Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cifração e Autenticação utilizando Funções Fisicamente não Clonáveis (PUFs)

Amanda Cristina Davi Resende

Dissertação apresentada como requisito parcial
para conclusão do Programa de Pós-graduação em Informática

Orientador
Prof. Dr. Diego de Freitas Aranha

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-graduação em Informática

Coordenadora do programa: Prof.ª Dr.ª Alba Cristina M. A. de Melo

Banca examinadora composta por:

Prof. Dr. Diego de Freitas Aranha (Orientador) — CIC/UnB
Prof. Dr. Mauricio Ayala Rincón — CIC/UnB
Prof. Dr. Jeroen van de Graaf — DCC/UFMG

CIP — Catalogação Internacional na Publicação

Resende, Amanda Cristina Davi.

Cifração e Autenticação utilizando Funções Fisicamente não Clonáveis (PUFs) / Amanda Cristina Davi Resende. Brasília : UnB, 2014.
80 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. PUFs, 2. PAKE, 3. protocolo, 4. Luby-Rackoff, 5. autenticação, 6. cifração, 7. segurança

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Cifração e Autenticação utilizando Funções Fisicamente não Clonáveis (PUFs)

Amanda Cristina Davi Resende

Dissertação apresentada como requisito parcial
para conclusão do Programa de Pós-graduação em Informática

Prof. Dr. Diego de Freitas Aranha (Orientador)
CIC/UnB

Prof. Dr. Mauricio Ayala Rincón
CIC/UnB

Prof. Dr. Jeroen van de Graaf
DCC/UFMG

Prof.ª Dr.ª Alba Cristina M. A. de Melo
Coordenadora do programa do Programa de Pós-graduação em Informática

Brasília, 28 de Novembro de 2014

Agradecimentos

Em primeiro lugar, gostaria de agradecer a Deus, por ter me dado força para superar todos os obstáculos que surgiram, e por ter colocado pessoas tão especiais no meu caminho.

Em segundo, a minha imensa gratidão a minha família, especialmente meu pai José Pedro de Resende Filho, minha vó/mãe Eunice Maria de Jesus, minha dindinha Silvia Maria Correia Lima e minha madrinha Silvânia Davi Dias de Oliveira, que sempre me incentivaram e confiaram em mim para que eu pudesse concretizar esse trabalho.

Ao meu orientador, Diego de Freitas Aranha por ter me apresentado o tema, assim como pelo tempo disponibilizado para me auxiliar na realização desse trabalho. Agradeço também os professores, desde os da alfabetização até os da pós-graduação, que de uma forma ou outra, me proporcionaram vários conhecimentos e lições. Em especial aos professores da UFG, Liliane do Nascimento Vale e Vaston Gonçalves da Costa, pela ajuda e suporte na escrita deste trabalho.

A todos os meus amigos/amigas, que direta ou indiretamente contribuíram para a elaboração deste trabalho. Em especial às novas amizades feita neste período, dentre elas Paula Letícia, Jonathan Alis, Lucas Emmanoel e a Ariane Alves, que é uma amizade mais antiga, por sempre me ajudarem dentro ou fora do meio acadêmico.

À Intel, que financia o projeto intitulado Funções Fisicamente não Clonáveis Aplicadas a Sistemas Embarcados em *Chips* – 2012/05156, ao qual faz parte este trabalho. À Capes e a UnB, por todo apoio fornecido.

E a todos, que de alguma forma, contribuíram para que esse fosse concretizado.

Resumo

Este trabalho apresenta aplicações baseadas em uma recente primitiva intitulada Funções Fisicamente não Clonáveis (PUFs), e como elas podem ser utilizadas para estabelecer serviços de segurança. No trabalho, PUFs são empregadas para construir uma cifra de blocos e um autenticador, onde a primeira é construída a partir de uma cifra Luby-Rackoff com 4 rodadas envolvendo PUFs e funções de *hash* universal. A cifra aprimora o estado da arte de cifração baseada em PUFs tanto em segurança quanto no tamanho do criptograma resultante. Já no segundo, é construído um Código de Autenticação de Mensagem (MAC) pela combinação de um MAC clássico de tamanho fixo com uma função de *hash* universal. Em ambos os casos, análises de segurança são fornecidas considerando noções padronizadas na literatura. Como as PUFs codificam chaves criptográficas implícitas, as técnicas apresentadas podem ser empregadas em esquemas de cifração autenticada de discos rígidos ou dispositivos móveis, com incremento de segurança por resistência ao vazamento de *bits* da chave. Além da proposta de construção de uma cifra e um autenticador, é proposto um protocolo de autenticação para aplicações bancárias combinando PUFs com protocolos para acordo autenticado de chave baseado em senha (PAKE). O protocolo resultante fornece autenticação mútua entre cliente e servidor e estabelecimento de uma chave de sessão entre as partes autenticadas, importantes características que não foram encontradas simultaneamente na literatura de autenticação baseada em PUFs. A combinação aprimora o estado da arte, garantindo que a chave de sessão estará disponível apenas para detentores legítimos da PUF, reduzindo a possibilidade de vazamento de segredos armazenados explicitamente. O protocolo suporta autenticação multi-fator e fornece proteção contra ataque de dicionário *offline* sobre a senha de autenticação. Além disso, ele satisfaz noções usuais de segurança quando a saída da PUF é imprevisível, e permite o cliente notificar o servidor em caso de emergência.

Palavras-chave: PUFs, PAKE, protocolo, Luby-Rackoff, autenticação, cifração, segurança

Abstract

This work presents applications based on a recent primitive called Physically Unclonable Functions (PUFs), and how they can be used to establish security properties. In this work, PUFs are employed to construct a block cipher and authenticator, where the first is constructed from a Luby-Rackoff cipher with 4 rounds involving PUFs and universal hash functions. The cipher improves the state of the art of PUF-based encryption in two aspects: security and size of the resulting ciphertext. The second, a Message Authentication Code (MAC) is built by the combination of a classic fixed-size MAC with a universal hash function. In both cases, security analysis are provided considering standard notions in the literature. Since the PUFs implicitly encode cryptographic keys, the techniques presented can be used in authenticated encryption schemes of hard drives or mobile devices, increasing resistance against leakage of key bits. Besides the proposed construction of a cipher and an authenticator, we propose an authentication protocol for banking applications combining PUFs with protocols for Password-based Authenticated Key Exchange (PAKE). The resulting protocol provides mutual authentication between client and server and establishes a session key between the authenticated parties, important features that were not found simultaneously in the literature of PUF-based authentication. This combination improves the state of the art, ensuring that the session key is only available to legitimate holders of the PUF, reducing the possibility of leaking secrets stored explicitly. The protocol supports multiple authentication factors and provides protection against offline dictionary attacks on the password authentication. Moreover, it satisfies the usual security notions when the PUF output is unpredictable, and allows the client to notify the server in case of emergency.

Keywords: PUFs, PAKE, protocol, Luby-Rackoff, authentication, encryption, security

Sumário

Lista de Figuras

Lista de Siglas

AES *Advanced Encryption Standard*, 8

CCD *Charge-Coupled Device*, 23

DES *Data Encryption Standard*, 8

DH-EKE *Diffie-Hellman Encrypted Key Exchange*, 18

FPGAs *Field Programmable Gate Arrays*, 29

FTG *Find then Guess*, 10

IND-CCA *Indistinguishability under Chosen Ciphertext Attack*, 10

IND-CCA1 *Indistinguishability under non-adaptive Chosen Ciphertext Attack*, 4

IND-CCA2 *Indistinguishability under adaptive Chosen Ciphertext Attack*, 10

IND-CPA *Indistinguishability under Chosen Plaintext Attack*, 4

IP *Internet Protocol*, 29

LOR *Left or Right*, 10

MAC *Message Authentication Code*, 1

OEKE *One-Encryption Key Exchange*, 47

PAKE *Password-based Authenticated Key Exchange*, 2

PKI *Public Key Infrastructure*, 9

POWF *Physical One-Way Function*, 21
PRF *Pseudorandom Function*, 16
PUFs *Physically Unclonable Function*, 2

RAM *Random Access Memory*, 2
RO-PUF *Ring Oscillator PUF*, 24
ROM *Read-Only Memory*, 2
ROR *Real or Random*, 10

SEM *Segurança semântica*, 10
SPEKE *Simple Password Exponential Key Exchange*, 18
SRAM *Static Random Access Memory*, 25

wPRF *weak Pseudorandom Functions*, 16

Capítulo 1

Introdução

Atualmente, a preocupação com a segurança da informação se apoia em tecnologias e ferramentas que podem ser utilizadas para garantir propriedades de segurança, como disponibilidade, integridade, autenticidade e a confidencialidade da informação que trafega pela rede [75]. Para que essas informações sejam tratadas de forma rápida e segura, existem diversas abordagens que podem ser utilizadas, dentre elas, o emprego de protocolos criptográficos de comunicação que visam fornecer garantias de segurança.

Várias tecnologias e serviços se tornam viáveis graças à utilização destes protocolos, como por exemplo, operações governamentais, transações bancárias, comunicações militares e comércio eletrônico. Eles também podem ser utilizados na construção de mecanismos de identificação (protocolos de autenticação), para que uma ou ambas as partes de uma comunicação consigam provar a sua identidade para a outra.

Esses mecanismos são muito importantes para soluções bancárias *online* (de *e-banking*), pois permitem verificar se o serviço está sendo acessado por um dispositivo autorizado pelo usuário, bem como permite ao usuário atestar que os dados serão acessados apenas pelo banco. Sendo assim, é importante permitir que as partes comunicantes sejam capazes de detectar a alteração maliciosa de mensagens em trânsito. Assim, a autenticação é utilizada para verificar a identidade dos usuários e fornecer um canal seguro de comunicação.

Para que tais protocolos sejam utilizados de forma segura, pode-se empregar construções baseadas em criptografia de chave simétrica, onde a chave de decifração pode ser facilmente deduzida, em tempo polinomial, a partir da chave de cifração (podendo ser a mesma) [54] e criptografia de chave assimétrica, na qual a chave de decifração não pode ser facilmente deduzida em tempo polinomial da chave de cifração[1] [54].

A criptografia simétrica é rápida, porém a gerência e distribuição de chave é complexa. Já a criptografia assimétrica é lenta (já que depende de problemas complexos da Teoria

[1] A chave de cifração é dita chave pública e a chave de decifração é dita chave privada.

dos Números), mas a gerência e distribuição de chave é mais simples. Devido as vantagens e desvantagens de ambos, o mais comum é empregar uma construção híbrida. Assim, a criptografia simétrica é utilizada para o processo de cifração da informação, pois é mais rápida (já que não depende de problemas da Teoria dos Números) que a assimétrica, e a criptografia assimétrica é utilizada para o compartilhamento da chave secreta, que é necessária no processo simétrico de cifração. Um exemplo de construção híbrida pode ser visto em [56], onde é proposta uma construção baseada no problema de Diffie-Hellman [22] combinado com cifração simétrica, código autenticador de mensagem (do inglês, *Message Authentication Code* – MAC) e funções de *hash*.

Tanto em outras abordagens quanto no modelo híbrido, protocolos de autenticação dependem, na maioria da vezes, do conhecimento de uma longa chave criptográfica de posse exclusiva dos detentores legítimos. Por essa razão, é comum o emprego de um dispositivo para o armazenamento seguro dessa chave, uma vez que a memória humana, na maioria das vezes, é incapaz de memorizá-la sem erros. Protocolos para acordo autenticado de chaves baseado em senha (do inglês, *Password-based Authenticated Key Exchange*–PAKE) [6, 7, 15, 39, 42] relaxam esse requisito, na medida que exigem apenas o conhecimento de uma chave muito mais curta (senha), não sendo necessário nenhum dispositivo extra para o seu armazenamento, pois é possível memorizá-la sem erros.

No entanto, mesmo utilizando protocolos PAKEs, para o caso de protocolos de autenticação, ou em outras aplicações criptográficas, como por exemplo cifras de blocos, a chave é armazenada de maneira explícita. Esse armazenamento pode acontecer tanto em memória não-volátil (memória somente de leitura, do inglês (*Read-Only Memory* – ROM), memória flash, entre outras) quanto em memória volátil (na memória de acesso aleatório (do inglês, *Random Access Memory* – RAM)).

Em ambos os casos, há clara dificuldade em se armazenar chaves de forma segura, uma vez que, adversários com acesso físico ao equipamento, podem montar ataques de canal lateral [44] para recuperar *bits* da chave a partir do estado latente da memória [35], ou pelo monitoramento de características físicas do sistema [45]. Desta forma, os *bits* restantes podem ser recuperados por ataques de busca exaustiva de menor complexidade [9] ou por heurísticas mais avançadas [14, 37, 62].

Assim, para resolver o problema do armazenamento da chave explicitamente, podemos empregar funções fisicamente não clonáveis (*Physically Unclonable Functions* – PUFs), introduzidas em 2002 por Gassend *et al.* e Pappu *et al.* [30, 60], como uma nova primitiva criptográfica, graças às suas propriedades (não clonabilidade, saída imprevisível, entre outras), uma vez que a chave está armazenada implicitamente em sua estrutura, através da correspondência entre desafios (entradas) e respostas (saídas), determinados pelo seu processo de fabricação.

Conforme serão apresentados nos Capítulos 4 e 5, PUFs podem ser empregadas para a cifração de disco ou dispositivos móveis, bem como combinadas com protocolos PAKE, respectivamente. PUFs incorporadas em cifras de bloco, tem como grande vantagem o não armazenamento da chave na memória, além de que combinada com uma cifra de Luby Rackoff atinge uma noção de segurança maior que o estado da arte. O cenário ideal para uma cifra baseada em PUFs é a cifração de disco ou dispositivos móveis, pois em uma aplicação convencional (duas ou mais partes comunicantes) exigiria a transmissão da PUF por um meio físico.

Já PUFs combinadas com protocolos PAKEs fornecem um protocolo para transações bancárias, que reúne algumas características úteis que não foram encontradas simultaneamente em outros protocolos. Além disso, aprimora o estado da arte, garantindo que a chave de sessão estará disponível apenas para detentores legítimos da PUF. Ademais, o protocolo também pode ser facilmente adaptado para suportar senha de pânico, que permite o cliente notificar ao servidor em caso de emergência.

1.1 Objetivos

O objetivo geral deste trabalho é a construção de protocolos criptográficos eficientes baseados em PUFs, com noções e reduções formais de segurança. Para alcançar esse objetivo tem-se os seguintes objetivos específicos:

- Construção de uma cifra simétrica autenticada segura nas noções de segurança padronizadas na literatura que envolvem a cifra de bloco com PUFs proposta em [2] e a estrutura de uma cifra de Luby-Rackoff com 4 funções de rodada proposta em [61, 65]. A cifra é ideal para aplicações que exijam garantias de sigilo e autenticidade de discos ou dispositivos móveis.
- Construção de um MAC baseado em PUFs, para poder detectar qualquer alteração maliciosa no texto claro resultante da decifração ou corrupção do conteúdo armazenado sem possibilidade de detecção posterior, garantindo assim a integridade da mensagem.
- Construção de um protocolo seguro combinando PUFs com protocolos PAKE que satisfaçam noções de segurança criptográfica padronizadas, como por exemplo a semântica (explicada na Seção 2.7.1), para soluções onde os clientes já estão acostumados a ter um dispositivo de autenticação adicional, de modo que os dispositivos atuais poderiam ser incrementados com uma PUF para fornecer autenticação com múltiplos fatores com base em hipóteses computacionais e físicas.

- Adaptar o protocolo proposto no item acima para que ele possa ter suporte a senha de pânico para situações de emergência, onde o cliente é coagido a entregar a PUF e revelar a sua senha.

1.2 Contribuições do trabalho

As contribuições deste trabalho são as seguintes:

1 Uma cifra de bloco construída combinando uma cifra de Luby-Rackoff com 4 rodadas (que envolve funções de *hash* universal) com PUFs modeladas como funções pseudoaleatórias (fracas);

2 Análise formal de segurança da cifra proposta, considerando noções de segurança padronizadas que garantem tanto indistinguibilidade contra ataques de texto claro escolhido (do inglês, *Indistinguishability under Chosen Plaintext Attack* – IND-CPA) quanto a indistinguibilidade contra ataques não-adaptativos de criptograma escolhido (do inglês *Indistinguishability under non-adaptive Chosen Ciphertext Attack* – IND-CCA1);

3 Um MAC construído pela combinação de um MAC clássico de tamanho fixo com uma função de *hash* universal, com sua análise de segurança para garantir um MAC inforjável contra ataques de mensagem escolhida;

4 Um protocolo combinando PUF e PAKE para autenticação mútua com múltiplos fatores e estabelecimento de uma chave de sessão entre as partes autenticadas, com análises heurística e formal de segurança, considerando cenários realistas e noções de segurança padronizadas para acordo autenticado de chaves;

5 Uma adaptação do protocolo proposto, que permite notificação do servidor em caso de emergência;

6 Um ataque de personificação do servidor contra o protocolo apresentado em [18] demonstrando que a correção sugerida pelos autores para o protocolo proposto em [78] não é suficiente;

7 Um ataque de dicionário contra o protocolo [28], que é o estado da arte de soluções de autenticação baseadas em PUFs, que depende apenas da posse temporária da PUF e observação de um único traço de comunicação cliente/servidor;

As contribuições 1, 2 e 3 foram publicadas e apresentadas no XIII Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais – SBSeg 2013 na forma de

um artigo intitulado "Cifração autenticada utilizando PUFs". Já a contribuição 4 foi apresentada no *Third International Conference on Cryptology and Information Security in Latin America – Latincrypt* 2014 como um resumo estendido intitulado *"PUF+PAKE = Mutual multifactor authentication for secure banking"*.

1.3 Estrutura do trabalho

No Capítulo 2, serão apresentadas as notações e terminologias empregadas, bem como as definições necessárias para a compreensão do trabalho. No Capítulo 3 será apresentada a principal primitiva criptográfica utilizada para a elaboração deste trabalho, as PUFs, exibindo propriedades, tipos e modelagens.

Nos Capítulos 4 e 5 serão expostas as contribuições deste trabalho. No Capítulo 4 serão apresentadas uma proposta de cifra de bloco baseada em PUFs bem como a proposta de um autenticador que também utiliza PUFs. Já no Capítulo 5 será apresentada a proposta de um protocolo seguro combinando PUFs com protocolos PAKE. Também é mostrada a proposta de uma adaptação para o protocolo de forma que ele possa ter suporte a senha de pânico para situações de emergência. E por fim, no Capítulo 6, serão apresentadas as conclusões do trabalho bem como os possíveis trabalhos futuros.

Capítulo 2

Definições preliminares

Neste Capítulo serão descritas as notações utilizadas no decorrer do trabalho, assim como as definições relevantes. Nas Seções 2.1 e 2.2 serão definidas as notações básicas utilizadas bem como uma revisão sucinta de criptografia simétrica e criptografia assimétrica. Na Seção 2.3 serão apresentadas as noções de segurança que foram utilizadas. Nas Seções 2.4 e 2.5 serão mostradas as definições de funções de *hash* universal e funções pseudoaleatórias (fracas). Na Seção 2.6 serão apresentadas as funções ruidosas e como esses ruídos podem ser corrigidos para utilizar tais funções em aplicações criptográficas. E por fim, na Seção 2.7 serão definidos protocolos PAKE bem como suas noções de segurança.

2.1 Notação e terminologia

A seguir serão apresentadas as notações utilizadas no trabalho.

- **Texto claro** (do inglês, *plaintext*) é o texto (mensagem) original, inteligível.
- **Texto cifrado ou criptograma** (do inglês, *ciphertext*) é o texto resultante do processo de cifração aplicado por algum sistema criptográfico.
- **Chave criptográfica** é uma cadeia de *bits* que funciona junto com o algoritmo criptográfico para prover sigilo e autenticação, dentre outras propriedades de segurança.
- **Cifração** é o processo de converter o texto claro em um texto cifrado, dada uma chave criptográfica.
- **Decifração** é o processo reverso da cifração, onde se restaura o texto claro a partir do cifrado e da chave.
- O conjunto $\mathcal{K}$ denota o espaço de chaves criptográficas.

- O conjunto $\mathcal{M}$ denota o espaço de mensagens.
- ε é um algoritmo/função de cifração, onde $\varepsilon_k(m)$, com $k \in \mathcal{K}$ e $m \in \mathcal{M}$, denota a cifração de um texto claro m usando a chave k.
- Δ é um algoritmo/função de decifração, onde $\Delta_k(m)$, com $k \in \mathcal{K}$ e $m \in \mathcal{M}$, denota a decifração de um texto cifrado m usando a chave k.
- $||$ denota a concatenação de cadeias de caracteres.
- Para uma cadeia de caracteres x, $|x|$ denota seu comprimento.
- Para um conjunto $\mathcal{S}$, a notação $x \xleftarrow{R} \mathcal{S}$ indica a seleção de um elemento x de um conjunto $\mathcal{S}$ com distribuição uniforme de probabilidade (amostragem aleatória).
- $x \in \{0,1\}^n$ denota uma cadeia de *bits* x de comprimento n.
- A operação $a \stackrel{?}{=} b$ denota a comparação se a é ou não igual a b.
- **Oráculos aleatórios** são argumentos heurísticos que fornecem segurança para protocolos criptográficos modelando certos algoritmos, normalmente funções de *hash* criptográficas, como oráculos (teoricamente uma caixa preta), neste sentido chamados oráculos de *hash*. Os oráculos respondem cada pergunta com uma resposta verdadeiramente aleatória escolhida uniformemente no intervalo de saída (funções aleatórias perfeitas).
- Uma função f é uma **função desprezível** (do inglês, *negligible function*) se para todo polinômio $p(\cdot)$, existe $N > 0 \in \mathbb{Z}$ tal que para todo $n > N$, $|f(n)| < \frac{1}{p(n)}$, onde $|f(n)|$ é o módulo de $f(n)$.
- O **nível de segurança** n de uma primitiva criptográfica, indica que o melhor algoritmo conhecido para atacá-la exige complexidade 2^n.
- **Criptograma de desafio** é o criptograma utilizado para formalizar noções de segurança criptográficas, de modo a desafiar o adversário a responder uma pergunta, que pode variar em cada formalização, relacionada a este criptograma.
- **Ataque de dicionário** é um mecanismo de ataque criptográfico, normalmente utilizado para palpites de senhas, que emprega um dicionário regular que contêm palavras populares (por exemplo, data de nascimento e apelido).
- **Chave de sessão** é uma chave utilizada apenas uma vez para cifrar e decifrar todas as mensagens em uma sessão de comunicação entre duas ou mais partes.

2.2 Criptografia simétrica e criptografia assimétrica

Os sistemas criptográficos podem ser classificados quando se diz respeito à chave em dois tipos: criptografia simétrica e criptografia assimétrica. A criptografia é simétrica (de chave secreta ou convencional) se a chave usada para decifrar é facilmente deduzida, em tempo polinomial, da chave usada para cifrar e vice-versa [54]. Já a criptografia é dita assimétrica ou de chave pública, quando a chave usada para decifrar (chave privada) não pode ser facilmente deduzida, em tempo polinomial, a partir da chave usada para cifrar (chave pública) [54].

2.2.1 Criptografia convencional, simétrica ou de chave secreta

Na criptografia simétrica (Figura 2.1), a chave empregada deve ser mantida em segredo, pois quem conhece a chave de cifração consegue decifrar a mensagem, de forma que seja conhecida apenas pelas partes comunicantes. Essa chave tem que ser compartilhada através de um canal seguro para evitar que uma pessoa não autorizada a capture.

