ALGORITMOS PARA TOKENS DE AUTENTICAÇÃO

3 authors, including:

W.V. Ruggiero
University of São Paulo
129 PUBLICATIONS **760** CITATIONS

ALGORITMOS PARA TOKENS DE AUTENTICAÇÃO

Gustavo Yamasaki Martins Vieira , Wilson Vicente Ruggiero
LARC-PCS-POLI-USP
Av. Prof. Luciano Gualberto, trav. 3 - 158, sala C2-46, Cidade Universitária, São Paulo - SP - Brasil

RESUMO

A autenticação é a primeira etapa no acesso à página web de qualquer instituição financeira. Nela o usuário apresenta a sua identidade digital e os servidores concedem acesso se a identidade é válida no sistema. Com o crescente ataque de phishing, scams e spywares na internet, este texto propõe o uso de autenticação de múltiplos fatores, através do emprego de um dispositivo dedicado porém construído em cima de um hardware de baixo custo e algoritmos de criptografia conhecidos por todos. Aqui são analisados dois algoritmos: CMAC e HMAC. Não apenas autenticação de usuário é considerada, mas também a autenticação de transações on-line.

PALAVRAS CHAVE

Segurança, autenticação, token, CMAC, HMAC

1. INTRODUÇÃO

A autenticação é a primeira etapa no acesso à página web de qualquer instituição financeira. Nela, o usuário apresenta sua identidade digital e os servidores concedem acesso se a identidade é válida no sistema.

Atualmente a identidade digital é verificada com o uso de autenticação baseada em conhecimento, isto é, o usuário é autenticado se conhece o seu login e a senha no sistema. Entretanto, ataques como phishing, scams e spywares, hoje freqüentes na internet, têm como objetivo obter os dados de login e senha, e conseqüentemente a identidade digital dos usuários.

Com a finalidade de impedir esses ataques surgiram estudos que empregam objetos em conjunto com senhas para autenticar os usuários. Os objetos, conhecidos como tokens, são utilizados para gerar OTPs (one-time passwords), isto é, senhas utilizadas uma única vez e então descartadas. Desta forma, a cada autenticação, não basta para um atacante obter apenas a senha, mas também é preciso conhecer a OTP que será escolhida para a sessão.

Existem trabalhos que propõe o uso de celulares como token que recebem mensagens de SMS como forma de enviar as OTP ao cliente (Claessens, Preneel et al. 2002). Isso é interessante por utilizar canais separados, e por isso mais difíceis de serem controlados por um atacante, no entanto isso representa um custo adicional para cada autenticação.

Há trabalhos que aproveitam a capacidade computacional de dispositivos como PDAs, celulares e mais recentemente smartphones para fazer o papel do objeto na autenticação. Programas em Java, por exemplo, rodando nesses ambientes são responsáveis por gerar as OTP para o usuário (Me, Pirro et al. 2006). Freqüentemente eles se baseiam na hipótese de que esses dispositivos são considerados isolados do computador, isto é, seguros para serem adotados como tokens. Entretanto o descuido de seus usuários (Hong 2005) e pragas como vírus (Terra 2006) e cavalos-de-tróia (IDG 2006) em plataformas móveis vêm surgindo e com o maior emprego dos mesmos haverá uma tendência natural (Felitti 2006) de surgirem ataques para tais ambientes.

Uma alternativa para manter a isolação é a adoção de dispositivos dedicados a essa função. Este artigo apresenta rapidamente a base de um dispositivo de baixo custo de produção através do uso de componentes de prateleira, ou seja, com produção em larga escala. A seguir foca-se no uso de algoritmos de código aberto. É mostrado como algoritmos para geração de OTP e autenticação de transações podem ser empregados e o

desempenho dos mesmos no ambiente de baixo custo. Esses algoritmos são baseados em MAC (message authentication code) e duas alternativas são analisadas: o CMAC-AES128 e HMAC-SHA1.

A seção 2 faz uma definição de token e destaca os blocos mais importantes a serem considerados no projeto de um token. Dando continuidade, a seção 3 mostra o hardware utilizado para analisar os algoritmos de MAC e então fala-se brevemente de algoritmos de desafio-resposta na seção 4. A seção 5 mostra como os algoritmos foram utilizados para OTP e autenticação e a análise de desempenho do CMAC e do HMAC no ambiente de baixo custo. O artigo se encerra com conclusões do trabalho.

