UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GUILHERME NUNES MELO

SISTEMA DE IDENTIFICAÇÃO BIOMÉTRICA BASEADO NO CÓDIGO DE ÍRIS COMBINADO COM CÓDIGOS CORRETORES DE ERROS

RECIFE, JULHO DE 2016.

GUILHERME NUNES MELO

SISTEMA DE IDENTIFICAÇÃO BIOMÉTRICA BASEADO NO CÓDIGO DE ÍRIS COMBINADO COM CÓDIGOS CORRETORES DE ERROS

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Doutor em Engenharia Elétrica**

ORIENTADOR: PROF. VALDEMAR CARDOSO DA ROCHA JUNIOR, PH.D.

Recife, Julho de 2016.

Catalogação na fonte
Bibliotecária Valdicéa Alves, CRB-4 / 1260

M528s Melo, Guilherme Nunes.
Sistema de identificação biométrica baseado no código de íris combinado com códigos corretores de erros - Recife: O autor, 2016.
134 folhas, Il.; Tabs.; Abr.; Sigl. e Simb.

Orientador: Prof. Dr. Valdemar Cardoso da Rocha júnior.

Tese (Doutorado) - Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2016.
Inclui Referências e Apêndice.

1. Engenharia Elétrica. 2. Código íris. 3. Códigos corretores de erro. 4. Codificação. 5. Biometria. 6. Autenticação. I. Rocha Junior, Valdemar Cardoso da (Orientador). II. Titulo.

UFPE

621.3 CDD (22. ed.) BCTG/2017 - 41

Universidade Federal de Pernambuco
Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE
TESE DE DOUTORADO DE

GUILHERME NUNES MELO

TÍTULO

"SISTEMA DE IDENTIFICAÇÃO BIOMÉTRICA BASEADO NO CÓDIGO DE ÍRIS COMBINADO COM CÓDIGOS CORRETORES DE ERROS"

A comissão examinadora composta pelos professores: VALDEMAR CARDOSO DA ROCHA JÚNIOR, DES/UFPE; RAFAEL DUEIRE LINS, CIN/UFPE; RENATO JOSÉ DE SOBRAL CINTRA, DE/UFPE; DANIEL PEDRO BEZERRA CHAVES, DES/UFPE; MARIA DE LOURDES MELO GUEDES ALCOFORADO, POLI/UPE e FRANCISCO MADEIRO BERNARDINO JÚNIOR, POLI/UPE, sob a presidência do primeiro, consideram o candidato **GUILHERME NUNES MELO** **APROVADO**.

Recife, 18 de julho de 2016.

MARCELO CABRAL CAVALCANTI
Coordenador do PPGEE

VALDEMAR CARDOSO DA ROCHA JÚNIOR
Orientador e Membro Titular Interno

MARIA DE LOURDES MELO GUEDES ALCOFORADO
Membro Titular Externo

RAFAEL DUEIRE LINS
Membro Titular Interno

FRANCISCO MADEIRO BERNARDINO JÚNIOR
Membro Titular Externo

RENATO JOSÉ DE SOBRAL CINTRA
Membro Titular Interno

DANIEL PEDRO BEZERRA CHAVES
Membro Titular Interno

Aos meus pais,

Deófrio da Costa Melo e

Maria do Carmo Nunes Melo

e aos meus filhos,

Daniel Figueiredo Melo,

Letícia Figueiredo Melo e

Lorena Figueiredo Melo.

Agradecimentos

Agradeço ao meu orientador, Professor Valdemar Cardoso da Rocha Junior, que me acolheu desde o início da minha caminhada acadêmica, como aluno de iniciação científica, depois, como aluno de mestrado e como aluno de doutorado. Sem a sua grande contribuição, não sei se conseguiria chegar tão longe nesta difícil jornada.

Agradeço muito aos meus pais, Deófrio da Costa Melo e Maria do Carmo Nunes Melo, que sempre me apoiaram, seja financeiramente, seja moralmente, permitindo que eu pudesse ter acesso a bons colégios, os quais me proporcionaram o aprendizado inicial desta minha vida acadêmica.

Agradeço a meus filhos, em ordem cronológica, Daniel, Letícia e Lorena, que até mesmo sem saber, me deram forças para seguir em busca desta conquista.

Agradeço a minha esposa: Giseuda, que aceitou muitos "nãos" com o propósito de permitir que eu realizasse as minhas pesquisas e meu desenvolvimento, que resultaram na conclusão desta Tese.

Agradeço a Meiriédna Queiroz Mota, que me ajudou na correção ortográfica.

Agradeço a Deus que colocou todas estas pessoas na minha vida e que me permitiu chegar com saúde até este momento.

Guilherme Nunes Melo

Universidade Federal de Pernambuco

18 de Julho de 2016

"Descobrir consiste em olhar para o que todo mundo está vendo e pensar uma coisa diferente."

— Roger Von Oech

RESUMO

A identificação biométrica já é uma realidade; baseia-se no uso de características biométricas dos indivíduos, principalmente, a impressão digital, as características da face, as características da palma da mão e a íris. Esta tese aborda a identificação biométrica baseada no código de íris, cujo processamento está sujeito a erros aleatórios e erros em surto, os quais dificultam a identificação do usuário. A partir de modelos de identificação de iris propostos, foram realizados testes de identificação usando as seguintes bases de dados de códigos de íris: BIOSECURE, CASIA, NIST-ICE(exp1) e NIST-ICE(exp2). Os melhores resultados disponíveis na literatura para sistemas de identificação biométrica que usam uma única íris e que corrigem um único símbolo por quadro, apresentam uma taxa de falsa rejeição (FRR) em torno de 30% para a base de dados BIOSECURE; cerca de 49% para a base de dados CASIA; cerca de 49% para a base de dados NIST-ICE(exp1); e cerca de 52% para a base de dados NIST-ICE(exp2). Quando ambas as íris são usadas, os percentuais para a FRR são cerca de 12% para a base de dados BIOSECURE; cerca de 24% para a base de dados CASIA; e cerca de 17% para a base de dados NIST-ICE. Nesta tese, são propostos quatro sistemas de identificação biométrica e recuperação de chave criptográfica, que reduzem os percentuais da FRR. Para o sistema de identificação generalizado proposto, foi obtida uma FRR máxima em torno de 8% para uma única íris e FRR máxima em torno de 1% para ambas as íris. Para a correção de um erro de símbolo por quadro, o melhor resultado, para uma íris, foi obtido na base de dados NIST-ICE(exp1), com a FRR de 3,96% e, para ambas as íris, foi obtido na base de dados CASIA com a FRR de 0,05%.

Palavras-chaves: Código íris. Códigos corretores de erro. Codificação. Biometria. Autenticação.

ABSTRACT

Biometric identification is already a reality; biometric identification systems employ individual characteristics such as fingerprints, facial features, the palm and iris characteristics. This thesis is concerned with biometric identification based on iris code, the processing of which is subject to random errors and burst errors, which difficult identification. By employing iris identification models proposed in this thesis, identification tests were performed using the following iris code databases: BIOSECURE, CASIA, NIST-ICE (exp1) and NIST-ICE (exp2). The best results available in the literature for identification systems employing a single iris and correcting a single symbol per frame, present a false rejection rate (FRR) around 30% for the BIOSECURE database, around 49% for the CASIA database, around 49% for the NIST-ICE(exp1) database and around 52% for the NIST-ICE(exp2) database. When both irides are employed, the percentages for the FRR are about 12% for the BIOSECURE database; about 24% to CASIA database; and about 17% for the NIST-ICE database. This thesis proposed four biometric identification and cryptographic key recovery systems to reduce the percentage of FRR. For the generalized identification system, FRR maximum rates were obtained around 8% for a single iris, and a maximum FRR rates around 1% for both irides. The obtained reduction in the percentage of FRR is due to the development of new search techniques and manipulations performed in the iris codes. For a correction of a symbol error per frame, the best result for an iris was obtained in NIST-ICE(exp1) database with FRR of 3.96% and for both irides, was obtained in CASIA database with FRR of 0.05%.

Keywords: Iris code. Error correction codes. Coding. Biometry. Authentication.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

Lista de Abreviaturas, Siglas e Símbolos

BIOSECURE *Biometrics for Secure Authentication*

CASIA *Chinese Academy of Sciences*

ECC Código corretor de erros

FAR *False Acceptance Rate*

FRR *False Rejection Rate*

h(K) Hash da chave K

h(K') Hash da chave K'

HD *Hamming Distance*

I_{ref} Imagem de Referência

I_{sam} Imagem de Teste

K Chave criptográfica gerada aleatoriamente

K' Chave criptográfica recuperada

kper Chave de permutação

k_{RS} Comprimento de bloco de entrada do código RS

k_{shuf} Chave de embaralhamento

LFSR *Linear Feedback Shift Register*

MB *Mega Bytes*

m_{RS} Parâmetro m do código RS

NIST-ICE *National Institute of Standards and Technology - Iris Challenge Evaluation*

n_{RS} Comprimento de bloco de saída do código RS

OSIRIS *Open Source Independent Review and Interpretation System*

Rand_num Sequência de números pseudo aleatórios

RMP Sistema de codificação baseado nos códigos produto de Reed Muller

RS Reed Solomon

RSH Sistema de codificação sequencial Reed Solomon, depois Hadamard

θ_{lock} Sequência codificada do código de íris

θ_{ps} Sequência de saída do codificador RSH

θ^{*}_{ps} Sequência de entrada do decodificador RSH

θ_{ref} Código de íris da imagem de referência

$\theta^{'}_{ref}$ Código de íris da imagem de referência, após manipulações

θ_{sam} Código de íris da imagem de teste

$\theta^{'}_{sam}$ Código de íris da imagem de teste, após manipulações

t_{RS} Capacidade de correção de erros do código RS

t_S Tempo total de processamento de um experimento, empregando apenas uma íris

t_T Tempo total de processamento de um experimento, empregando ambas as íris

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

"Todas as vitórias ocultam uma abdicação."

