Avaliação de assinaturas baseadas em *hash* para a Internet das Coisas

Jéssica C. Carneiro[1], **Leonardo B. Oliveira**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

{jessicacarneiro,leonardo.barbosa}@dcc.ufmg.br

Abstract. *With the advent of quantum computing, traditional signatures such as RSA, DSA, and ECDSA will become unsafe thanks to Shor's algorithm. Thus, cryptosystems resilient to quantum computing must be studied, specially for devices in the Internet of Things which limitations may hinder the use of secure cryptosystems. Alternatives are the hash-based signatures (HBS) which are post-quantum and provide a trade-off between performance and security level. This paper presents an implementation of HBS schemes on an Arduino Due platform and a performance evaluation.*

Resumo. *Com o advento da computação quântica, assinaturas tradicionais como RSA, DSA e ECDSA tornarão-se inseguras graças ao algoritmo de Shor. Dessa forma, criptossistemas resilientes à computação quântica devem ser estudados, especialmente para dispositivos da Internet das Coisas cujas limitações podem dificultar a utilização de criptossistemas seguros. Uma alternativa são as assinaturas baseadas em* hash *(*hash-based signatures - *HBS) que são pós-quânticas e proveem um compromisso entre desempenho e nível de segurança. Este trabalho apresenta a implementação de esquemas HBS sobre a plataforma Arduino Due e uma avaliação de desempenho.*

1. Introdução

Assinaturas digitais são uma forma de associar informações a uma entidade, seja ela uma pessoa ou um dispositivo [Menezes et al. 1996]. Elas fornecem autenticação, integridade e irretratabilidade de informações trocadas em uma rede. Atualmente, assinaturas digitais são amplamente utilizadas em serviços na Web, computadores pessoais, dispositivos IoT, etc. Alguns esquemas de assinatura tais como RSA [Rivest et al. 1978], DSA [ElGamal 1985] e ECDSA [Johnson et al. 2001] estão entre os criptossistemas de chave pública mais conhecidos e utilizados [Buchmann et al. 2009].

Os criptossistemas citados têm sua segurança pavimentada na intratabilidade de problemas matemáticos tais como fatoração de números inteiros (RSA), logaritmos discretos (DSA) e logaritmos discretos em curvas elípticas (ECDSA). No entanto, a descoberta por Shor [Shor 1994] de algoritmos polinomiais quânticos para resolução de tais problemas os tornaria inseguros. Consequentemente, alternativas devem ser avaliadas para o período pós-quântico.

É especialmente necessário avaliar criptossistemas pós-quânticos adequados para a Internet das Coisas pois espera-se que o número de dispositivos cresça significantemente[1]. Dispositivos para a IoT possuem recursos computacionais restritos o que impõe

[1]http://www.gartner.com/newsroom/id/2636073

uma limitação a quais criptossistemas são viáveis para tais plataformas. As assinaturas baseadas em *hash* (*hash-based signatures* - HBS) são alternativas possíveis para a IoT pois dependem apenas de funções *hash* e não são necessárias operações caras como no esquemas RSA e ECDSA, por exemplo.

As HBS permitem um compromisso entre desempenho e nível de segurança. Também, caso a função *hash* empregada seja considerada insegura, o criptossistema pode continuar sendo utilizado apenas trocando-se a função, o que fornece uma maior flexibilidade às assinaturas. Além disso, sua segurança depende somente da propriedade de resistência à colisão da função escolhida [Buchmann et al. 2009].

A escolha pelas HBS se deu pelo fato de elas serem rápidas, possuírem chaves menores e sua segurança ser melhor entendida em relação a outras alternativas pós-quânticas como Criptografia baseada em Reticulados, Criptografia baseada em Códigos Corretores de Erro e Criptografia Multivariada [Hülsing et al. 2015].

As contribuições deste trabalho são:

1. Uma explicação didática das HBS na seção 2;
2. Implementação dos esquemas Winternitz (W-OTS) e Merkle *Signature Scheme* (MSS) na seção 3;
3. Avaliação de desempenho em termos de armazenamento, processamento e comunicação na seção 4;
4. Reunião de outros trabalhos sobre esquemas pós-quânticos e sobre HBS em dispositivos restritos na seção 5.

