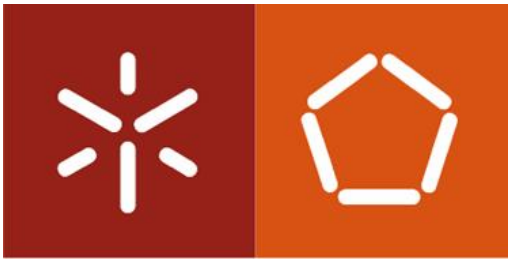

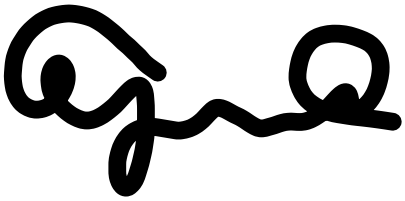

Universidade do Minho

MIEBIOM

DSA e tecnologias associadas

Criptografia

Docente:

José Manuel Valença

Discentes:

Ana Sofia Quintas, 65078

Janeiro 2015

Resumo

Neste trabalho apresenta a descrição do algoritmo DSA e as suas tecnologias associadas. Aborda ainda as escolhas dos parâmetros envolvidos, assim como o seu mecanismo de geração e validação de assinatura. Apresenta ainda uma análise de ataque ao algoritmo por redução dos parâmetros.

1. Introdução

O DSA (*Digital Signature Algorithm*) trata-se de um algoritmo assimétrico para assinaturas digitais criado pela NSA e patenteado pelo governo americano. Foi proposto pelo NIST em agosto de 1991, para utilização no seu padrão Digital Signature Standard (DSS). Este algoritmo está assente na segurança conseguida através da dificuldade de computar logaritmos discretos. [1]

Um algoritmo de assinatura digital permite que uma entidade autentifique a integridade dos dados assinados e da identidade do signatário. O destinatário de uma mensagem assinada pode usar uma assinatura digital como meio de prova para demonstrar a terceiros que a assinatura foi, de fato, gerado pela entidade reivindicada. Um algoritmo de assinatura digital é destinado ao uso do correio eletrônico, transferência eletrônica de fundos, intercâmbio eletrônico de dados, distribuição de software, armazenamento de dados e outras aplicações que exigem garantia de integridade de dados e autenticação de origem de dados. [2]

Este artigo teve como suporte teórico principal a publicação da NIST em 2013 o standard FIPS PUB 186-4, intitulado “Digital Signature Standard (DSS)”, onde descreve todos os pormenores de para obtenção de uma assinatura digital recorrendo ao DSA. Teve também como como referencia o livro “Network Security with OpenSSL” que especifica a forma de como implementar o algoritmo do DSA através do OpenSSl. Por último recorrendo ao Standard PKCS#11 onde se encontra disponível a forma de implementar o algoritmo DSA, assim como as funções especificas para tal.

Este documento encontra-se estrutura por sete capítulos. No primeiro capitulo, encontra-se uma breve introdução e contextualização ao algoritmo. No Segundo capitulo apresenta-se uma visão geral do que se trata o algoritmo, e do seu próprio mecanismo. No terceiro capitulo encontram-se descritos os parâmetros necessários para a assinatura digital segundo o DSA. No quarto capitulo apresenta-se o mecanismo de geração da assinatura e no quinto de validação e verificação da mesma. No capitulo seis será abordada a temática do ataque ao algoritmo por redução dos parâmetros. No sétimo capítulo encontra-se algumas aplicações do DSA, e por último encontra-se as conclusões e o conhecimento apreendido.

2. DSA: Visão geral

O DSA trata-se de algoritmo que se pode dividir em três fases fundamentais. A primeira fase é a geração da assinatura, a segunda trata-se da verificação e por último obtêm-se a validação da mesma. Em jogo estão um vasto conjunto de parâmetros, incluindo uma chave privada x, e uma chave publica y e ainda uma função de hash. Todos os parâmetros têm de ser gerados e verificados sob métodos especificados em standards oficiais, de forma a se obter uma assinatura válida e resistente a ataques.