— Simone de Beauvoir

A identificação de usuários empregando dados biométricos é uma realidade em muitos sistemas computacionais, incluindo acesso a bancos, votações, acesso a cofres, entre outros [4], [5], [6] e [7]. O sucesso de sistemas de identificação que empregam dados biométricos depende do desenvolvimento de técnicas que garantam uma maior confiabilidade na operação, permitindo acesso aos usuários genuínos e negando acesso aos impostores com alta probabilidade. Muitos sistemas de identificação que usam biometria trabalham com impressões digitais, com a palma da mão, com reconhecimento de face ou com a íris [4]. Esta tese é restrita aos sistemas que empregam os dados digitais da íris (código de íris).

1.1 ÁREAS DE PESQUISA USANDO A IDENTIFICAÇÃO BIOMÉTRICA DA ÍRIS

Basicamente existem três abordagens principais, distintas, para pesquisas envolvendo identificação biométrica da íris. A primeira abordagem consiste em melhorar os métodos de captura de imagens, usando equipamentos com câmeras de melhor definição de imagens, ou seja, câmeras que possuam capacidade de capturar mais pontos de imagem por

unidade de área. A segunda abordagem consiste em melhorar a identificação dos usuários, manipulando as imagens já capturadas, alterando o tipo de filtro usado para converter as imagens em código binário. A terceira abordagem consiste em manipular o código binário que representa uma imagem de uma íris, buscando principalmente, com o uso de códigos corretores de erro, melhorar a identificação dos usuários [4].

Nas bases de dados empregadas nesta tese [8] e [1], já existe disponível o código binário que representa a imagem da íris, o qual é composto por 1.188 *bits*, que representam a imagem de uma íris, de um determinado usuário. Esta tese aborda a manipulação do código de íris, usando códigos corretores de erros e técnicas de pré-processamento destes códigos.

Na literatura especializada encontram-se vários trabalhos que empregam identificação biométrica usando a íris, como por exemplo: reconhecimento da íris em dispositivos móveis [9], técnicas de reconhecimento de padrões em imagens de vídeo da íris [10], segmentação da íris usando imagens de alta qualidade [11], identificação da íris usando vasos sanguíneos da retina [12], reconhecimento da íris usando novos filtros de Gabor [13], reconhecimento da íris usando *oblivious* RAM [14], características de movimento dos olhos, constrição da íris e parâmetros de dilatação [15], projeção espacial probabilística da trajetória de movimento dos olhos [16], reconhecimento periocular [17], identificação usando segmentação do contorno modificado da íris [18], características do piscar dos olhos [19] e o uso de pequenas partes da íris ao invés de toda a íris [20] e [21]. Ao contrário da abordagem desta tese, a qual envolve codificação, os artigos [9]- [21] abordam, basicamente, novos métodos de captura ou de conversão da imagem da íris em código binário.

1.2 MOTIVAÇÃO

A análise dos sistemas utilizados nos testes que empregam o código de íris e o tipo de codificação usada nos mesmos, fez com que fosse possível acreditar que a codificação utilizada é muito poderosa para resolver um problema de identificação relativamente "simples". Muito provavelmente existe a possibilidade de fazer modificações nos esquemas de codificação, com a expectativa de alcançar taxas de falsa rejeição - *False Rejection Rate* (FRR), melhores do que as encontradas até o momento.

1.3 OBJETIVOS

O objetivo desta tese é propor um novo sistema de codificação do código de íris que permita diminuir a FRR, aumentando a probabilidade de identificação de usuários genuínos, e recuperar uma chave criptográfica com maior comprimento que os sistemas propostos até o momento.

1.3.1 OBJETIVOS ESPECÍFICOS

- Desenvolver um método de identificação biométrica que empregue não só as rotações das imagens de teste da íris, mas, também, as rotações das imagens de referência da íris;
- Investigar o uso da técnica do voto de maioria visando melhorar o desempenho dos esquemas usados na identificação de usuários que empregam o código íris;
- Analisar o código íris para identificar a possível presença de redundâncias ou a presença de correlação entre os seus *bits*;
- Apresentar um sistema generalizado que inclua todas as propostas apresentadas ao longo da tese.

1.4 ORGANIZAÇÃO DA TESE

O conteúdo desta tese está dividido em sete Capítulos e um apêndice. As referências encontram-se nas páginas finais. A seguir, a apresentação do conteúdo dos Capítulos da tese.

Capítulo 2. Neste Capítulo serão apresentadas as fundamentações teóricas sobre os assuntos abordados ao longo da tese.

Capítulo 3. O objetivo principal deste Capítulo é apresentar a nova técnica de identificação de usuários, chamada de busca por rotação, que emprega a rotação das imagens de referência e das imagens de teste do código íris e compará-la com a técnica de identificação de usuários usada atualmente, que é chamada de busca padrão e é usada em [2] e em [3].

Capítulo 4. Neste Capítulo, será empregada a técnica de voto de maioria aplicada ao código íris usando um novo sistema de regeneração de chave criptográfica e os resul-

tados desta proposta serão comparados com os obtidos nos sistemas utilizados atualmente.

Capítulo 5. O objetivo deste Capítulo é analisar o código íris em busca de redundância entre os *bits* do código e a alteração na proporção de *bits* inseridos, de modo que seja possível utilizar eventuais descobertas para melhorar o sistema de regeneração de chave criptográfica proposto.

Capítulo 6. Após analisar cada modificação individualmente sugerida nos Capítulos 3, 4 e 5, as mesmas serão empregadas em conjunto em um novo sistema, que será chamado de sistema generalizado, que inclua todas as modificações sugeridas nos Capítulos anteriores, e eventuais melhorias.

Capítulo 7. Este Capítulo é dedicado às considerações finais da tese, juntamente com as contribuições e sugestões para trabalhos futuros.

Apêndice A. Neste apêndice serão detalhadas as bases de dados e os códigos corretores de erros empregados nesta tese.

Capítulo 2

Fundamentos teóricos e técnicas usadas para manipulação da íris

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original."

— Albert Einsten

Neste Capítulo serão explicados os fundamentos teóricos que serão usados nos sistemas de regeneração de chave criptográfica e identificação do usuário, assim como também serão detalhadas as técnicas usadas nestes sistemas.

2.1 Biometria

A biometria consiste em usar um dado próprio de um indivíduo em um sistema de identificação, é baseada em dados físicos e comportamentais específicos do indivíduo, como a impressão digital, a íris, a voz, a geometria da mão, a geometria da face, entre outros. [5], [6]

A vantagem do uso da biometria em relação o uso de cartões magnéticos, consiste no fato de que estes podem ser perdidos, roubados, distribuídos e forjados, enquanto que a

biometria está isenta destes problemas, além do fato de que, com o uso da biometria, a delegação não é permitida, uma vez que o indivíduo precisa estar presente no momento da autenticação. Já se consegue copiar a impressão digital, esse é um dos motivos pelo qual nesta tese, foi optado o uso da íris. Até o momento não há relato de sucesso em cópia da íris.

De acordo com [22] O uso de técnicas biométricas de reconhecimento já existe há décadas, proporcionando identificação/verificação de indivíduos baseado em suas características únicas. Em particular, o uso da biometria tem aumentado significativamente nas últimas décadas trazendo preocupações relacionadas à manutenção das liberdades civis e privacidade do indivíduo, uma vez que as soluções biométricas tradicionais requerem o armazenamento direto dos dados pessoais do indivíduo.

A Criptografia [23], [24], [25], [26], [27], [28], é capaz de proporcionar alta privacidade aos dados e possui a necessidade de exatidão dos dados a fim de funcionar corretamente. Esta necessidade de exatidão dos dados é um dos principais problemas que precisa ser enfrentado pelos sistemas de identificação biométrica, visto que os dados biométricos variam de uma coleta a outra devido a diversos fatores.

A pesquisa visando à combinação da criptografia com a biometria tem trazido soluções não só com relação à proteção dos dados biométricos, mas também abriu a possibilidade de gerar chaves criptográficas a partir de dados biométricos, com a principal vantagem de fornecer uma forte ligação entre usuário e chave criptográfica, o que não ocorre com as chaves criptográficas tradicionais [22].

Alguns dos artigos que consideram a combinação biometria/criptografia são [2], [3], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45] e [46]. A fim de tornar viável esta combinação, diferentes abordagens têm sido usadas, dentre elas a técnica de regeneração de chave criptográfica é a que tem se mostrado mais promissora [2], [3], [39], [40], [41], [42], [43], [44], [45] e [46].