Os algoritmos na criptografia simétrica podem ser divididos em dois tipos: cifra de fluxo (ou contínua), onde a cifração é realizada *bit* a *bit* da mensagem e cifra de bloco, onde a mensagem é dividida em blocos de tamanhos que são aceitos pelas cifras. São exemplo de cifras de fluxo a HC-128 [81], Rabbit [13], Salsa20 [10] e Sosemanuk [8] que fazem parte do The eSTREAM Project [68]. Já as cifras de blocos mais conhecidas são o *Data Encryption Standard* (DES) [27] e o *Advanced Encryption Standard* (AES) [21], que é a cifra adotada como padrão de criptografia pelo governo dos Estados Unidos em 2002 que substituiu o, até então padrão, DES.

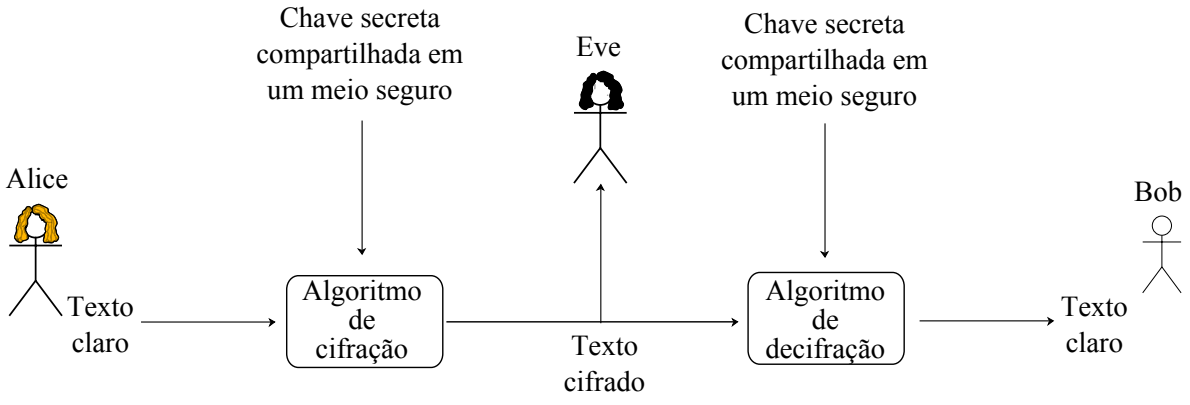


Figura 2.1: Criptografia simétrica.

2.2.2 Criptografia assimétrica ou de chave pública

Na criptografia assimétrica (Figura 2.2), a chave utilizada para cifrar é chamada de chave pública e a chave para decifrar é a chave privada. A chave pública do destinatário é distribuída livremente para qualquer pessoa, enquanto a chave privada é mantida em segredo.

Os algoritmos na criptografia assimétrica são normalmente construídos a partir da dificuldade computacional de problemas da Teoria dos Números. O RSA [67] e o Rabin [64], por exemplo, são algoritmos baseados no problema da fatoração de inteiros, enquanto o ElGamal [25] é baseado no problema do logaritmo discreto e no problema de Diffie-Hellman [22].

Um problema em criptografia assimétrica é garantir que a chave pública de uma pessoa é realmente dela e não de outra pessoa. Por exemplo, Alice quer receber mensagens cifradas e então divulga sua chave pública. Qualquer pessoa que possua a chave pública de Alice pode lhe enviar uma mensagem, entretanto, outra pessoa, por exemplo Eve, pode divulgar uma chave pública (que ela conheça a respectiva chave privada) alegando ser de Alice. Assim, quando alguém enviar uma mensagem criptografada pensando ser para Alice (utilizando a chave pública que foi divulgada por Eve), Eve poderá decifrá-la, pois a mensagem havia sido cifrada com a chave pública de Eve, e desse modo ela possui a chave privada correspondente.

No entanto, se Alice possuir um certificado de sua chave púbica através de uma entidade confiável, qualquer pessoa que confia na entidade também pode confiar em Alice e assim sucessivamente. Este mecanismo é chamado de infraestrutura de chave pública (do inglês, *Public Key Infrastructure* – PKI).

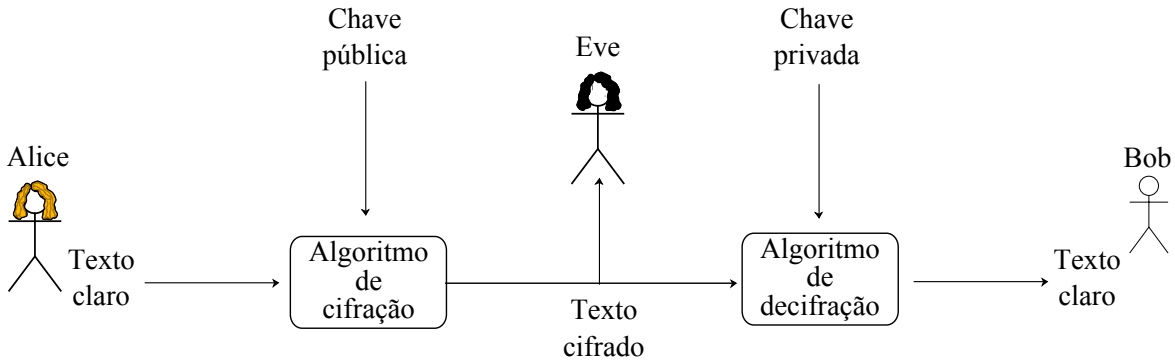


Figura 2.2: Criptografia assimétrica.

2.2.3 Modelo híbrido

A grande limitação da criptografia simétrica é o compartilhamento da chave através de um canal seguro, o que não se pode garantir sempre, uma vez que ataques podem

ser efetuados. Assim, a chave poderá ser descoberta e outras pessoas que não sejam o destinatário podem decifrar a mensagem. Esse problema é conhecido como problema de distribuição de chaves.

Para evitar esse problema, pode-se utilizar criptografia de chave assimétrica. Assim, apenas o destinatário correspondente terá a posse da chave, sem precisar de nenhum tipo de compartilhamento. Entretanto, a criptografia simétrica é computacionalmente mais simples, e consequentemente mais rápida do que a assimétrica, pois não depende de problemas da Teoria dos Números e, assim, pode ser projetada a partir de operações bem mais simples. Por isso, geralmente constrói-se um modelo híbrido utilizando as vantagens de ambas, onde se utiliza a criptografia assimétrica para a distribuição de chave e a criptografia simétrica para a cifração e decifração da mensagem.

2.3 Noções de segurança

As principais noções de segurança utilizadas na literatura são a IND-CPA e a indistinguibilidade contra ataques de criptograma escolhido (do inglês, *Indistinguishability under Chosen Ciphertext Attack* – IND-CCA). Existem outras noções de segurança, tais como: esquerda ou direita (do inglês, *Left or Right* – LOR), encontra então adivinha (do inglês, *Find then Guess* – FTG) e segurança semântica (do inglês, *semantic security* – SEM) [4].

Essas noções podem ser equivalentes ou implicarem umas nas outras. Por exemplo, sabe-se que segurança semântica e indistinguibilidade são noções equivalentes [32]. Neste trabalho, se utiliza a noção de segurança real ou aleatória (do inglês, *Real or Random* – ROR) (Definição 2.3.2), por sua conveniência na análise de construções de cifras de bloco e equivalência com a noção de indistinguibilidade que será definida na Seção a seguir.

2.3.1 Indistinguibilidade

A noção de indistinguibilidade contra ataques de texto claro escolhido (Definição 2.3.1) reflete a dificuldade computacional de um adversário em diferenciar a cifração de duas mensagens de sua escolha, mesmo que ele tenha acesso a um oráculo de cifração para cifrar mensagens arbitrárias.

Na definição para ataques de criptograma escolhido, IND-CCA, o adversário tenta realizar a mesma tarefa, mas agora com o incremento do acesso a um oráculo de decifração que decifra criptogramas também arbitrários. Na versão não adaptativa, IND-CCA1, o adversário pode fazer consultas aos oráculos somente até que receba o criptograma de desafio. Já na definição adaptativa, indistinguibilidade contra ataques adaptativos de criptograma escolhido (do inglês, *Indistinguishability under adaptive Chosen Ciphertext Attack* – IND-CCA2), o adversário pode continuar a fazer consultas aos oráculos mesmo

após ter recebido o criptograma de desafio, com a restrição de não poder consultar o oráculo de decifração com o criptograma de desafio pois, caso contrário, a tarefa seria trivial.

Definição 2.3.1. Indistinguibilidade sob ataques de texto claro escolhido – IND-CPA. [41, Adaptada da Definição 3.21].

Sejam $\mathcal{A}$ um adversário com poder computacional polinomial, $\Gamma = (\varepsilon, \Delta)$ uma cifra simétrica e $Privk^{cpa}_{\mathcal{A},\Gamma}(n)$ a execução de um experimento com $\mathcal{A}$ no nível de segurança n:

- A chave k é gerada fazendo-se $k \xleftarrow{R} \mathcal{K}$;
- $\mathcal{A}$ recebe o oráculo de cifração $\varepsilon_k(\cdot)$ e produz m_0, $m_1 \in \mathcal{M}$ com $|m_0| = |m_1|$;
- Um *bit* $b \xleftarrow{R} \{0, 1\}$ é escolhido, e então o criptograma de desafio $c = \varepsilon_k(m_b)$ é calculado e entregue a $\mathcal{A}$;
- O adversário $\mathcal{A}$ continua a ter acesso ao oráculo $\varepsilon_k(\cdot)$ e produz o *bit* b';
- A saída do experimento é 1 se $b' = b$ e 0 caso contrário. $\mathcal{A}$ tem sucesso quando $Privk^{cpa}_{\mathcal{A},\Gamma}(n) = 1$.

Uma cifra Γ é indistinguível sob ataques de texto claro escolhido se para todo adversário $\mathcal{A}$, existe função λ desprezível tal que:

$$\Pr[Privk^{cpa}_{\mathcal{A},\Gamma}(n) = 1] \leq \frac{1}{2} + \lambda(n).$$

Essa definição pode ser facilmente adaptada para ataques de criptograma escolhido (IND-CCA), acrescentando o acesso ao oráculo de decifração por parte do adversário e restrições correspondentes.

2.3.2 Real ou Aleatório

Na noção de segurança ROR (Definição 2.3.2), o adversário não pode diferenciar a cifração de uma mensagem de sua escolha da cifração de uma mensagem escolhida aleatoriamente com vantagem não desprezível. Deve-se considerar dois diferentes jogos. No jogo real (*real*), inicia-se escolhendo uma chave aleatória $a \xleftarrow{R} \mathcal{K}$. Então é fornecido para o adversário $\mathcal{A}$ um oráculo que, quando consultado com uma cadeia de caracteres $x \in \mathcal{M}$, responde com uma cifração de x sob a chave a.

Já no jogo aleatório (*random*), inicia-se escolhendo uma chave aleatória $a \xleftarrow{R} \mathcal{K}$, como no jogo real. Então é fornecido para o adversário $\mathcal{A}$ um oráculo que quando consultado

com $x \in \mathcal{M}$, responde com uma cifração, sob a chave a, de uma cadeia de caracteres aleatória de tamanho $|x|$.

Um sistema de cifração é considerado seguro para essa noção de segurança se nenhum adversário pode obter vantagem significativa, ou seja, não desprezível, em distinguir entre o jogo real e aleatório com complexidade polinomial no parâmetro de segurança n.

Definição 2.3.2. *Real ou Aleatório* (ROR). [4, Adaptada da Definição 1].

Sejam $\mathcal{A}$ um adversário com poder computacional polinomial, $\Gamma = (\varepsilon, \Delta, \mathcal{K})$ uma cifra simétrica e $Exp^{ror-atk}_{\mathcal{A},\Gamma}(n)$ a execução de um experimento com $\mathcal{A}$ no nível de segurança n, onde atk é a noção de segurança IND-CPA ou IND-CCA:

- A chave k é gerada fazendo-se $k \xleftarrow{R} \mathcal{K}$.
- Um *bit* $b \xleftarrow{R} \{0, 1\}$ é escolhido.
- $\mathcal{A}$ instância o oráculo de cifração $\varepsilon_k(m^i_b)$ para responder consultas m^i de cifração que gera $\varepsilon_k(m^i_1)$ se $b = 1$ ou escolhe $m'^i \xleftarrow{R} \{0,1\}^{|m|}$ e então retorna $\varepsilon_k(m'^i_0)$ se $b = 0$. Se atk = IND-CCA, então o oráculo de decifração $\Delta_k(m^i_b)$ também é instanciado e criptogramas são decifrados, salvo criptogramas gerados anteriormente pelo oráculo de cifração.
- O adversário $\mathcal{A}$ produz o *bit* b'.
- A saída do experimento é 1 se $b' = b$ e 0 caso contrário. $\mathcal{A}$ tem sucesso quando $Exp^{ror-atk}_{\mathcal{A},\Gamma}(n) = 1$.

A cifra simétrica Γ possui segurança $(t, q, \mu; \epsilon)$-ROR se qualquer adversário que executa em tempo no máximo t, faz no máximo q consultas ao oráculo totalizando no máximo μ *bits* e possui vantagem limitada por uma função ϵ desprezível no parâmetro de segurança n tal que:

$$\Pr[Exp^{ror-atk}_{\mathcal{A},\Gamma}(n) = 1] \leq \frac{1}{2} + \epsilon(n).$$

2.3.3 Inforjabilidade

As noções de segurança para primitivas de autenticação são ligeiramente diferentes, pois refletem a dificuldade computacional enfrentada por um adversário que objetiva forjar um autenticador para uma mensagem de sua escolha. O adversário tipicamente possui acesso a um oráculo de cálculo de autenticadores, sob a restrição de que a mensagem oferecida pelo adversário como evidência de forja não pode ter sido consultada como entrada

do oráculo. Antes de definir inforjabilidade é necessário definir um código autenticador de mensagem.

Definição 2.3.3. Codigo Autenticador de Mensagem – MAC. [41, Adaptada da Definição 4.1].

Uma MAC é uma tupla de algoritmos probabilísticos $(MAC, Vrfy)$ tais que:

- A chave k é gerada fazendo-se $k \xleftarrow{R} \mathcal{K}$.
- O algoritmo de geração de *tag* MAC recebe com entrada a chave k e a mensagem $m \in \{0,1\}^*$ e gera a *tag* $\mathcal{T} \leftarrow Mac_k(m)$.
- O algoritmo de verificação $Vrfy$ recebe como entrada a chave k, a mensagem m e a *tag* $\mathcal{T}$. Então é gerado um *bit* $b = Vrfy_k(m, \mathcal{T})$, com $b = 1$ para um autenticador válido e $b = 0$ caso contrário.

Definição 2.3.4. Inforjabilidade. [41, Adaptada].

Sejam $\mathcal{A}$ um adversário com poder computacional polinomial, $\psi = (Mac, Vrfy)$ um código de autenticação de mensagem e $Mac\text{-}forge_{\mathcal{A},\psi}(n)$ a execução de um experimento com $\mathcal{A}$ no nível de segurança n:

- A chave k é gerada fazendo-se $k \xleftarrow{R} \mathcal{K}$;
- $\mathcal{A}$ recebe acesso ao oráculo Mac_k e produz $(m, \mathcal{T})$ após $\mathcal{Q}$ consultas ao oráculo;
- A saída do experimento é 1 se $Vrfy\ (m,\ Mac(m, \mathcal{T})) = 1$, com $m \neq \mathcal{Q}$, e 0 caso contrário.

Um código de autenticação de mensagens ψ é existencialmente inforjável contra ataque adaptativo de mensagem escolhida se, para todo adversário $\mathcal{A}$, existe função ϕ desprezível no parâmetro de segurança n tal que:

$$\Pr[Mac\text{-}forge_{\mathcal{A},\psi}(n) = 1] \leq \phi(n).$$

2.4 Funções de *hash* universal

Uma função de *hash* (Definição 2.4.1) é uma candidata à função de sentido único (do inglês, *One-way Function*) [80] que tem como entrada uma sequência de tamanho arbitrário e gera uma sequência de tamanho fixo. A função de *hash* é dita candidata porque é fácil calcular o *hash* de uma mensagem, mas o processo reverso parece ser difícil, ou seja, calcular qualquer uma das infinitas mensagens que geram o *hash*.

Definição 2.4.1. Função de *hash* [41, Adaptada da Definição 4.11].

Uma função de *hash* é um algoritmo probabilístico de tempo polinomial H que satisfaz o seguinte:

- A chave k é gerada fazendo-se $k \xleftarrow{R} \mathcal{K}$;
- Existe um l polinomial tal que H toma como entrada k e uma sequência $x \in \{0,1\}^*$ e gera uma sequência $H_k(x) \in \{0,1\}^{l(n)}$ (onde n é o valor do parâmetro de segurança).

Se H_k é definido apenas para entradas $x \in \{0,1\}^{l'(n)}$ e $l'(n) > l(n)$, é dito que H é uma função de *hash* de tamanho fixo para entradas de tamanho $l'(n)$.

Funções de *hash* tem inúmeras aplicações em criptografia, tais como armazenamento de senha, derivação de chaves, verificação de integridade, assinaturas digitais e códigos autenticadores de mensagem. Esta última aplicação é proposta neste trabalho.

As funções de *hash* podem ser divididas em duas famílias quanto à chave: família de funções de *hash* chaveadas (Definição 2.4.2) e família de funções de *hash* não chaveadas.

Definição 2.4.2. Família de funções de *hash* chaveadas [73, Adaptada da Definição 4.1].

Uma família de funções de *hash* chaveadas é uma quádrupla $(\mathcal{M}, \mathcal{Y}, \mathcal{K}, H)$, onde:

- $\mathcal{Y}$ é um conjunto finito de *hash* de mensagem ou *tags* de autenticação;
- Para cada $k \in \mathcal{K}$, existe uma função de *hash* $h_k \in H$, onde cada $h_k : \mathcal{M} \to \mathcal{Y}$.

Já a segunda família, é uma função $h : \mathcal{M} \to \mathcal{Y}$, onde $\mathcal{M}$ e $\mathcal{Y}$ são iguais aos da Definição 2.4.2.

Em [54], são enumeradas 3 propriedades para uma função de *hash* h que não é chaveada: 1 resistência à pré-imagem, 2 resistência à segunda pré-imagem e 3 resistência à colisões. É importante ressaltar que cada propriedade implica à anterior ($3 \Rightarrow 2 \Rightarrow 1$) [41], sendo definidas como as seguintes:

1 **Resistência à pré-imagem:** Dado um valor de *hash* y, deve ser computacionalmente inviável encontrar x tal que $y = h(x)$;

2 **Resistência à segunda pré-imagem:** Dado um valor de *hash* y e uma mensagem x tal que $y = h(x)$, deve ser computacionalmente inviável encontrar uma mensagem $x' \neq x$ tal que $h(x) = h(x')$;

3 **Resistência a colisões:** Deve ser computacionalmente inviável encontrar duas mensagens distintas x e x' tais que $h(x) = h(x')$.

Em [19], Carter e Wegman introduzem o conceito de funções de *hash* universal (Definição 2.4.3) (do inglês, *Universal Hash Function*). Essas funções são selecionadas aleatoriamente de uma família H de funções cuidadosamente escolhidas, de forma que, raramente irão mapear elementos distintos do domínio D para o mesmo elemento da imagem R.

Em [79], foi definido formalmente um conjunto de famílias de *hash* fortemente universal (do inglês, *strongly universal hash families*) e também foi introduzido, mesmo que não formalmente, o conceito de família de *hash* quase fortemente universal (do inglês, *Almost Strongly Universal*).

Stinson [71, 72] define famílias ϵ-quase universal de *hash* (Definição 2.4.4) (do inglês, ϵ-*almost universal hash families*) que é uma das famílias utilizadas neste trabalho e também apresenta uma família ϵ-quase fortemente universal de *hash* (do inglês, ϵ-*almost strongly universal hash families*). Outra família que é utilizada neste trabalho é a família ϵ-bissimétrica de funções de *hash* (Definição 2.4.5) proposta por Ramzan [65].

Definição 2.4.3. Funções de *Hash* Universal [53, Definição 11].

Seja H um conjunto de funções de *hash* de $\mathcal{U}$ em 1, 2, ..., Υ associada a uma distribuição de probabilidade que é chamada também de H. É dito que H é universal se para todo par de valores $x \neq y$ de $\mathcal{U}$ tem-se

$$\Pr_{h \overset{R}{\leftarrow} H}[h(x) = h(y)] \leq \frac{1}{\Upsilon}$$

.

onde $\Pr_{h \overset{R}{\leftarrow} H}[h(x) = h(y)]$ é a probabilidade de $h(x) = h(y)$.

Considere agora que H é uma família de funções de *hash* chaveadas, com domínio D e imagem R.

Definição 2.4.4. Família ϵ_1-quase-universal de funções de *hash* [65, Definição 13].

É dito que H é uma família ϵ_1-universal de funções de *hash* se para todo $x, y \in D$ tal que $x \neq y$, $z \in R$,

$$\Pr[a \overset{R}{\leftarrow} \mathcal{K}_H : h_a(x) - h_a(y) = z] \leq \epsilon_1(n).$$

onde $\Pr[a \overset{R}{\leftarrow} \mathcal{K}_H : h_a(x) - h_a(y) = z]$ é a probabilidade de $h_a(x) - h_a(y) = z$, com $a \overset{R}{\leftarrow} \mathcal{K}_H$.

Definição 2.4.5. Família ϵ_2-bissimétrica de funções de *hash* [65, Definição 15].

É dito que H é uma família ϵ_2-bissimétrica de funções de *hash* se para todo x, $y \in D$ (aqui permite-se que $x = y$), $z \in R$,

$$\Pr[a_1 \xleftarrow{R} \mathcal{K}_H, a_2 \xleftarrow{R} \mathcal{K}_H : h_{a_1}(x) + h_{a_2}(y) = z] \leq \epsilon_2(n).$$

onde $\Pr[a_1 \xleftarrow{R} \mathcal{K}(H), a_2 \xleftarrow{R} \mathcal{K}_H : h_{a_1}(x)+h_{a_2}(y) = z]$ é a probabilidade de $h_{a_1}(x)+h_{a_2}(y) = z$, com $a_1 \xleftarrow{R} \mathcal{K}(H), a_2 \xleftarrow{R} \mathcal{K}_H$.

Os valores ϵ_1 e ϵ_2 são desprezíveis em função de um parâmetro de segurança n.

2.5 Funções pseudoaleatórias (fracas)

Um função pseudoaleatória (do inglês, *Pseudorandom Function – PRF*) é uma função que não pode ser distinguida de uma função verdadeiramente aleatória em tempo polinomial [17]. As análises em sistemas baseados em PRF, de maneira geral, primeiramente reduzem sua segurança para a suposição de uma função verdadeiramente aleatória, dependendo apenas de uma análise probabilística, em vez de limites de poder computacional ou dificuldade de problemas computacionais [41].

A Definição formal 2.5.1 é a clássica de uma PRF dada em [41]. Nessa definição é usado um distinguidor $\mathfrak{D}$ cujo objetivo é determinar dada uma entrada, se ela foi gerada por uma função F_k (com $k \xleftarrow{R} \{0,1\}^n$) ou por uma função f que é escolhida aleatoriamente entre todas as funções que mapeiam cadeias de n *bits* em n *bits*.

Definição 2.5.1. Função Pseudoaleatória – PRF [41, Definição 3.23].