2. Fundamentação Teórica

As HBS foram criadas por Ralph Merkle em [Merkle 1989]. Elas ganharam atenção como alternativas no cenário pós-quântico aos esquemas usados atualmente. Sua construção requer somente a existência de funções *hash* criptográficas e sua segurança depende da propriedade de resistência à colisão da função usada [Buchmann et al. 2009].

Dentre as principais vantagens das HBS estão: (i) a simplicidade de sua implementação; (ii) sua segurança é bem entendida [Buchmann et al. 2011]; (iii) suas chaves e assinaturas são menores que outras opções pós-quânticas (exemplo: Criptografia baseada em Reticulados); (iv) o tempo gasto na geração de assinaturas e verificação é competitivo mesmo em relação a esquemas tradicionais [Hülsing et al. 2012]. Algumas desvantagens são o número limitado de assinaturas que podem ser geradas com um par de chaves e a impossibilidade de utilizá-las para encriptação.

2.1. Tipos de HBS

Os esquemas HBS são divididos em dois tipos principais: *One-Time Signatures* (OTS) e *Multi-Time Signatures* (MTS). O primeiro permite a geração de uma única assinatura por par de chaves. Portanto, as OTS não são úteis na prática. No entanto, elas são utilizadas para a construção de esquemas MTS que permitem a assinatura de um número maior de mensagens.

A figura 1 mostra como é gerada uma assinatura no esquema LD-OTS, a mais simples OTS. Na figura 1, n representa o tamanho do resumo criptográfico produzido pela função *hash* f em bits, SK é a chave privada, PK a chave pública e b_i representa

o i-ésimo bit do resumo criptográfico da mensagem a ser assinada para $0 \leq i < n$. sk_i é uma sequência de bits de tamanho n selecionada aleatoriamente. Obtém-se PK aplicando-se f sobre cada sequência de SK. A assinatura é gerada através da seleção da sequência $sk_{i,0}$ de SK caso o i-ésimo bit de b seja 0 ou $sk_{i,1}$ caso seja 1. Como é possível perceber, um esquema OTS revela parte de sua chave privada, portanto, uma segunda assinatura poderia revelar a um adversário toda a SK. É importante salientar que o nível de segurança das HBS apresentadas é definido como $n/2$ [Buchmann et al. 2011].

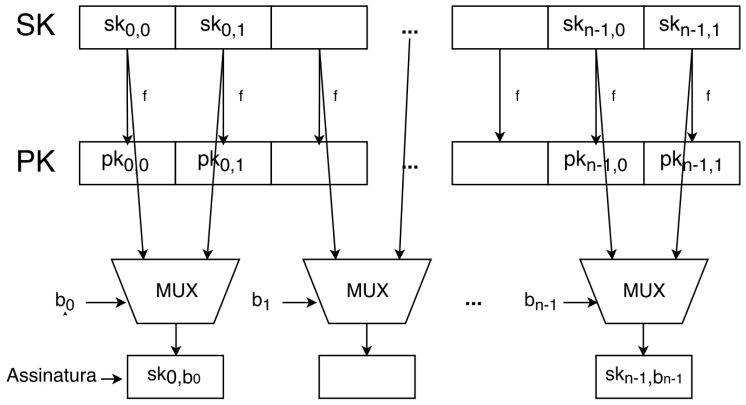

Figura 1. Geração de assinatura em LD-OTS. Fonte: PQCRYPTO17[2]

Para a verificação de uma assinatura LD-OTS, um processo similar ao de sua geração é feito. No entanto, para cada bit da mensagem a ser verificada, obtém-se a respectiva sequência (0 ou 1) de PK ao invés de SK. Então, f é calculado para todas as sequências sk_i, n da assinatura e esse resultado deve ser comparado ao do passo anterior. Como $f(SK)$ = PK, os dois resultados serão iguais se a assinatura for válida para a mensagem correspondente.