2.2 SISTEMAS DE REGENERAÇÃO DE CHAVE

De acordo com [22], dentre as soluções propostas até o momento, a que tem se mostrado mais apropriada é a abordagem denominada regeneração de chave. Nesta abordagem, uma sequência de *bits* aleatória (chave) é combinada com o dado biométrico do usuário, a chave portanto, não é gerada diretamente pelos dados biométricos do usuário. Esta técnica per-

mite que a chave possa então ser regenerada, apresentando-se o resultado desta combinação e outra amostra biométrica genuína. Esta ideia denomina-se *biometric locking*, foi introduzida por Soutar et al. em [47].

Para lidar com a variabilidade dos dados biométricos, os sistemas de regeneração de chave criptográfica usam códigos corretores de erros.

Clancy et al. [42] propuseram uma aplicação similar baseada em impressões digitais usando a técnica chamada *fuzzy vault* introduzida por Juels e Sudan em [43]. Um código Reed-Solomon é usado nesta técnica a fim de possibilitar a regeneração da chave.

A técnica *Fuzzy Commitment* foi apresentada por [40], e consiste em gerar uma chave aleatória, adicionar redundância e combinar o resultado do código íris através de uma operação ou-exclusivo, o que torna a chave completamente independente do dado biométrico. Juels et al., neste trabalho, mostraram como utilizar códigos corretores de erros, mas não propuseram o uso de um código específico.

Hao et al. [39] propuseram um sistema de regeneração de chave baseado na íris, onde a concatenação dos códigos Reed-Solomon e Hadamard é usada para lidar com as variabilidades da íris. Os testes feitos por Hao et al. usaram uma base de dados proprietária, não sendo possível fazer uma comparação muito justa com outras pesquisas.

Kanade et al. propuseram em [46] um sistema de regeneração de chave baseado no trabalho de Hao et al., porém os testes foram feitos em uma base de dados pública (NIST-ICE [1]) e mostraram que este sistema é capaz de produzir chaves de 198 *bits* de comprimento, com *False Acceptance Rate* (FAR) igual a 0,0550% e FRR igual a 1,0400%.

2.3 CAPTURA DE IMAGEM DA ÍRIS

A captura das imagens da íris e feita usando câmeras que tiram fotos em infravermelho, na Figura 2.1 é possível ver o equipamento que foi usado para capturar as imagens da base de dados NIST-ICE, os três pontos em vermelho são as posições dos LEDs infravermelhos que funcionam como flash [48].

As imagens da íris capturadas estão sujeitas a alguns problemas, seguem seis principais:

- **Desvio da íris**. Quando uma imagem da íris está sendo capturada [49], se não houver controle, é possível que o olho esteja desviado para qualquer direção. Dependendo de como for o desvio, ainda é possível processar a íris diretamente. Este tipo de problema

Figura 2.1: *LG 2200 equipamento de captura da iris utilizado para gerar as imagens da base de dados NIST-ICE [1].*

pode ser corrigido usando técnicas de trigonometria para estimar o angulo do desvio e fazer uma transformação na imagem, rotacionando o olho até que o mesmo esteja direcionado para frente [49].

- **Obstrução da íris**. Este é um dos problemas mais frequentes, pode ser causado pela pálpebra ou pelos cílios, principalmente por causa do piscar dos olhos. Neste casos uma máscara deve ser usada para prevenir que *pixels* não válidos possam afetar o código íris. A quantidade da íris que está obstruída deve ser levada em conta, pois poderá ter pouca informação para ser considerada uma imagem válida para ser usada nos experimentos.
- **Midríase**. A midríase consiste na dilatação excessiva da pupila, que pode ser causada por alguma doença, trauma, uso de drogas ou uso de álcool. A midríase também pode ser provocada artificialmente usando colírio. No caso da ocorrência de midríase, deformações não elásticas da íris ocorrerão quando a pupila dilatar. Devido a este tipo de deformação a forma circular da íris é afetada. Se a midríase estiver em um estado avançado, poderá ser difícil identificar o usuário.
- **Uso de óculus**. Considerando o grande número de pessoas que usam óculos, é importante que o ruído causado pelo seu uso não afete o desempenho do sistema de reconhecimento. Reflexões são um dos efeitos que poderão ocorrer com o uso dos óculos, e se esta

reflexão cobrir parte da íris, uma máscara precisará ser usada para prevenir que *pixels* não válidos possam afetar o código íris.

- **Uso de lentes de contato**. Devido a algumas vantagens do uso de lentes de contato, mais pessoas estão usando-as, e dependendo do material que as mesmas são feitas, alguns problemas podem afetar mais ou menos os sistemas de reconhecimento que utilizam a íris. As lentes de contato podem ser rígidas ou flexíveis e algumas ainda poderão ter marcas impressas nas lentes para orientar o uso correto das mesmas.
- **Rotação da íris**. A rotação da íris ocorre quando o posicionamento da cabeça do usuário não está disposto na posição vertical. Uma pequena alteração na disposição da cabeça pode gerar um ruído na captura da íris que, dependendo do grau, poderá afetar o reconhecimento do usuário.

Nas Figuras 2.2, 2.3, 2.4, 2.5, 2.6 e 2.7, tem-se respectivamente exemplos de ruídos causados pelo desvio da íris, pela obstrução da íris, pela midríase, pelo uso de óculos, pelo uso de lentes de contato e pela rotação da íris.

Figura 2.2: *Exemplo de imagens com desvios na captura.*

Figura 2.3: *Exemplo de imagens com obstrução da íris.*

Figura 2.4: *Exemplo de imagens com midríase.*

Figura 2.5: *Exemplo de imagens com o uso de óculos.*

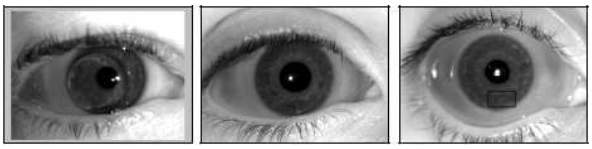


Figura 2.6: *Exemplo de imagens com lentes de contato.*

Figura 2.7: *Exemplo de imagens com rotação da íris.*

O ruído causado pela rotação da imagem da íris quando a mesma é capturada, pode ser compensado fazendo a rotação das imagens de referência da íris, que estão disponíveis na base de dados. Os outros problemas de ruído, também podem ser afetados pela rotação, além do seu problema específico.

Todos os problemas de ruído relatados, poderão ser minimizados com o uso de filtros que determinam a melhor área da imagem da íris a ser extraída, melhorando o desempenho dos sistemas de reconhecimento. Deste modo, quando uma imagem da íris é capturada, ela passa por um filtro e por um processo de rotação.

A melhoria do filtro é uma das áreas que concentra um grande número de pesquisas. Nas bases de dados utilizadas nesta tese, tem-se a imagem central da íris, 10 rotações para a esquerda e 10 rotações para a direita, totalizando 21 rotações para cada imagem capturada.

Todas as imagens da base de dados já estão convertidas para dados binários, que são conhecidos como código de íris. Nesta tese, não será feita a captura da íris ou a busca por melhoria nos filtros, como os dados já estão em binário, serão utilizados códigos corretores de erros e manipulações nestes códigos para buscar melhorar o desempenho do sistema de reconhecimento que será proposto.

2.4 Sistema de regeneração de chave proposto por Kanade

No trabalho de Kanade et al. [39], o sistema proposto usa códigos íris gerados pelo software *Open Source Independent Review and Interpretation System* (OSIRIS) e a base de dados NIST-ICE. Nele são obtidos chaves de comprimento de 42 *bits*, com $t_{RS} = 13$ e FRR de 19,4100%, que é uma taxa alta, o t_{RS} é a capacidade de correção do código de Reed-Solomon.

No trabalho de Bringer et al [45], o sistema proposto por Hao et al. também foi testado e foram obtidas chaves de comprimento igual a 14 *bits*, para a FRR igual a 10%.

O sistema de regeneração proposto por [46] utiliza o software OSIRIS para a extração do código íris de comprimento 1.188 *bits*. Este software é inspirado no sistema proposto por Daugman [50], foi desenvolvido durante o projeto Biosecure e incorpora máscaras estáticas, ou seja, apenas características em certos locais na imagem da íris são calculados a fim de evitar erros que ocorrem devido a cílios e pálpebras. Além disso, um ajuste para possíveis rotações é feito através do deslocamento da imagem normalizada da íris em ambas as direções [22].

A Figura 2.8 apresenta o diagrama esquemático do sistema de regeneração de chave proposto por Kanade et al. [46].