Seja $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ uma função chaveada eficientemente computável. É dito que $\mathcal{F}$ é uma função pseudoaleatória se para todo distinguidor probabilístico de tempo polinomial (do inglês, *polynomial-time distinguisher*) $\mathfrak{D}$, existe uma função desprezível Φ tal que:

$$|\Pr[\mathfrak{D}^{\mathcal{F}_k(\cdot)}(1^n) = 1] - \Pr[\mathfrak{D}^{f(\cdot)}(1^n) = 1]| \leq \Phi(n),$$

onde $k \xleftarrow{R} \{0,1\}^n$ é escolhida de maneira uniformemente aleatória e f é escolhida de maneira uniformemente aleatória entre todas as funções que mapeiam cadeias de n *bits* em n *bits*.

Para o desenvolvimento deste trabalho, é utilizada um versão limitada de PRF, pois suas entradas são escolhidas aleatoriamente e não pelo adversário, intitulada função pseudoaleatória fraca (Definição 2.5.2) (do inglês, *weak Pseudorandom Functions* – wPRF).

Definição 2.5.2. Função Pseudoaleatória Fraca – wPRF. [2, Adaptada da Definição 4].

Considere uma família de funções $\mathcal{F}$ com entrada $\{0,1\}^l$ e saída $\{0,1\}^m$. É dito que $\mathcal{F}$ é $(q_{prf}, \epsilon_{prf})$-*pseudoaleatória em relação a uma distribuição*[1] $\tilde{\mathbb{D}}$ sobre $\{0,1\}^m$, se a vantagem de distinguir entre as duas distribuições a seguir, fazendo no máximo q_{prf} consultas para escolhas adaptativas distintas entre si de entradas x_1, ..., $x_{q_{pfr}}$, é no máximo ϵ_{prf}:

- $y_i = f(x_i)$ onde $f \xleftarrow{R} \mathcal{F}$.
- $y_i \leftarrow \tilde{\mathbb{D}}$.

$\mathcal{F}$ é chamada de pseudoaleatória fraca em relação a uma distruibuição $\tilde{\mathbb{D}}$, se suas entradas não são escolhidas por um adversário, mas sim escolhidas aleatoriamente de $\{0,1\}^l$. Essas funções diferem da definição clássica [41] que considera a distribuição $\mathbb{D}$ como a distribuição uniforme $\mathbb{U}_m$.

2.6 Extrator difuso

Uma das características da PUF (Seção 3) é que para a mesma entrada, a mesma PUF gera saídas diferentes devido a ruídos inseridos no meio, ou seja, PUFs se comportam como funções ruidosas (Definição 2.6.1). Em aplicações criptográficas, por questões de segurança, é importante que a chave utilizada seja exatamente a mesma, ou seja, sem ruídos. Para que se obtenha, para a mesma PUF, saídas iguais quando as entradas forem iguais, faz-se necessário o emprego de ferramentas que possam extrair esses ruídos, tais como, esboços seguros [24, 47] (do inglês, *secure sketch*) ou extratores difusos [23, 24] (do inglês, *fuzzy extractor*).

Definição 2.6.1. Função ruidosa [2, Definição 2].

Para três inteiros positivos l, m, $\delta \in \mathbb{N}$ com $0 \leq \delta \leq m$, uma (l, m, δ)-função ruidosa f^* é um algoritmo probabilístico que aceita entradas (desafios) $x \in \{0,1\}^l$ e gera saídas (respostas) $y \in \{0,1\}^m$ tais que a distância de Hamming[2] entre duas saídas para a mesma entrada seja no máximo δ.

O esboço seguro resolve o problema da tolerância a erro, ou seja, dada uma entrada r, é emitido um valor público v, que embora revelando pouco sobre r, permite a reconstrução exata de qualquer entrada r' ruidosa que é suficientemente próxima (distância de Hamming entre r e r' no máximo δ). No entanto, essa abordagem soluciona apenas o problema de

[1] Distribuição qualquer, pode ou não ser igual a $\mathbb{D}$.

[2] A distância de Hamming entre duas sequências de mesmo comprimento é o número de posições nas quais elas diferem entre si.

tolerância a erro, mas não a não uniformidade [24], por isso esse trabalho utiliza extratores difusos, que resolvem tanto o problema de tolerância a erro, quanto o problema da não uniformidade.

Extratores difusos (Definição 2.6.2) extraem de forma confiável uma sequência aleatória e uniforme s de uma entrada r que tem tolerância a ruído. A primeira vez que s é gerado, o extrator difuso produz uma sequência auxiliar ω que pode ser pública sem comprometer a segurança de s, sendo utilizado para recuperar s dada uma entrada ruidosa r' que é próxima de r (distância de Hamming entre r e r' no máximo δ).

Definição 2.6.2. Extrator Difuso. [2, Definição 5].

Um $(m, n, \delta, \epsilon_{FE})$ extrator difuso E é um par de procedimentos probabilísticos de geração Gen: $\{0,1\}^m \rightarrow \{0,1\}^n \times \{0,1\}^*$ (ver Figura 3.6) e de reprodução Rep: $\{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^n$ (ver Figura 3.7), onde o procedimento de geração produz informação auxiliar ω para que sua saída possa ser recuperada a partir do procedimento de reprodução.

A propriedade de correção garante que para a saída z e sua correspondente informação auxiliar ω $(z,\ \omega) \leftarrow Gen(y)$ e uma entrada ruidosa $y' \in \{0,1\}^m$ com a distância de Hamming $dist(y,\ y') \leq \delta$, leva ao procedimento $Rep(y', \omega) = z$. Se $dist(y, y') > \delta$, então nenhuma garantia é fornecida sobre a saída de Rep. A propriedade de segurança garante que, mesmo para adversários com acesso à informação auxiliar ω, a distância entre a distribuição observada de z e a distribuição uniforme é no máximo ϵ_{FE}.

2.7 Protocolo de acordo autenticado de chaves baseado em senha (PAKE)

Um protocolo para acordo autenticado de chaves baseado em senha (PAKE) estabelece uma chave compartilhada sobre um canal inseguro dependendo apenas de um pequeno segredo compartilhado, chamado senha. Essa é uma característica bastante útil, pois não é necessário nenhum dispositivo extra para armazenar uma longa chave criptográfica, mas apenas a capacidade da memória humana para guardar um curto segredo.

Os primeiros PAKEs surgiram na década de 90, com o protocolo de acordo de chaves Diffie-Hellman cifrado (do inglês, *Diffie-Hellman Encrypted Key Exchange – DH-EKE*) em 1992 [7] e com o protocolo de acordo de chaves exponencial simples com senha (do inglês, *Simple Password Exponential Key Exchange* – SPEKE) em 1996 [39]. Ambos possuem algumas variantes que são descritas resumidamente em [40].

Na atualidade, começando em 2000, o principal PAKE é o protocolo **AuthA** [6] de Bellare e Rogaway, que fornece as mesmas propriedades de segurança dos anteriores, contudo sendo mais simples, com menor custo de comunicação e mais versátil. Os trabalhos

[1, 16] propõem protocolos que são baseados em **AuthA** [6]. Outros protocolos PAKE são propostos em [15, 42]. Neste trabalho, será utilizado o protocolo proposto por Bresson *et al.* [16] como escolha de PAKE para fins de ilustração, mas qualquer PAKE seguro com a propriedade de *freshness* (Seção 2.7.1) pode ser utilizado.

Um protocolo para acordo de chave autenticada precisa satisfazer duas noções de segurança: a segurança semântica (também chamada de propriedade *freshness*, nesse contexto) e a propriedade de autenticação. Ambas as noções de segurança são definidas a seguir.

2.7.1 Segurança semântica (*Freshness*)

Protocolos estão sujeitos a vários tipos de ataques, pois as trocas de mensagens ocorrem em um canal inseguro. Deste modo, as informações transmitidas também estão propensas a diversas ameaças tais como espionagem e adulteração. Um agente malicioso pode espionar conversas honestas entre duas entidades e gravar todas as trocas de mensagens para no futuro tentar se passar por alguma das duas partes, simplesmente enviando as mensagens que foram anteriormente obtidas em uma execução honesta. Esse tipo de ataque é chamado de ataque de *replay* ou repetição.

Uma maneira de evitar esse tipo de ataque é garantir que os protocolos forneçam a propriedade de *freshness*. Essa propriedade garante que as mensagens provavelmente pertencem apenas à execução atual e não a repetições de mensagens anteriores trocadas em alguma conexão honesta. A propriedade de *freshness* pode ser obtida de várias maneiras, como por exemplo, com o uso de relógios ou com o uso de operações de desafios/respostas [46].

A utilização de relógio para assegurar *freshness* é possível através do incremento de marcas de tempo (do inglês, *timestamp*) em cada criptograma. A mensagem será aceita se conter a marca de tempo perto o suficiente do conhecimento do tempo atual. Um exemplo do uso do relógio para alcançar este objetivo é o sistema de autenticação Kerberos [57, 70].

Mecanismos de desafios/respostas são geralmente mais utilizados, pois não necessitam de sincronização dos relógios entre as entidades e, consequentemente são muito robustos e populares em projetos de protocolos criptográficos. Neste método, uma entidade A envia para uma entidade B um valor aleatório (desafio) e requer que esse valor esteja em uma próxima mensagem (resposta) recebida de B e vice versa.

O desafio deve ser protegido de forma que possa ser lido apenas pelo verdadeiro destinatário. Isso pode ser feito de algumas maneiras como, por exemplo, em [16], onde o cliente escolhe um valor aleatório $x \in [1, q-1]$ onde q é o tamanho do grupo cíclico finito $\mathbb{G}$, calcula o valor de g^x para um gerador $g \in \mathbb{G}$ e envia para o servidor que, por sua vez, escolhe um valor aleatório $y \in [1, q-1]$, calcula g^y, cifra o resultado utilizando o

segredo compartilhado e envia o resultado para o cliente. Depois, cada uma das partes verifica implicitamente se a outra parte recebeu o valor correto. Outros exemplos de tal uso podem ser encontrados em [15, 39].

2.7.2 Autenticação

Um dos objetivos de protocolos de acordo de chaves autenticado é garantir a propriedade de autenticação, que assegura que a chave criptográfica compartilhada seja obtida apenas pelas partes que satisfazem os requisitos de autenticação. A probabilidade do adversário conseguir personificar o cliente ou o servidor em uma execução do protocolo P é denotada por $Adv_{\mathcal{A},P}^{m-auth}(n)$ e, assim, o protocolo P é dito *seguro* se tal probabilidade é desprezível no parâmetro de segurança n.

Capítulo 3

Funções fisicamente não clonáveis

A ideia de utilizar características físicas difíceis de duplicar para a identificação de objetos, sistemas e pessoas não é nova. Em [38], William Herschel apresenta o uso de impressões digitais para a identificação de humanos desde o século XIX. A formalização do conceito sobre o uso de características físicas difíceis de duplicar foi introduzida primeiramente como função física unidirecional (do inglês, *Physical One-Way Function* – POWF) [60, 66], depois como função aleatória física (do inglês, *Physical Random Function*) [30] e finalmente como funções fisicamente não clonáveis (PUF).

PUFs foram introduzidas por Gassend *et al.* [30] e Pappu *et al.* [60] para se construir primitivas criptográficas que dependam de hipóteses tanto físicas quanto computacionais. Como as funções são geralmente implementadas a partir de um efeito físico imprevisível intrinsicamente ligado à instanciação de cada PUF individual, PUFs não podem ser facilmente duplicadas ou manipuladas sem alterar seu comportamento consideravelmente, apresentando propriedades que são bastante úteis em aplicações criptográficas.

O mecanismo de acesso a uma PUF é do tipo desafio-resposta, conforme Figura 3.1, onde as entradas são os desafios e as saídas constituem as respostas [52]. A correspondência entre desafios e respostas, determinada pelo processo de fabricação da PUF, codifica implicitamente um segredo permanente. Por essa razão, diversos trabalhos na literatura já utilizaram PUFs para construir primitivas criptográficas ou aplicações de segurança, como por exemplo: cifra de bloco resistente ao vazamento de chave [2], autenticação de dispositivos [74], geração de chaves criptográficas [51], geração de números aleatórios [59], proteção de propriedade intelectual [34] e construção de protocolos de autenticação, com foco em aplicações bancárias [28, 78].

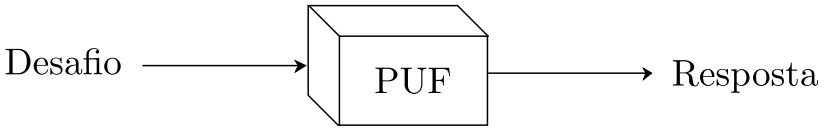


Figura 3.1: Mecanismo de acesso a PUF.

Algumas PUFs, com certas propriedades de segurança, estão sendo contestadas na literatura [36, 43, 50]. No entanto, este trabalho depende da escolha adequada de uma PUF que satisfaça propriedades de segurança rigorosas.

Neste Capítulo serão apresentados nas Seções 3.1 e 3.2 as propriedades de uma PUF, bem como os diferentes tipos de PUFs existentes, com uma breve descrição, respectivamente. A Seção 3.3 modela a PUF em três diferentes tipos e por fim, na Seção 3.4 é apresentado o jogo de respostas e o jogo de indistinguibilidade de resposta da PUF, que mostram seu comportamento imprevisível.

3.1 Propriedades

Abaixo estão as principais propriedades de PUFs que são importantes em aplicações criptográficas.

- **Independência**: Duas diferentes PUFs apresentam comportamentos completamente independentes [2];
- **Não clonabilidade**: Nenhum processo físico eficiente é conhecido que permite a clonagem física de PUFs, ou seja, cada PUF é única;
- **Evidência de violação**: Violações físicas na PUF provavelmente irão destruir a sua estrutura física, tornando-a inútil, ou transformando-a em uma nova PUF;
- **Saídas ruidosas**: A mesma entrada pode gerar saídas diferentes, desde que a distância entre as possíveis saídas seja limitada superiormente;
- **Distribuição não uniforme**: As saídas da PUFs não são uniformes, ou seja, a distribuição de probabilidade não é igual.

3.2 Tipos de PUFs

Aqui serão descritas brevemente alguns tipos de PUFs e seu funcionamento. Em [52], são divididas todas as PUFs propostas na literatura até o momento, baseando-se nos princípios de construção e funcionamento. A primeira categoria engloba as PUFs nas

quais sua construção e/ou funcionamento são intrinsecamente não eletrônicos, o que não impede que algumas técnicas eletrônicas e digitais sejam usadas para um processamento e armazenamento mais eficiente de suas respostas.

A segunda categoria são as PUFs cujo funcionamento básico depende de uma medição analógica de uma quantidade elétrica ou eletrônica. A terceira e a quarta categorias são baseadas em PUFs intrínsecas, onde uma é baseada em atraso e a outra é baseada no estado de sedimentação de células da memória digital.

E por fim, a quinta e última categoria é composta de conceitos propostos que estão intimamente relacionados com PUFs. A seguir será apresentado um breve exemplo das quatro primeiras categorias. Para maiores detalhes ver [52].

3.2.1 PUF ótica

PUFs óticas (Figura 3.2) baseadas em transparência média foram propostas em [60, 66], como uma POWF. Sua construção não depende de operações elétricas e são bastantes trabalhosas devido às configurações do *laser* e de mecanismos de posicionamento do sistema. Seu principal elemento é um *token* ótico que contém uma micro estrutura ótica construída pela mistura de esferas de vidro microscópicas (50μm) de refração em uma pequena (10mm, 10mm, 2,54mm) placa de epóxi transparente.

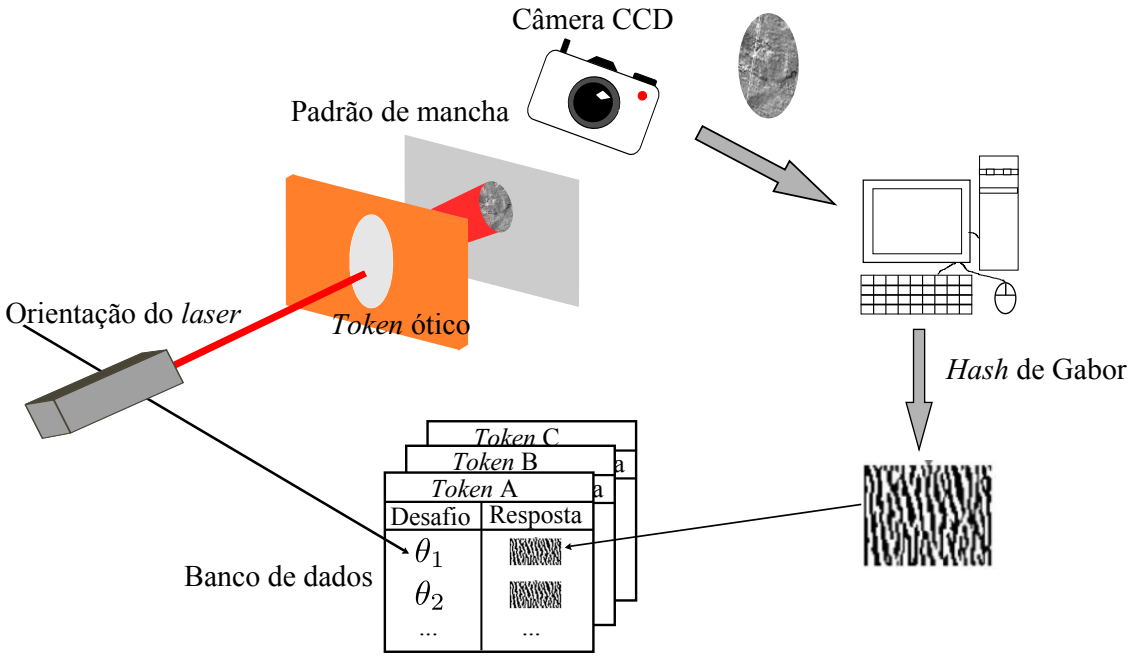


Figura 3.2: Descrição do funcionamento básico de uma PUF ótica [52, Adaptada].

O *token* é irradiado com um *laser Hellium-Neon* e a frente da onda emergente torna-se muito irregular devido ao espalhamento múltiplo do feixe de partículas de refração. O

padrão de mancha que surge é capturado por uma câmera *Charge-Coupled Device* (CCD) para processamento digital. *Hash* de Gabor[1] é aplicado ao padrão de mancha observada como um processo de extração de características. O resultado é uma sequência de *bits* que representam o valor de *hash*. O desafio da PUF é a exata orientação do *laser* e a resposta é o resultado do *hash* de Gabor de um padrão de mancha.

3.2.2 PUF de revestimento

PUFs de revestimentos (Figura 3.3), introduzidas em [77], tem como funcionamento básico a medição analógica de uma quantidade elétrica ou eletrônica. Ela considera a aleatoriedade das medições de capacitância em sensores em forma de pente no topo da camada de metal de um circuito integrado. Em vez de confiar apenas no efeito aleatório da variabilidade de fabricação, os elementos aleatórios são explicitamente introduzidos por meio de um revestimento passivo dielétrico pulverizado diretamente em cima dos sensores. E, uma vez que esse revestimento é opaco e quimicamente inerente, ele oferece forte proteção contra ataques físicos. O desafio da PUF é a seleção do sensor e a resposta é a medição da capacitância quantizada.

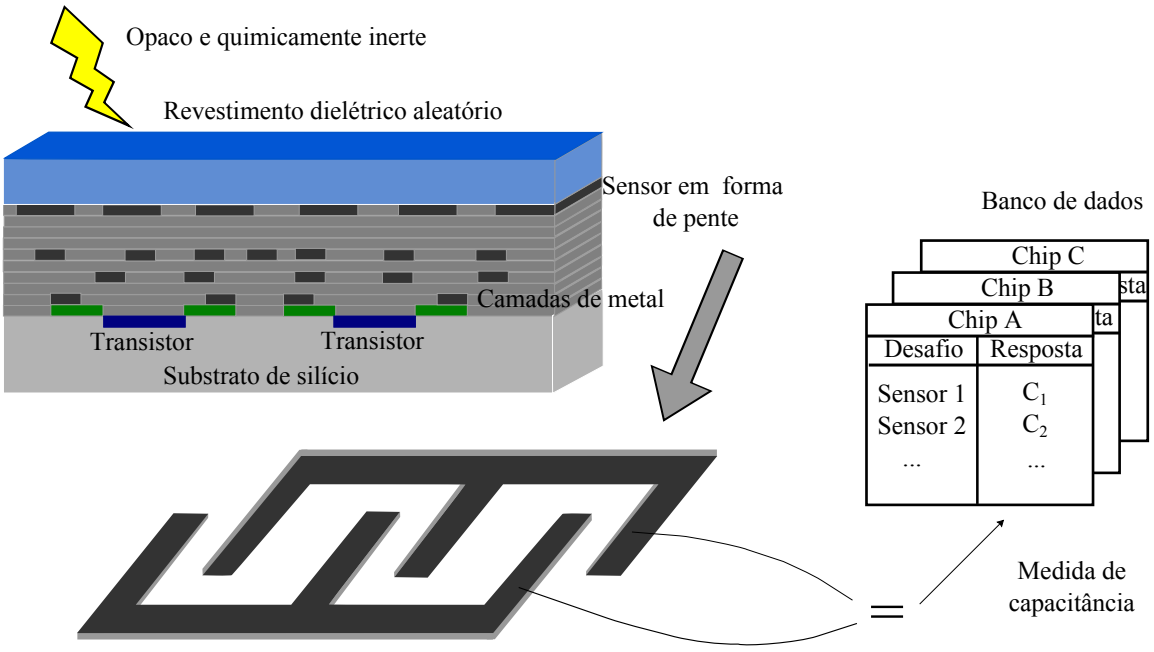


Figura 3.3: Descrição do funcionamento básico de uma PUF de revestimento [52, Adaptada].

[1] É um algoritmo de *hash*, cuja função é capturar os dados de uma mancha bruta e reduzi-lo a uma sequência de *bits*.

3.2.3 PUF oscilador em anel

PUFs construídas a partir de osciladores em anel (do inglês, *Ring Oscillator PUF – RO-PUF*) (Figura 3.4) introduzidas em [29, 30], são baseadas em atrasos intrínsecos. A saída de uma linha digital de atraso é invertida e alimentada de volta para a sua entrada (conforme mostrado pela linha vermelha na Figura 3.4), criando um laço chamado oscilador em anel. A frequência f do oscilador (linha roxa da Figura 3.4) é precisamente determinada pelo exato atraso da linha de atraso.

Assim, medir a frequência f é medir o atraso e devido às variações aleatórias de fabricação sobre o atraso, a frequência exata também será parcialmente aleatória ($\sim f$) e dependente do dispositivo. Para a medição da frequência um detector de bordas encontra as bordas crescentes (linha laranja na Figura 3.4) em uma oscilação periódica e um contador digital conta o número de arestas ao longo de um período de tempo. O desafio de uma RO-PUF é a configuração de um atraso em particular e a resposta é o valor do contador.

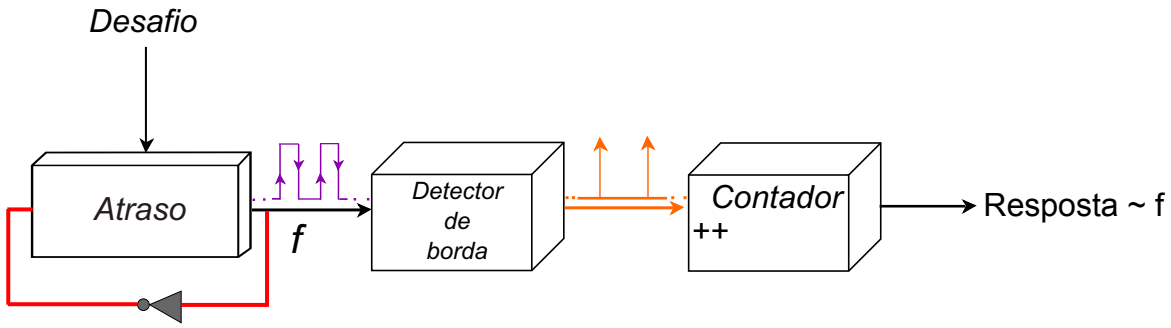


Figura 3.4: Descrição do funcionamento básico de uma PUF oscilador de anel. [52, Adaptada].