O algoritmo do DSA, inicia-se com a obtenção de um resumo da mensagem inicial através da função de hash. Após determinar os parâmetros de domínio procede-se à assinatura digital, com a chave privada x. Este passo tem como resultado final os parâmetros r, e s. A mensagem inicial, o parâmetro r e s são então passados para o destinatário, assim como os parâmetros de domínio. Novamente recorrendo a uma função de hash e com os parâmetros que lhe foram passados e a chave pública y procede-se à verificação da assinatura. Caso o parâmetro r passado ao destinatário, seja igual a um v obtido da verificação a assinatura é considerada válida, caso contrário a assinatura é considerada inválida. [1][2]

Nos próximos capítulos, todas as etapas e especificações dos parâmetros serão discutidas, assim como modo de geração e verificação de cada um deles.

3. Parâmetros do DSA

O ponto de partida para o do algoritmo de assinatura digital é sem dúvida, a definição dos parâmetros iniciais. Desta forma, a assinatura digital DSA é calculada utilizando um conjunto de parâmetros de domínio identificados pelas letras *p*, *q* e *g*, uma chave privada *x*, uma chave pública *y*, um número secreto *k*, as informações a serem assinadas, e uma função hash. Assim, os parâmetros de partida do algoritmo encontram-se abaixo descritos:

p Um número primo entre $2^{L-1}<p<2^{L}$, onde L é tamanho de bits de *p*;

q Um divisor primo de (*p*-1), entre $2^{N-1}<q<2^{N}$, onde N é o tamanho de bits de *q*;

g Gerador de um subgrupo de ordem q, em que 1<g<p;

x Uma chave privada, que deve permanecer secreta, onde *x* é um inteiro gerado de forma aleatória ou pseudo-aleatória no intervalo $0<x<q$;

y Uma chave pública onde $y=g^{x} \bmod p$;

k Um número secreto único para cada mensagem. *k* é gerado de forma aleatória ou pseudo-aleatória

É de referir que os parâmetros de domínio *p*, *q* e *g* poderão ser públicos a um grupo de usuários. *k* é um parâmetro individual de cada assinatura sendo que deve ser mantido em segredo, por sua vez a chave privada x, também. [2][1]

3.1 Seleção dos parâmetros e funções de hash

O standard FIPS 186-4 especifica quais os tamanhos, em bits, mais adequados para os pares de L e N usados no cálculo de *p e q* respectivamente. Assim, os pares recomendados apresentam-se a baixo :

L=1024, N=160;

L=2048, N=224;

L=2048,N=256;

L=3072,N=256.

É de realçar que o uso incorreto destes pares pode colocar em causa toda a segurança do algoritmo.[2]

Os pares anteriormente mencionados, estão diretamente relacionados com a força de segurança de um algoritmo. A força de segurança de um algoritmo trata-se do trabalho necessário para "quebrar" esse mesmo algoritmo criptográfico. Na tabela 1 pode-se observar a relação entre os pares recomendados e a sua força de segurança associada, também denominada como bits de segurança. [2],[3]

Tabela 1: Relação entre os bits de segurança e o Par(L,N) [3]

Bits de segurança	**Par (L,N)**
80	$L = 1024$ $N = 160$
112	$L = 2048$ $N = 224$
128	$L = 3072$ $N = 256$

Uma função de hash aprovada conforme especificado no FIPS 180-4, deve ser utilizada durante a geração de assinaturas digitais. A força de segurança associada a uma assinatura digital, recorrendo ao algoritmo de DSA, não é maior do que o mínimo de segurança da força do par (L, N) e da força da função hash que é usada.