2. AUTENTICAÇÃO DE MÚLTIPLOS FATORES

Tradicionalmente a identidade digital é verificada através de "*algo que o usuário sabe*", ou seja, é baseada em conhecimento. Nesse método, o usuário é autenticado quando o sistema atesta que o usuário tem o conhecimento de um segredo previamente combinado. Outros métodos, ou fatores, empregados para efetuar autenticação são "*algo que o usuário possui*", isto é, baseada em objetos e "*algo que o usuário é*", ou seja, baseada em biometria (O'Gorman 2003).

O uso simultâneo de diferentes fatores é conhecido como autenticação de múltiplos fatores. A combinação de diferentes fatores resulta em uma atenuação mútua de suas vulnerabilidades e, em conseqüência, um método mais seguro de verificação de identidade.

A verificação da posse do token é feita através de algoritmos de desafio-resposta. O servidor propõe um desafio ao usuário e o usuário entra com esse desafio no token. O token responde o desafio e o usuário pode se autenticar no sistema. O desafio possui duas funções. A primeira é verificar a posse do token. A segunda é servir como uma senha de uso único, ou OTP. A cada nova execução do protocolo de autenticação um novo desafio é empregado, resultando em uma nova OTP a cada acesso.

Apesar de ser comumente utilizado na autenticação de usuários, um token também pode ser empregado na autenticação de transações. Seja qual for o uso, a grande vantagem de um dispositivo externo de autenticação é a sua isolação. A chave de criptografia, os algoritmos, a entrada de dados bem como exibição de dados não estão sujeitos aos ataques realizados no computador com a função de roubar dados. Tanto as operações quanto os dados críticos ficam isolados do computador do usuário. Assim mesmo um sistema executando um spyware ainda pode ser utilizado para autenticação sem o comprometimento da identidade digital do usuário. Além da isolação, o roubo do dispositivo físico se torna mais um fator a ser considerado por um adversário atacante.

Blocos Funcionais

A apresenta os quatro blocos mais importantes no projeto e construção de um token:

- Construção do hardware
- Escolha do tipo de algoritmo desafio-resposta
- Software para o controle do token: é como se fosse um sistema operacional para controlar o hardware do sistema e os programas que executam o sistema.
- Escolha de um algoritmo de criptografia

Esse artigo tem como foco principal a análise de algoritmos para a geração de OTP e autenticação de transações, porém antes de chegar a esse ponto será rapidamente apresentado o hardware do token para que se possa contextualizar o ambiente adotado.

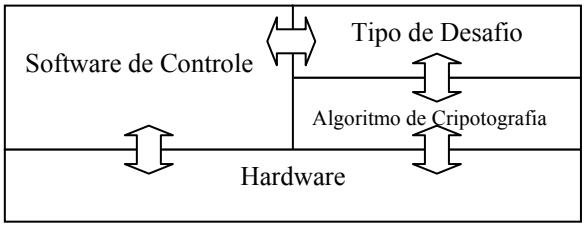

Figura 1. Blocos funcionais de um token

3. PLATAFORMA

Como token de baixo custo considerou-se que o custo de produção fosse inferior a cinco dólares por unidade. Dentro desse requisito, foi empregado um microcontrolador como a unidade responsável por executar os algoritmos de criptografia. Um microcontrolador é um circuito integrado que possui em uma única pastilha processador para uso geral, memória e periféricos como, por exemplo, porta serial ou controladores de LCD. Devido a essa flexibilidade e por ser produzido em grandes quantidades, um microcontrolador se tornou um candidato natural para esse sistema.

O microcontrolador considerado foi o PIC18F65J90 (Microchip 2007). O dispositivo é capaz de controlar LCD, tem portas para fazer interface com um teclado matricial (tipo de teclado utilizado em calculadoras portáteis) sem componentes adicionais, possui oscilador interno tornando a gerência de energia mais fácil, pois é possível acelerar ou reduzir a velocidade do sistema conforme o tipo de operação executada. O dispositivo ainda apresenta memória suficiente para armazenar o algoritmo de criptografia e os demais programas para gerenciar as funções do microcontrolador.

A interface do sistema é constituída de um display numérico e um teclado, sendo muito semelhante a calculadoras portáteis. Como o microcontrolador possui periféricos para controlar ambos os elementos eles podem ser diretamente conectados ao microcontrolador.