3.2.4 PUF SRAM

PUFs de memória estática de acesso aleatório (do inglês, *Static Random Access Memory* – SRAM), introduzidas em [33], são baseadas no estado de sedimentação de células da memória digital.

Uma SRAM é um tipo de memória digital que consiste de células que são capazes, cada uma, de armazenar um dígito binário. Sua célula (Figura 3.5) é logicamente construída com dois inversores cruzados acoplados, conduzindo assim a dois estados estáveis. A partir da descrição lógica da célula, não é claro em qual estado ela deverá estar depois que a memória for inicializada. Observa-se que algumas células tem preferência por armazenar 0 quando são inicializadas, outras tem preferência de armazenar 1 e em algumas não há preferência.

No entanto, esses três tipos de células estão distribuídas de forma aleatória na memória. Assim, a aleatoriedade da distribuição das células na memória, causada pela variação na fabricação, determina o comportamento da inicialização. O desafio da PUF é o endereço da SRAM e a resposta é o estado de inicialização das células endereçadas.

Até 2013, acreditava-se que este tipo de PUF possuía as características necessárias de uma PUF ideal, entretanto, Helfmeier *et al.* [36] demonstra que PUFs SRAM não atendem a vários requisitos que constituem uma PUF ideal, como por exemplo a não clonabilidade, e assim não são adequadas. Isso acontece devido a natureza compacta da SRAM, as interconexões padrão e a resiliência aos efeitos ambientais, tornando-a fácil de clonar.

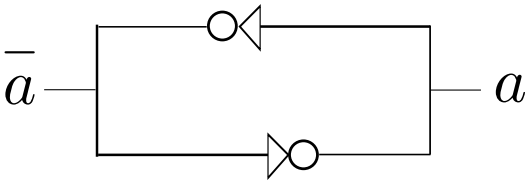


Figura 3.5: Circuito lógico de uma célula PUF SRAM. [52, Adaptada].

3.3 Modelagem

PUFs podem ser modeladas de três formas: como função ruidosa (Definição 3.3.1), como pós-processada (Definição 3.3.2) e como uma função pseudoaleatória fraca (Definição 3.3.3). A primeira forma, como funções ruidosas, é onde a saída da PUF é bruta.

Definição 3.3.1. PUF como função ruidosa.

Uma PUF é uma função **PUF**: $\{0,1\}^m \to \{0,1\}^n$ que sob entrada de tamanho m produz uma saída de tamanho n. A saída "bruta" da PUF é ruidosa e não uniformemente distribuída.

A segunda forma, como pós-processada, é quando a saída da PUF passa por um extrator difuso (Seção 2.6). Esta modelagem é útil para que a saída da PUF possa ser empregada em aplicações criptográficas, uma vez que é necessário a eliminação de ruídos inseridos e, assim, a mesma entrada para a mesma PUF, gere a mesma saída em instantes de tempo distintos.

Definição 3.3.2. PUF pós-processada.

Sejam η uma família de PUFs e E um extrator difuso. Uma família η_E é um conjunto de pares de procedimentos probabilísticos de geração (Figura 3.6) e reprodução (Figura 3.7),

com geração $Gen(\Pi)$, para $\Pi \in \eta$ com entrada $x \in \{0,1\}^l$ e saída $(z, \omega_x) \stackrel{def}{=} Gen(\Pi(x)) \in \{0,1\}^n \times \{0,1\}^*$; e reprodução $Rep(\Pi)$ com entrada $(x', \omega_x) \in \{0,1\}^l \times \{0,1\}^*$ e saída $z = Rep(\Pi(x'), \omega_x)$, onde x' é a entrada ruidosa.

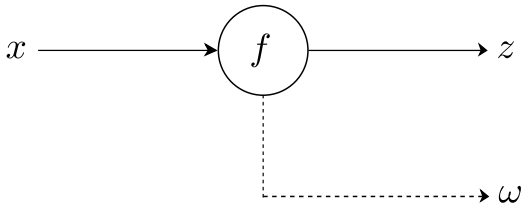


Figura 3.6: Processo de geração.

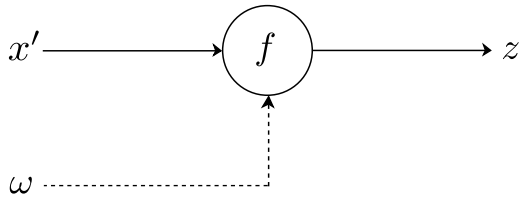


Figura 3.7: Processo de reprodução.

Após a definição de uma PUF pós-processada, que é a composição de PUF e extratores difusos, resta agora enunciar sua propriedade de segurança, derivada em [2].

Proposição 3.3.1. *Pseudoaleatoriedade da composição PUF com extrator difuso E. [2, Teorema 1]. Seja $\mathcal{P}$ uma família de PUFs $(q_{prf}, \epsilon_{prf})$-pseudoaleatória com respeito a uma distribuição $\mathbb{D}$ e E = (Gen, Rep) um extrator difuso. A vantagem de qualquer adversário que escolhe entradas adaptativas distintas entre si $x_1, \ldots, x_{q_{prf}}$ e recebe como saídas $(z_1, \omega_1), \ldots, (z_{qprf}, \omega_{qprf})$ para diferenciar entre as duas distribuições a seguir é no máximo $\epsilon_{prf} + q_{prf} \times \epsilon_{FE}$:*

- *$(z_i, \omega_i) = Gen(\Pi(x_i))$, onde $\Pi \stackrel{R}{\leftarrow} \mathcal{P}$.*
- *(z_i, ω_i) onde $z_i \leftarrow \mathbb{U}_n$, $(z'_i, \omega_i) = Gen(\Pi(x_i))$ e $\Pi \stackrel{R}{\leftarrow} \mathcal{P}$.*

E, por fim, a terceira e ultima modelagem, como função pseudoaleatória fraca, intitulada PUF-wPRF, proposta por Armknecht *et al.* [2], e utiliza a segunda modelagem.

Definição 3.3.3. PUF-wPRF. [2, Definição 6].

Considere a família $\mathcal{V}$ de funções PUF-wPRF, construída a partir de uma família $\mathcal{P}$ de PUFs pseudoaleatórias fracas e um extrator difuso E. Uma família $\mathcal{V}$ é um conjunto de

pares de procedimentos probabilísticos de geração (Figura 3.6) e reprodução (Figura 3.7), com geração $Gen(\Pi)$, para $\Pi \in \mathcal{P}$ com entrada $x \in \{0,1\}^l$ e saída $(z, \omega_x) \stackrel{def}{=} Gen(\Pi(x)) \in \{0,1\}^n \times \{0,1\}^*$; e reprodução $Rep(\Pi)$ com entrada $(x', \omega_x) \in \{0,1\}^l \times \{0,1\}^*$ e saída $z = Rep(\Pi(x'), \omega_x)$ onde x' é a entrada ruidosa.

3.4 Imprevisibilidade

Nesta Seção serão apresentados dois jogos que estão relacionados com a característica de imprevisibilidade da PUF. O primeiro jogo é o jogo de resposta contra um adversário polinomial $\mathcal{A}$ e o segundo é sua versão decisional, chamado *jogo de indistinguibilidade de respostas* da PUF. Ambos são definidos por Frikken *et al.* [28].

Dado o comportamento imprevisível, uma PUF pode ser modelada matematicamente como uma função **PUF** : $\{0,1\}^m \rightarrow \{0,1\}^n$, para a qual pode ser definido o seguinte *jogo de respostas* para um adversário $\mathcal{A}$ polinonimal [28]:

- **Fase 1**: O adversário $\mathcal{A}$ solicita e recebe respostas (r_i, ω_i) para qualquer desafio da PUF d_i de sua escolha;
- **Desafio da PUF**: $\mathcal{A}$ escolhe um desafio da PUF d não consultado anteriormente e recebe a informação auxiliar ω produzida por $Gen(PUF(d))$, mas não sua saída;
- **Fase 2**: $\mathcal{A}$ pode fazer mais solicitações para a PUF para quaisquer outros desafios da PUF que não sejam d;
- **Resposta**: Eventualmente, $\mathcal{A}$ gera seu palpite para r' para a resposta $r = Rep(PUF(d), \omega)$.

O adversário $\mathcal{A}$ vence caso $r = r'$. Para uma PUF imprevisível, tem-se a probabilidade $Adv_{\mathcal{A}}^{PUF}(n) = Pr[r = r']$ de $\mathcal{A}$ vencer como uma função desprezível. Para o jogo de respostas acima, pode-se definir sua versão decisional como o *jogo de indistinguibilidade de respostas* da PUF como a seguir [28]:

- **Inscrição**: $\mathcal{A}$ executa a fase de inscrição para qualquer valor d_i de sua escolha, recebendo o ω_i correspondente. Define-se o conjunto desses pares (d_i, ω_i) como $\mathcal{C}$;
- **Fase 1**: $\mathcal{A}$ solicita e recebe as respostas r_i da PUF para qualquer $(d_i, \omega_i) \in \mathcal{C}$ de sua escolha;
- **Desafio da PUF**: $\mathcal{A}$ escolhe um desafio da PUF d que foi consultado na fase de **Inscrição** mas não na **Fase 1**. Um *bit* aleatório b é escolhido. Se $b = 0$, $\mathcal{A}$ recebe $R = Rep(PUF(d), \omega)$, onde $(d, \omega) \in \mathcal{C}$; caso contrário, ele recebe uma sequência uniformemente escolhida a partir de $\{0,1\}^n$;

- **Fase 2**: $\mathcal{A}$ é permitido consultar a PUF para desafios da PUF em $\mathcal{C}$ exceto (d, ω);
- **Resposta**: Eventualmente, $\mathcal{A}$ gera um *bit* b'.

O adversário $\mathcal{A}$ vence quando $b = b'$. Temos $Adv_{\mathcal{A}}^{PUF-ind} = Pr[b = b']$ como a probabilidade de $\mathcal{A}$ vencer o jogo. Assume-se que $Adv_{\mathcal{A}}^{PUF-ind}(n) - \frac{1}{2}$ é desprezível.

3.5 Trabalhos relacionados

PUFs são utilizadas no escopo de vários trabalhos graça às suas propriedades e o fato da chave criptográfica estar armazenada implicitamente em sua estrutura. Para gerar números aleatórios, Odonnell *et al.* [59] descrevem como usar PUFs, especificamente PUFs de silício, para criar um *hardware* candidato à gerador de números aleatórios. O *hardware* tenta extrair aleatoriedade de sistemas físicos complexos, neste caso a PUF. Eles apresentam um argumento a favor da viabilidade dos métodos e fornecem um breve estudo da aleatoriedade dos números gerados usando PUFs, através de uma série de testes estatísticos, que mostram um promissor caminho para o uso de PUFs em um *hardware* gerador de números aleatórios.

Para autenticação de dispositivo, Suh e Devadas [74] propõem um projeto com PUFs que explora características de atrasos (por exemplo, Seção 3.2.3) inerentes de fios e transistores, que diferem de *chip* para *chip*. Além disso, descrevem como PUFs podem permitir autenticação de baixo custo para circuitos integrados individuais e como gerar chaves secretas voláteis para operações criptográficas.

Guarjado *et al.* [34], propõem o uso de PUFs para a proteção de propriedade intelectual. Eles apresentam novos protocolos para o problema de proteção de um protocolo de internet (do inglês, *Internet Protocol* – IP) em arranjo de portas programável em campo (do inglês, *Field Programmable Gate Arrays* – FPGA) baseado em criptografia de chave pública, analisando as vantagens e custos da abordagem e descrevem uma PUF intrínseca para FPGAs baseada em PUF-SRAM. Entretanto, conforme visto na Seção 3.2.4, em 2013 Helfmeier *et al.* [36] demonstram que SRAM não possui as propriedades necessárias para ser uma PUF.

Maes *et al.* [51], apresenta o uso de PUF, especificadamente RO-PUF, para geração de chaves criptográficas. O projeto que é intitulado PUFKY, é prático e modular, e sua implementação foi desenvolvida e avaliada em um conjunto de dispositivos PFGA.

Outra forma de empregar PUF é para projetar cifras de blocos resistentes ao vazamento de *bits* da chave. Essa é uma forma no qual PUFs foram empregadas para o desenvolvimento deste trabalho (Capítulo 4). Em [2], Armknecht *et al.* propõem uma cifra combinando uma cifra de Luby-Rackoff, com 3 funções de rodada, (que será expli-

cado na Seção 4.1.1) com PUF-wPRF (Seção 3.3.3). Na cifra cada função de rodada é uma PUF-wPRF diferente e possui o valor aleatório ρ para tornar aleatória a primeira função de rodada e, assim, a PUF-wPRF se comportará como uma PRF. A análise da cifra mostra que ela oferece segurança IND-CPA.

E, por fim, PUFs podem ser empregadas na construção de protocolos de autenticação, com foco em aplicações bancárias. Essa é outra forma na qual PUF foram empregadas neste trabalho (Capítulo 5). Em [28], Frikken *et al.* apresentam a construção de um protocolo apenas de autenticação, sem o estabelecimento de uma chave de sessão, onde o usuário se autentica usando uma I-PUF (PUF integrada) emitida pelo banco e uma senha. Além disso, eles mostram uma adaptação do protocolo para suportar senhas de pânico (definidas na Seção 5.3).

Outra proposta é feita por Tuyls e Škorić [78]. Eles propõem um protocolo de autenticação com estabelecimento de uma chave de sessão baseado em PUFs. O protocolo é similar ao anterior, sendo utilizado para autenticação entre o cliente e o servidor, com o incremento do estabelecimento de uma chave de sessão. Entretanto, para um modelo de atacante mais realista no cenário de autenticação, onde o atacante tem o controle físico da PUF e de seu respectivo leitor por um curto espaço de tempo, é possível personificar o servidor, conforme demonstrado por Bush *et al.* [18].

Capítulo 4

Cifração autenticada utilizando PUFs

Neste Capítulo será apresentada a proposta de uma cifra de bloco baseada em PUFs, que é construída a partir de uma cifra de Luby-Rackoff com 4 rodadas envolvendo tanto PUFs (Seção 3) quanto função de *hash* universal (Seção 2.4) e aprimora o estado da arte [2] tanto em segurança quanto no comprimento do criptograma resultante. Além disso, apresenta a proposta de um autenticador que também utiliza PUFs e é construído combinando um MAC clássico de tamanho fixo com uma função de *hash* universal. Para ambas as propostas são fornecidas análises de segurança que consideram as noções padronizadas na literatura.

Cifras de blocos baseadas em PUFs são utilizadas para cifração de disco ou dispositivos, pois o proprietário da PUF irá cifrar dados para ele próprio futuramente decifrar. Entretanto, o emprego dessas cifras para cifrar informações entre duas ou mais entidades não é viável, pois seria necessário que a entidade emissora da mensagem enviasse sua PUF, por um meio seguro, à entidade receptora, para que ela seja capaz de decifrar a mensagem. E, caso fosse possível enviar a PUF através de um canal seguro, seria melhor enviar a mensagem diretamente através desse canal.

O Capítulo está dividido da seguinte forma: Na Seção 4.1 será apresentada uma cifra de bloco que é construída a partir de uma cifra de Luby-Rackoff (Seção 4.1.1) com 4 rodadas envolvendo PUFs (Seção 3) e funções de *hash* universal (Seção 2.4) que aprimora o estado da arte de cifração baseada em PUFs tanto em segurança quanto no comprimento do criptograma resultante. Na Seção 4.2 será apresentado um MAC construído pela combinação de um MAC clássico de tamanho fixo com uma função de *hash* universal. Em ambas as Seções, as análises de segurança são fornecidas considerando noções padronizadas na literatura.

4.1 Proposta de cifra de bloco

Nesta Seção, será apresentada a construção de uma cifra de bloco baseada em PUFs. Mas antes de apresentar a cifra proposta, será mostrado na Seção 4.1.1, uma breve descrição da cifra de Luby-Rackoff, bem como suas melhorias. Nas Seções 4.1.2 e 4.1.3, será apresentada a cifração e decifração da cifra proposta, respectivamente. Na Seção 4.1.4, serão demonstradas as provas de segurança que garantem segurança IND-CPA e IND-CCA1. E por fim, na Seção 4.1.5 é feita uma comparação da cifra com os trabalhos propostos em [2] e [61, 65].

4.1.1 Cifra de Luby-Rackoff

Redes de Feistel (Figura 4.1) são as principais ferramentas para construir permutações pseudoaleatórias sobre sequências de *bits* a partir de funções que preservam o comprimento sobre uma sequência bits [65]. A Figura 4.1 ilustra uma rede de Feitel com i rodadas, onde a entrada é divida em duas metades L_0 e R_0. Se a entrada tem comprimento n *bits* então L_i e R_i, cada um, tem comprimento $\frac{n}{2}$. Cada função de rodada f_i é uma função não-inversível e L_i e R_i são as saídas geradas pela rede depois de i rodadas.

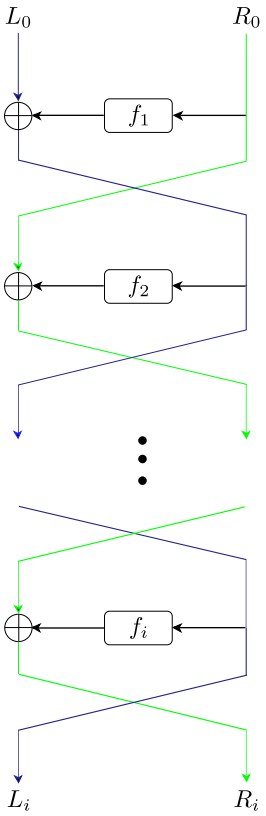
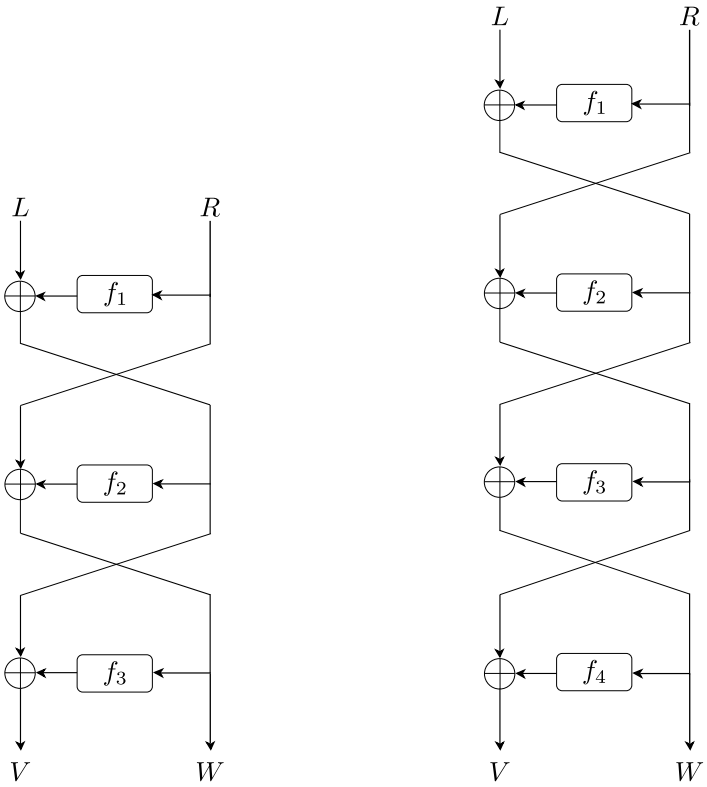


Figura 4.1: Rede de Feistel, onde f_i são funções de rodada, L_0 e R_0 são as entradas e L_i e R_i as saídas. [65, Adaptada]

As rede de Feistel foram usadas em várias cifras de blocos conhecidas como, por exemplo, o DES. Uma de suas grandes vantagens é que a cifração e a decifração são bastante semelhantes e, em alguns casos até iguais, onde é necessário apenas utilizar a cifra de forma inversa fazendo como que sua implementação seja mais resumida, pois não precisa implementar algoritmos diferentes para cifração e decifração. Vale destacar que uma Rede de Feistel é inversível mesmo que suas funções de rodada não sejam [41].

Devido à sua praticidade e eficiência, Michael Luby e Charles Rackoff em [48, 49] analisaram a construção de uma rede de Feistel (Figura 4.1), e provaram que apenas 3 funções de rodada (Figura 4.2 (a)) eram suficientes para tornar a rede uma permutação pseudoaleatória se as funções de rodada f_1, f_2 e f_3 fossem PRFs, conforme definidas na Seção 2.5, enquanto que 4 funções de rodada (Figura 4.2 (b)) eram necessárias para tornar a rede uma permutação super pseudoaleatória, que é quando a rede continua segura mesmo na presença de um adversário que tem acesso ao oráculo de decifração. Devido a esse importante resultado, uma rede de Feistel que contenha 3 ou 4 funções de rodada é chamada de Cifra de Luby-Rackoff.



(a) Cifra de Luby-Rackoff com 3 funções de rodada

(b) Cifra de Luby-Rackoff com 4 funções de rodada

Figura 4.2: Cifra de Luby-Rackoff, onde f_1, f_2, f_3 e f_4 são PRFs, L e R são as entradas e V e W são as saídas.

Em [58], Naor e Reingold melhoraram a cifra de Luby-Rackoff que contém 4 funções de rodada, trocando a primeira e a última rodada da Cifra de Luby-Rackoff por funções de *hash* fortemente universal, h_1 e h_2, conforme Figura 4.3, onde tais funções tem que consistir de permutações para que a cifra continue inversível.

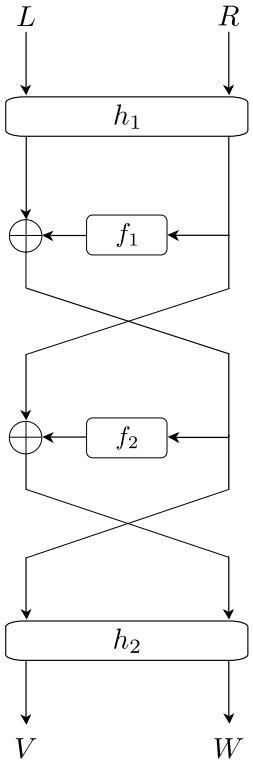


Figura 4.3: Construção de Naor-Reingold, onde h_1 e h_2 são funções de *hash* fortemente universal. [58, 65, Adaptada]

Essa construção é mais eficiente porque utiliza funções de *hash* fortemente universal que envolvem apenas propriedades estatísticas específicas, e por isso podem ser implementadas de forma mais eficiente que boas PRFs e também tais funções não fazem qualquer suposição criptográfica [65]. Assim, Naor e Reingold [58] alcançaram um importante resultado reduzindo de 4 para apenas 2 chamadas à PRF, garantindo, portanto, uma implementação mais eficiente.

Além de propor a utilização de funções de *hash* fortemente universal na primeira e na última função de rodada no lugar das PRFs, Naor e Reingold [58] também propõe duas otimizações. A primeira é utilizar a mesma PRF na segunda e terceira rodada, o que economiza material de chave. E a segunda é operar apenas com metade dos dados, onde as funções de *hash* universal h_i, são funções de *hash* quase universal, o que economiza tempo de execução e material de chave. Infelizmente as duas otimizações não podem ser utilizadas juntas, pois nem sempre produz uma cifra segura.