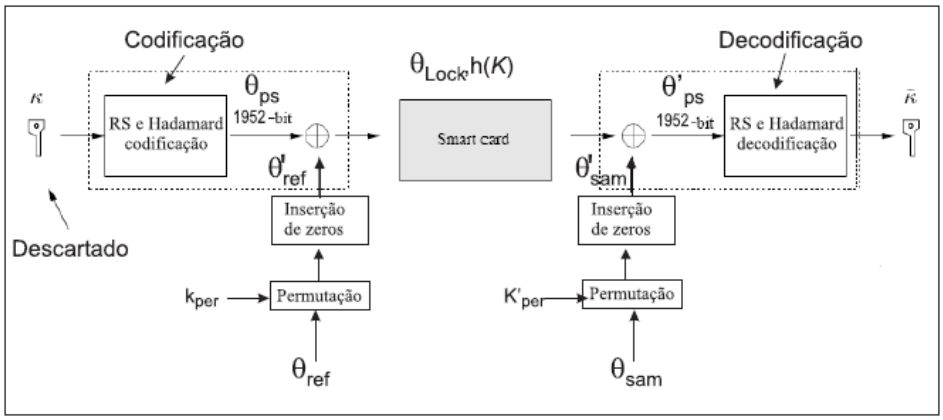


Figura 2.8: *Sistema de Regeneração de Chave utilizando três fatores de autenticação (cartão magnético, íris e senha).*

2.4.1 DESCRIÇÃO DO SITEMA PROPOSTO POR KANADE

Nesta subseção a descrição do sistemas proposto por Kanade foi baseado nas explicações dadas por Câmara em [22]. No sistema proposto por Kanade, um vetor aleatório K (chave criptográfica) é gerado e codificado pela concatenação dos códigos Reed-Solomon e Hadamard, gerando θ_{ps}. O pseudo-código θ_{ps}, é então operado ou-exclusivo com θ'_{ref} , o código íris de referência modificado. Produzindo, $\theta_{lock} = \theta_{ps} \oplus \theta'_{ref}$

O código íris de referência modificado, θ'_{ref} , consiste no código íris obtido após a permutação de θ_{ref} e inserção de zeros. θ_{lock} e o *hash* da chave criptográfica h(K), são armazenados em um *smart card*. Detalhes sobre a permutação e sobre a inserção de zeros serão descritos

respectivamente nas seções 2.5 e 2.6.

O usuário apresenta o *smart card* e uma amostra da sua íris quando precisar ser reconhecido pelo sistema de identificação. O código íris, θ_{sam} é gerado, passando pelos mesmos mecanismos de permutação e inserção de zeros usados na geração dos dados que estão armazenados no *smart card*, produzindo o código íris modificado, θ'_{sam}. O código íris obtido na fase de reconhecimento θ_{sam} é considerado, a priori, diferente do código íris obtido na fase de cadastro do usuário θ_{ref}, pois devido a fatores como iluminação, posicionamento, dilatação da pupila, entre outros, é pouco provável obter dois códigos idênticos.

O código íris θ'_{sam} é então operado ou-exclusivo com θ_{lock}:

$$\theta'_{ps} = \theta_{lock} \oplus \theta'_{sam} \quad (2.1)$$

$$= \theta_{ps} \oplus \theta'_{ref} \oplus \theta'_{sam} \quad (2.2)$$

$$= \theta_{ps} \oplus e; \quad (2.3)$$

onde e representa as diferenças entre θ'_{ref} e θ'_{sam}. Então, θ'_{ps} passa pelo decodificador resultando em K'. Se o processo de decodificação é bem sucedido, K' = K, e portanto, o h(K') = h(K). Neste caso, o usuário é considerado legítimo e sua chave autêntica. Caso contrário, o usuário é considerado impostor.

Esta técnica de codificação é a mesma utilizada por Hao et. al. [39] e é muito conveniente para lidar com erros aleatórios e erros em surto, presentes no código íris, porém os parâmetros dos códigos não são os mesmos usados por Hao et al. pois, $||\theta'_{ref}||$ = $||\theta_{ps}||$ = 1952 [22].

Seguindo o mesmo raciocínio apresentado por Hao et. al. e levando em conta que $||\theta_{ps}||$ = 1952 e que $k_H = 5$ se mostrou o valor mais conveniente; os parâmetros para os códigos Reed-Solomon e Hadamard são: $n_{RS} = 61$ blocos e $m_{RS} = k_H + 1 = 6$ *bits* [22].

Assim como em [39], t_{RS} foi variado durante os experimentos observando qual valor resultaria em melhor desempenho biométrico e comprimento de chave. Considerando o desempenho biométrico do sistema e o comprimento da chave, $t_{RS} = 14$, é o que produz melhores resultados, gerando chaves de comprimento 198 *bits* com FRR de 1,4600% [22].

O método proposto em [39] foi testado numa base de dados proprietária sendo capaz de corrigir 27% dos erros no código íris, porém experimentos em base de dados públicas bem conhecidas, tais como NIST-ICE [1], mostraram a necessidade de aumentar esta capacidade de correção [45], [46] e [22].

Desta forma, com o intuito de aumentar a capacidade de correção da técnica proposta em [39], dois mecanismos foram adicionados, resultando no sistema apresentado em [46]. Com essas modificações o novo sistema tornou possível a regeneração de chaves de comprimento 198 *bits* com FAR igual a 0,0550% e FRR igual a 1,0400% [22].

Os dois mecanismos adicionados ao sistema de Hao et al. foram [22]:

- permutação do código íris baseado numa chave de permutação específica por usuário, k_{per};
- inserção de zeros.

2.5 PERMUTAÇÃO (USADA NO SISTEMA PROPOSTO POR KANADE)

A permutação do código de íris proposta por [46], segundo [22], consiste em permutar o código de íris tendo como base uma chave de permutação de 198 *bits* de comprimento, específica por usuário, kper. O código íris de 1.188 *bits* é dividido em 198 blocos de 6 *bits*, sendo então alinhado a kper. Os blocos para os quais o *bit* de kper é "1" são deslocados para frente enquanto que os outros são enviados para trás. Este mecanismo é ilustrado na Figura 2.9.

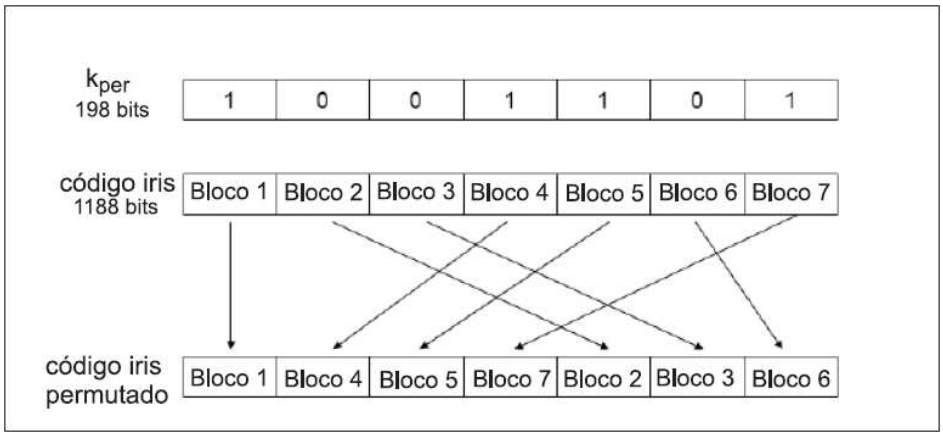


Figura 2.9: *Permutação do código íris.*

A permutação do código íris melhora a capacidade de correção de erros do sistema através da difusão dos erros, diminuindo a densidade de erros e auxiliando o mecanismo de correção de erros. Além disso, como a permutação depende da chave de permutação do usuário, kper, se o usuário é genuíno a mesma permutação é aplicada tanto na fase de registro quanto na fase de reconhecimento e, portanto, erros não são adicionados. Porém, se

um impostor usa sua chave de permutação, kper*, diferente de kper, o código é permutado de forma distinta na fase de reconhecimento e erros são adicionados [22]. Para o sistema proposto nesta tese, o kper foi renomeado para k_{shuf}, ou chave de embaralhamento.

2.6 INSERÇÃO DE ZEROS (USADA NO SISTEMA PROPOSTO POR KANADE)

Após a permutação, 2 zeros são inseridos a cada 3 *bits* do código íris permutado, ou seja, são acrescentados 764 *bits*, resultando num vetor de comprimento 1.980 *bits*. A fim de possibilitar a operação ou-exclusivo com θ_{ps}, 28 *bits* são retirados do final do vetor de 1.980 *bits*, resultando no código íris de referência modificado, θ'_{ref} , de 1.952 *bits* de comprimento [22].

Observe que o código de Hadamard, que é definido pelos parâmetros $(2^k, k+1, 2^{k-2}-1)$, utilizado, é o código (32,6,7), ou seja, tem $k = 5$ [22], e tem capacidade de correção $t_{HC} = 7$, portanto, pode corrigir até 7 *bits* em 32 *bits*, ou seja, uma taxa de correção de 21,8700%, que não é suficiente para lidar com as variações observadas nos códigos íris, (a variável k é o parâmetro que define o código de Hadamard). Porém, ao adicionar zeros, similaridades são inseridas, não produzindo erros. Desta forma, em cada bloco de 32 *bits*, efetivamente, 18 ou 20 *bits* estão vulneráveis a erros. Considerando 20 *bits* vulneráveis a erros, a taxa efetiva de correção é de $7/20$, ou seja, 35%. Assim, é possível lidar com a quantidade de erros presentes usualmente nos códigos íris [22].

2.7 INSERÇÃO DE SEQUÊNCIA PSEUDO ALEATÓRIA (USADA NO SISTEMA PROPOSTO POR CÂMARA)

No esquema proposto por Câmara [3], ao invés de serem adicionados zeros no código íris, são adicionados números pseudo aleatórios, denominados de **Rand_num**. Na proposta de [3], são usadas ambas as íris, ou seja, o código íris possui 2.376 *bits*, de modo que a quantidade de *bits* adicionados são 1.528, ou seja, ||**Rand_num**||=1.528 bits.