Sendo assim a escolha do parâmetros de forma a obter a força de segurança do par (L,N) e a resistência da função de hash devem satisfazer ou ultrapassar a segurança necessária requerida para a assinatura em questão. Desta forma é recomendável que a força de segurança do par (L, N) e a força da função hash utilizada para a geração de assinaturas digitais de segurança ser a mesma, a menos que tenha sido feita a um acordo entre entidades participantes utilizar uma função hash mais forte. Na tabela 2 podemos verificar as funções de hash que podem ser utilizadas para assinatura em função da sua força. [1],[2],[3]

Tabela 2: bits de segurança associados às funções de hash [3]

Bits de segurança	**Funções de Hash para assinatura digital**
80	SHA-1 ; SHA-224 ; SHA-256 ; SHA-384, SHA-512
112	SHA-224, SHA-256 ; SHA-384 ; SHA-512
128	SHA-256, SHA-384, SHA-512
192	SHA-384, SHA-512
256	SHA-512

É de realçar que uma função hash que proporciona uma força de segurança menor do que o par(L, N), não deve ser utilizada uma vez que isso irá reduzir a força do processo de assinatura digital para um nível de segurança não maior do que o previsto pela função hash. [1]

3.2 Parâmetros do Domínio

Os parâmetros do domínio para o DSA tratam-se de *p*, *q* e *g*, e opcionalmente o *domain_parameter_seed* e *counter* utilizados na geração de *p* e *q*.

O DSA requer que os pares de chaves privadas e chaves públicas utilizados para a geração e verificação de assinatura digital sejam gerados relativamente um conjunto particular de parâmetros do domínio, sendo que estes podem ser públicos e comuns a um grupo de utilizadores, tal como já foi anteriormente referido.

Um usuário de um conjunto de parâmetros do domínio deve ter a garantia de que estes são válidos antes de os usar. Assim, embora os parâmetros de domínio possam ser uma informação pública, devem de ser geridos de modo a que a correspondência correta entre um determinado par de chaves e seu conjunto de parâmetros de domínio seja mantida para todas as partes que usam o par de chaves. Um conjunto de parâmetros de domínio podem ser fixos por um período de tempo prolongado. [1]

- **Geração dos parâmetros de domínio.**

A geração dos parâmetros do domínio pode ser uma responsabilidade atribuída a uma terceira parte confiável, TTP ou AC, por exemplo.

Com a publicação do Standard FIPS 186-4, o método de geração dos parâmetros *p* e *q* com base na função SHA-1 e métodos probabilísticos tornou-se desatualizado, sendo que foi substituído sendo que o novo método, considerado válido. Este "novo" método recomendado trata-se da geração dos parâmetros *p* e *q* utilizando uma função hash aprovada, sendo que esta deve respeitar o pressuposto de possuir uma segurança igual ou maior que a força do par (L,N).

A geração dos parâmetros *p* e *q* inicia-se com a definição do input. Este método recebe como input, o par(L,N) e ainda um parâmetro *seedlen* que deve ser igual ou superior a N. O *seedlen* trata-se do tamanho desejável para o parâmetro seed, sendo que este por sua vez pode ser qualquer número, como por exemplo o favorito do usuário ou um numero escolhido aleatoriamente. O output por sua vez consiste no *p*, no *q*, no *domain_parameter_seed* e ainda um *counter*, sendo que os dois últimos

são opcionais. Posto isto o processo de geração das *p* e *q* pode-se iniciar. O processo possui algumas fases no entanto são de realçar que a fase inicial é a verificação do par(L,N) segundo as recomendações. Caso este não seja válido a geração é considerada inválida. O mesmo acontece caso *seedlen* < *N* . As fases de maior importância para a obtenção dos valores de *p e q* são sem dúvida a sua verificação enquanto valores primos aceitáveis.

Após a geração de *p*, *q* , *domain_parameter_seed, e counter*, ainda se procede à validação dos mesmos. Esta validação tem como objectivo principal a verificação se o *q* e *p* anteriormente gerado é um primo válido.

Para finalizar a geração dos parâmetros de domínio apenas falta abordar a geração de *g*. Dois métodos para a geração de *g* são possíveis, um deles por não possuir validação do output, é apenas recomendável quando a geração do *g* é confiável.

O primeiro método denomina-se de "Geração do gerador *g* não verificada". Este método determina o valor de *g* com base nos valores de *p* e *q*, e tal como o nome indica este é método que não implica a verificação posterior do *g* obtido. Os passos deste método encontram-se abaixo descritos:

1. *e*=(*p*–1)/*q*.
2. h= é um qualquer inteiro que satisfaz a condição 1 < *h* < (*p* – 1), h será sempre diferente de qualquer valor que já foi tentado;
3. *g*=h^e mod p.
4. Se (*g* = 1), retorna ao passo 2
5. Devolve *g*.

