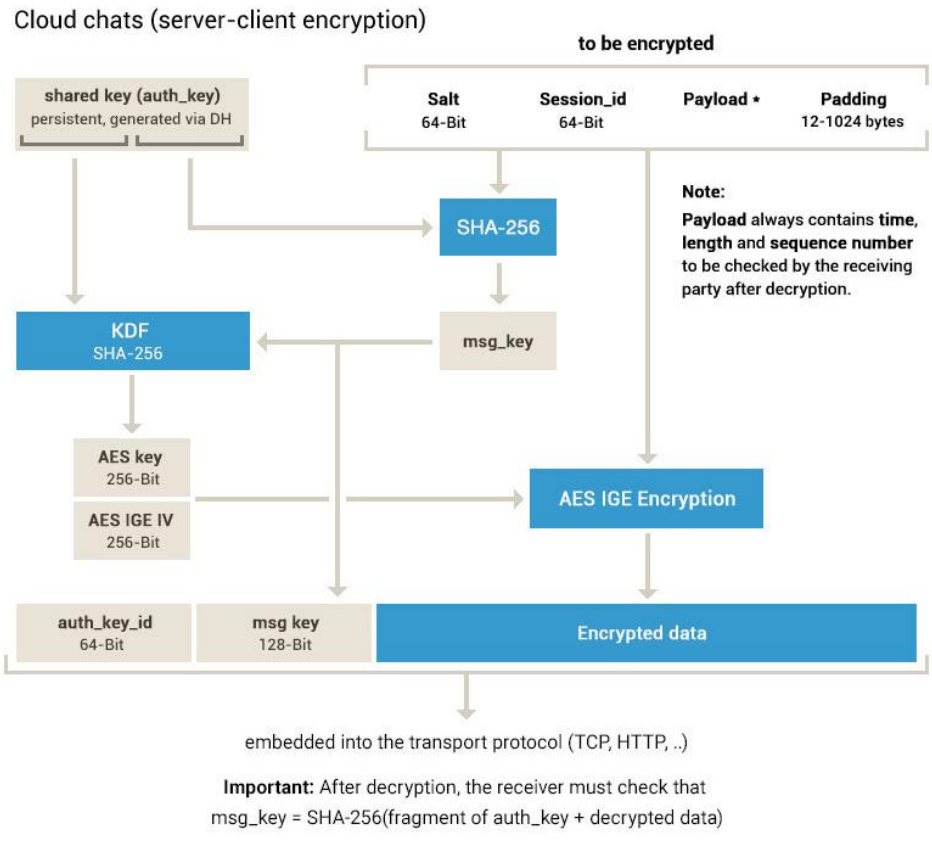
Foi desenvolvido em 1993 por *Bruce Schneier* como uma alternativa grátis mais rápida para os algoritmos criptográficos existentes. Desde então ele vem sendo analisado de forma considerável e está conquistando a aceitação do mercado como um algoritmo forte.

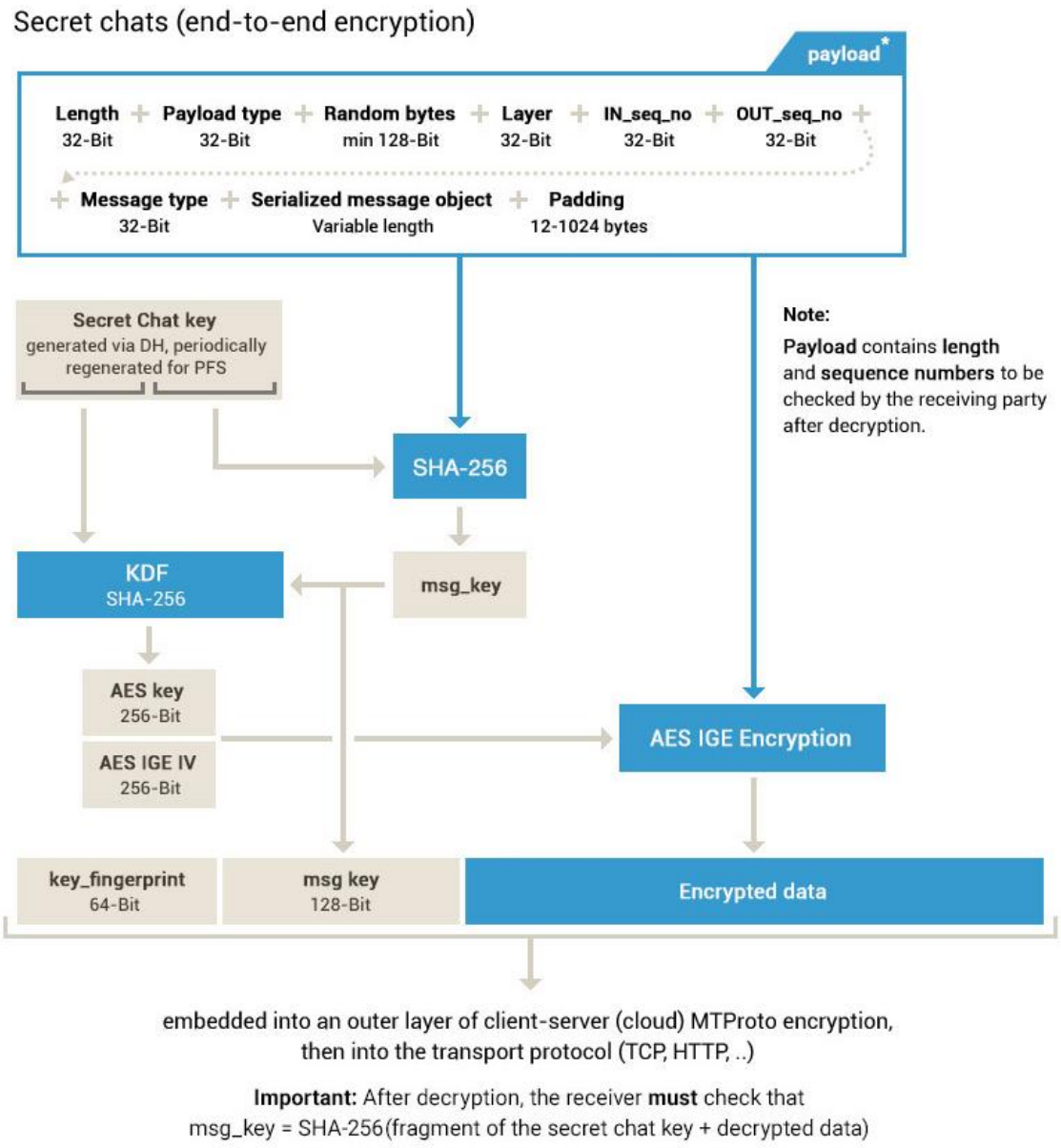
Existem duas derivações do *Blowfish*:

- *Twofish*
- *Threefish*

Ambas muito seguras, sendo a *Threefish* mais segura devido à utilização de blocos maiores e de chaves também maiores.

MTProto 2.0





CAST-128

Foi criado em 1996 e é um algoritmo em bloco de chave simétrica.

É utilizado em diversas aplicações e foi aprovado pelo Governo do Canadá para ser utilizado pelo Estabelecimento de Comunicação Segura.

As chaves variam entre 40 e 128 *bits*, mas sempre em incrementos de 8.

Conclusão

AxCrypt e ***Telegram*** são aplicações cujos focos aproximam-se da encriptação. A primeira encripta ficheiros e tem a capacidade de armazenar passwords. Já o *Telegram* é uma *app* especializada na troca de mensagens de forma segura.

Google Authenticator é uma aplicação, criada a pensar na autenticação de dois-passos para fortalecer a segurança das contas dos seus utilizadores.

SSH é um serviço que usa técnicas criptográficas para garantir que todas as comunicações do servidor remoto são encriptadas, foi criado com o intuito de substituir o *Telnet*, que não era criptografado.

Microsoft Intune é um serviço baseado em nuvem que se foca na gestão de dispositivos e aplicações móveis. Utiliza encriptação nas suas comunicações e armazenamento. ***Addigy*** é um aplicativo de *device management*. Uma das funcionalidades desta aplicação é encriptar e guardar ficheiros em *cloud*.

Algoritmos criptográficos como AES, RSA, …, são de extrema importância e necessários no ramo da segurança de informação e cibersegurança. É necessário conhecer totalmente um criptosistema destes antes de implementá-lo, seja num sistema ou numa aplicação/serviço, tendo cada um deles a sua finalidade e utilidade.

Referências

https://everything.explained.today/Advanced_Encryption_Standard/

https://www.axcrypt.net/

https://everything.explained.today/substitution%e2%80%93permutation_network/

https://everything.explained.today/AES_implementations/

https://www.cyclonis.com/what-is-aes-256-encryption/

https://www.axcrypt.net/downloads/3432/

https://everything.explained.today/side-channel_attack/

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

https://en.wikipedia.org/wiki/HMAC

https://www.pbkdf2.com/

https://www.nist.gov/publications/advanced-encryption-standard-aes
https://en.wikipedia.org/wiki/Google_Authenticator

https://cryptography.io/en/latest/hazmat/primitives/twofactor/

https://github.com/google/google-authenticator-android

https://crypto.stackexchange.com/questions/12087/how-does-googles-authenticator-number-generator-work

https://en.wikipedia.org/wiki/CAST-128

https://pt.wikipedia.org/wiki/SHA-1

https://en.wikipedia.org/wiki/HMAC

https://www.hostinger.com.br/tutoriais/como-funciona-o-ssh/

https://pt.wikipedia.org/wiki/Secure_Shell

https://www.schneier.com/academic/blowfish/

https://telegram.org/privacy#3-what-personal-data-we-use

https://support.addigy.com/support/solutions/articles/8000041028-addigy-security-compliance

https://docs.microsoft.com/pt-pt/mem/intune/fundamentals/what-is-intune

https://docs.microsoft.com/en-us/legal/intune/microsoft-intune-privacy-statement

https://addigy.com/privacy-policy/

Aula Teórica 04 (v2) – Criptografia Simétrica, *Gonçalo Hermenegildo* – Disponível no *Moodle*