2.2. W-OTS

É possível notar que as chaves de LD-OTS possuem um tamanho consideravelmente grande ($2n^2$). Isso ocorre pois cada bit da mensagem é assinado individualmente. O esquema W-OTS diminui o tamanho da chave permitindo a assinatura de mais bits simultaneamente. O parâmetro w define o número de bits assinados em conjunto. Essa melhoria pode ser feita às custas de um aumento no número de avaliações da função f, isto é, no tempo de processamento. A partir de w e n outros parâmetros são definidos:

$$t_1 = \left\lceil \frac{n}{w} \right\rceil, t_2 = \left\lfloor \frac{\lceil log_2 t_1 \rceil + 1 + w}{w} \right\rfloor, t = t_1 + t_2 \quad (1)$$

A chave privada SK é composta por t sequências de bits de tamanho n. A chave pública PK é gerada aplicando-se f $2^w - 1$ vezes à SK. Para a geração da assinatura é calculado o resumo criptográfico da mensagem $f(M) = B = (b_0, b_1, ..., b_{n-1})$. B é então dividido em t_1 sequências de tamanho w tal que $B = b_{t-1}||...||b_{t-t_1}$, onde $||$ denota concatenação.

[2] https://huelsing.files.wordpress.com/2017/06/20170619_pq_summer_school.pdf

A seguir, cada b_i é considerado como um inteiro de valor entre 0 e $2^w - 1$ e um *checksum* C é calculado.

$$C = \sum_{i=t-t_1}^{t-1} (2^w - b_i) \tag{2}$$

Em seguida, C é dividido em t_2 sequências de bits de tamanho w tal que $C = b_{t_2-1}||...||b_0$. Caso o tamanho de B e C não seja divisível por w, zeros são acrescentados no início da sequência. Por fim, a assinatura SIG é calculada utilizando $B||C$:

$$SIG = (f^{b_{t-1}}(sk_{t-1}), ..., f^{b_1}(sk_1), f^{b_0}(sk_0)) \tag{3}$$

A verificação da assinatura é feita calculando-se o $checksum$ novamente conforme descrito anteriormente e comparando o resumo da assinatura calculado abaixo com a chave pública.

$$(f^{2^w-1-b_{t-1}}(sig_{t-1}), ..., f^{2^w-1-b_0}(sig_0)) = (pk_{t-1}, ..., pk_0) \tag{4}$$

2.3. MSS

O esquema MSS utiliza uma árvore binária de *hashes* (árvore de Merkle) para assinar até 2^h mensagens onde h é a altura da árvore. A figura 2 mostra uma árvore que permite a assinatura de até 8 mensagens. São geradas 2^h chaves OTS e o *hash* das chaves públicas OTS $PK_0, ..., PK_{2^h}$ são armazenados nas folhas da árvore. Cada nó acima das folhas é construído gerando-se o *hash* da concatenação de seu filho esquerdo com o direito. A chave pública de um esquema MSS é a raiz da árvore. Portanto, para validar uma assinatura, o verificador deve receber, além da chave pública OTS, o índice da chave utilizada e todos os nós da árvore que permitam o cálculo da raiz. Uma assinatura MSS é válida se a verificação da chave OTS resultar em sucesso e a reconstrução da árvore gerar a mesma raiz recebida.

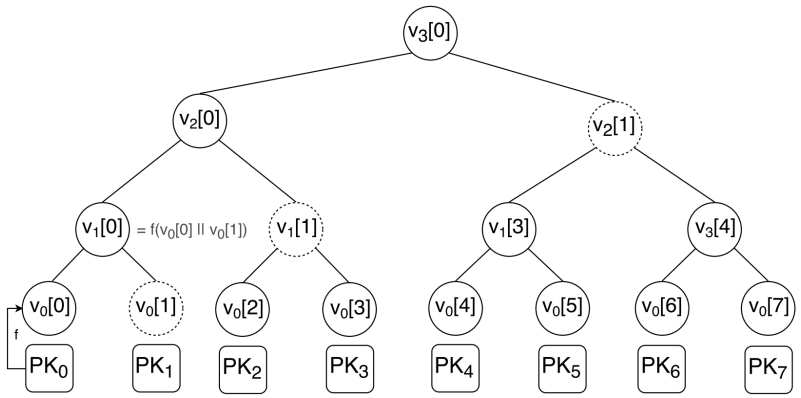

Figura 2. MSS h=3. Os nós de bordas tracejadas são usados no cálculo da raiz em uma assinatura com $v_0[0]$.