O segundo método trata-se de um método que envolve uma maior complexidade. Este denomina-se de "geração canonical verificável de gerador g". O input para além dos valores de *p* e *q*, ainda inclui *domain_parameter_seed e índex*. Este é ainda um método em que se pode determinar vários valores de *g* para o mesmo *p* e *q*. A vantagem deste método prende-se no facto em que se pode definir, por exemplo, *índex*=1 para assinatura digital e *índex*=2 para o estabelecimento da chave. Um outra característica é a utilização da mesma função de hash usada para a determinação de *p* e *q*. Os passos envolvidos neste método são:

1. Se (*index* for incorreto), retornar INVALIDO**.**
2. N = **len**(q).
3. e=(p–1)/q.
4. *count* = 0.
5. *count* = *count* + 1.
6. Se (*count* = 0), Retorna INVALIDO
7. *U = domain_parameter_seed* || "ggen" || *index* || *count.*
8. W = **Hash**(*U)*.
9. $g=W^e \bmod p$
10. Se (g < 2), volta ao passo 5
11. Retornar VALIDO e valor de g.

A verificação deste método recebe todos os parâmetros de input do próprio método e ainda o próprio *g* gerado. E resume-se aos seguintes passos:

1. Se (*index* for incorreto), retorna INVALIDO**.**
2. Verifica-se que $2 \leq g \leq (p-1)$. Se não for verdade retorna-se INVALIDO
3. Se ($g^q \neq 1 \bmod p$) retorna INVALIDO
4. N = **len**(q).
5. e=(p–1)/q.
6. *count* = 0.
7. *count* = *count* + 1.
8. Se (*count* = 0), retorna INVALIDO
9. *U = domain_parameter_seed* || "ggen" || *index* || *count.*
10. W = Hash(U).
11. Computed_g = We mod p.
12. Se (computed_g < 2 retorna ao passo 7.
13. Se (computed_g = g) , Retorna VÁLIDO [1]

- ## Gestão dos parâmetros de domínio.

Cada par de chaves de assinatura digital deve ser corretamente associado a um conjunto específico de parâmetros de domínio. Os parâmetros de domínio devem ser protegidos contra modificações não autorizadas até que o conjunto seja desativado, ou seja quando este deixar de ser necessário.

Os mesmos parâmetros de domínio podem ser usado para mais do que uma finalidade (por exemplo, os mesmos parâmetros de domínio podem ser usados para ambas as assinaturas digitais e de estabelecimento de chave). No entanto, usando diferentes valores para o gerador *g* reduz o risco de que os pares de chaves gerados para uma finalidade possam ser usados acidentalmente (ou não) para outra. [1]

3.3 Parâmetros das Chaves

Cada assinatura possuiu um par de chaves: uma chave privada *x* e uma chave pública *y,* que estão matematicamente relacionadas entre si. A chave privada deve ser utilizada apenas por um período fixo de tempo na qual podem ser geradas assinaturas digitais. Por sua vez, a chave pública pode continuadamente utilizada. [1]

- ### Geração do par de chaves

A criptografia de logaritmos discreto é divido em FFC(criptografia de corpos finitos) e ECC(criptografia de curvas elípticas. O DSA um exemplo de FFC.

Com base no pressuposto anteriormente mencionado, pode-se partir, então para a geração do par de chaves *x* e *y*. Um par de chaves FFC é obtido a partir dos inputs de *p*, *q*, g, *domain_parameter_seed* e do *counter.* As chaves podem ser obtidas a partir de dois métodos, num entanto deve-se optar por um. O primeiro método denomina-se de "geração de pares de chaves utilizando bits 'extra-aleatórios' " e o segundo denomina-se, por sua vez, por "geração de chaves por teste dos candidatos".