Entretanto, Ramzan *et al.* [61, 65] demonstraram que as duas otimizações podem ser utilizadas simultaneamente se as funções de *hash* universal forem funções de *hash* quase universal bissimétricas [1]. A cifra de Luby-Rackoff com essas otimizações é conhecida como cifra Patel-Ramzan-Sundaram conforme Figura 4.4.

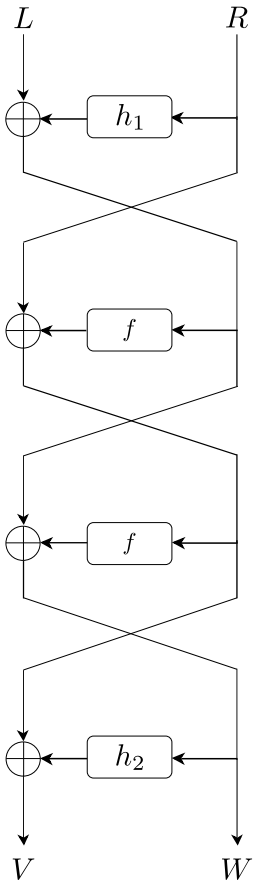


Figura 4.4: Construção de Patel-Ramzan-Sundaram, onde h_1 e h_2 são funções de *hash* quase universal bissimétricas. [61, 65, Adaptada]

4.1.2 Cifração

Sejam $\mathcal{V}$ uma família de PUF-wPRFs com entrada e saída em $\{0,1\}^n$, H uma família de funções de *hash* ϵ_1-quase universal ϵ_2-bissimétricas com entrada e saída também de comprimento n e a entrada para a cifra em $\{0,1\}^{2n}$.

A cifra de bloco $\varepsilon^{\mathcal{V}H}$ possui 4 rodadas e se baseia na construção Luby-Rackoff [48, 49] com modificações de [58, 61, 65]. Na construção original de [61, 65], as funções de rodada consistiam em PRFs ou funções de *hash* universal. Na construção proposta, são realizadas duas invocações a uma PUF-wPRF $f \in \mathcal{V}$ e a duas funções de *hash* universal distintas $h_i \in H$, $i = 1, 2$. Como pode ser visto na Figura 4.5, a primeira e a última funções de rodada

[1] Que são definidas separadamente na Seção 2.4.

são as funções de *hash* universal h_i. O texto cifrado é (V, W, ω_1, ω_2 e ρ) e compreende os blocos propriamente ditos, as informações auxiliares ω_1, ω_2 produzidas pelo extrator difuso e o vetor de inicialização aleatório $\rho \in \{0,1\}^n$.

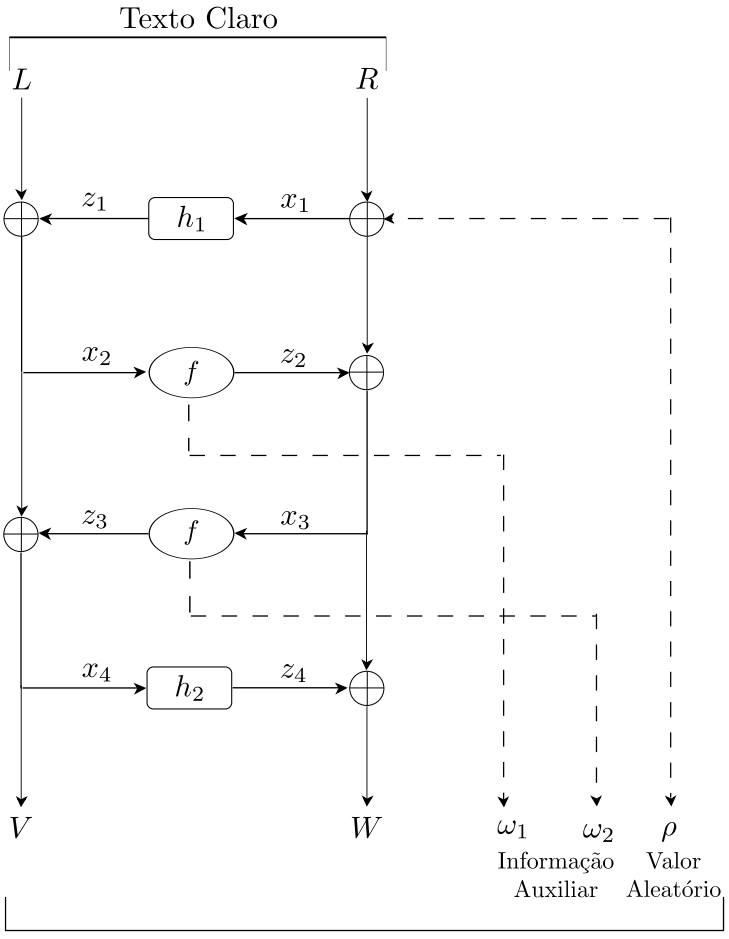


Figura 4.5: Proposta de cifração baseada na construção de Luby-Rackoff, onde f é uma PUF-wPRF.

4.1.3 Decifração

A decifração, que pode ser vista na Figura 4.6, é similar para o caso de uma cifra de Luby-Rackoff tradicional, onde a informação auxiliar ω_i é utilizada junto com o procedimento de reprodução para reconstruir as saídas z_2 e z_3 das PUF-wPRFs da segunda e terceira funções de rodada e o valor ρ para reverter a entrada utilizada na primeira função de rodada durante a cifração.

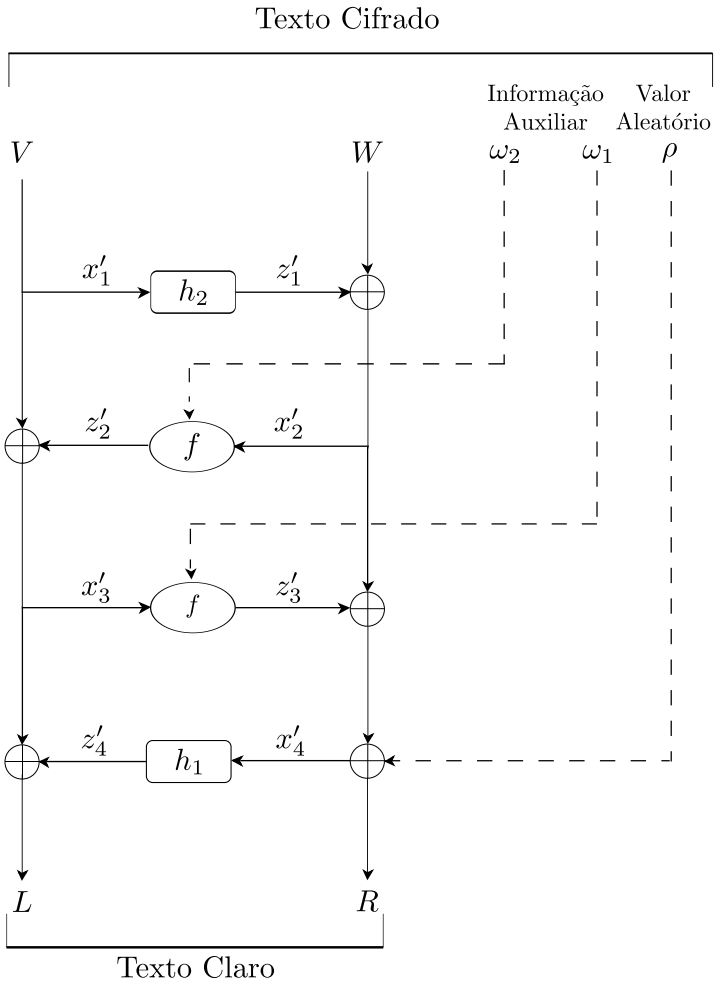


Figura 4.6: Decifração da construção baseada na cifra de Luby-Rackoff, onde f é uma PUF-wPRF.

4.1.4 Análise de segurança

Nesta Seção são apresentadas as provas de segurança que garantem tanto segurança IND-CPA quanto segurança IND-CCA1.

Segurança contra ataques de texto claro escolhido

A intuição para a demonstração a seguir é baseada na noção de segurança ROR-CPA, onde o jogo 1 consiste na randomização da saída da primeira ocorrência de f, o que equivale a randomizar R, e o jogo 2 consiste na randomização da entrada da primeira ocorrência de f, o que equivale a randomizar L.

Como a vantagem de distinguir L de L' (lado esquerdo da mensagem escolhida aleatoriamente) é desprezível, a vantagem de distinguir R de R' (lado direito da mensagem escolhida aleatoriamente) também é desprezível, e os jogos 1 e 2 são indistinguíveis entre

si, o adversário tem vantagem desprezível em distinguir a cifração de um texto claro de sua escolha da cifração de uma mensagem aleatória.

Teorema 4.1.1. *Seja $\varepsilon^{\mathcal{V}H}$ a cifra da Figura 4.5 usando a família $\mathcal{V}$ de PUF-wPRFs e a família H de funções de hash ϵ_1-quase universal ϵ_2-bissimétricas. Então a vantagem do adversário que realiza até q_{prf} consultas é no máximo:*

$$4\epsilon_{prf} + 2q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^{n-1}} + 2q_{prf}^2 \times \epsilon_1(n)$$

Demonstração. Seja $(L^{(i)}, R^{(i)})$, $i = 1, \ldots, q_{prf}$; a sequência de escolhas adaptativas de textos claros e $x_j^{(i)}$, $z_j^{(i)}$ a entrada e saída respectivamente para a função de rodada j dentre (h_1, f, f e h_2), e ρ é um valor uniformemente aleatório utilizado para tornar aleatório a parte R do texto claro. O teorema será provado pela definição de uma sequência de jogos e estimativa da vantagem de se distinguir entre eles. O jogo real é, portanto, o cenário onde o adversário recebe a cifração do texto claro escolhido por ele.

No jogo 1, as saídas $z_2^{(i)}$ da segunda função de rodada f são trocadas por valores uniformemente aleatórios $\tilde{z}_2^{(i)} \xleftarrow{R} \{0,1\}^n$. Sob a hipótese de que os valores $x_2^{(i)}$ são distintos entre si, a vantagem de distinguir entre os dois casos de acordo com a Proposição 3.3.1 é no máximo $\epsilon_{prf} + q_{prf} \times \epsilon_{FE}$. Além disso, temos que considerar a probabilidade de colisão entre $R^{(i)} \oplus \rho$ para dois valores $R^{(i)}$ distintos que é igual a $\frac{q_{prf}^2}{2^n}$ e a probabilidade de colisão em h_1 para dois valores $x_1^{(i)}$ distintos que é $q_{prf}^2 \times \epsilon_1(n)$. Como consequência, a vantagem para distinguir entre o jogo real e o jogo 1 é limitada superiormente por $\epsilon_{prf} + q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^n} + q_{prf}^2 \times \epsilon_1(n)$.

O jogo 2 é definido como o jogo 1, onde agora as entradas $x_2^{(i)}$ para a segunda função de rodada são trocadas por $\tilde{x}_2^{(i)} \xleftarrow{R} \{0,1\}^n$ selecionados aleatoriamente. Observe que os valores de $x_2^{(i)}$ são usados em dois diferentes contextos: i) para calcular o lado esquerdo do criptograma (pela adição módulo 2 com a saída da terceira função de rodada) e ii) como entrada para a segunda função de rodada.

Em relação a i), observe que as saídas da terceira função de rodada são independentes dos valores de $x_2^{(i)}$, com os valores de $\tilde{z}_2^{(i)}$ (e portanto as entradas da terceira função de rodada) uniformemente escolhidos por definição, e que os valores $x_2^{(i)}$ são independentes do texto claro (por causa da escolha da função de *hash* universal). Assim, i) e ii) representam duas características independentes, possibilitando distinguir entre o jogo 1 e o jogo 2, e assim são examinadas separadamente.

A vantagem de distinguir entre o jogo 1 e o 2 baseado em i) é equivalente a decidir se os valores $L^{(i)} \oplus z_1 \oplus z_3$ são uniformementes aleatórios ou pertencem às saídas da terceira

função de rodada. Com os mesmos argumentos acima utilizados, a vantagem é limitada superiormente por $\epsilon_{prf} + q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^n} + q_{prf}^2 \times \epsilon_1(n)$.

A vantagem de distinguir entre o jogo 1 e o jogo 2 baseado em ii) é no máximo a vantagem de distinguir $(\Pi(x_1), \ldots, \Pi(x_{qprf}))$ de $(\Pi(\tilde{x}_1), \ldots, \Pi(\tilde{x}_{qprf}))$ onde Π denota a PUF usada na segunda função de rodada. Pela definição de PUFs, a vantagem de distinguir entre $(\Pi(x_1), \ldots, \Pi(x_{qprf}))$ e $y_1, \ldots, y_{qprf}$, onde $y_i \leftarrow \mathbb{D}$ para $\mathbb{D}$ uma distribuição apropriada é no máximo ϵ_{prf} [2]. Na verdade, o mesmo vale para $(\Pi(\tilde{x}_1), \ldots, \Pi(\tilde{x}_{qprf}))$ (o fato de que os valores $\tilde{x}_i^{(1)}$ são desconhecidos não pode aumentar a vantagem). Assim, por desigualdade triangular, segue-se que a vantagem em relação a ii) é no máximo $2\epsilon_{prf}$. No total a vantagem de distinguir entre o jogo 1 e o jogo 2 é menor ou igual a $3\epsilon_{prf} + q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^n} + q_{prf}^2 \times \epsilon_1(n)$.

Finalmente, observamos que é indistinguível se $x_2^{(i)}$ ou $L^{(i)}$ é aleatório e também se $z_2^{(i)}$ ou $R^{(i)}$ são aleatórios. Assim, o jogo 2 é indistinguível de um jogo aleatório onde os textos claros são aleatórios. Por transitividade, a vantagem de um adversário real ou aleatório é no máximo $4\epsilon_{prf} + 2q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^{n-1}} + 2q_{prf}^2 \times \epsilon_1(n)$. □

Segurança contra ataques de criptograma escolhido

A intuição para a demonstração a seguir é baseada na noção de segurança ROR-CCA1 e segue a demonstração do Teorema 4.1, com a única diferença que nesse teorema o adversário pode fazer consultas ao oráculo de decifração.

Teorema 4.1.2. *Sejam $\varepsilon^{\mathcal{V}H}$ a função de cifração (Figura 4.5) e $\Delta^{\mathcal{V}H}$ a função decifração (Figura 4.6) usando a família $\mathcal{V}$ de PUF-wPRFs e a família H de funções de hash ϵ_1-quase universal ϵ_2-bissimétrica. Então a vantagem do adversário que realiza até q_{prf} consultas é no máximo:*

$$8\epsilon_{prf} + 4q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^{n-2}} + 4q_{prf}^2 \times \epsilon_1(n) + q_{prf}^2 \times \epsilon_2(n).$$

Demonstração. Para o caso de cifração temos que a vantagem de um adversário distinguir entre o jogo real ou aleatório é no máximo $4\epsilon_{prf} + 2q_{prf} \times \epsilon_{FE} + \frac{q_{prf}^2}{2^{n-1}} + 2q_{prf}^2 \times \epsilon_1(n)$, de acordo com o Teorema 4.1.1.

Agora será considerado o caso em que o adversário pode fazer consultas de decifração. Seja $(V^{(i)}, W^{(i)})$, $i = 1, \ldots, q_{prf}$; uma sequência de escolhas adaptativas de criptogramas, onde $x'^{(i)}_j$, $z'^{(i)}_j$ são entradas e saídas respectivas para a função da rodada j dentre (h_2, f, f, h_1) e o valor ρ reverte a entrada utilizada na primeira função de rodada durante a cifração. O teorema será provado pela estimativa da vantagem de se distinguir entre uma sequência de jogos, na verdade os mesmos utilizados no Teorema 4.1, com a diferença que agora o adversário possui acesso ao oráculo de decifração $\Delta^{\mathcal{V}H}$.

Temos que o jogo 1 consiste na randomização da saída da primeira ocorrência de f, ou seja, as saídas $z'^{(i)}_2$ da segunda função de rodada f são trocadas por alguns valores uniformemente aleatórios $\tilde{z'}^{(i)}_2 \xleftarrow{R} \{0,1\}^n$ o que equivale a randomizar W. Já o jogo 2 consiste na randomização da entrada da primeira ocorrência de f, ou seja, as entradas $x'^{(i)}_2$ para a segunda função de rodada são trocadas por $\tilde{x'}^{(i)}_2 \xleftarrow{R} \{0,1\}^n$, o que equivale a randomizar V.

Como a vantagem de distinguir V de V' (lado esquerdo do criptograma) aleatório é desprezível, a vantagem de distinguir W de W' (lado direito do criptograma) aleatório também é desprezível, e os jogos 1 e 2 são indistinguíveis entre si, assim o adversário tem vantagem desprezível em distinguir a decifração do criptograma normal de um criptograma aleatório.

Assim, utilizando o mesmo argumento do Teorema 4.1.1, tem-se que a vantagem do adversário real ou aleatório fazendo até q_{prf} consultas é no máximo $4\epsilon_{prf} + 2q_{prf} \times \epsilon_{FE} + \frac{q^2_{prf}}{2^{n-1}} + 2q^2_{prf} \times \epsilon_1(n)$. Como agora o adversário tem acesso ao oráculo de decifração, tem-se que analisar o evento onde $1 \leq i, j \leq q$ tal que $h_1(R^{(i)}) \oplus L^{(i)} = W^{(j)} \oplus h_2(V^{(j)})$. De acordo com [65], temos que:

$$\begin{aligned}
&\Pr[\exists\ 1 \leq i,j \leq q_{prf} : h_1(R^{(i)}) \oplus L^{(i)} = W^{(j)} \oplus h_2(V^{(j)})] \\
&\quad\leq \sum_{1\leq i,j\leq q_{prf}} \Pr[h_1(R^{(i)}) \oplus L^{(i)} = W^{(j)} \oplus h_2(V^{(j)})] \\
&\quad\leq \sum_{1\leq i,j\leq q_{prf}} \Pr[h_1(R^{(i)}) \oplus h_2(V^{(j)}) = W^{(j)} \oplus L^{(i)}] \\
&\quad\leq q^2_{prf} \times \epsilon_2(n).
\end{aligned}$$

Assim, a vantagem do adversário distinguir entre o jogo real ou aleatório baseado na noção de segurança ROR-CCA1 é no máximo: $8\epsilon_{prf} + 4q_{prf} \times \epsilon_{FE} + \frac{q^2_{prf}}{2^{n-2}} + 4q^2_{prf} \times \epsilon_1(n) + q^2_{prf} \times \epsilon_2(n)$.

□

Desta forma, a segurança da construção fica reduzida às propriedades de segurança da composição PUF com extrator difuso, tanto para ataques de texto claro escolhido quanto ataques de criptograma escolhido.

4.1.5 Comparação com trabalhos propostos

A cifra proposta em [2], mostrada na Figura 4.7, é uma combinação da construção Luby-Rackoff com PUF-wPRF e possui 3 funções de rodada, onde cada rodada (f_1, f_2 e f_3), é uma PUF-wPRF diferente e possui o valor aleatório ρ para tornar aleatória a

entrada da primeira função de rodada. O criptograma resultante contém 3 informações auxiliares (ω_1, ω_2 e ω_3), que são geradas pela invocação da PUF-wPRF em cada uma das 3 funções de rodada, além do valor aleatório ρ, V e W. A análise mostra que a cifra oferece segurança IND-CPA.

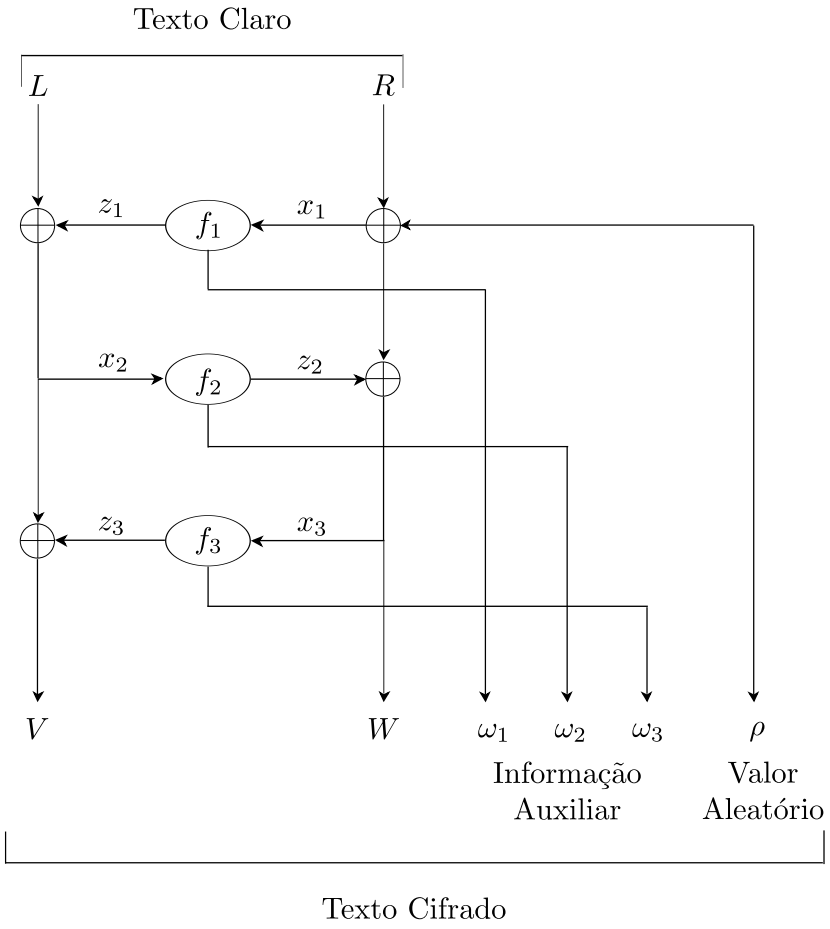


Figura 4.7: Cifra proposta em [2] onde f_1, f_2 e f_3 são PUF-wPRF.

Diferentemente da cifra mencionada acima, a construção proposta utiliza 4 funções de rodada, onde apenas a segunda e a terceira funções de rodada utilizam PUF-wPRFs, reduzindo assim o tamanho do criptograma, uma vez que cada chamada a uma PUF-wPRF gera informação auxiliar (ω_i) que precisa ser fornecida durante a decifração. Como a cifra de Luby-Rackoff com 4 rodadas possui a vantagem de fornecer segurança IND-CCA [61, 65], a cifra proposta preserva essa vantagem, de modo a possuir uma noção de segurança maior do que a cifra apresentada em [2], que possui segurança IND-CPA. Desse modo a cifra proposta aprimora o estado da arte tanto em segurança quanto no comprimento do criptograma resultante.

A estrutura da cifra proposta com 4 rodadas é praticamente a mesma apresentada em [61, 65], com algumas diferenças. A segunda e terceira funções de rodada em [61, 65] são PRFs, diferentemente da construção proposta que utiliza PUF-wPRFs. A principal

vantagem de se utilizar PUF-wPRFs é não ser necessário armazenar uma chave passível de captura, visto que a chave está implicitamente embutida na configuração da PUF. É importante observar que essa substituição nem sempre é trivial e exige tanto adaptações na construção, para refletir a pseudoaleatoriedade fraca de uma PUF-wPRF [2], quanto modificações nos argumentos formais de segurança. A primeira e quarta funções de rodada consistem em uma função de *hash* universal em ambos os casos.