3. Implementação

A implementação dos esquemas apresentados (W-OTS e MSS) foi feita em C com a biblioteca criptográfica RELIC[3]. Os esquemas foram implementados seguindo as especificações contidas em [Buchmann et al. 2009] e descritos na seção 2 deste trabalho.

[3] `https://github.com/relic-toolkit`

Para a geração das chaves foi utilizado um pseudo-gerador de números aleatórios implementado pela RELIC. Desta forma, as chaves do esquema MSS não precisariam ser armazenadas, apenas a semente do gerador seria suficiente para gerar as chaves à medida que fossem utilizadas.

A implementação foi feita de maneira a permitir que a função $hash$ utilizada seja trocada facilmente. O trecho abaixo mostra uma função genérica para calcular o $hash$ de uma mensagem.

```
1  void hash_msg(uint8_t out[MD_LEN], uint8_t *msg, size_t size)
2  {
3  #if MD_MAP == SHONE
4    md_map_shone(out,msg,size);
5  #elif MD_MAP == SH224
6    md_map_sh224(out,msg,size);
7  #elif MD_MAP == SH256
8    md_map_sh256(out,msg,size);
9  #elif MD_MAP == SH384
10   md_map_sh384(out,msg,size);
11 #elif MD_MAP == SH512
12   md_map_sh512(out,msg,size);
13 #endif
14 }
```

4. Avaliação e Resultados Experimentais

4.1. Avaliações Analíticas

As avaliações analíticas foram feitas de acordo com as definições em [Buchmann et al. 2009]. A tabela 1 apresenta como se dá o compromisso entre memória e processamento no esquema W-OTS de acordo com a escolha do parâmetro w. O tamanho da chave diminui linearmente enquanto o número de avaliações da função *hash* aumenta exponencialmente. Portanto, é preciso encontrar o compromisso que satisfaça às limitações do dispositivo.

Tabela 1. Compromisso entre tamanho da assinatura e chave e o tempo de processamento

n	w	**Tamanho assinatura/ chave (bytes)**	**Avaliações de f**
256	2	4256	399
	4	2144	1005
	8	1088	8670
	16	576	1179630
512	2	16768	786
	4	8384	1965
	8	4224	16830
	16	2176	2228190

A tabela 2 apresenta a avaliação analítica do esquema MSS considerando $w = 4$. O tamanho da assinatura muda em relação à tabela 1 pois os nós da árvore de Merkle para a validação da chave pública também devem ser considerados.

Tabela 2. Avaliação do esquema MSS para w = 4

h	**n**	**Tam. chave (bytes)**	**Tam. assinatura (bytes)**	**Máximo de assinaturas**
14	256	2144	2,592	16,384
	384	4752	5,424	
	512	8384	9,280	
16	256	2,144	2,656	65,536
	384	4752	5,520	
	512	8384	9,408	
18	256	2,144	2,720	262,144
	384	4752	5,616	
	512	8384	9,536	

4.2. Avaliações Experimentais

As avaliações experimentais foram feitas na plataforma Arduino Due que possui processador ATSAM3X8E 84 MHz, 96kB de SRAM e 512kB de memória Flash. Os experimentos foram executados 100 vezes para cada operação (geração de chaves, assinatura da mensagem e verificação da mensagem). Os gráficos 3(a) e 3(b) apresentam os tempos por operação utilizando as funções SHA256, SHA384 e SHA512 implementadas na biblioteca RELIC. Os valores escolhidos para w foram 2 e 4 pois estes possibilitam um tamanho de chave aceitável para o Arduino Due sem que os tempos de execução tornem-se impraticáveis.