A inserção é então feita da seguinte forma: dois dígitos binários são inseridos após três *bits* do código íris, até os primeiros 2.208 *bits* e um dígito binário do **Rand_num** é inserido após cada três *bits*, nos próximos 168 *bits* restantes, resultando no código de íris modificado

de comprimento 3.904 *bits*:

$$\left(\frac{2.208}{3}\right) * 2 + \left(\frac{168}{3}\right) * 1 = 1.528. \tag{2.4}$$

Neste esquema proposto por [3], não é necessário truncar a sequência concatenada, pois o comprimento do código íris, 2.376 *bits*, mais os 1.528 *bits* do **Rand_num** acrescentados, são exatamente 3.904 *bits*, que é o mesmo comprimento do θ_{ps}. Vale ressaltar que o processo usado na codificação é o mesmo usado na decodificação.

Uma vantagem que se obtém com a inserção da sequência de *bits* pseudo aleatórios ao invés da sequência de *bits* iguais a zeros é que o peso de Hamming é aumentado no primeiro caso e mantido no segundo caso, este fato se reflete mais claramente quando são feitos os testes de impostores, no qual é possível observar que as curvas dos pesos de Hamming relativos são separadas, entre os testes para identificação de usuários genuínos, e entre os testes para identificação de impostores, quando os números pseudo aleatórios são acrescentados, enquanto que quando são acrescentados zeros, as curvas não são separadas. Estas curvas poderão ser vistas no Capítulo 3, nas seções 3.3.2 e 3.3.3.

2.8 TEOREMA DE NEYMAN-PEARSON (TESTE DE HIPÓTESE)

O teorema de Neyman-Pearson [51], usado em teoria da decisão para a detecção de sinais, é a seguir aplicado a um exemplo de teste de hipótese. A notação $N(\mu, \sigma^2)$ é usada para representar uma função densidade de probabilidades Gaussiana com média μ e variância σ^2, referente a uma variável aleatória X, denotando por x[0] uma única observação de X. Assumindo que foi observada a ocorrência de duas variáveis aleatórias cujas funções densidade de probabilidades são dadas por $N(0,1)$ ou $N(1,1)$, é necessário determinar se $\mu = 0$ ou $\mu = 1$, baseando-se apenas na observação de uma amostra $x[0]$. Cada possível valor de μ pode ser avaliado como uma hipótese, então o problema passa a ser: escolher entre duas hipóteses, $H_0 : \mu = 0$ ou $H_1 : \mu = 1$, em que H_0 é referido como sendo a hipótese nula e H_1 como a hipótese alternativa. Este problema é conhecido como sendo o teste de hipótese binário, uma vez que é escolhida uma entre duas hipóteses. As funções densidade de probabilidades associadas a cada hipótese são mostradas na Figura 2.10.

Ao analisar uma única amostra na Figura 2.10 pode ser difícil definir qual função de densidade de probabilidade gerou a mesma. No entanto uma abordagem pode ser decidir por H_1 se $x[0] > 0,5$. Isto porque se $x[0] > 0,5$, a amostra observada provavel-

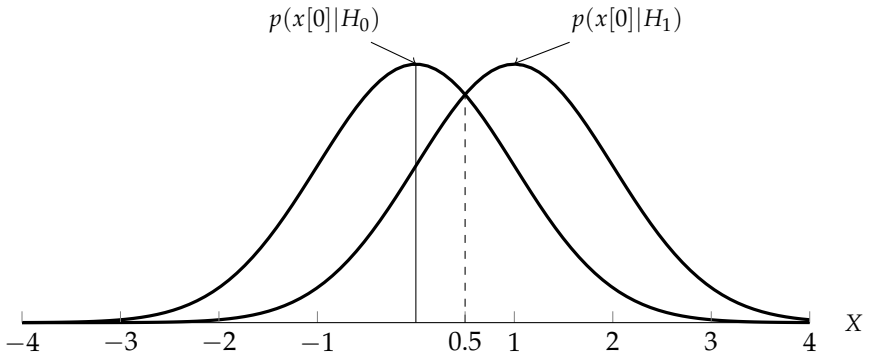


Figura 2.10: *Funções de densidade de probabilidades para o problema de teste de hipótese.*

mente ocorrerá se H_1 for verdadeira. Observando a Figura 2.11, se $x[0] > 0,5$, tem-se que $p(x[0]|H_1) > p(x[0]|H_0)$. O detector então pode comparar o valor observado com $0,5$, este valor é chamado de limiar de decisão. Note que com este esquema é possível cometer dois tipos de erros. Se for escolhido H_1 mas H_0 era verdadeiro, comete-se o erro do tipo I, por outro lado, se for escolhido H_0 mas H_1 era verdadeiro, comete-se o erro do tipo II. Estes erros estão ilustrados na Figura 2.11. A notação $P(H_i|H_j)$ indica a probabilidade de decidir por H_i quando H_j é verdadeiro. Por exemplo, $P(H_1|H_0) = Pr(x[0] > 0,5|H_0)$ e é mostrado como a área mais escura na Figura 2.11 [51].

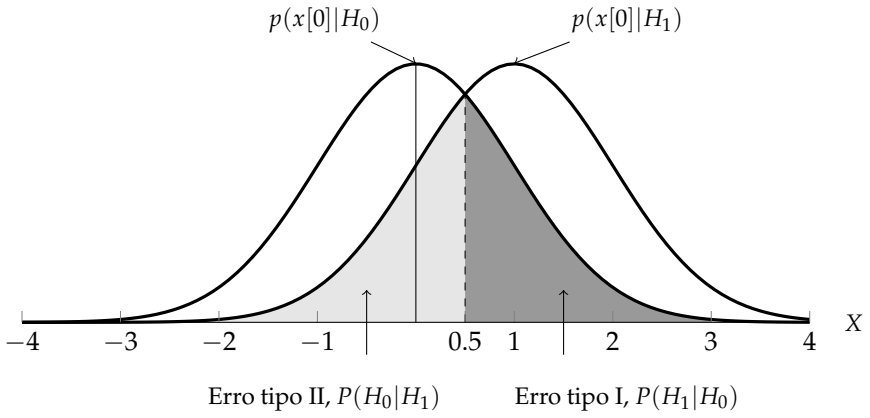


Figura 2.11: *Possíveis erros do teste de hipótese e suas probabilidades.*

Quando não é dada nenhuma probabilidade a priori, não é possível determinar a perda esperada ou a probabilidade de erro total, uma vez que não é possível estabelecer um teste para minimizar nenhuma destas quantidades. O princípio de verossimilhança pode ser usado como um critério no qual um teste pode ser baseado. Um outro critério, que se

aplica em diferentes circunstâncias, mas é provavelmente mais convincente, é manter a probabilidade de erro de um tipo menor ou igual a um valor pré-definido e minimizar a probabilidade de erro do outro tipo [52].

Erros do tipo I e erros do tipo II são inevitáveis, porém podem ser "negociados" entre si, uma vez que é possível deslocar o limiar para um dos lados, aumentando um dos erros e diminuindo o outro e vice-versa [51].

Para o sistema de identificação proposto nesta tese, o erro tipo I é referido como sendo a FAR e o erro tipo II é referido como a FRR. Para o sistema proposto por Kanade et al. [46], no qual é realizada a inserção de zeros, as curvas de H_0 (usuários genuínos) e de H_1 (usuários impostores) continuam sobrepostas para o intervalo do $1 \leq t_{RS} \leq 22$. Neste sistema é necessário definir o limiar de decisão, de modo a minimizar a FRR, mantendo a FAR menor que um valor aceitável pré-definido.

Para o sistema proposto por Câmara em [3], é realizada a inserção de uma sequência binária pseudo aleatória ao invés da sequência de zeros. Esta técnica proposta por Câmara é a técnica usada nesta tese. As variáveis aleatórias empregadas nesta tese são discretas, isto, aliado ao uso da técnica proposta por Câmara, para a faixa $1 \leq t_{RS} \leq 22$, resulta na separação das curvas de H_0 (usuários genuínos) e de H_1 (usuários impostores), não sendo necessário se preocupar com o limiar de decisão. As curvas para usuários genuínos e para usuários impostores podem ser vistas no Capítulo 3, nas seções 3.3.2 e 3.3.3.

Vale ressaltar que não foram realizados experimentos para valores de $t_{RS} > 22$. Caso o valor do t_{RS} seja aumentado acima de 22, passa a existir a possibilidade de interseção das curvas para usuários genuínos e para usuários impostores, sendo necessário utilizar o teste de hipótese e a definição do limiar de decisão.

2.9 BASES DE DADOS

Tipicamente, as bases de dados contêm um conjunto de imagens para cada usuário, no qual tem-se tanto imagens de referência (I_{ref}) como imagens de teste (I_{sam}). Uma imagem de referência é aquela que foi obtida em um ambiente controlado, enquanto que uma imagem de teste é aquela que foi obtida com um equipamento de identificação de um usuário, ou seja, em condições abaixo da ideal. De acordo com [46], para cada imagem na base de dados, uma sequência binária de comprimento 1.188 *bits* é obtida a partir de uma imagem infra-vermelho de uma íris. Essas sequências são denominadas de código íris. Os códigos

íris obtidos a partir das imagens de referência e a partir das imagens de teste são respectivamente representados por θ_{ref} e θ_{sam}. Nesta tese, os códigos íris usados em todos os testes foram obtidos a partir das seguintes bases de dados: BIOSECURE [8], CASIA [8] e NIST-ICE [1].