4.2 Proposta de autenticação

Nesta Seção será apresentada a proposta de um autenticador baseado em PUFs e função de *hash* universal combinados com um MAC clássico de tamanho fixo. Em aplicações de cifração, a confidencialidade não é o único serviço de segurança desejável, visto que a simples cifração não impede um adversário ativo de manipular texto cifrado com o objetivo de causar alterações maliciosas no texto claro resultante da decifração ou corrupção do conteúdo armazenado sem possibilidade de detecção posterior.

Para que essa detecção seja possível, utiliza-se tipicamente construções para MAC, o que permite alcançar a mais alta noção de segurança, IND-CCA2. A combinação de um esquema de cifração com IND-CPA com um autenticador inforjável contra ataques de mensagem escolhida, a partir de construções genéricas (por exemplo, [5]), permite atingir a noção de segurança IND-CCA2.

É importante observar que cifração autenticada de discos, quando realizada na granularidade natural de bloco, apresenta limitações intrínsecas do ponto de vista de autenticação, visto que um atacante ativo pode sempre substituir um bloco cifrado por um bloco autêntico anteriormente escrito sem ser detectado. A proposta de autenticação aqui apresentada se concentra apenas em detectar tentativas maliciosas de se alterar blocos cifrados do disco que não reutilizam blocos autênticos anteriores.

Em [11], Rogaway *et al.* propõem uma construção de MAC seguro utilizando funções de *hash* universal e PRFs. O MAC proposto a seguir é baseado nessa construção com a diferença que, ao invés de utilizar PRFs, emprega-se uma PUF-wPRF. Vale ressaltar que, como a saída da cifra proposta possui comprimento fixo (saída da quarta função de rodada) e a entrada do MAC tem comprimento fixo, a construção do MAC se torna muito mais simples.

Definição 4.2.1. Seja uma família de funções de *hash* universal $H = \{h : \{0,1\}^m \rightarrow \{0,1\}^m\}$, uma família de PUF-wPRF $\mathcal{V} = \{\{0,1\}^m \rightarrow \{0,1\}^m \times \{0,1\}^*\}$ e um vetor de inicialização $\sigma \in \{0,1\}^m$. O código de autenticação de mensagem FMAC(H, $\mathcal{V}$) = TAG, onde a TAG é composta pelo autenticador $\mathcal{T}$ e pela informação auxiliar ω conforme

a Figura 5.6, é definido pelos algoritmos de geração de parâmetros, cálculo do MAC e verificação como a seguir, respectivamente:

- *GenP*: escolher $\sigma \xleftarrow{R} \{0,1\}^m$ uniformemente aleatório e $h \xleftarrow{R} H$;
- *Mac*: ao receber como entrada $\sigma \in \{0,1\}^m$, $h \in H$ e a mensagem M, calcular:

$$\mathcal{T} = Mac(M) = f(h(M) \oplus \sigma).$$

- *Vrfy*: ao receber a mensagem M, $h \in H$ e o autenticador $\mathcal{T}$, produz a saída 1 se e somente se:

$$\mathcal{T} \stackrel{?}{=} f(h(M)) \oplus \sigma).$$

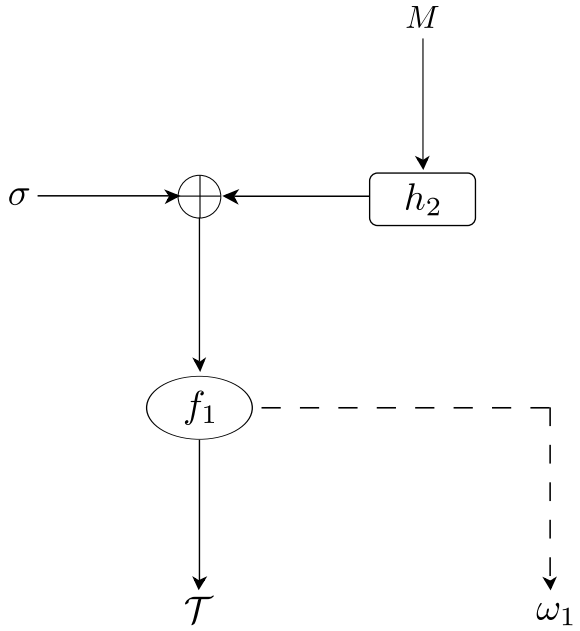


Figura 4.8: MAC utilizando função de *hash* universal e PUF-wPRF.

4.2.1 Análise de segurança

Nesta Seção será demonstrada a prova de segurança para o autenticador. Fornecer o argumento formal de segurança da construção de MAC envolve demonstrar que a mesma é inforjavél contra ataque adaptativo de mensagem escolhida.

Segurança contra ataque adaptativo de mensagem escolhida

Teorema 4.2.1. *Sejam* $H = \{h : \{0,1\}^* \to \{0,1\}^m\}$ *uma família de funções de hash* ϵ*-universal e* $\mathcal{V} = \{\{0,1\}^m \to \{0,1\}^m \times \{0,1\}^*\}$ *uma família de PUF-wPRFs. A vantagem*

de um adversário $\mathcal{A}$ que realiza até q_{prf} consultas é no máximo:

$$\Pr[\textit{Mac-forge}_{\mathcal{A}}(m) = 1] \leq q_{prf} \times \lambda(n) + \frac{q_{prf}^2}{2^m},$$

com $\lambda(n)$ uma função desprezível do parâmetro de segurança n.

Demonstração. Em [41] é apresentada uma construção segura de MAC a partir de uma PRF F_k proposta por [31]. A redução de segurança é obtida ao considerar-se primeiramente um MAC calculado como uma função verdadeiramente aleatória f' e com isso observar que o adversário não ganha vantagem, além da desprezível, de forjar, em tempo polinomial, um MAC válido para uma mensagem de sua escolha, ao se substituir f' por um MAC calculado com F_k. Neste caso, a probabilidade do adversário construir um MAC válido continua desprezível.

Conforme a formulação de PUF-wPRFs apresentada em [2] e na Seção 3.3, uma wPRF possui o mesmo comportamento de uma PRF se suas entradas são escolhidas de maneira uniformemente aleatória. Ou seja, a vantagem de se distinguir entre uma PRF e uma wPRF em tempo polinomial é limitada por uma função desprezível $\lambda(n)$. Assim, por transitividade, uma wPRF é indistinguível de uma função ideal em tempo polinomial, desde que suas entradas sejam uniformemente aleatórias. Como o adversário pode realizar até q_{prf} consultas, a vantagem do adversário é de $q_{prf} \times \lambda(n)$, é uma função desprezível.

Para que as entradas da wPRF sejam uniformemente aleatórias, temos que considerar a probabilidade de colisão de $h(M) \oplus \sigma$ que é no máximo $\frac{q^2}{2^m}$ [41, Lema A.9]. Assim temos que a probabilidade de um adversário construir um MAC válido construído a partir de uma PUF-wPRF é no máximo $q_{prf} \times \lambda(n) + \frac{q^2}{2^m}$. □

Fica, portanto, demonstrada a segurança da construção proposta para autenticação, a partir das propriedades de segurança fornecidas pela formalização da PUF.

Capítulo 5

Proposta de protocolo combinando PUF com protocolos PAKE

Neste Capítulo será apresentada a proposta de um protocolo seguro combinando PUFs com protocolos PAKE. O protocolo oferece autenticação mútua entre cliente e servidor e estabelece uma chave de sessão entre as partes autenticadas, características importantes que não foram encontradas simultaneamente na literatura de autenticação baseada em PUF.

A combinação aprimora o estado da arte, garantindo que a chave de sessão esteja apenas disponível para detentores legítimos da PUF, sem a possibilidade de vazamento de segredos armazenados explicitamente e também garante proteção contra ataques *offline* de dicionário sobre a senha de autenticação. É proposta uma adaptação do protocolo para que ele possa ter suporte a senha de pânico em situações de emergência, onde o cliente é coagido a entregar a PUF e revelar a sua senha.

O Capítulo está dividido da seguinte forma: na Seções 5.1 e 5.2 será descrito todo o protocolo, que é composto por duas fases: a de inscrição e a de autenticação, bem como sua análise de segurança tanto heurística quanto formal, respectivamente. Na Seção 5.3, é mostrado o protocolo da Seção 5.1 com algumas modificações para que ele possa ter suporte a senha de pânico. E por fim, na Seção 5.4, é feita uma comparação do protocolo proposto com outros da literatura, bem como os ataques realizados.

5.1 O protocolo

O protocolo demonstrado a seguir utiliza a combinação de uma PUF com um protocolo PAKE. A ideia é utilizar a saída da PUF para ser o segredo compartilhado requisitado no PAKE. Na fase de inscrição o servidor utiliza uma PUF para gerar um par de desafio-resposta da PUF que será utilizado na fase de autenticação. Nesta fase o servidor gera

o desafio da PUF d composto por um desafio[1] c concatenado com a senha do usuário (pwd), e então armazena esta saída juntamente com c. Observe que outros fatores de autenticação podem ser concatenados a c como entrada da PUF, como sugerido em [28].

A fase de autenticação é composta de três etapas. Na primeira etapa é estabelecido o segredo compartilhado (que é a saída da PUF para desafio da PUF d) entre o cliente e o servidor. Na segunda etapa é executado um protocolo PAKE que possua a propriedade de *freshness* para o estabelecimento de uma chave de sessão. E por fim, na terceira etapa é executado o passo de confirmação de chave para uma autenticação mútua, conforme a transformação genérica estudada em [6].

5.1.1 Fase de inscrição

A fase de inscrição é uma etapa onde o servidor vai gerar e armazenar o desafio da PUF para o cliente e sua respectiva resposta, e é considerada que ela seja executada através de um canal seguro. A fase é descrita abaixo e ilustrada na Figura 5.1.

Fase de Inscrição

1 Servidor $\mathcal{S}$ gera um desafio c e o envia para o usuário $\mathcal{U}$;

2 Usuário $\mathcal{U}$ envia $H(c||pwd)$ para o dispositivo $\mathcal{D}$, onde pwd é a senha do usuário $\mathcal{U}$ e H é uma função de *hash* criptográfico;

3 Dispositivo $\mathcal{D}$ calcula o desafio da PUF $d = H(c||pwd)$ e executa o procedimento Gen sobre esse valor e obtém (s, ω) como resposta. O dispositivo $\mathcal{D}$ envia (s, ω) para o usuário $\mathcal{U}$;

4 Usuário $\mathcal{U}$ envia (s, ω) para o servidor $\mathcal{S}$ que armazena esse valor juntamente com c.

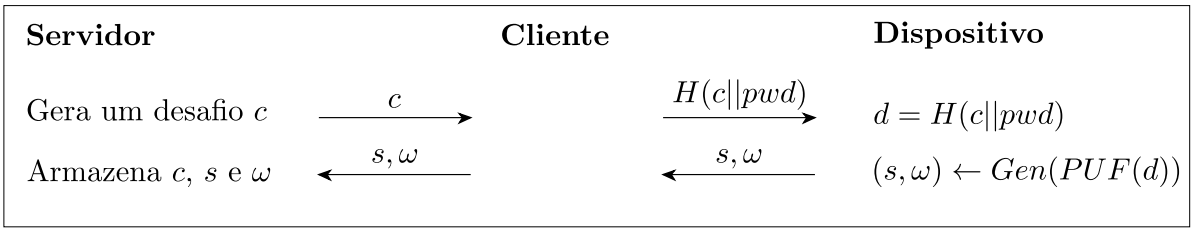


Figura 5.1: Fase de inscrição.

[1] A diferença entre o desafio aqui utilizado e um desafio em um contexto mais geral (*nonce*) é que o desafio aqui pode ser fixo em todas as execuções do protocolo, enquanto o *nonce* é sempre diferente (aleatório).

5.1.2 Fase de acordo autenticado de chaves

Nesta fase de autenticação e geração de chave de sessão ilustrada na Figura 5.2, a primeira etapa, que estabelece o segredo compartilhado K, é padrão para a combinação de PAKE com PUF. A segunda etapa é flexível, onde o protocolo utilizado pode ser qualquer PAKE ou similar, como por exemplo qualquer protocolo dentre [7, 16, 39] que possua a propriedade de *freshness*.

Na Figura 5.2 o protocolo utilizado é o proposto por Bresson *et al.* [16], com algumas adaptações, que apresenta um protocolo de acordo de chave com uma cifração (do inglês, *One-Encryption Key Exchange – OEKE*), o qual é uma variante "simplificada" do protocolo PAKE proposto em [6]. A terceira etapa também é flexível, onde são possíveis outros tipos de protocolos para a confirmação de chave, como discutido em [39]. As funções H_0 e H_1 representam funções de *hash* criptográficas que podem ser construídas a partir de H com prefixos, ε e Δ as funções de cifração e decifração de uma cifra de bloco segura, respectivamente.

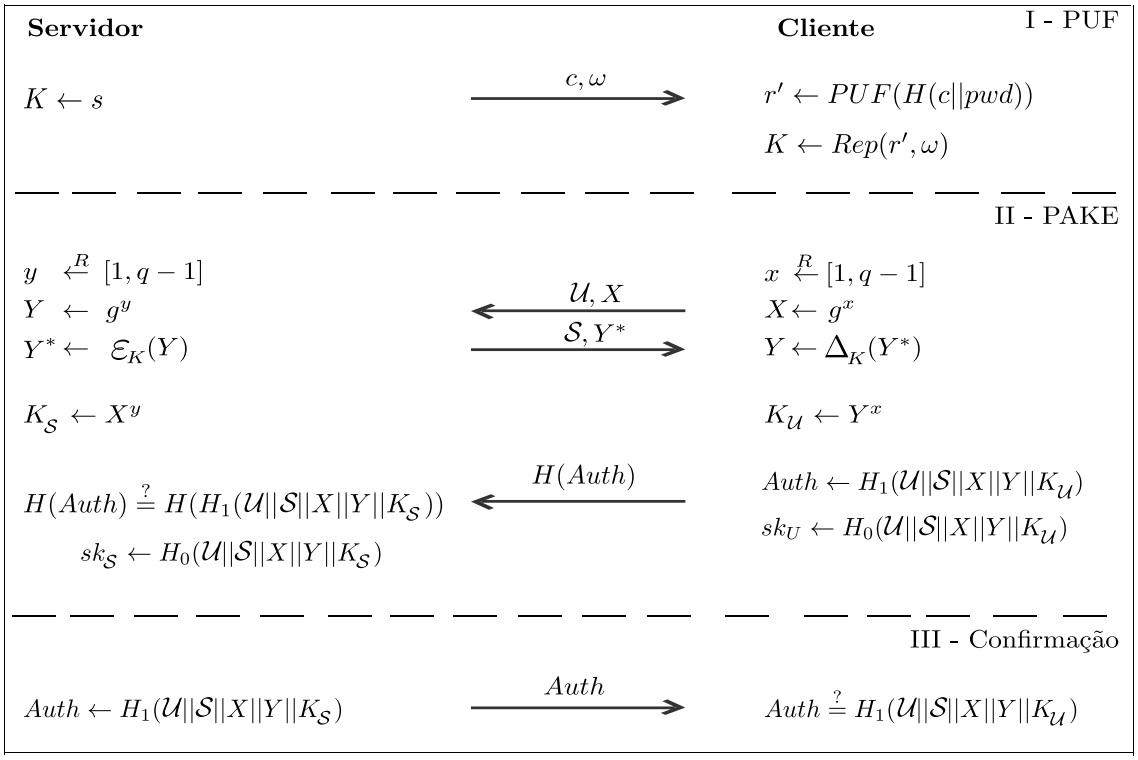


Figura 5.2: Fase de autenticação.

5.2 Segurança

Em qualquer instante de tempo, o adversário não tem acesso simultâneo a todos os fatores de autenticação, caso contrário, torna-se trivial um ataque. Assim, o adversário é incapaz de capturar diretamente o segredo K. Como resultado, em cada tentativa de autenticação entre o cliente e o servidor pode-se usar o mesmo par de desafio-resposta da PUF, cuja resposta deve ser armazenada de forma segura pelo servidor. Portanto, em cada comunicação, o segredo compartilhado entre o cliente e o servidor é o mesmo.

Essa decisão de projeto não parece ter impacto sobre a segurança do protocolo, uma vez que o PAKE proporciona *freshness*, mesmo para uma senha fixa, mas reduz significativamente a exigência de armazenamento do servidor. O desafio constante c pode também ser armazenado na memória do dispositivo durante a Fase de Inscrição afim de salvar a primeira rodada de comunicação. No entanto, uma corrupção da memória tornaria o dispositivo inutilizável.

Na prática, a fim de evitar vazamento progressivo do valor de K por interações contínuas, a restrição pode ser removida por randomizar o desafio c do protocolo à custa de reduções de segurança mais complexas e maior exigência de armazenamento do servidor para o desafio da PUF $d_i = H(c_i||pwd)$ e as respostas e informações auxiliares correspondentes. Isto é diferente de outros trabalhos que assumem a existência de uma forte PUF ligada inseparavelmente à um *chip* capaz de realizar cálculos (I-PUF) [76], onde a comunicação entre PUF e *chip* é inacessível para um atacante e, portanto, não pode ser adulterada.

Assim, considera-se que o adversário $\mathcal{A}$ tem acesso temporário a PUF, quando ele pode executar um número limitado de consultas para construir um conjunto de pares desafio-resposta da PUF. Quando o número de tentativas for ultrapassado, o servidor pode impor um limite de tempo que impede um ataque de busca exaustiva na senha, sem bloquear completamente o cliente/servidor. O adversário também tem acesso ao tráfego de rede entre o cliente e o servidor, correspondente às tentativas de autenticações bem-sucedidas.

5.2.1 Análise heurística

- O adversário não tem acesso a PUF e não tem acesso a senha

  Neste cenário o adversário não tem acesso a PUF em nenhum momento e não conhece a senha (pwd) do cliente. O adversário apenas possui tráfego antigo de comunicações honestas entre o cliente e o servidor.

  - **Cliente**: uma vez que o PAKE é seguro, possuindo a propriedade de *freshness*, a probabilidade de um adversário personificar o cliente é a mesma probabili-

dade do adversário conseguir descobrir K. Para isso, o adversário teria de ser capaz de adivinhar qual é a resposta gerada pela PUF, o que acontece com vantagem desprezível, conforme a Seção 3.1. Assim, a probabilidade do adversário conseguir personificar o cliente com sucesso é desprezível.

  - **Servidor**: a vantagem do adversário de personificar com sucesso o servidor é igual ao do cliente, ou seja, o adversário também teria de ser capaz de adivinhar o valor de K, o que acontece com probabilidade $Adv_{\mathcal{A}}^{PUF}(n)$ desprezível. Assim, a probabilidade do adversário personificar o servidor com sucesso é desprezível.

- O adversário tem acesso a PUF e não tem acesso a senha

  Este cenário onde o adversário tem acesso a PUF por um determinado tempo, mas não tem conhecimento da senha (pwd) do cliente é o cenário de autenticação mais realista. Por exemplo, um estabelecimento comercial onde um funcionário possui acesso temporário ao cartão de crédito do cliente para faturamento. Nele, por um curto período de tempo, o cliente não tem total controle sobre o dispositivo e o adversário pode ler vários dados da memória do cartão e obter informações importantes. Este cenário é igual ao caso de qualquer protocolo PAKE seguro que possua a propriedade de *freshness*, onde a segurança está em uma senha menos complexa que uma chave simétrica;

  - **Cliente**: neste cenário, para o adversário personificar o cliente com sucesso ele teria que adivinhar qual é a senha do usuário. Como o adversário está de posse da PUF e observou pelo menos o tráfego de uma comunicação honesta bem sucedida entre o cliente e o servidor para saber qual desafio c e ω são utilizados, ele pode calcular um número limitado (lembrando que isso depende do tempo de posse da PUF e da quantidade de tentativas permitidas pelo dispositivo) de d_i desafios/respostas da PUF, para $d_i = H(c||pwd_i)$. Assim o adversário vai gerar um conjunto $\mathcal{K}$ de valores K'_i, onde $K'_i = Rep(PUF(H(c||pwd_i), \omega))$. Então, após o servidor enviar o desafio c e ω, o adversário escolhe um $K' \in \mathcal{K}$ e executa o protocolo com o servidor. Se o protocolo for encerrado com sucesso, o adversário conseguiu personificar o cliente e termina descobrindo qual é a senha do usuário; caso contrário, ele aborta o protocolo e o inicia novamente com outro valor de K'. Neste caso a probabilidade[2,3] do adversário personificar o servidor com sucesso é $\Pr[K = K' \wedge K' \in \mathcal{K}]$ que é considerada

[2]Lembrando que a distribuição de probabilidade de senhas é geralmente longe de ser uniforme, devido ao fato de algumas senhas serem comumente escolhidas e estarem propícias a ataques de dicionários

[3]Vale ressaltar que o valor de $|\mathcal{K}|$ pode ser limitado por algum mecanismo que impeça, por exemplo, mais de 3 tentativas de comunicação que não foram bem sucedidas.

desprezível se a distribuição de senhas é próxima de uniforme ou se existir um limite no número de tentativas não bem sucedidas de autenticação. Observe que a utilização do PAKE exige que cada tentativa de autenticação envolva interação entre cliente e servidor.

  - **Servidor**: a vantagem do adversário de personificar com sucesso o servidor é igual ao do cliente, ou seja, o adversário também teria de ser capaz de adivinhar qual é a senha do usuário, que acontece com probabilidade $\Pr[K = K' \wedge K' \in \mathcal{K}]$.

- O adversário não tem acesso a PUF e tem acesso a senha

  Neste caso o adversário não tem acesso a PUF, mas conhece a senha do cliente.

  - **Cliente**: para o adversário personificar o cliente, ele precisa adivinhar a saída K da PUF, que tem como entrada o c enviado pelo servidor concatenado com pwd. A probabilidade de o adversário adivinhar o valor de K é desprezível, conforme Seção 3.1. Assim sendo, a vantagem do adversário personificar o cliente com sucesso também é desprezível.
  - **Servidor**: para o adversário personificar o servidor, ele teria que adivinhar o valor de K de maneira análoga ao item anterior. Ou seja, a probabilidade que o adversário adivinhe o valor de K é desprezível. Assim, a vantagem do adversário personificar o servidor com sucesso também é desprezível.

- O adversário tem acesso a PUF e tem acesso a senha

  Neste caso o adversário tem acesso a PUF por um determinado tempo (por exemplo, o garçom que pega o cartão de crédito do cliente) e também tem conhecimento da senha (pwd) do cliente.

  - **Cliente**

    Este caso é trivial, pois o adversário estando de posse da PUF e conhecendo a senha pwd, ele será capaz de personificar o usuário com 100% de probabilidade, pois quando o servidor enviar c para o cliente (adversário), o adversário poderá calcular a entrada da PUF, pois ele conhece a senha e também obterá a resposta, pois, estará de posse da PUF e assim conhecerá K.
  - **Servidor**

    Estando de posse da PUF e da senha, o adversário não teria motivos para personificar o servidor.

5.2.2 Análise formal

Um protocolo para acordo de chave autenticada precisa satisfazer duas noções de segurança: a segurança semântica (também chamada de propriedade *freshness* nesse contexto) e a propriedade de autenticação. A primeira noção de segurança garante que o protocolo produz como resultado uma chave criptográfica compartilhada. A segunda noção de segurança garante que a chave criptográfica compartilhada é obtida apenas para as partes que satisfizerem certos requisitos de autenticação.