Mais da metade de todo o custo do sistema se dá pela presença do microcontrolador, sendo o mesmo cotado no site de seu fabricante a US$ 2,65. Os demais custos são atenuados devido à presença de muitos periféricos no próprio microcontrolador, minimizando hardware e conexões externas e por conseqüência o custo final.

Naturalmente outros aspectos do hardware podem ser discutidas nesse ponto, como por exemplo adequação ao FIPS 140-2 (NIST 2001), porém essa seção tem como objetivo apenas mostrar em que tipo de equipamento os algoritmos são executados e ao mesmo tempo dar uma idéia de como a interface do token pode ser concebida.

Não será mostrado nesse artigo porém o token baseado nesse ambiente pode atender vários dos requisitos do FIPS 140-2. Requisitos não atendidos incluem ambiente operacional (verificação de integridade de memória) e prevenção a ataques de análise de energia e análise de tempo. Os demais requisitos podem ser atendidos até o nível 2. Não foi verificado se o custo de tornar o dispositivo tamper-evident pode ser atendido e ainda assim manter o custo de produção inferior a 5 dólares.

4. ALGORITMOS DESAFIO-RESPOSTA

A verificação de posse de um token é feita através de algoritmos desafio-resposta. Nesses algoritmos o servidor propõe um desafio que apenas o cliente é capaz de responder através do uso de seu token. Esses desafios podem ser implícitos ou explícitos (Menezes, Oorschot et al. 2001):

- Desafio baseado em tempo: Utiliza a hora atual como entrada para o desafio. O servidor não precisa enviar o desafio, porém necessita que o relógio do token e do servidor estejam sincronizados.
- Desafio baseado em eventos: Utiliza uma seqüência numérica como desafio e é incrementada a cada uso. Também pode apresentar problemas de sincronismo pois o cliente pode gerar desafios no token e não utilizá-los.
- Desafio explícito: O servidor passa um desafio que o usuário deve digitar no token. Não apresenta problemas de sincronismo porém a interface é mais complicada por depender de um método de entrada, como um teclado.

Para esse sistema foi utilizado o algoritmo de desafio-resposta baseado em desafios explícitos, para evitar preocupações com sincronismo. Apesar disso, independente do tipo de algoritmo, o hardware e o uso do algoritmo de criptografia são muito semelhantes em todos os casos. O desafio baseado em tempo ainda exigiria um cristal externo para minimizar o erro de sincronismo dos relógios com o passar do tempo.

5. ALGORIMOS DE CRIPTOGRAFIA

O token baseado em microcontrolador deve executar um algoritmo de criptografia que atenda a critérios de baixo consumo de memória e pouca necessidade de processamento.
Além dos critérios de desempenho, selecionou-se algoritmos conhecidos e públicos para que qualquer um que deseje implementar o sistema conheça o seu funcionamento e ao mesmo tempo evitar que vulnerabilidades conhecidas estejam presente em seu código. Isso também o torna livre do pagamento de royalties e o seu uso não pesa no custo final do sistema.

Algoritmos Avaliados

Foi avaliado o desempenho de dois algoritmos de MAC para realizar autenticação. O primeiro foi o CMAC-AES128 e o outro algoritmo verificado foi o HMAC-SHA1.

O CMAC é baseado em cifras de blocos sendo descrito originalmente com o nome de OMAC em (Iwata and Kurosawa 2003). Seu nome foi oficializado em (NIST 2005). Este projeto utilizou como cifra de blocos o AES com chave e blocos de 128 bits. O HMAC utiliza funções de hash para calcular o MAC. Sua descrição pode ser encontrada em (NIST 2002) e a implementação aqui utilizada é baseada no SHA1. Ambos foram escolhidos por serem funções autenticação de mensagens recomendadas em (NIST 2001).

Uso dos Algoritmos

Neste trabalho os algoritmos de MAC são utilizados para garantir integridade e prover autenticação unilateral. Como os dados devem ser digitados por uma pessoa, por razões de ordem prática, é preciso utilizar um MAC reduzido, isto é, o valor de MAC truncado. Tanto o tamanho do desafio quanto o tamanho da resposta podem ser configurados e foram escolhidos de forma a balancear segurança e facilidade de uso. Por isso foi definido que tanto desafio quanto resposta podem assumir 10^6 valores diferentes, isto é, um número decimal de 6 dígitos.