No método de "geração de pares de chaves utilizando bits 'extra-aleatórios' " é necessário recorrer a um gerador de bits aleatório (RBG) com segurança igual ou superior à segurança associada ao par(L,N). O processo baseia-se nas seguintes etapas:

1. N = tamanho(q), L = tamanho(p).

2. Se o par(L, N) for considerado inválido, segundo aqueles que são recomendados, retorna ERRO;
3. *requested_security_strength* = segurança associada ao par (L, N); (especificada na tabela 1)

4. Obter a *string N*+64 *returned_bits* apartir de RBG, em que RBG tem uma segurança associada, igual ou superior a *requested_security_strength,* Caso RBG não seja maior, retorna ERRO;
5. Converter *returned_bits* para um inteiro não negativo *c;*
6. $x = (c \bmod (q-1)) + 1$
7. $y=g^x \bmod p$
8. Retornar o estado de SUCESSO e *x*, and *y*.

No segundo método, trata-se de certa forma de um método por iterações. O que difere do algoritmo anterior é que a obtenção do RBG é o valor atribuído a *returned_bits*. Os passos 1,2 e 3 são iguais ao primeiro método, e é acrescentado um passo 6 de comparação.

4. Obter a *string N returned_bits* a partir de **RBG** em que RBG tem uma segurança associada, igual ou superior a *requested_security_strength,* Caso RBG não seja maior, retorna ERRO;

5. Converter *returned_bits* para um inteiro não negativo *c;*

6. Se ($c > q-2$), retornar ao passo 4.

7. $x=c+1$.

8. $y=g^x \bmod p$

9. Retornar o estado de SUCESSO e *x*, and *y*. [1]

Gestão do par de chaves

O uso seguro de assinaturas digitais depende da uma boa gestão do par de chaves para assinatura digital. Algumas recomendações são especificadas em FIPS 180-4.

- A validade de todos os parâmetros do domínio deverá de ser assegurada antes da geração do par de chaves, ou da verificação e da validação de uma assinatura digital.
- Cada par de chaves devem de ser associado aos parâmetros de domínio em que o par de chaves foi gerado.
- Um par de chaves só podem ser utilizados para gerar e verificar assinaturas utilizando os parâmetros de domínio associados a esse par de chaves.

- A chave privada deve ser usado apenas para a geração de assinatura, conforme especificado e deve ser mantido em segredo; a chave pública só pode ser utilizada para a verificação de assinatura, podendo ser tornada pública.
- A chave privada deve ser protegida contra acesso não autorizado, divulgação e modificação.
- A chave pública deve ser protegida contra modificações não autorizadas (incluindo substituição).
- O destinatário deve obter chaves públicas de forma confiável ;
- Um destinatário e um assinante deve ter a garantia da validade da chave pública.

3.4 Número secreto (*k*)

Um número secreto aleatório *k* deve ser gerado, por cada mensagem, antes da geração de cada assinatura digital para o uso durante o processo de geração de uma assinatura. Este número secreto deve ser protegido de divulgação e modificação não autorizada.

k^{-1} é o inverso multiplicativo de *k* em relação à multiplicação do módulo *q*; isto é, $0 < k^{-1} < q$ e $1 = (k^{-1} k) \bmod q$. Este inversa é necessário para o processo de geração de uma assinatura.

k e k^{-1} podem ser pré-computados, desde que o conhecimento da mensagem a ser assinada não seja necessário para a computação. Quando *k* e k^{-1} são pré-computados, a sua confidencialidade e integridade deve ser protegida.[1]

- **Geração do K**

Tal como para a geração das chaves, a geração do k tem dois métodos, sendo que apenas um deve ser escolhido um algoritmo de entre os dois sugeridos. O primeiro algoritmo sugerido "Geração do numero secreto usando bits aleatórios extra". Este método tem como input p,q e g, e o k e k^{-1}. Os passos envolvidos neste processo são:

1. N = tamanho(q), L = tamanho(p).

2. Se o par(L, N) for considerado inválido, segundo aqueles que são recomendados, retorna ERRO;
3. *requested_security_strength* = segurança associada ao par (L, N); (especificada na tabela 1)
4. Obter a *string* N+64 *returned_bits* apartir de RBG, em que RBG tem uma segurança associada, igual ou superior a *requested_security_strength,* Caso RBG não seja maior, retorna ERRO;
5. Converter *returned_bits* para um inteiro não negativo *c;*
6. $k = (c \bmod (q-1)) + 1$
7. Cálculo de k^{-1}
8. Retornar o estado de k e k^{-1}

No segundo método intitulado "Geração do numero secreto por teste de candidatos", um número aleatório é obtido e testado para determinar se o k obtido se encontra na gama de valores admissíveis. Se k é "fora-de-gama", ou seja não se encontra nos valores admissíveis, um outro número aleatório é criado. Isto é obtido por teste do passo 6 e novas iterações caso a condição não se verifique.

1. N = tamanho(q), L = tamanho(p).
2. Se o par(L, N) for considerado inválido, segundo aqueles que são recomendados, retorna ERRO;
3. requested_security_strength = segurança associada ao par (L, N); (especificada na tabela 1)
4. Obter a string N+64 returned_bits apartir de RBG, em que RBG tem uma segurança associada, igual ou superior a requested_security_strength, Caso RBG não seja maior, retorna ERRO
5. Converter returned_bits para um inteiro não negativo c;
6. Se c>q-2, retornar ao passo 4.
7. (estado, k-1)= inverso(k,q)
8. Retorna estado, k, e k-1

4. Geração da assinatura do DSA

A assinatura de uma mensagem M consiste na obtenção par de números *r* e *s* que é calculado de acordo com as seguintes equações:

$r = (g^k \bmod p) \bmod q$.

$s = (k^{-1} (H(M) + xr)) \bmod q$.

Para criar uma assinatura, um usuário calcula o valor de r e s, que são funções de componentes de chave pública (*p*, *q* e *g*), e da chave privada do usuário *x*, e das função de hash H(M), e do inteiro *k* adicional individual para da assinatura.

Os valores de *r* e *s* devem ser verificados para determinar se $r = 0$ ou $s = 0$. Se qualquer razão $r = 0$ ou $s = 0$, um novo valor de *k* deve ser gerado, bem como a assinatura deve ser recalculada, isto porque é extremamente improvável que *r* e *s* sejam valores nulos, quando as chaves são bem geradas.

5. Verificação e validação da assinatura

A assinatura digital é verificada utilizando os mesmos parâmetros de domínio, *y* chave pública que está matematicamente relacionado com a chave privada *x* usada para gerar a assinatura digital, a informação a ser verificado, e a mesma função hash que foi usada utilizado durante a geração de uma assinatura.

O último passo do processo é a verificação e validação da assinatura, que comprovarão a validade da mesma. Esta pode ser efectuada por qualquer parte envolvida no processo seja elas o usuário ou o destinatário da mensagem, ou ainda outra qualquer parte. Antes de verificar a mensagem assinada, o destinatário da mensagem, ou aquele que a vai decifrar terá de receber de maneira autenticada os valores *p*, *q*, *g* e a chave pública do emissor.

Seja M ', r' e s' as versões que foram recebidas de M, r, e s, respectivamente, e seja *y* a chave pública do assinante. O primeiro passo da verificação e validação passa por verificar se 0 <r '<q e 0 <s' <q. Caso qualquer uma das condições não se verifique a assinatura deve ser considerada inválida.

Se o primeiro passo for verificado pode-se então prosseguir para o segundo passo, e os seguintes parâmetros são então calculados.

$w = (s')^{-1} \bmod q;$

$u1 = ((H(m'))w) \bmod q;$

$u2 = ((r')w) \bmod q;$

$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q.$

No final, caso se prove que v=r' então a assinatura é considerava válida.