As bases de dados BIOSECURE e CASIA são formadas por 1.200 imagens cada, originadas de 60 usuários distintos, cada um com 20 imagens, onde 10 imagens são as imagens de referência e as outras 10 imagens restantes são as imagens de teste. Outra possibilidade de interpretação é considerar 30 usuários distintos, sendo 10 imagens de referência para o olho direito, 10 imagens de teste para o olho direito, 10 imagens de referência para o olho esquerdo, e 10 imagens de teste para o olho esquerdo. Usando estas duas bases de dados, é possível realizar 6.000 testes para os usuários genuínos em cada base de dados, considerando apenas uma íris de cada vez, ou 3.000 testes usando as íris de ambos os olhos, ao mesmo tempo.

A base de dados NIST-ICE é formada por 2.953 imagens, que são divididas em dois testes distintos, que são chamados de ICE-exp1 e ICE-exp2. O teste ICE-exp1 é composto por 124 usuários, totalizando 1.425 imagens e se refere ao olho direito, enquanto que o teste ICE-exp2 é composto por 120 usuários totalizando 1.528 imagens e se refere ao olho esquerdo. O ICE-exp1 permite realizar 12.214 testes para os usuários genuínos enquanto que o ICE-exp2 permite realizar 14.653 testes. Na base de dados NIST-ICE, o número de imagens por usuário não é fixo, nela existem usuários que possuem desde apenas 1 imagem até o máximo de 31 imagens. Consequentemente é necessário usar um arquivo de controle para verificar quais imagens poderão ser utilizadas em cada teste. Para os experimentos nesta base de dados, será definida a primeira imagem de um usuário como sendo a imagem de referência, tornando todas as demais imagens deste usuário como sendo imagens de teste. Ao concluir este passo, será escolhida a segunda imagem deste mesmo usuário como sendo a imagem de referência e esta será comparada com todas as demais imagens deste usuário. Os experimentos com este usuário terminam quando for escolhida a sua última imagem como sendo a imagem de referência. As bases de dados que possuem o mesmo número de imagens por usuário fixo, será chamada de *bases de dados regular* (BIOSECURE e CASIA), enquanto que as bases de dados cujo número de imagens por usuário não é fixo, será chamada de *base de dados irregular* (NIST-ICE). Na Tabela 2.1 é possível observar a quantidade de imagens por base de dados e a quantidade de testes que poderão ser realizados nas mesmas. O con-

junto completo de testes realizados em uma determinada base de dados será chamado de experimento.

Tabela 2.1: *Quantidade de testes realizados em cada base de dados.*

	BIOSECURE			CASIA			NIST-ICE			
							exp_1		exp_2	
	Usuários	Imagens		Usuários	Imagens		Usuários	Imagens	Usuários	Imagens
Uma íris	60	$10(I_{ref})$	$10(I_{sam})$	60	$10(I_{ref})$	$10(I_{sam})$	120	variável	124	variável
Testes	6.000			6.000			12.214		14.653	
	Usuários	Imagens		Usuários	Imagens		Usuários		Imagens	
Ambas as íris	30	$10(I_{ref})$	$10(I_{sam})$	30	$10(I_{ref})$	$10(I_{sam})$	69		variável	
Testes	3.000			3.000			6.229			

As informações mais detalhadas sobre as bases de dados e sobre os códigos corretores de erros usados nos sistemas propostos nesta tese, podem ser encontradas no Apêndice A.

2.10 A FUNÇÃO *hash*

A função *hash* [53] tem grande utilidade em segurança da informação, opera mapeando "grandes" domínios para "pequenos" domínios e é usada para garantir integridade dos dados e autenticação de mensagens.

As funções *hash* operam sobre uma mensagem como entrada e produzem uma saída que é chamada de código *hash*, resultado *hash*, valor do *hash*, ou simplesmente *hash*. Mais precisamente, uma função *hash* h(K) mapeia uma sequência de *bits* de um comprimento finito para uma outra sequência de comprimento também finito, de n_h *bits*. A ideia básica da função *hash* é que o valor do *hash* serve como uma representação compacta da sequência original, muitas vezes é chamada de impressão, impressão digital, ou síntese da mensagem, e pode ser usada como se fosse uma identificação única da sequência original [53].

As funções *hash* são usadas para garantir a integridade dos dados em conjunto com esquemas de assinatura digital, nos quais, por diversas razões uma mensagem tem seu *hash* calculado previamente e este é usado em substituição da mensagem original [53].

No sistema de identificação de usuário proposto nesta tese, a função *hash* é utilizada para garantir que a chave recuperada K' seja comparada com a chave original K, comparando-se o h(K') com o h(K), em caso de igualdade, o processo de identificação obteve sucesso. Esta comparação é obrigatória em um caso real de identificação de um usuário, porém, nos experimentos, esta comparação não foi realizada, pois devido ao fato de não necessitar

fazer a decodificação por RS, uma vez que é possível saber quantos erros ocorreram após a decodificação de Hadamard, é possível identificar se houve erro ou acerto no processo de identificação do usuário, sem a necessidade da comparação entre as duas funções *hash*.

2.11 CRIPTOGRAFIA DE CHAVE-SECRETA

Um esquema é dito ser de chave secreta se a chave usada na codificação é a mesma chave usada na decodificação. Este esquema, que pode ser visto na Figura 2.12, também é conhecido como sistema criptográfico simétrico, de chave única ou de chave secreta. Neste esquema, a chave secreta é usada para cifrar um texto claro e também é usada para decifrar uma mensagem criptografada, recuperando o texto claro [53].

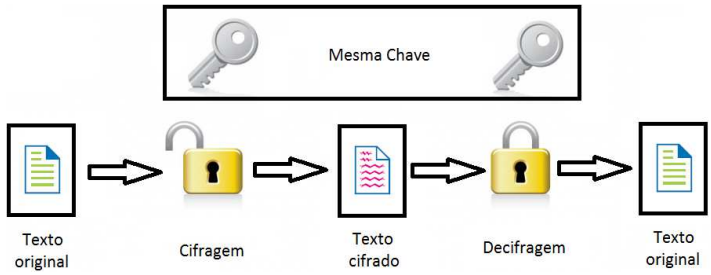


Figura 2.12: *Criptografia de chave secreta.*

2.11.1 O COMPRIMENTO DA CHAVE CRIPTOGRÁFICA

Um dos fatores determinantes para dificultar a descoberta da chave criptográfica é o seu comprimento, que irá definir o espaço das chaves. O espaço das chaves é o número de pares de cifragem/decifragem que existem disponíveis no sistema de cifragem. Uma chave é, normalmente, uma forma compacta de especificar uma transformação criptográfica que será utilizada [53].

Por exemplo, uma cifra de transposição de comprimento t_c, tem $t_c!$ possibilidades de ser selecionada. Cada uma delas pode ser descrita por uma permutação, que pode ser chamada de chave. É uma grande tentação relacionar a segurança do sistema criptográfico com a cardinalidade do espaço das chaves [53].

Um fato necessário, mas nem sempre suficiente, é fazer com que o espaço das chaves seja grande o suficiente para dificultar o ataque realizado por busca exaustiva, no qual todas as chaves são testadas até se encontrar qual delas é a correta [53].

Para o sistema de regeneração proposto, quanto maior o comprimento da chave criptográfica, maior será a segurança do sistema, por este motivo o t_{RS} será escolhido de modo a maximizar este comprimento, sem aumentar muito a FRR.

2.12 CÓDIGOS CORRETORES DE ERROS

Os códigos de íris são afetados por dois tipos de erros, que são os erros aleatórios e os erros em surto. Devido a esta característica, é necessário que sejam usados códigos corretores de erro que tenham a capacidade de lidar com ambos os tipos de erros. Hao et al. [39] foi o primeiro a propor uma combinação de códigos que pudesse lidar com esta característica. Na proposta de Hao [39] foi sugerido um sistema de regeneração de chave criptográfica, baseado no código de íris que usa a técnica de correção de erros concatenando os códigos de Reed-Solomon (RS) e os códigos de Hadamard. Os códigos de Hadamard são usados para lidar com os erros aleatórios, causados, por exemplo por ruído na câmera, distorção da íris, efeitos de captura de imagem que não possam ser corrigidos por fases de pré processamento, enquanto que os códigos de RS tem a capacidade de lidar com erros em surto, causados, por exemplo pelo piscar das pálpebras, pelos cílios ou por reflexões.

2.12.1 O CÓDIGO REED-SOLOMON

Os códigos de Reed-Solomon são uma sub classe dos códigos BCH q-ários. Esta sub classe é a mais importante dos códigos BCH q-ários e o nome foi dado em homenagem aos seus descobridores [54].

Os códigos de Reed-solomon com símbolos em GF(q) tem os seguintes parâmetros [54]:

- Comprimento de bloco: $n_{RS} = q - 1$;
- Número de dígitos de paridade: $n_{RS} - k_{RS} = 2 * t_{RS}$;
- Distância mínima: $d_{min} = 2 * t_{RS} + 1$.