O valor do desafio é um número de 9 dígitos decimais sendo composto por 6 dígitos aleatórios. Os outros 3 dígitos são o MAC reduzido dos outros 6 dígitos com o intuito de que o cliente possa fazer verificação de integridade e detectar erros de digitação. O valor da resposta é dada pelos 6 últimos dígitos do MAC desafio enviado. Como se tratam de dígitos decimais, o MAC calculado é convertido para o seu formato decimal. O resultado é então truncado de forma a apresentar o número de dígitos necessários em cada um dos usos.

Como tanto desafio quanto resposta nascem a partir do MAC de 6 dígitos, a distinção entre ambos é feita através de um sal anexado aos 6 primeiros dígitos do desafio de forma transparente ao usuário. O sal do desafio é 0xFF e o sal da resposta é 0x01.

A escolha dos bits a serem utilizados no MAC reduzido foi feita de forma arbitrária já que os algoritmos garantem que os bits aparecem de forma equiprovável dentro do valor do MAC. Em (M'Raihi, Bellare et al. 2005) é utilizada uma forma diferente para escolha de um MAC reduzido.

Uma característica importante nas OTP é que uma OTP anterior não apresente relação com a OTP atual. Para verificar essa propriedade, foi feito o teste de qui-quadrado (Montgomery and Runger 2003) para diferentes chaves onde se geravam 1000 OTP por chave. Os testes apontaram para uma distribuição uniforme, indicativo de pouca relação entre as OTP geradas.

A Tabela 1 apresenta como as mensagens são trocadas entre o usuário e o token e entre o usuário e o servidor.

Tabela 1. Uso dos algoritmos de MAC para OTP

	Cliente		Servidor	Descrição
1			Gera RND	O servidor gera um número aleatório de 6 dígitos (RND)
2			Calcula INT: INT = CMAC(0xFF | RND)	Servidor calcula o CMAC de 0xFF (sal) concatenado com RND. Pega-se os 3 últimos algarismos do CMAC para se gerar o código de integridade do desafio (INT).
3	Cliente	← Desafio	Desafio = (RND | INT)	O desafio enviado ao cliente é composto de 9 dígitos: RND e INT.
4	Cliente (*Token* ← Desafio)			O cliente digita o desafio no *token*.
5	*Token*: INT=CMAC(RND)?			O *token* verifica se INT corresponde ao RND e pára o processo em caso de erro.
6	*Token*: RSP=CMAC(0x01 | RND)			O *token* utiliza o 0x01 (sal) concatenado com RND como entrada para um novo CMAC, gerando RSP.
7	*Token*: RSP			O *token* pega os 6 últimos algarismos de RSP e exibe o resultado para o usuário.
8	Cliente	→ RSP	Servidor	O usuário envia os dados para o servidor.
9			CMAC(0x01 | RND) = RSP?	O servidor verifica RSP e finaliza a autenticação se o valor estiver correto.

* O uso da chave está implícito nas chamadas do MAC

Mesmo com o uso de autenticação de múltiplos fatores, um sistema de autenticação ainda pode ser burlado como apresentado em (Schneier 2005) e por isso não basta o token gerar OTP, mas também autenticar cada transação considerada crítica, como transferências de dinheiro, executadas no ambiente da internet.

O processo de autenticação deve ser tal que o usuário digite as transações no token de forma que ele seja capaz de associar a transação com o que está sendo efetivamente digitado. Abaixo se tem o exemplo de como seria uma possível transação financeira e a forma como ela poderia ser digitada no *token*. Primeiro deve-se notar que qualquer transação pode ser representada facilmente apenas por números:

- Favorecido: João da Silva
- Tipo de transação: 8 (Depósito)
- Banco: 034
- Conta Corrente: 0197
- Valor: $ 1.500,00
- Data: 03/04/2007
- Nonce: 1
- INT: 2914 (Código de integridade)

Para esta transação, pode-se ter o número destacado na Tabela 2.

Tabela 2. Exemplo de autenticação de transações

Transação	Banco	Conta	Valor	Data (dia/mês/ano)		Nonce	INT
8	034	0197	1500	0304	2007	1	2914

No exemplo anterior, "favorecido" não é necessário para autenticação, pois banco e conta já identificam o cliente, assim, o identificador da transação (TRS) que deve ser digitado no token é 8 034 0197 1500 0304 2007 1 2914. A informação foi dividida em blocos de no máximo 4 dígitos para facilitar a entrada no token. Com a informação visualmente clara para o usuário, ele pode ter certeza da informação que está sendo autenticada. O valor INT funciona como um MAC reduzido para que o token possa verificar a integridade da transação e assim o cliente tem a certeza de estar autenticando a transação adequada. Além desses números, associar hora e data e/ou algum número único à transação é importante para evitar ataques do tipo replay.