Na imagem 1 oferece uma visão geral sobre o processo de assinatura (à esquerda), verificação e validação (à direita). [1],[2]

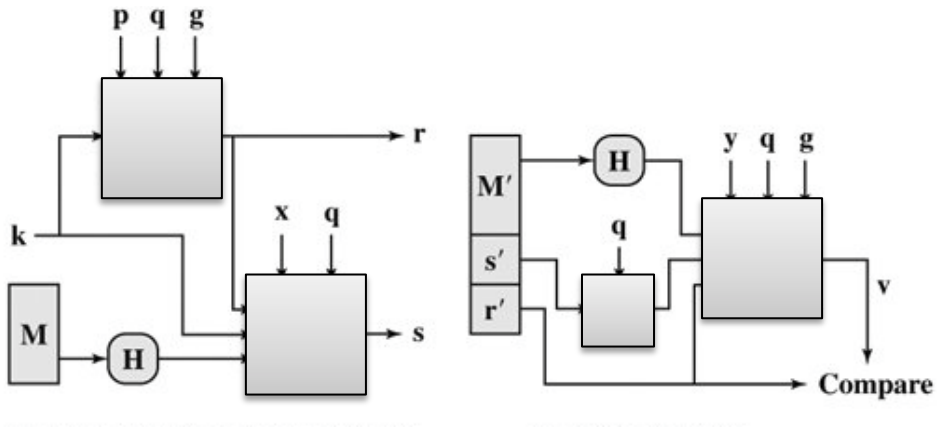

Figura 1: Geração e validação da assinatura digital

6. Ataques ao DSA por redução dos parâmetros.

Tal como a maioria dos algoritmos também o DSA está sujeito aos mais variados tipos de ataques. Alguns exemplos dos mesmo são, o ataque *p* de Pollard, extremamente eficiente para valores pequenos de q; o ataque λ de Pollard, eficiente quando se sabe de à priori que a chave privada localiza-se num intervalo de tamanho reduzido; o ataque de Ataque de Vaudenay com base na observação que a função de hashing efetivamente utilizada não é SHA-1(M), mas SHA-1(M) mod q, entre outros.[5] No entanto será analisada a influencia da redução dos parâmetros p, q e k, na possibilidade de se abrir possibilidades a ataques.

No DSA, *p* varia de 512 a 1024 bits. Reduzir o parâmetro *p* implica em abrir possibilidades para o ataque ao logaritmo discreto, quebrando a chave privada *x* pela equação de $y = (g^x) \bmod p$.

Num segundo cenário o parâmetro *q* é reduzido. Isto implica que o tamanho da assinatura também diminua, dado que as equações *r* e *s* são da ordem de *q*. Supondo que q tendo 16 bits, gerando, assim, assinatura de 32 bits. Como a chave privada *x* é da ordem de *q*, ela poderá ser quebrada com no máximo 65535 tentativas pela equação de $y = (g^x) \bmod p$, ou seja, atribuindo no máximo 65535 valores para *x*, no intuito de encontrar o valor que gere o *y* correspondente. Uma outra consequência da redução de *q* diz trata-se da redução do número de assinaturas (combinações de *r* e *s*), uma vez que tais equações são da ordem de *q*. Com o *q* menor, a chance de colisões de assinatura também aumenta. A colisão ocorre quando mensagens distintas resultam em mesma assinatura

Um outro problema encontra-se na redução do parâmetro *k*, que também é da ordem de q, que é possível de quebrar por $r = (g^k \bmod p) \bmod q$, sendo que obter o valor de k, implica aceder ao à chave privada através do valor da equação. [2],[4]

7. Tecnologias associadas ao DSA

- ## OpenSSL

SSL é uma sigla de *Secure Sockets Layer*. É o padrão por trás da comunicação segura na Internet, integrando criptografia de dados e protocolos. Os dados são encriptados antes mesmo de sair do computador de origem e são descriptados apenas ao chegarem a seu destino. O OpenSSL é uma implementação de código aberto dos protocolos SSL e TLS.

Geralmente de forma a obter uma autenticação bidirecional é comum usar-se protocolos que conjugam o DSA conjuntamente com o Diffie-Hellman. Para que duas partes possam utilizar este algoritmo primeiramente cada parte terá de assinar a sua mensagem, após isto poderão enviar as mensagens.

A livraria do OpenSSL permite, através das funções que fornece criar os parâmetros necessários, gerar chaves, assinar a mensagem, verificar e descodificar a mensagem. Apesar de não se entrar em grandes pormenores nesta monografia sobre a especificação das funções necessárias, estas serão brevemente referenciadas.