Teorema 5.2.1. *O protocolo combinado PUF+PAKE utilizando um protocolo PAKE semanticamente seguro continua semanticamente seguro sob as premissas do protocolo PAKE e supondo a imprevisibilidade da PUF.*

Demonstração. A prova é trivial. Um protocolo PAKE é semanticamente seguro dado um conjunto de premissas computacionais e a escolha uniformemente aleatória de uma senha compartilhada. Uma PUF modelada como função imprevisível satisfaz esse requisito. Logo, a construção combinada apenas transfere a segurança semântica e demais propriedades de segurança do protocolo PAKE para o protocolo PUF+PAKE, sob as mesmas premissas computacionais que garantem a segurança semântica do protocolo PAKE isolado. □

Teorema 5.2.2. *Um adversário polinomial com acesso à PUF (com parâmetro de segurança ℓ) possui probabilidade desprezível de sucesso no protocolo de autenticação PUF+PAKE para um cadastro prévio de usuário, supondo que H é um oráculo aleatório, o protocolo PAKE satisfaça a propriedade de autenticação e senhas são escolhidas de um conjunto suficientemente grande para a probabilidade de acerto ser também desprezível.*

Demonstração. A redução de segurança é uma simples adaptação do Teorema 5.2.1 em [28]. Supondo que um adversário $\mathcal{A}$ com probabilidade de sucesso não-desprezível existe para a propriedade de autenticação, vamos construir um adversário $\mathcal{B}$ para a indistinguibilidade da PUF que utiliza $\mathcal{A}$ como caixa preta. O adversário $\mathcal{B}$ escolhe um desafio aleatório c e uma senha aleatória pwd, calcula $d = H(c||pwd)$ e escolhe d como seu desafio da PUF para o jogo da indistinguibilidade. O adversário $\mathcal{B}$ recebe um par (r_b, ω) com probabilidade $\frac{1}{2}$ determinada pelo bit b para $r_0 = Rep(PUF(d), \omega)$, ou o valor escolhido aleatoriamente r_1 caso contrário; e instancia o adversário $\mathcal{A}$ com os valores (c, ω), fornecendo oráculos para H e PUF.

Para simular H, $\mathcal{B}$ cria um conjunto de tuplas inicializado como $H_S = (c||pwd, h)$ para um valor h escolhido aleatoriamente. Quando $\mathcal{A}$ consulta o oráculo com entrada x, $\mathcal{B}$ verifica se um par (x, y) já existe em H_S e retorna y; caso contrário adiciona (d', h') ao conjunto H_S e retorna h' como resultado, para um valor h' escolhido aleatoriamente.

Para simular a PUF para uma consulta (d', ω'), $\mathcal{B}$ verifica se $d = d'$ e retorna FALHA em caso positivo. Caso contrário, $\mathcal{B}$ retorna $(r', \omega') = Gen(d')$ como resultado. Desta forma, $\mathcal{A}$ possui visão indistinguível do protocolo real e eventualmente produz uma prova de autenticação a partir da chave possivelmente compartilhada. Se esta prova de autenticação for correta, $\mathcal{B}$ retorna 0 como resultado, ou um bit aleatório b' caso contrário.

Agora resta analisar a probabilidade $\Pr[b = b']$. Seja F o evento em que $\mathcal{B}$ retorna FALHA. Este evento ocorre com probabilidade apenas desprezível, pois exige que o adversário $\mathcal{A}$ acerte a senha pwd ou o desafio da PUF d. Pode-se dividir o caso restante $\Pr[b = b'|\overline{F}]$ nos dois valores de b:

$$\Pr[b = b'|\overline{F}] = \frac{1}{2}\Pr[b = b'|\overline{F}, b = 0] + \frac{1}{2}\Pr[b = b'|\overline{F}, b = 1].$$

Agora seja G o evento em que $\mathcal{A}$ produz uma prova correta de autenticação e a probabilidade associada ao caso $b = 1$:

$$\Pr[b = b'|\overline{F}, b = 1] = \Pr[b = b'|\overline{F}, b = 1, G]\Pr[G|\overline{F}, b = 1] + \Pr[b = b'|\overline{F}, b = 1, \overline{G}]\Pr[\overline{G}|\overline{F}, b = 1].$$

Temos que $\Pr[b = b'|\overline{F}, b = 1, G] = 0$, pois $b' = 0$ na ocasião do evento G; $\Pr[b = b'|\overline{F}, b = 1, \overline{G}] = \frac{1}{2}$, pois $b = 1$ ocorre com 50% de chance na ocasião do evento $\overline{G}$; e $\Pr[G|\overline{F}, b = 1]$ é uma função $\lambda(\ell)$ desprezível pela propriedade de autenticação do protocolo PAKE. Assim, para $\mathcal{B}$ possuir probabilidade de sucesso não-desprezível:

$$\Pr[b = b'|\overline{F}, b = 1] > \frac{1}{2} - \lambda(\ell).$$

O caso $b = 0$ é similar:

$$\Pr[b = b'|\overline{F}, b = 0] = \Pr[b = b'|\overline{F}, b = 0, G]\Pr[G|\overline{F}, b = 0] + \Pr[b = b'|\overline{F}, b = 0, \overline{G}]\Pr[\overline{G}|\overline{F}, b = 0].$$

Agora, $\Pr[b = b'|\overline{F}, b = 0, G] = 1, \Pr[b = b'|\overline{F}, b = 0, \overline{G}] = \frac{1}{2}$ e $\Pr[G|\overline{F}, b = 0] > \frac{1}{f(\ell)}$ para algum polinômio f, sendo não-desprezível pela premissa da probabilidade de sucesso do adversário $\mathcal{A}$. Logo:

$$\Pr[b = b'|\overline{F}, b = 0] > \frac{1}{f(\ell)} + \frac{1}{2} \cdot \left(1 - \frac{1}{f(\ell)}\right) = \frac{1}{2} + \frac{1}{2f(\ell)}.$$

Substituindo, temos:

$$\Pr[b = b'|\overline{F}] > \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2f(\ell)}\right) + \frac{1}{2}\left(\frac{1}{2} - \lambda(\ell)\right)$$

Como temos $\Pr[b = b'|\overline{F}] - \frac{1}{2}$ não-desprezível, $\Pr[b = b'] - \frac{1}{2}$ é também não-desprezível e o atacante $\mathcal{A}$ precisa existir, contrariando a hipótese de que o protocolo PAKE é seguro.

□

5.3 Protocolo de emergência

Senhas de pânico são mecanismos que permitem que os usuários usem um tipo especial de senha, neste sentido chamado de senha de pânico, para sinalizar para o servidor (ou qualquer outra parte comunicante) que sua senha está sendo inserida como resultado de uma ação coerciva [20]. Elas também são conhecidas com senhas ou códigos de socorro na literatura.

Devido ao uso de tais senhas em cenários militares e de inteligência, não é possível determinar exatamente a dimensão dos estudos sobre o seu uso, pois podem existir vários trabalhos que são mantidos em sigilo. Assim, a literatura não é muito extensa.

Existem sistemas que utilizam senhas de pânico que são patenteados. Entre eles incluem um sistema onde caixas eletrônicos exibem uma lista de palavras a partir do qual o usuário pode escolher um opção predefinida para autenticar-se normal ou qualquer das outras palavras para sinalizar que está sendo coagido e, assim, limitar a quantidade disponível de dinheiro para saque [69].

Outro exemplo de senhas de pânico, é um dispositivo onde o usuário tenta se autenticar em uma rede e quando é coagido a revelar/digitar sua senha, ele insere a senha de pânico que retransmite o usuário para um banco de dados diferente do normal, de uma forma aparentemente comum, que é desconhecida pelo atacante [3].

Um exemplo mais popular do uso de senhas de pânico foi empregado em versões antigas do dispositivo RSA SecurID [63], cujas versões recentes não possuem mais essa característica. Este dispositivo tem como finalidade executar uma autenticação de dois fatores para um usuário acessar um recurso de rede como, por exemplo, um *firewall*, onde para o usuário acessa-ló pode ser necessário tanto um número de identificação pessoal quanto um número que estará sendo exibido naquele momento no seu *token* RSA SecurID.

O protocolo apresentado na Figura 5.2 pode ser facilmente adaptado para suportar senhas de pânico [20]. Utilizaremos a mesma técnica apresentada em [28], onde existem duas senhas válidas para cada usuário, ambas com um prefixo compartilhado (pwd_1) e sufixos distintos que indicam situação normal (pwd_2) e situação de emergência (pwd_3) e tendo pwd^* sendo o prefixo compartilhado pwd_1 com o sufixo pwd_2 ou pwd_3. A fase de inscrição sofre algumas adaptações e é descrita abaixo e ilustrada na Figura 5.3.

Fase de inscrição:

1 Servidor $\mathcal{S}$ gera um desafio c e envia para o usuário $\mathcal{U}$;

2 Usuário $\mathcal{U}$ envia $H(c||pwd_1)$, pwd_2 e pwd_3 para o dispositivo $\mathcal{D}$, onde $pwd_i's$ são as 3 partes da senha do usuário $\mathcal{U}$ e H é uma função de *hash* criptográfico;

3 Dispositivo $\mathcal{D}$ calcula o desafio da PUF $d = H(c||pwd_1)$ e executa o protocolo Gen sobre esse valor e obtém (s, ω) como resposta. O dispositivo $\mathcal{D}$ envia $(H(s||pwd_2), H(s||pwd_3), \omega)$ para o usuário $\mathcal{U}$;

4 Usuário $\mathcal{U}$ envia $(H(s||pwd_2), H(s||pwd_3), \omega)$ para o servidor $\mathcal{S}$ que armazena esse valor juntamente com c.

Servidor		**Cliente**		**Dispositivo**
Gera um desafio c	$\xrightarrow{c}$		$\xrightarrow{H(c\|\|pwd_1),\ pwd_2 \text{ e } pwd_3}$	$d = H(c\|\|pwd_1)$
Armazena $H(s\|\|pwd_2)$, $H(s\|\|pwd_3), \omega$ e c	$\xleftarrow{H(s\|\|pwd_2),\ H(s\|\|pwd_3), \omega}$		$\xleftarrow{H(s\|\|pwd_2),\ H(s\|\|pwd_3), \omega}$	$(s, \omega) \leftarrow Gen(PUF(d))$

Figura 5.3: Fase de inscrição com suporte a senhas de pânico.

A análise formal de segurança da variante será omitida, visto que apenas modificações simples são necessárias no protocolo. Na fase de autenticação, serão invertidos os papéis de cliente e servidor no PAKE ilustrativo, o que não provoca impacto de segurança no PAKE quando a transformação genérica de autenticação mútua [6] é aplicada. Esta inversão é necessária para transferir o passo de cifração para o cliente, que enviará a cifração sob uma de duas chaves possíveis, novamente indicando situação normal ou situação de emergência. O servidor pode, então, decifrar a mensagem recebida e verificar para qual tipo de situação a próxima mensagem do cliente corresponde.

Quando o servidor detecta uma situação de emergência, ele executa o protocolo normalmente, de modo que o adversário não perceba que a senha de pânico foi utilizada, ou seja, indistinguibilidade entre a execução do protocolo com a senha normal e a senha de pânico. Neste caso o servidor pode emitir algum sinal para a policia, avisando que está acontecendo algo de suspeito, limitar a quantidade de dinheiro disponível para saque e até mesmo marcar quais notas estão sendo retiradas.

No mais, quando não houver situação de emergência, o servidor prossegue normalmente e envia sua mensagem para autenticação mútua. O protocolo resultante encontra-se descrito na Figura 5.4. Naturalmente, quando a notificação de emergência é desejável, todos os clientes sob qualquer situação devem utilizar o protocolo abaixo, para que instâncias normais do protocolo possuam o mesmo padrão de comunicação que situações de emergência.

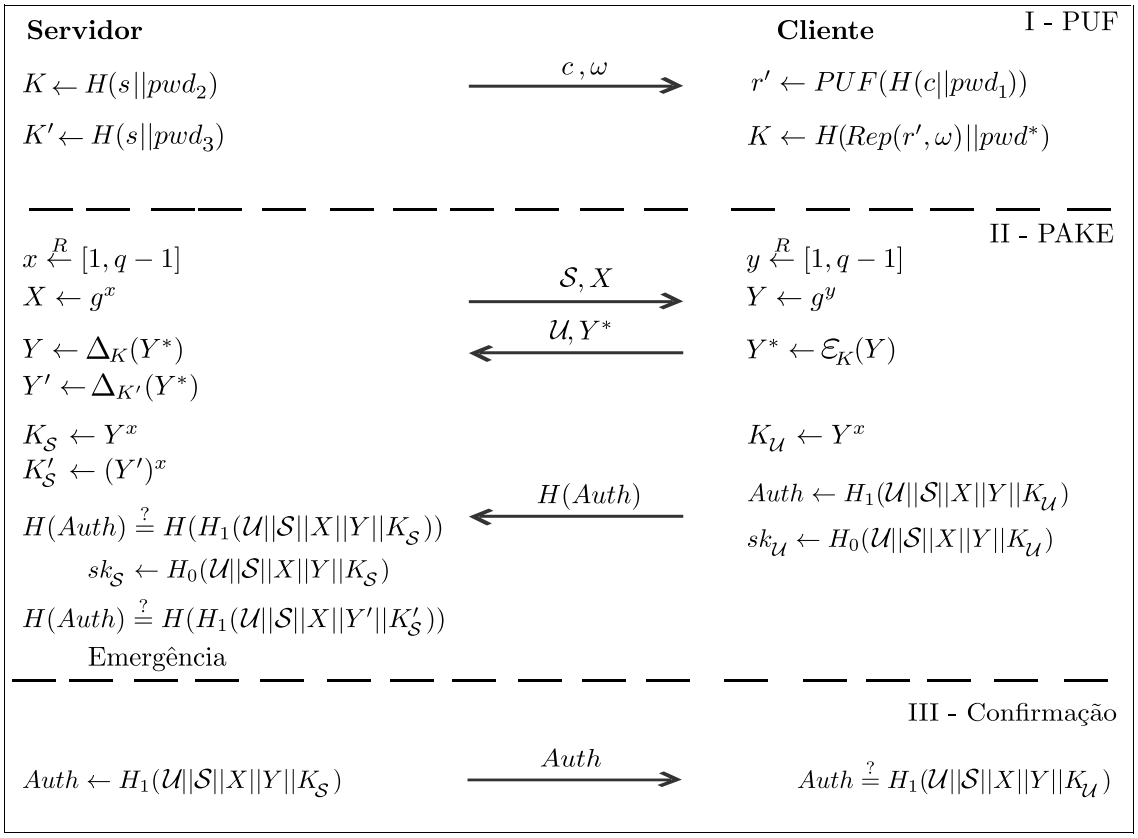


Figura 5.4: Fase de autenticação com suporte a senhas de pânico.

5.4 Ataques propostos contra protocolos da literatura

Nesta Seção são apresentados os trabalhos propostos em [28] e [78]. Para [28], é demonstrado um ataque de dicionário contra o protocolo, que depende apenas da posse temporária da PUF e observação de um único traço de comunicação cliente/servidor. É também apresentado um ataque de personificação do servidor contra o protocolo exibido em [18], que demonstra que a correção sugerida pelos autores para o protocolo proposto [78] não é suficiente. No decorrer desta Seção, é conservada a maioria das notações usadas nos trabalhos relacionados para compatibilidade.

5.4.1 Autenticação robusta usando PUFs

Em [28], PUFs são utilizadas na construção de um protocolo apenas de autenticação (Figura 5.5) sem o estabelecimento de uma chave de sessão, onde um usuário se autentica usando uma I-PUF emitida pelo banco e uma senha adicional. Levando em consideração os cenários mostrados na Seção 5.2.1 esse protocolo tem as mesmas propriedades de segurança que o protocolo proposto, salvo o cenário em que o adversário possui acesso temporário à PUF, mas desconhece a senha do usuário.

No protocolo PUF+PAKE, o adversário personifica com sucesso o cliente ou o servidor sobre esse cenário com probabilidade $\Pr[K' \in \mathcal{K} \wedge K = K']$ para cada tentativa de autenticação. Será mostrado que a personificação do servidor em [28] tem uma maior probabilidade (equivalente a $Pr[K = \mathcal{K}] = \sum_{K' \in \mathcal{K}} \Pr[K = K']$).

Como o desafio c é fixo em [28], o desafio da PUF $d = H(H(c||pwd), g, P)$ também o é, para alguma informação auxiliar[4] P. De posse da PUF, o adversário calcula um conjunto $\mathcal{R}$ de valores g^{r_i} correspondentes para desafios da PUF $d_i = H(H(c||pwd_i), g, P)$, onde pwd_i é uma candidato a senha (ataque de dicionário). Observe que pela hipótese de uma I-PUF não é permitido a um atacante obter a resposta r^i diretamente, mas sim o resultado do cálculo g^{r_i}.

Posteriormente, o adversário repete para o cliente o desafio c, a descrição do grupo $\langle G_q \rangle$, q, a informação auxiliar P e um *nonce* N observado em uma comunicação honesta anterior entre cliente e servidor. O usuário então envia $(H(c||pwd), \langle G_q \rangle, q, P, N)$ para o dispositivo que calcula $d = H(H(c||pwd), g, P)$ e executa o procedimento Rep para obter r. O dispositivo escolhe um valor aleatório $v \in \mathbb{Z}_q$ e calcula $t = g^v$.

Seguindo o protocolo, o dispositivo calcula $c' = H(g, g^r, t, N)$ e $w = v - c'r \bmod q$, e envia c' e w para o cliente, que envia esses valores para o adversário. O adversário então calcula um conjunto $\mathfrak{T}$ de valores t'_i, onde $t'_i = g^w g^{r_i c'}$, pois ele conhece o valor de g, w, c', e possui um conjunto $\mathcal{R}$ de g^{r_i} candidatos. Então o adversário checa se $c' = H(g, g^{r_i}, t_i, N)$ para algum valor de i e aprende a senha do usuário se $pwd = pwd_i$.

A probabilidade do adversário personificar o servidor em [28] é maior do que a do protocolo proposto, pois o adversário apenas precisa personificar o servidor uma vez para verificar se a senha do usuário está incluída no conjunto de senhas candidatas consultadas através da PUF e o restante do trabalho pode ser feito *offline*. Já o PAKE no protocolo proposto restringe o adversário personificar o cliente/servidor para verificar uma única senha candidata por tentativa de autenticação, permitindo ainda o cliente ou o servidor monitorar o comportamento malicioso da outra parte.

Além deste ataque, é possível construir um outro onde uma fonte, funcionário do banco (agente interno) por exemplo, que rouba os registros de autenticação em [28], consegue recuperar o valor g^r para algum usuário e é capaz de montar um ataque *offline* da mesma maneira se, posteriormente ele obtém acesso ao dispositivo e consulta a PUF com vários desafios $d_i = H(H(c||pwd_i), g, P)$, na esperança de receber $g^{r_i} = g^r$ como resposta.

Esta observação contradiz diretamente uma afirmação de segurança indicada no Resumo e Introdução de [28], onde é afirmado que o protocolo é resistente contra ataques internos. Vale ressaltar que este ataque também pode ser feito no protocolo proposto com a recuperação de K por algum agente interno.

[4] Vale lembrar que ω é a notação utilizada como padrão, neste trabalho, para informação auxiliar.

Comparando o protocolo proposto com o apresentado em [28], ele fornece um *trade-off* de segurança interessante: enquanto o protocolo proposto fornece maior resistência contra ataques de dicionário, um ataque interno é mais fácil de montar, porque a resposta K da PUF é armazenada diretamente no servidor.

Inscrição:

1 Servidor $\mathcal{S}$ envia desafio c para o usuário $\mathcal{U}$ junto com a descrição de um grupo G_q, denotado por $\langle G_q \rangle$ e que deve consistir de um par (p, q) e seu gerador g.

2 Usuário $\mathcal{U}$ envia $H(c||pwd)$, $\langle G_q \rangle$, g onde pwd é a senha do usuário, para o dispositivo $\mathcal{D}$ para o protocolo Gen modificado.

3 Dispositivo $\mathcal{D}$ calcula o desafio $d = H(H(c||pwd), \langle G_q \rangle, g)$ e executa Gen sobre esse valor para obter uma resposta (r, P). $\mathcal{D}$ entao envia para o usuário (g^r, P).

4 Usuário entrega (g^r, P) para o servidor $\mathcal{S}$, que armazena essa informação junto com c, g, $\langle G_q \rangle$.

Autenticação:

1 Servidor $\mathcal{S}$ envia desafio c, $\langle G_q \rangle$, g, P e um *nonce* N para o usuário $\mathcal{U}$;

2 $\mathcal{U}$ envia $(H(c||pwd)$, $\langle G_q \rangle$, g, P, $N)$ para o dispositivo $\mathcal{D}$ para o protocolo Rep.

3 Dispositivo $\mathcal{D}$ calcula o desafio $d = H(H(c||pwd), g, P)$ e executa Rep sobre esse valor para obter a resposta r. $\mathcal{D}$ escolhe um valor aleatório $v \in \mathbb{Z}_q$ e calcula $t = g^v$. $\mathcal{D}$ então calcula $c' = H(g, g^r, t, N)$ e $w = v - c'r \ mod \ q$, e envia c', w para $\mathcal{U}$.

4 O usuário $\mathcal{U}$ envia esse valor para o servidor $\mathcal{S}$. $\mathcal{S}$ calcula $t' = g^w g^{rc'}$ e aceita a autenticação se $c' = H(g, g^r, t', N)$, caso contrário rejeita o valor.

Figura 5.5: Protocolo de autenticação utilizando PUF de [28].

5.4.2 Autenticação forte usando PUFs

Tuyls e Škorić [78] propõem um protocolo de autenticação com estabelecimento de chave baseado em PUFs, que é similar ao protocolo anterior, utilizado para autenticação entre o cliente e o servidor, com o incremento de estabelecimento de uma chave de sessão.

O protocolo é dividido em duas partes, sendo a primeira chamada de fase de inscrição na qual é atribuído um identificador ID_{PUF} para a PUF, é escolhida uma sequência aleatória rd, é também criado um conjunto de desafios $\mathcal{C}$, onde para cada desafio $c_i \in \mathcal{C}$ é gerada uma saída r_i e informação auxiliar ω_i (formando o conjunto $\mathcal{W}$) que são

entradas para o procedimento de reprodução[5] $G(r_i, \omega_i)$ para gerar o valor correspondente s_i (formando o conjunto $\mathcal{S}$).

Na memória do cartão, são armazenados ID_{PUF}, $n = 0$, $m = rd$ e, por fim, os valores ID_{PUF}, $n' = 0$, $m' = rd$ e $\{\mathcal{C}, \mathcal{W}, \mathcal{S}\}$ são armazenados no banco de dados do servidor. Já a segunda parte do protocolo funciona conforme descrito na Figura 5.6. No modelo de ataque desse protocolo, o adversário tem acesso apenas a PUF.