Na Figura 2.13 [54] é possível observar o esquema usado para codificar uma mensagem com o código Reed-Solomon (RS). Trata-se de um registrador de deslocamento linear com realimentação (LFSR - *Linear Feedback Shift Register*). Os coeficientes de $g(x)$ são obtidos a partir dos coeficientes de

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3) \cdots (x - \alpha^{n_{RS} - k_{RS}}). \tag{2.5}$$

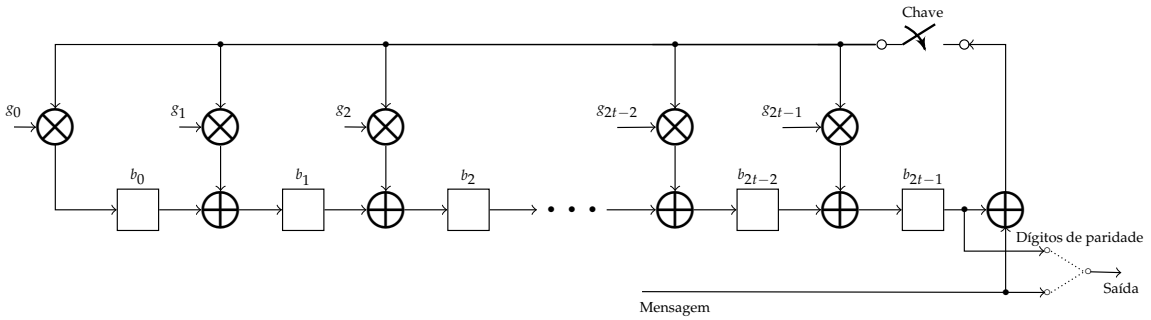


Figura 2.13: *Circuito para codificar um código cíclico não binário - LFSR codificador RS.*

A partir da equação 2.5 obtém-se os coeficientes de $g(x)$ que serão utilizados para calcular os termos b_j presentes na Figura 2.13. Mais detalhes sobre como implementar o código de Reed-Solomon estão disponíveis no Apêndice A.

2.12.2 O CÓDIGO DE HADAMARD

Uma matriz **H**=$\left[h_{ij}\right]_{n\times n}$ é uma matriz de Hadamard se $h_{ij} \in \{+1, -1\}$ para todo i, j e $\mathbf{HH^T} = n\mathbf{I}$, $n \in \mathbb{N}^*$ e **I** é a matriz identidade. Visto de forma apropriada, as colunas da matriz de Hadamard n x n formam um código binário de comprimento n, com n palavras código. Para enxergar esta informação, note que cada coluna de **H** é um vetor binário, no qual os símbolos são $\{+1, -1\}$ ao invés de $\{0, 1\}$. Um dos aspectos mais interessantes deste código é o fato de que, para todo n par, quaisquer duas palavras código tem distância de Hamming de exatamente $n/2$ entre si [55] e [54].

Dada uma matriz **H**, $n \times n$, o código de Hadamard de comprimento n, Had_n, é formado pelas palavras código das colunas de **H**, em que os $+1$'s são substituídos por 0's e os -1's são substituídos por 1's, e pelo complemento das colunas de **H**. Existem alguns métodos para encontrar a matriz de Hadamard e, consequentemente, o código de Hadamard [55] e [54]. No *software* em C++ é usado o método de Sylvester para construção da matriz de Hadamard. Mais detalhes sobre como implementar o código de Hadamard estão disponíveis no Apêndice A.

2.12.3 A CONCATENAÇÃO REED-SOLOMON-HADAMARD (RSH)

A codificação RSH

Para realizar o processo de codificação é necessário ter o vetor chave K, a partir do qual será gerada a codificação Reed-Solomon (RS). A matriz do código de Hadamard (Had) é obtida pelo método de Sylvester, detalhado no Apêndice A. Neste exemplo, por questões de simplificação: o código de Reed-Solomon empregado tem parâmetros iguais a (7,5,1); o código de Hadamard tem parâmetros iguais a (4,3,2); o vetor de embaralhamento foi escolhido todo nulo; K foi escolhido como uma sequência de 1 até 5; o usuário escolhido é o primeiro usuário da base de dados BIOSECURE; são inseridos zeros a cada 3 bits de dados; a imagem de referência é a terceira, centralizada; a imagem de teste é a primeira, também centralizada. Estas escolhas não afetam a generalidade do processo de reconhecimento e regeneração da chave criptográfica, apenas simplificam o processo para melhor entendimento do leitor.

Tem-se que:

$$K = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix};$$

$$RS = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 3 \end{bmatrix};$$

$$Had = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}.$$

O código RSH (θ_{ps}) é obtido a partir da concatenação do código RS com o código de Had, da seguinte maneira: a posição dada por cada coordenada do vetor RS é obtida partir da matriz Had, como por exemplo a primeira coordenada de RS é 1, portando a primeira

4-upla de θ_{ps} será a segunda linha da matriz Had, a matriz de Had é numerada desde a linha 0 até a linha 7, colocando-se zero no lugar do -1. Na Tabela 2.2 é possível ver este exemplo de codificação. O θ_{ref} e o θ_{sam}, são respectivamente os primeiros 18 *bits* da base de dados BIOSECURE do usuário 1, acrescidos de 10 zeros, dois zeros a cada 3 *bits* de dados.

Tabela 2.2: *Exemplo de codificação RSH para o usuário 1 - imagem de referência 3 (rotação: 11) com imagem de teste 1 (rotação: 11)*

Posição	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
θ_{ref}	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0
θ_{sam}	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	0	0
θ_{ps}	1	0	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	1	1	0	0	1
θ_{lock}	0	1	0	0	1	0	0	1	1	0	1	1	1	0	0	1	1	0	0	1	0	0	1	1	1	1	0	1
θ'_{ps}	1	0	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1
θ'_{ps}(-1)	1	-1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	1	1	1	-1	-1	1
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0

É possível observar que houve um erro, pois o vetor e possui um *bit* igual a 1 na posição 22, indicando que nesta posição houve uma diferença entre θ_{ps} e θ'_{ps}. Este erro precisa ser corrigido pelo processo de decodificação, caso o mesmo possua esta capacidade de correção, do contrário, um erro será computado.

A decodificação RSH

Para realizar a decodificação RSH, é necessário ter a matriz inversa de de Hadamard: Had_{inv}, que no caso das matrizes geradas a partir do método de Sylvester, são calculadas fazendo-se a matriz transposta de Hadamard: Had^t:

$$Had_{inv} = Had^t = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

O vetor θ'_{ps} é então separado de 4-uplas em 4-uplas e multiplicado por cada coluna da matriz Had_{inv}, calculando-se o valor da síndrome obtida é possível saber se houve erro e quantos erros ocorreram no total:

$$\theta'_{ps_{4x4}} * Had_{inv} = \begin{bmatrix} 0 & 4 & 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & -4 \\ -4 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 & 4 & 0 & 0 \\ 2 & -2 & -2 & -2 & -2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & -4 \end{bmatrix}.$$

O que se busca nesta operação matricial são as posições em que houve erros, as quais podem ser identificadas facilmente pelas coordenadas do vetor $\theta'_{ps_{4x4}} * Had_{invmax}$, nas posições em que o valor foi diferente do máximo 4, cada coordenada do vetor corresponde ao valor máximo obtido em cada linha da matriz, neste exemplo:
$\theta'_{ps_{4x4}} * Had_{invmax} = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 & 2 & 4 \end{bmatrix}$.

Deste modo, para o exemplo dado, é possível observar que houve um único erro, na sexta 4-upla, pois o valor máximo encontrado foi 2, quando o valor máximo esperado, para acerto, seria 4. Também é possível identificar que para este exemplo, ocorreu um único erro, o que poderá ser corrigido pela decodificação do código de Reed-Solomon, uma vez que o código RS empregado possui capacidade de correção de exatamente um erro.

Para os experimentos computacionais, por questão de ganho de tempo na decodificação, o processo de identificação de erros compara a quantidade de erros encontrada após a decodificação de Hadamard com a capacidade de correção de erros do código de Reed-Solomon, de modo que se a quantidade de erros encontrada após a decodificação por Hadamard for menor ou igual à capacidade de correção do código RS, um acerto é considerado. em caso contrário, um erro é decretado. A decodificação por RS não foi implementada. O processo de codificação é um processo de baixo custo computacional, sendo realizado, relativamente, em pouco tempo de processamento, enquanto que o processo de decodificação é de alto custo computacional, demorando muito tempo para ser concluído. Nos experimentos realizados, o processo de decodificação leva em média, cerca de 75% do tempo total de processamento.

Capítulo 3

Autenticação com busca por rotação

"O ignorante afirma, o sábio duvida, o sensato reflete."

— Aristóteles

Um novo teste chamado de "busca por rotação" é proposto para identificação de usuários e regeneração de uma chave criptográfica em sistemas que empregam a representação digital da íris *(iris code)*. Quando aplicado às bases de dados *Biometrics for Secure Authentication* (BIOSECURE), *Chinese Academy of Sciences* (CASIA) e *National Institute of Standards and Technology - Iris Challenge Evaluation* (NIST-ICE) a busca por rotação mostrou, na média, uma redução de duas vezes na taxa de falsa rejeição FRR, com a taxa de falsa aceitação - FAR igual a zero, em comparação com o método de busca padrão empregado em outros sistemas. A melhoria mais significativa alcançada na FRR pela busca por rotação, se comparado com a busca padrão, foi de cerca de 100 vezes para uma única íris, para $t_{RS} = 15$, na base CASIA e de cerca de 85 vezes para ambas as íris, para $t_{RS} = 10$, na base NIST-ICE, e em muitos casos, a medição da FRR foi igual a zero. Neste Capítulo, será mostrado como é possível obter este ganho tão significativo.