A primeira função a ser utilizada trata-se da *DSA_generate_parameters* que inicializa um novo objecto de DSA com novos parâmetros de *p, q*, e *g*. Uma vez gerados os parâmetros anteriormente especificados são gerados os pares de chaves. O OpenSSL disponibiliza uma função especifica para tal, a *DSA_generate_key*. Esta função recebe como argumento um objecto de DSA. Se as chaves forem geradas com sucesso ela retornará um valor diferente de zero, enquanto que em caso de erro retorna zero. As chaves geradas são armazenas como *pub_key* e *prive_key*. Para a assinatura do documento a libraria disponibiliza duas opções de funções o int *DSA_sign_setup* e int *DSA_sign* . Por último a verificação é efectuada pela função *DSA_verify*.[7]

- ## PCKS

O PKCS é uma coleção protocolos proposta pela RSA Data Security, Inc, utilizando sistemas assimétricos para realizar as seguintes tarefas: assinatura digital, certificação digital.

No Standard 11 é possível encontrar especificado os mecanismos de implementação do DSA através de funções especificas da API do próprio PCKS.

À semelhança do OpenSSL existem funções especificas para cada um dos passos envolvidos na geração dos parâmetros, das chaves, assinatura e verificação. Assim, a função CKM_DSA_KEY_PAIR_GEN é definida com o intuito de gerar os pares de chaves. Por sua vez CKM_DSA_PARAMETER_GEN tem a intenção de gerar os parâmetros. Por último existem ainda dois mecanismos CKM_DSA e CKM_DSA_SHA1. O primeiro não utiliza a função de hash e destina-se apenas a assinaturas únicas, enquanto que o segundo já faz uso da mesma e destina-se a assinaturas únicas e simples. [6]

8. Conclusão

O algoritmo DSA trata-se de uma assinatura credível e que assegura a autenticidade do documento e da entidade que o gerou. Este algoritmo garante a autenticidade da assinatura através de gerações credíveis e verificadas de todos os parâmetros envolvidos, e não só durante os passos de verificação e validação da assinatura propriamente dita. É de notar que qualquer desrespeito à recomendações pode implicar uma redução drástica da segurança e um comprometimento da assinatura. A escolha adequada da força de segurança da função de hash, dos par(L,N) deve ser realizada em função da segurança desejada, sendo que a força de segurança da função de hash e par(L,N) devem ser sempre iguais superiores à segurança desejada, e por sua vez a segurança da função de hash nunca deve ser inferior à segurança estabelecida para o par(L,N).

Tal como é esperado, também o DSA está susceptível aos mais variados ataques, neste artigo foram analisados os efeitos da redução dos parâmetros p, q e k. A redução do parâmetro p abre lugar ao ataque à chave privada através do ataque ao logaritmo discreto. A redução de q possibilita o ataque a k e x, e ainda a diminuição do universo de chaves geradas abrindo possibilidade a colisões.

Bibliografia

[1] Digital Signature Standard", National Institute of Standards and Technology. FIPS PUB 186-4, NIST, 2013.W. Stalings.

[2] Criptografia e segurança de redes. *Person*, 4 edição

[3] Recommendation for Key Management – Part 1: General (Revised). Special Publication 800-57, NIST, (2007)

[4] P. Barreto, Sobre a segurança de assinaturas digitais baseadas no logaritmo discreto em subgrupos de tamanho reduzido. *Laboratório de Arquitetura e Redes de Computadores. Escola Politécnica da Universidade de São Paulo*

[5] A.C. Silva, D.S Freitas,L.C. Zancanella. Análise da vulnerabilidade do DSA com parâmetros curtos, *Laboratório de Segurança em Computação, Universidade Federal de Santa Catarina (UFSC)*.

[6] "PKCS #11 Mechanisms v2.30: Cryptoki – Draft 7", RSA laboratories

[7] C. Pravir, M.Massier, J.Viega. Network Security with OpenSSL. O'Reilly, 2002, pag.195-200