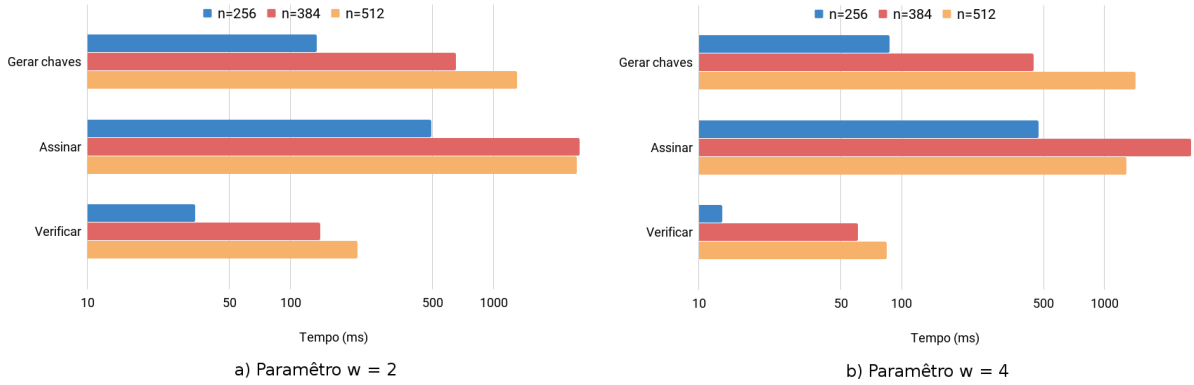

Figura 3. Tempo por operação em W-OTS

O gráfico 4 apresenta o tempo por operação para o MSS usando o W-OTS. O maior custo do MSS é a geração de chaves pois é preciso gerar todas as chaves OTS para calcular a PK. No entanto, para valores de h pequenos é possível gerá-las no próprio dispositivo. As operações de geração e verificação da assinatura têm como a maior parte de seu custo ligados ao esquema OTS. Isso decorre do fato de que gerar e validar o caminho de autenticação é barato pois apenas h nós são utilizados.

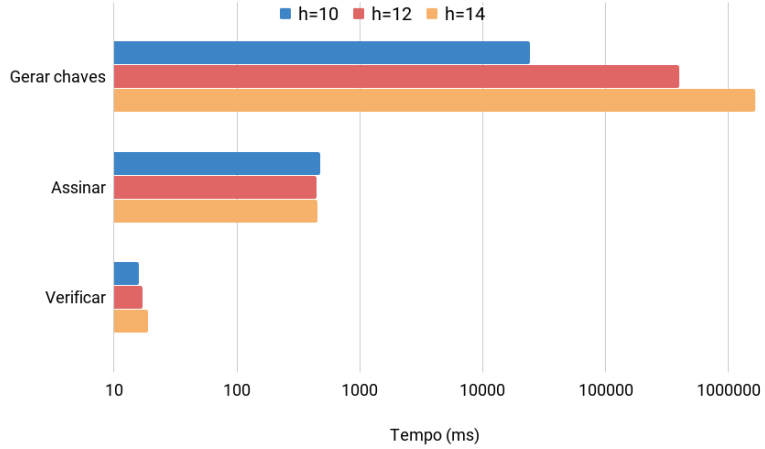

Figura 4. Tempo por operação em MSS

5. Trabalhos Relacionados

O trabalho de [Barreto et al. 2014] apresenta um panorama da criptografia pós-quântica. Exitem trabalhos que implementam esquemas pós-quânticos citados na seção 1 para dispositivos restritos. Em [Maia et al. 2010] é apresentada uma implementação de um criptossistema baseado em sistemas quadráticos multivariados para redes de sensores sem fio. No trabalho de [Biasi et al. 2014] é feita a implementação de um criptossistema baseado em códigos corretores de erro.

Na literatura, existem também trabalhos que implementam e avaliam HBS em dispositivos limitados como *smart cards* com microprocessadores 8-bit [Rohde et al. 2008] e

16-bit [Hülsing et al. 2012], microprocessadores AVR ATxmega [Eisenbarth et al. 2013] e microcontroladores ARM Cortex M3 [Hülsing et al. 2016]. No entanto, ainda existem poucos trabalhos tratando deste assunto.

Em [Rohde et al. 2008], o esquema MSS com o W-OTS foram implementados em *smart cards* com microprocessadores 8-bit da família AT90SCxxx utilizando a cifra de bloco AES. Os autores mostraram que sua implementação para *smart cards* possui código menor se comparado a implementações dos esquemas RSA e ECDSA para a mesma plataforma, além de ser mais eficiente que o RSA e comparável ao ECDSA. No entanto, neste trabalho as chaves não são geradas diretamente no dispositivo devido às suas limitações.

O trabalho de [Hülsing et al. 2012] apresenta uma versão baseada no esquema XMSS à qual dão o nome de XMSS$^+$ e sua implementação para *smart cards* 16-bit Infineon SLE78. Neste trabalho, o problema de geração das chaves no próprio dispositivo foi solucionado através da redução do tempo de geração das chaves em XMSS$^+$.