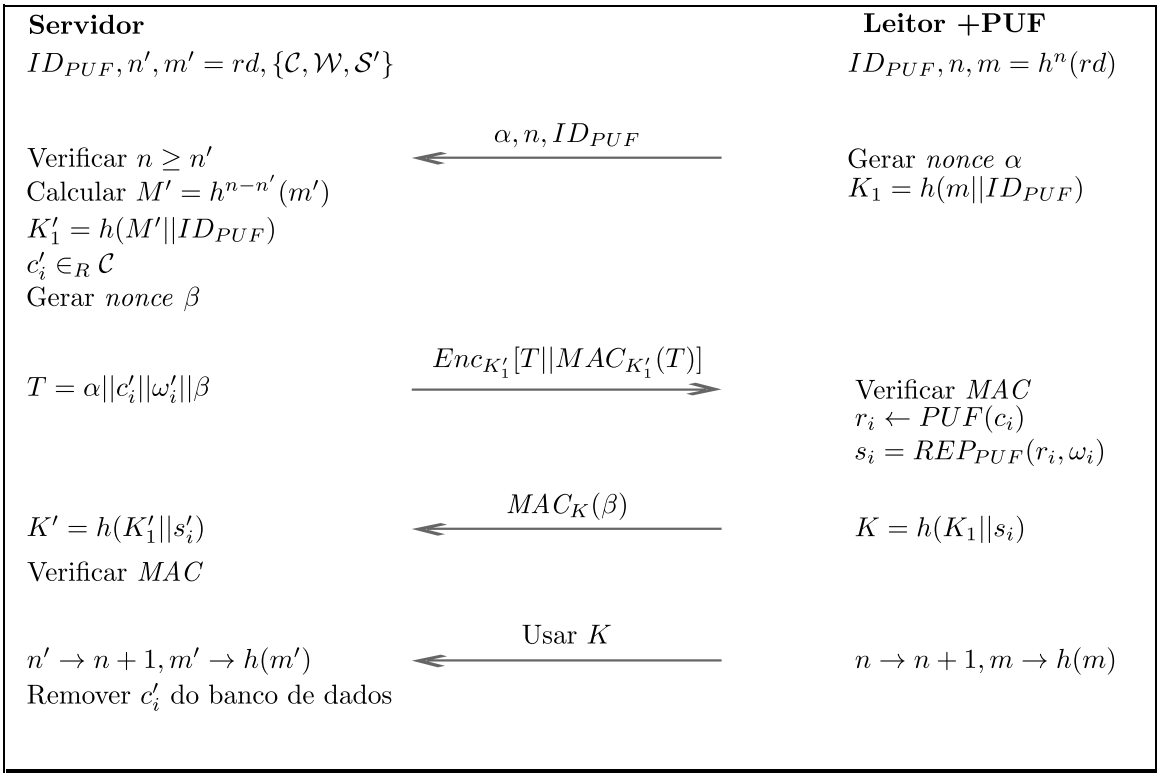


Figura 5.6: Protocolo de autenticação e estabelecimento de chave utilizando PUF de [78].

Bush *et al.* [18] propõem um modelo de atacante diferente de [78], onde o atacante tem o controle físico da PUF e de seu respectivo leitor por um curto espaço de tempo. Esse tipo de atacante é bastante realista no cenário de autenticação, como por exemplo, na situação em que o funcionário de um estabelecimento comercial leva o cartão de crédito do cliente para longe de seu campo visual para faturamento. Durante esse tempo o funcionário (adversário) pode ler dados armazenados em sua memória, como fazer algumas consultas de pares de desafio-resposta.

Com esse modelo de atacante, Bush *et al.* [18] propõem um ataque onde o adversário consegue personificar o servidor no protocolo proposto em [78]. O ataque funciona da seguinte maneira: tem-se $\mathcal{A}$ como um adversário que tem acesso ao cartão de crédito, incluindo a PUF uma única vez. O adversário $\mathcal{A}$ escolhe um pequeno número de desafios

[5] Procedimento *Rep* descrito na Seção 3.1

$\mathcal{C}^*$ e calcula suas respectivas respostas $\mathcal{R}^*$. Além disso, ele lê o identificador ID_{PUF}, o contador de uso n e o atual valor de *hash* de m que estão armazenado na memória do cartão.

De posse dessas informações o adversário pode personificar o servidor da seguinte maneira: o adversário calcula o valor $M^* = h^{n-n^*}(m)$, que é possível porque $\mathcal{A}$ obtém o contador n e o valor de hash $m = h(M)$ a partir da memória. Então, $\mathcal{A}$ calcula $K_1^* = h(M^*||ID_{PUF})$ e gera um *nonce* aleatório β^*. Assim $\mathcal{A}$ escolhe um $c_i^* \in \mathcal{C}^*$ e gera sua respectiva saída r_i juntamente com a informação auxiliar ω_i e então executa o algoritmo de reprodução G para obter $s_i \in \mathcal{S}*$.

Em seguida, $\mathcal{A}$ produz o MAC sobre a tupla $(\alpha, c_i^*, \omega_i^*, \beta^*)$ usando a chave K_1^*, e cifra o MAC com K_1^* e envia $Enc_{K_1^*}[(\alpha||c_i^*||\omega_i^*||\beta^*)||MAC_{K_1^*}(\alpha||c_i^*||\omega_i^*||\beta^*)]$ para o leitor. O leitor subsequentemente calcula o valor de $K_1 = (m||ID_{PUF})$, e decifra $Enc_{K_1^*}[(\alpha||c_i^*||\omega_i^*||\beta^*)||$ $MAC_{K_1^*}(\alpha||c_i^*||\omega_i^*||\beta^*)]$ e verifica se o MAC é válido.

Assim, uma vez que o MAC e o *nonce* α decifrados são válidos, o protocolo não aborta com uma mensagem de erro. E desse modo o restante do protocolo é executado até o seu final, onde uma chave simétrica K^* (respectivamente K) é estabelecida entre o leitor e o adversário e, desse modo, o adversário consegue personificar o servidor com sucesso.

Para tentar sanar esse tipo de personificação, [18] propõe o uso de filtros de Bloom [12] ou árvores de *hash* [55] para poder armazenar apenas na memória do cartão um subconjunto de desafios que foi inicialmente consultado pelo servidor pois, devido à natureza simétrica do protocolo, o leitor não pode decidir se ele foi consultado pelo servidor ou pelo adversário, dado um desafio durante a execução.

Segundo [18], esse armazenamento é feito de forma compacta e não permite que o adversário com poder computacional limitado ganhe informações úteis sobre os desafios. Veremos a seguir que a sobrecarga adicional de armazenamento imposta ao cliente não resolve a vulnerabilidade contra personificação do servidor.

Conforme mencionado a solução proposta feita por [18] não resolve o problema de [78] para o modelo de atacante criado. O atacante que tem acesso ao cartão de crédito com a PUF por um curto período de tempo ainda consegue personificar o servidor. Assume-se que a fase de inscrição do protocolo, que vai desde a fabricação da PUF até a entrega do cartão de crédito para o usuário, seja segura.

Para demonstrar que é possível personificar o servidor com sucesso mesmo utilizando filtro de Bloom ou árvore de *hash* o seguinte adversário é descrito. Tem-se o adversário $\mathcal{A}$ que tem acesso ao cartão de crédito incluindo a PUF apenas uma vez. Com acesso ao cartão, como descrito em [18], $\mathcal{A}$ consegue ler através da memória do cartão o ID_{PUF}, o contador de uso n e o atual valor de *hash* de m e armazena esses dados para uso posterior. Depois, o adversário estabelece uma comunicação "honesta" com o servidor e

consegue obter α^*, n^*, ID_{PUF}, $Enc_{K_1^*}[T^*||MAC_{K_1^*}(T^*)]$, onde $T^* = \alpha^*||c_i'^*||\omega_i^*||\beta^*$.

Assim, com esses dados $\mathcal{A}$ consegue encontrar o desafio utilizado nesta conversa da seguinte forma: $\mathcal{A}$ calcula o valor de $M^* = h^{n-n^*}(m)$, que é possível, pois $\mathcal{A}$ tem o contador de uso n e o valor de $m = H(M)$. Então, $\mathcal{A}$ calcula $K_1^* = h(M^*||ID_{PUF})$ e assim consegue decifrar $Enc_{K_1^*}[T^*||MAC_{K_1^*}(T^*)]$.

Com o valor de T^*, o adversário consegue obter o c_i e ω_i que foram utilizados na conversa espionada. Com o desafio c_i e sua correspondente informação auxiliar ω_i válidos, o adversário pode personificar o servidor com sucesso da mesma forma que foi feita em [18], com exceção de escolher o c_i descoberto anteriormente ao invés de $c_i \in \mathcal{C}$.

Este ataque funciona porque o cliente não verifica o reuso de um mesmo c_i. Para que essa "verificação" ocorra, é necessário remover cada c_i utilizado da memória do cartão do cliente e para esse fim é obrigatório a remoção do c_i armazenado no filtro de Bloom $\mathcal{B}$ ou na árvore de *hash*, agregando um custo adicional de complexidade. Por exemplo, uma variante de filtros de Bloom permite a remoção de um elemento com custo maior de armazenamento [26].

Capítulo 6

Conclusão

A utilização de PUFs como primitiva criptográfica é um campo promissor de pesquisa e promete fornecer soluções elegantes e inovadoras para diversos problemas de ordem criptográfica, como autenticação de transações e identificação de dispositivos.

Neste trabalho foram propostas construções para cifração e autenticação baseadas na combinação de PUFs e funções de *hash* universal. O objetivo das construções é fornecer serviços de segurança resistentes ao vazamento de *bits* da chave, uma vez que não há necessidade de armazenamento de material criptográfico em memória, cenário ideal para aplicações que exijam garantias de sigilo e autenticidade de discos ou dispositivos móveis. A cifra de bloco proposta aprimora a melhor construção presente na literatura, fornecendo tanto segurança contra ataques de criptograma escolhido, IND-CCA1, quanto redução do fator de expansão provocado pelo processo de cifração. A construção de autenticação, por sua vez, é bastante simples e envolve apenas uma chamada à PUF. Ambas as construções foram analisadas do ponto de vista de segurança utilizando noções de segurança padronizadas na literatura.

O trabalho também apresenta a proposta de um protocolo de autenticação flexível, baseado em uma combinação de PUF com protocolo PAKE e que fornece autenticação mútua entre cliente e servidor e o estabelecimento de uma chave de sessão, algumas das principais características de um protocolo de autenticação útil que não foram encontradas juntas na literatura. Este protocolo pode ser utilizado em diversos ambientes, sendo um deles a autenticação de aplicações bancárias, em que a chave de sessão pode ser usada para autenticar transações posteriores ou proteger informação financeira em trânsito. Em particular, a combinação de PUF+PAKE melhora o estado da arte de soluções de autenticação baseadas em PUFs, de acordo com a análise heurística e formal apresentadas. Além disso, é proposta uma adaptação do protocolo com suporte a senha de pânico para situações de emergência, onde o cliente é coagido a entregar a PUF e revelar sua senha.

6.1 Trabalhos futuros

Como possíveis trabalhos futuros do Capítulo 4 são propostas as seguintes sugestões:

- **Reduzir o tamanho do criptograma**: Diversas alternativas são possíveis para reduzir o tamanho do criptograma resultante da construção de cifração apresentada;
  - Adotar extratores determinísticos no lugar de extratores difusos pode tornar as informações auxiliares ω_i constantes para todas as cifrações;
  - Relaxar os requisitos de segurança dos vetores de inicialização, para que os mesmos sejam determinísticos para um mesmo bloco sendo cifrado. Desta forma, a PUF pode ser utilizada para derivar os vetores de inicialização a partir de metadados que identificam o bloco fisicamente.
- **Integração da cifra de bloco e MAC**: Integrar as construções de cifra de bloco e MAC propostos permitindo unificar os dois vetores de inicialização;
- **Implementação**: Implementar a cifra de bloco e o autenticador para obtenção de medidas de desempenho e comparações com cifras convencionais.

Para o Capítulo 5, o primeiro tópico está em andamento enquanto o segundo segue com possível trabalho futuro:

- **Construir um protocolo que ofereça segurança contra agentes internos**: O protocolo proposto em [28] foi adaptado para fornecer tanto segurança contra agentes internos quanto contra ataques de dicionário *offline*. Essa adaptação aparentemente descarta o ataque montado na Seção 5.4.1. Resta enunciar a prova formal de segurança desta adaptação;
- **Implementação e medidas de desempenho**: É importante implementar a proposta para a obtenção de medidas de desempenho, como o tempo de execução e o consumo de energia para a confirmação de sua viabilidade prática.

Referências

[1] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework. In Tal Malkin, editor, *Topics in Cryptology (CT-RSA 2008)*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351. Springer Berlin Heidelberg, 2008. 19

[2] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions. In Mitsuru Matsui, editor, *Advances in Cryptology (ASIACRYPT 2009)*, volume 5912 of *Lecture Notes in Computer Science*, pages 685–702. Springer Berlin Heidelberg, 2009. ix, 3, 17, 18, 21, 22, 27, 29, 31, 32, 39, 40, 41, 42, 44

[3] Leemon Baird III, Mance Harmon, Reed Young, and James Armstrong Jr. Apparatus and method for authenticating access to a network resource, May 4 2004. US Patent 6,732,278. 53

[4] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings., 38th Annual Symposium on Foundations of Computer Science (FOCS 1997).*, pages 394–403. IEEE, Oct 1997. 10, 12

[5] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer Berlin Heidelberg, 2000. 42

[6] Mihir Bellare and Phillip Rogaway. The AuthA protocol for password-based authenticated key exchange. Technical report, Citeseer, 2000. 2, 18, 19, 46, 47, 54

[7] Steven Bellovin and Michael Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Proceedings., IEEE Computer Society Symposium on Research in Security and Privacy.*, pages 72–84. IEEE, May 1992. 2, 18, 47

[8] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a Fast Software-Oriented Stream Cipher. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 98–118. Springer Berlin Heidelberg, 2008. 8

[9] Daniel Bernstein. Cache-timing attacks on AES. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf, 2005. 2

[10] Daniel Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer Berlin Heidelberg, 2008. 8

[11] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and Secure Message Authentication. In Michael Wiener, editor, *Advances in Cryptology (CRYPTO 1999)*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer Berlin Heidelberg, 1999. 42

[12] Burton Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, July 1970. 59

[13] Martin Boesgaard, Mette Vesterager, Thomas Pedersen, Jesper Christiansen, and Ove Scavenius. Rabbit: A New High-Performance Stream Cipher. In Thomas Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 307–329. Springer Berlin Heidelberg, 2003. 8

[14] Joseph Bonneau and Ilya Mironov. Cache-Collision Timing Attacks Against AES. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems (CHES 2006)*, volume 4249 of *Lecture Notes in Computer Science*, pages 201–215. Springer Berlin Heidelberg, 2006. 2

[15] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In Bart Preneel, editor, *Advances in Cryptology (EUROCRYPT 2000)*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer Berlin Heidelberg, 2000. 2, 19, 20

[16] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security Proofs for an Efficient Password-based Key Exchange. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, (CCS 2003), pages 241–250, New York, NY, USA, 2003. ACM. 19, 47

[17] Johannes Buchmann. *Introduction to cryptography*. Springer, 2004. 16

[18] Heike Busch, Stefan Katzenbeisser, and Paul Baecher. PUF-Based Authentication Protocols – Revisited. In HeungYoul Youm and Moti Yung, editors, *Information Security Applications*, volume 5932 of *Lecture Notes in Computer Science*, pages 296–308. Springer Berlin Heidelberg, 2009. 4, 30, 55, 58, 59, 60

[19] Lawrence Carter and Mark Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. 15

[20] Jeremy Clark and Urs Hengartner. Panic Passwords: Authenticating Under Duress. In *Proceedings of the 3rd Conference on Hot Topics in Security*, (HOTSEC 2008), pages 8:1–8:6, Berkeley, CA, USA, 2008. USENIX Association. 53

[21] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. In *First Advanced Encryption Standard (AES) Conference*, 1998. 8

[22] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Nov 1976. 2, 9

[23] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal on Computing*, 38(1):97–139, 2008. 17

[24] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology (EUROCRYPT 2004)*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer Berlin Heidelberg, 2004. 17, 18

[25] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin Heidelberg, 1985. 9

[26] Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking (TON 2000)*, 8(3):281–293, jun 2000. 60

[27] (FIPS PUB) Federal Information Processing Standards Publication. Data encryption standard. *National Institute of Standards and Technology*, 1977. 8

[28] Keith Frikken, Marina Blanton, and Mikhail Atallah. Robust Authentication Using Physically Unclonable Functions. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio A. Ardagna, editors, *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 262–277. Springer Berlin Heidelberg, 2009. x, 4, 21, 28, 30, 46, 51, 53, 55, 56, 57, 62

[29] Blaise Gassend. *Physical random functions*. PhD thesis, Massachusetts Institute of Technology, 2003. 25

[30] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, (CCS 2002), pages 148–160, New York, NY, USA, 2002. ACM. 2, 21, 25

[31] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the Cryptographic Applications of Random Functions (Extended Abstract). In George Robert Blakley and David Chaum, editors, *Advances in Cryptology (CRYPTO 1985)*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer Berlin Heidelberg, 1985. 44

[32] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 10

[33] Jorge Guajardo, Sandeep Kumar, Geert-Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer Berlin Heidelberg, 2007. 25

[34] Jorge Guajardo, Sandeep Kumar, Geert-Jan Schrijen, and Pim Tuyls. Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection. In *International Conference on Field Programmable Logic and Applications (FPL 2007)*, pages 189–195. IEEE, Aug 2007. 21, 29

[35] Alex Halderman, Seth Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph Calandrino, Ariel Feldman, Jacob Appelbaum, and Edward Felten. Lest We Remember: Cold-boot Attacks on Encryption Keys. *Commun. ACM*, 52(5):91–98, May 2009. 2

[36] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, and Jean-Pierre Seifert. Cloning Physically Unclonable Functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2013).*, pages 1–6. IEEE, June 2013. 22, 26, 29

[37] Nadia Heninger and Hovav Shacham. Reconstructing RSA Private Keys from Random Key Bits. In Shai Halevi, editor, *Advances in Cryptology (CRYPTO 2009)*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2009. 2

[38] William James Herschel. *The origin of finger-printing*. H. Milford, Oxford University Press, 1916. 21

[39] David Jablon. Strong Password-only Authenticated Key Exchange. *SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, October 1996. 2, 18, 20, 47

[40] David Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings., Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.*, pages 248–255. IEEE, Jun 1997. 18

[41] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall, 2008. 11, 13, 14, 16, 17, 33, 44

[42] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology (EUROCRYPT 2001)*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer Berlin Heidelberg, 2001. 2, 19

[43] Stefan Katzenbeisser, Ünal KocabaKocabaş, Vladimir Rožić, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 283–301. Springer Berlin Heidelberg, 2012. 22

[44] Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology (CRYPTO 1996)*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin Heidelberg, 1996. 2

[45] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology (CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Berlin Heidelberg, 1999. 2

[46] Kwok-yan Lam and Dieter Gollmann. Freshness assurance of authentication protocols. In Yves Deswarte, Gérard Eizenberg, and Jean-Jacques Quisquater, editors, *Computer Security (ESORICS 1992)*, volume 648 of *Lecture Notes in Computer Science*, pages 261–271. Springer Berlin Heidelberg, 1992. 19

[47] Qiming Li, Yagiz Sutcu, and Nasir Memon. Secure Sketch for Biometric Templates. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology (ASIACRYPT 2006)*, volume 4284 of *Lecture Notes in Computer Science*, pages 99–113. Springer Berlin Heidelberg, 2006. 17

[48] Michael Luby and Charles Rackoff. Pseudo-random Permutation Generators and Cryptographic Composition. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, (STOC 1986), pages 356–363, New York, NY, USA, 1986. ACM. 33, 35

[49] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. 33, 35

[50] Roel Maes. An Accurate Probabilistic Reliability Model for Silicon PUFs. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *Lecture Notes in Computer Science*, pages 73–89. Springer Berlin Heidelberg, 2013. 22

[51] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 302–319. Springer Berlin Heidelberg, 2012. 21, 29

[52] Roel Maes and Ingrid Verbauwhede. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 3–37. Springer Berlin Heidelberg, 2010. ix, 21, 22, 23, 24, 25, 26

[53] Armando Matos. "Hash universal e perfeito". `http://www.dcc.fc.up.pt/~acm/t1.pdf`, 2008. 15

[54] Alfred Menezes, Paul Van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC press, 2010. 1, 8, 14

[55] Ralph Merkle. Protocols for Public Key Cryptosystems. In *IEEE Symposium on Security and privacy*, volume 1109, pages 122–134. IEEE Computer Society, 1980. 59

[56] Mihir Bellare Michel Abdalla and Phillip Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. Cryptology ePrint Archive, Report 1999/007, 1999. http://eprint.iacr.org/. 2

[57] Steven Miller, Clifford Neuman, Jeffrey Schiller, and Jermoe Saltzer. Kerberos authentication and authorization system. In *Project Athena Technical Plan*. Citeseer, 1987. 19

[58] Moni Naor and Omer Reingold. On the Construction of Pseudorandom Permutations: Luby—Rackoff Revisited. *Journal of Cryptology*, 12(1):29–66, 1999. ix, 34, 35

[59] Charles Odonnell, Edward Suh, and Srinivas Devadas. PUF-based random number generation. MIT CSAIL CSG Technical Memo 481 (http://csg.csail.mit.edu/pubs/memos/Memo-481/Memo-481.pdf), 2004. 21, 29

[60] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One Way Functions. *Science*, 297(5589):2026–2030, 2002. 2, 21, 23

[61] Sarvar Patel, Zulfikar Ramzan, and GanapathyS. Sundaram. Towards Making Luby-Rackoff Ciphers Optimal and Practical. In Lars Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 171–185. Springer Berlin Heidelberg, 1999. ix, 3, 32, 35, 41

[62] Constantinos Patsakis. RSA private key reconstruction from random bits using SAT solvers, 2013. http://eprint.iacr.org/. 2

[63] Nicolas Popp, Siddharth Bajaj, and Phillip Hallam-Baker. Hybrid authentication, jan 2005. US Patent App. 10/864,501. 53

[64] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. 1979. 9

[65] Zulfikar Amin Ramzan. *A study of Luby-Rackoff ciphers*. PhD thesis, Massachusetts Institute of Technology, 2001. ix, 3, 15, 32, 34, 35, 40, 41

[66] Pappu Srinivasa Ravikanth. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, 2001. 21, 23

[67] Ronald Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. 9

[68] Matthew Robshaw and Olivier Billet. *New stream cipher designs: the eSTREAM finalists*, volume 4986. Springer, 2008. 8

[69] Ronald Russikoff. Computerized Password Verification System and Method for ATM Transactions, March 22 2005. US Patent 6,871,288. 53

[70] Jennifer Steiner, Clifford Neuman, and Jeffrey Schiller. Kerberos: An Authentication Service for Open Network Systems. In *USENIX Winter*, pages 191–202, 1988. 19

[71] Douglas Stinson. Universal hashing and authentication codes. In Joan Feigenbaum, editor, *Advances in Cryptology (CRYPTO 1991)*, volume 576 of *Lecture Notes in Computer Science*, pages 74–85. Springer Berlin Heidelberg, 1992. 15

[72] Douglas Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 4(3):369–380, 1994. 15

[73] Douglas Stinson. *Cryptography: Theory and Practice*, volume 36. CRC press, 2006. 14

[74] Edward Suh and Srinivas Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th Annual Design Automation Conference*, (DAC 2007), pages 9–14, New York, NY, USA, 2007. ACM. 21, 29

[75] Andrew Tanenbaum. *Sistemas Operacionais Modernos*. Editora Prentice-Hall, 2^a edition, 2003. 1

[76] Pim Tuyls and Lejla Batina. RFID-Tags for Anti-counterfeiting. In David Pointcheval, editor, *Topics in Cryptology (CT-RSA 2006)*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer Berlin Heidelberg, 2006. 48

[77] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-Proof Hardware from Protective Coatings. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems (CHES 2006)*, volume 4249 of *Lecture Notes in Computer Science*, pages 369–383. Springer Berlin Heidelberg, 2006. 24

[78] Pim Tuyls and Boris Škorić. Strong Authentication with Physical Unclonable Functions. In Milan Petković and Willem Jonker, editors, *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, pages 133–148. Springer Berlin Heidelberg, 2007. x, 4, 21, 30, 55, 57, 58, 59

[79] Mark Wegman and Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265 – 279, 1981. 15

[80] Maurice Vincent Wilkes. *Time-sharing computer systems*. Macdonald, 1972. 13

[81] Hongjun Wu. The Stream Cipher HC-128. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 39–47. Springer Berlin Heidelberg, 2008. 8