O algoritmo de OTP anterior foi empregado modificando-se o sal da transação de 0xFF para 0xFE e na autenticação de 0x01 para 0x02. Assim a mesma chave não irá gerar números válidos para uma OTP. O algoritmo é descrito na Tabela 3. O tamanho do código de autenticação da transação ainda é de 6 dígitos.

Tabela 3. Uso dos algoritmos de MAC para autenticação

	Cliente		Servidor	Descrição
1			Recebe TRS	O servidor recebe a solicitação de uma transação (TRS)
2			Calcula INT: INT = CMAC(0xFE | TRS)	Servidor calcula o CMAC de 0xFE (sal) concatenado com TRS. Pega-se os 4 últimos algarismos do CMAC para se gerar o código de integridade da transação (INT).
3	Cliente	← TRSI	TRSI = TRS | INT	A transação é formatada de forma que o cliente associe a transação com o número que será digitado.
4	Cliente (*Token* ← TRSI)			O cliente digita a transação no *token*.
5	*Token*: INT=CMAC(0xFE | TRS)?			O *token* verifica se INT corresponde ao TRS e pára o processo em caso de erro.
6	*Token*: RSP=CMAC(0x02 | TRS)			O *token* calcula CMAC de 0x02 (sal) concatenado com TRS.
7	*Token*: RSP			Pega-se os 6 últimos algarismos do CMAC (RSP) e exibe o resultado para o usuário.
8	Cliente	→ RSP	Servidor	O usuário envia o código de autenticação para o servidor.
9			CMAC(0x02 | TRS) = RSP?	O servidor verifica RSP e autentica a transação se o valor estiver correto.

* O uso da chave está implícito nas chamadas do CMAC

Desempenho

Ambos os algoritmos foram implementados no PIC18F65J90 e foram analisados sob dois aspectos: tempo de execução percebido pelo usuário (não se avaliou apenas o algoritmo, mas o tempo entre o usuário terminar de digitar o desafio e o tempo para o token exibir alguma resposta) e consumo da memória de programa do microcontrolador.

Como critérios para considerar um algoritmo adequado foi exigido que o algoritmo fosse capaz de devolver uma resposta em no máximo três segundos para evitar desconforto no uso diário de um token e que ele coubesse dentro da memória disponível do dispositivo adotado (no caso 32 Kbytes).

O HMAC consumiu quase 16 kbytes de memória de programa do microcontrolador e para obter uma resposta dentro dos critérios adotados, o processador funcionou a 4 MHz para verificar a validade de um desafio e gerar a resposta do mesmo.

O CMAC ocupou menos de 2 kbytes da memória de programa do microcontrolador. Rodando a 1 MHz para verificar a validade de um desafio e gerar a resposta do mesmo, foram necessários pouco menos de 2 segundos.

A tabela a seguir compara o desempenho de dois algoritmos apresentando exatamente os valores obtidos. A partir dela pode-se afirmar que, para o caso de um sistema de chaves simétricas, um algoritmo seguro é suficientemente rápido (dentro do critério de três segundos adotado) para ser executado em um microcontrolador sem a necessidade de qualquer tipo de hardware dedicado para essa função.

Tabela 4. Comparação HMAC x CMAC em ambiente de 8 bits

Algoritmo	Tempo de Execução	Memória
HMAC	2,84±0,04s (4 MHz)	16.056 bytes
CMAC	1,52±0,05s (1 MHz)	1.674 bytes
Observações	CMAC 7,4 vezes mais rápido	CMAC 9,5 vezes menor

Ao comparar os números é preciso ter em mente que os testes de desempenho foram realizados dentro de uma plataforma de 8 bits. Enquanto o SHA1 é otimizado para ser executado em plataformas de 32-bits, o AES pode ser implementado de forma a aproveitar a característica de diversas plataformas, inclusive a adotada aqui nesse comparativo. O SHA1 ainda apresenta carries em suas somas que não aparecem no AES. Além do problema da plataforma, é preciso lembrar que SHA1 foi enfraquecido por ataques em (Wang, Yin et al. 2005). Apesar dos ataques, o algoritmo não pode ser considerado “quebrado”.