No trabalho de [Eisenbarth et al. 2013], uma nova abordagem para a computação do caminho de autenticação para o MSS foi sugerida para evitar aumentar a resiliência ao vazamento das chaves sob o custo de maior armazenamento necessário. A implementação desta nova abordagem para o MSS foi feita para microprocessadores AVR ATxmega.

Em [Hülsing et al. 2016], os autores implementaram o esquema SPHINCS em um microcontrolador ARM Cortex M3. Este esquema não armazena estados, isto é, ao contrário do esquema avaliado neste trabalho, o SPHINCS não lembrar-se da última chave utilizada. A desvantagem de esquemas *stateless* é que eles possuem chaves significantemente maiores que os esquemas *stateful*. No entanto, os autores demonstraram ser possível implementar um esquema *stateless* em um dispositivo limitado.

6. Conclusão

Este trabalho mostrou a aplicabilidade das HBS em dispositivos IoT como uma alternativa aos esquemas tradicionais no período pós-quântico. As HBS devem ser consideradas como possíveis substitutas do RSA, DSA, ECDSA e outros. O bom entendimento do nível de segurança e facilidade de implementação das HBS tornam esses criptossistemas vantajosos em relação a outros esquemas pós-quânticos.

Para trabalhos futuros, serão implementados outros esquemas HBS para avaliação e comparação com os já implementados. Além disso, outras métricas podem ser utilizadas nas avaliações tais como consumo de energia, tamanho de código e número de ciclos.

Agradecimentos

Esse trabalho foi parcialmente financiado pelo CNPq, FAPEMIG, CAPES e RNP.

Referências

Aranha, D. F. and Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

Barreto, P. S., Biasi, F. P., Dahab, R., López-Hernández, J. C., de Morais, E. M., de Oliveira, A. D. S., Pereira, G. C., and Ricardini, J. E. (2014). A panorama of post-quantum

cryptography. In *Open Problems in Mathematics and Computational Science*, pages 387–439. Springer.

Biasi, F. P., Barreto, P. S., Misoczki, R., and Ruggiero, W. V. (2014). Scaling efficient code-based cryptosystems for embedded platforms. *Journal of Cryptographic Engineering*, 4(2):123–134.

Buchmann, J., Dahmen, E., and Szydlo, M. (2009). Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–93. Springer.

Buchmann, J. A., Dahmen, E., Ereth, S., Hülsing, A., and Rückert, M. (2011). On the security of the winternitz one-time signature scheme. *Africacrypt*, 11:363–378.

Dods, C., Smart, N. P., and Stam, M. (2005). Hash based digital signature schemes. In *IMA International Conference on Cryptography and Coding*, pages 96–115. Springer.

Eisenbarth, T., Von Maurich, I., and Ye, X. (2013). Faster hash-based signatures with bounded leakage. In *Selected Areas in Cryptography*, pages 223–243.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

Hülsing, A., Busold, C., and Buchmann, J. A. (2012). Forward secure signatures on smart cards. In *Selected Areas in Cryptography*, volume 7707, pages 66–80. Springer.

Hülsing, A., Gazdag, S.-L., Butin, D., and Buchmann, J. (2015). Hash-based signatures: An outline for a new standard.

Hülsing, A., Rijneveld, J., and Schwabe, P. (2016). Armed sphincs: Computing a 41 kb signature in 16 kb of ram.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63.

Maia, R. J. M., Barreto, P. S. L. M., and de Oliveira, B. T. (2010). Implementation of multivariate quadratic quasigroup for wireless sensor network. In *Transactions on computational science XI*, pages 64–78. Springer.

Meier, A. V. (2005). The elgamal cryptosystem.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Merkle, R. C. (1989). A certified digital signature. In *Proceedings on Advances in cryptology*, pages 218–238. Springer-Verlag New York, Inc.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Rohde, S., Eisenbarth, T., Dahmen, E., Buchmann, J., and Paar, C. (2008). Fast hash-based signatures on constrained devices. In *International Conference on Smart Card Research and Advanced Applications*, pages 104–117. Springer.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE.