Como apresentado na Tabela 4, o CMAC tem um melhor desempenho e sugere-se o seu uso para o propósito aqui descrito. Apesar disso, o HMAC ainda é uma alternativa viável dado que o mesmo ainda atende aos requisitos apresentados inicialmente. Isso é possível pois o microcontrolador escolhido tem a capacidade de variar a sua velocidade em tempo de execução, isto é, tarefas que necessitem de pouco poder

de processamento são executadas em baixas velocidades, por exemplo 250KHz, e quando necessário executar o algoritmo de criptografia alterna-se para um modo mais veloz sem grande impacto na vida útil da bateria do sistema.

6. CONCLUSÕES E TRABALHOS FUTUROS

A autenticação de dois fatores tem se mostrado como uma alternativa para tornar a identidade digital de usuário mais difícil de ser roubada através de ataques comuns na internet como phishing e o spywares. Nesse contexto, o artigo apresentou brevemente o hardware para um token de baixo custo e focou em algoritmos para a geração de OTP.

Como outros trabalhos já mostraram, apenas a autenticação de usuário não é suficiente devido a ataques man-in-the-middle. Por isso apresentou-se também uma alternativa de uso do token para que ele também execute autenticação de transações.

A seguir dois algoritmos de código aberto são avaliados para fornecer o suporte a esses serviços. Verifica-se nessa análise tempo de execução dos algoritmos e consumo de memória. A partir dessa análise mostrou-se que é possível implementar um MAC dentro do ambiente de baixo custo apresentado com tempo de resposta aceitável.

Naturalmente, os algoritmos não precisam se limitam apenas à plataforma apresentada e podem ser adotados em outras situações como smartphones.

Este trabalho pode ainda ser expandindo detalhando-se a implementação do hardware do token e dos programas para suporte do mesmo e como ele pode atender a requisitos de nível 2 do FIPS140-2. A explicação do funcionamento do lado do servidor do sistema também seria de grande valor.

REFERÊNCIAS

Claessens, J., B. Preneel, et al. (2002). "Combining World Wide Web and wireless security." Informatica: 123-132.

Felitti, G. (2006). "Vírus para celulares dobram durante 2006, segundo análise da McAfee." Retrieved 27/Jul/2007, from http://idgnow.uol.com.br/seguranca/2006/12/21/idgnoticia.2006-12-21.4137157066.

Hong, J. I. (2005). Minimizing security risks in ubicomp systems. Computer. **38:** 118-119.

IDG. (2006). "F-Secure alerta para 1ª praga que rouba dados de celulares." Retrieved 10/Jun/2007, from http://idgnow.uol.com.br/seguranca/2006/03/30/idgnoticia.2006-03-30.0073893494.

Iwata, T. and K. Kurosawa (2003). OMAC: One-Key CBC MAC, Springer**:** 24-26.

M'Raihi, D., M. Bellare, et al. (2005). "RFC4226 - HOTP: An HMAC-Based One-Time Password Algorithm."

Me, G., D. Pirro, et al. (2006). A mobile based approach to strong authentication on Web. Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on.

Menezes, A. J., P. C. v. Oorschot, et al. (2001). Handbook of Applied Cryptography, CRC Press.

Microchip (2007). PIC18F85J90 Family Data Sheet, Microchip.

Montgomery, D. C. and G. C. Runger (2003). Estatística aplicada e probabilidade para engenheiros. Rio de Janeiro, LTC Editora.

NIST (2001). "FIPS140-2 Security requirements for cryptographic modules."

NIST (2002). "FIPS198 - The Keyed-Hash Message Authentication Code (HMAC)." 20.

NIST (2005). "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication."

O'Gorman, L. (2003). "Comparing passwords, tokens, and biometrics for user authentication." Proceedings of the IEEE **91**(12): 2021-2040.

Schneier, B. (2005). Two-factor authentication: too little, too late. Communications of the ACM. **48**.

Terra. (2006). "Descoberto vírus que se transfere do PC para o celular." Retrieved 01/Jun/2006, from http://tecnologia.terra.com.br/interna/0,,OI901779-EI4805,00.html.

Wang, X., Y. L. Yin, et al. (2005). Finding Collisions in the Full SHA-1. Crypto'05.