3.1 SISTEMA DE REGENERAÇÃO DE CHAVE PROPOSTO

A Figura 3.1 apresenta o diagrama de blocos para o sistema de regeneração de chave proposto neste Capítulo. Este sistema será utilizado para realizar os experimentos de recuperação de chave, a partir das imagens das íris, disponíveis nas bases de dados BIOSECURE, CASIA e NIST-ICE. Este sistema proposto aqui é essencialmente o sistema que foi proposto em [2] exceto pela inserção de números pseudo aleatórios [3] ao invés de zeros e pelo bloco "Concatenação do código íris", o qual realiza a concatenação das duas íris, quando o experimento usa ambas as íris. Quando o experimento usa apenas uma íris, o bloco "Concatenação do código íris" irá apenas copiar o conteúdo da sua entrada para a sua saída. Os detalhes do sistema proposto são descritos a seguir.

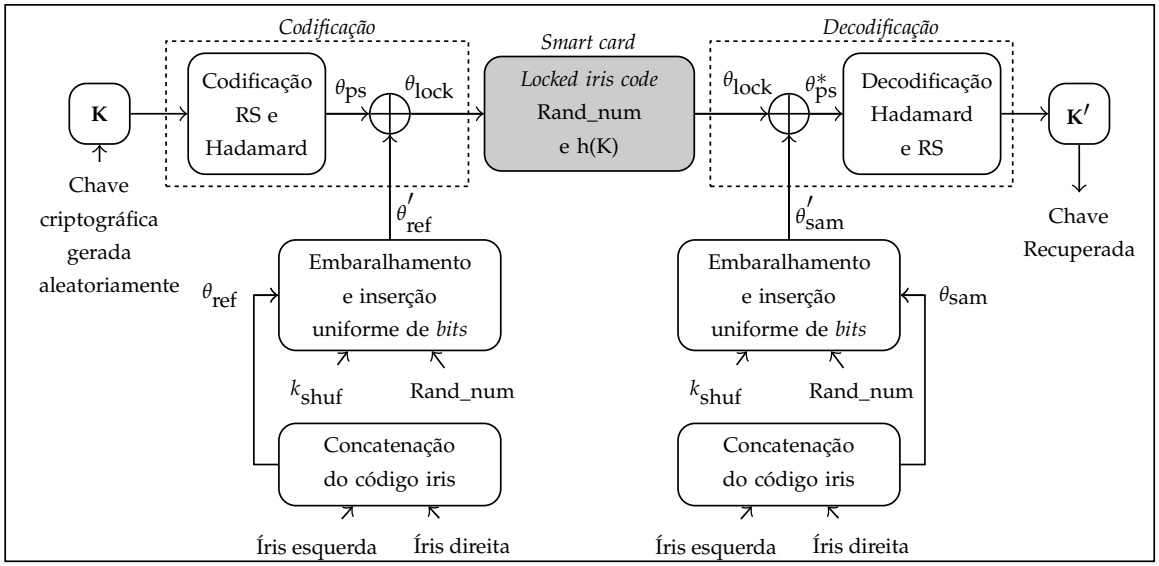


Figura 3.1: *Sistema de regeneração de chave uni ou multi biométrico, empregando smart card, íris e senha.*
Fonte: o autor, adaptado de [3]

Como ilustrado na Figura 3.1, a codificação da chave criptográfica **K** é feita sequencialmente, realizando primeiro a codificação usando o código de Reed Solomon (RS) [56, p.294] e em seguida codificando usando o código de Hadamard $(2^k, k+1, 2^{k-1})$ [56, p.44]. Do mesmo modo que [2], para o caso em que apenas uma íris é empregada, será utilizado o código de RS encurtado, de comprimento de bloco $n_{RS} = 61$ sobre GF(2^6) e o código de Hadamard $(32, 6, 16)$. Para o caso em que ambas as íris são empregadas, será utilizado o código de RS encurtado, de comprimento de bloco $n_{RS} = 61$ sobre GF(2^7) e o código de Hadamard $(64, 7, 32)$, o mesmo usado em [3]. A capacidade de correção de erros t_{RS} do código

RS é ajustada na faixa que varia de $1 \leq t_{\text{RS}} \leq 22$ satisfazendo a relação $k_{RS} = 61 - 2t_{\text{RS}}$, em que k_{RS} denota o número de símbolos de informação do código RS. Valores de t_{RS} maiores que 22 são evitados devido ao fato de que poderá ocorrer um aumento da FAR, ou seja, para $t_{\text{RS}} > 22$ o esquema pode começar a erroneamente considerar alguns impostores como sendo usuários genuínos.

A operação de embaralhamento, de acordo com [2], consiste em segmentar os 1.188 *bits* do código íris em 198 blocos de 6 *bits* cada, e reordená-los usando a chave de embaralhamento (k_{shuf}) gerada pseudo-aleatoriamente, de comprimento 198 *bits*. A reordenação dos blocos (operação de embaralhamento) pode ser descrita por meio de uma analogia com o embarque de passageiros em um avião. Originalmente os 198 blocos (passageiros) formam uma fila única e, sequencialmente, para cada bloco (passageiro) que está na fila, é dada uma ficha que pode ser 1 ou 0, obtida lendo-se sequencialmente os *bits* da chave de embaralhamento. Então, respeitando a ordem original dos blocos na fila, duas novas filas são formadas. Uma fila é formada pelos blocos que receberam a ficha 1 (passageiros da classe A) e outra fila é formada pelos blocos que receberam a ficha 0 (passageiros da classe econômica). Obedecendo a ordem de chegada, os blocos com a ficha 1 embarcam primeiro, seguidos pelos blocos com a ficha 0, ou seja, a sequência embaralhada contém primeiro os blocos que receberam a ficha 1, seguidos pelos blocos que receberam a ficha 0. Quando ambas as íris são empregadas, o bloco "Embaralhamento e inserção uniforme de *bits*" faz o mesmo procedimento descrito, com exceção de dois detalhes. Primeiro, $2 \times 1.188 = 2.376$ *bits* são empregados, que é o resultado da concatenação do código da íris do olho esquerdo com o código da íris do olho direito, e segundo, $k_{\text{shuf}} = 396$ *bits* são empregados para manter o comprimento de cada bloco da operação de embaralhamento igual a 6.

A inserção uniforme de *bits* consiste em concatenar sequências de blocos formadas por três *bits* do código íris seguida por dois *bits* de uma sequência aleatória ou pseudo aleatória. Uma vez que o código íris consiste de uma sequência binária de comprimento 1.188 *bits*, após o procedimento de inserção de *bits*, uma sequência de comprimento 1.980 *bits* é obtida. Devido ao fato de que o comprimento do bloco de θ_{ps} é $61 \times 32 = 1.952$, é necessário apagar 28 *bits* da sequência de comprimento 1.980 para obter uma sequência de comprimento 1.952 para θ'_{ref}. Dos 28 *bits* apagados, 16 *bits* são provenientes do código íris e 12 *bits* são provenientes da sequência inserida. Os *bits* perdidos do código íris correspondem a, aproximadamente, $1,35\%$ do total, e foi verificado que a capacidade de correção de

erros do esquema não é afetada significativamente por esta perda. Quando ambas as íris são empregadas, o código íris é uma sequência binária de comprimento 2.376 *bits*, e após a operação de inserção de *bits*, tem-se uma sequência de 3.960 *bits*. Entretanto, para o código de Hadamard com $k = 6$, o comprimento de θ_{ps} é $61 \times 64 = 3.904$, sendo necessário apagar os últimos 56 *bits* da sequência de comprimento 3.960, para obter uma sequência de comprimento 3.904 para θ'_{ref}. Dos 56 *bits* apagados, 33 *bits* são provenientes do código íris e 23 *bits* são provenientes da sequência inserida. Os *bits* perdidos do código íris correspondem a aproximadamente $1,3900\%$ do total, e, novamente, foi verificado que a capacidade de correção de erros do esquema não é afetada significativamente por esta perda.

A decodificação se inicia pelo código de Hadamard, o que significa que para cada palavra código de 6 *bits*, um *byte* é entregue como um símbolo de uma palavra código do código RS para uma única íris. Quando duas íris são usadas, o decodificador para o código de Hadamard (64,7,32) entrega uma palavra de 7 *bits* para formar um símbolo do código RS. Um artifício [2] que acelera o tempo da simulação computacional é então empregado. Uma vez que são conhecidas as palavras código do código de RS que foram geradas, estas são comparadas com as palavras que saem do decodificador do código de Hadamard e são contados o número de símbolos com erros. Este número é representado por, t, e é, então, comparado com o número de erros t_{RS} que é a capacidade de correção de erros do código de RS. Se $t \leq t_{\text{RS}}$, então, é possível recuperar a chave criptográfica $\mathbf{K}$, evitando a decodificação do código de RS e reduzindo o tempo de processamento da simulação.

3.2 NOVO TESTE PROPOSTO

Nesta seção, será proposto um novo teste de identificação de usuários chamado busca por rotação. Antes de apresentar o novo teste proposto, será descrito o teste usado em [46] e [2], que será chamado de busca padrão, o qual, será comparado com o teste de busca por rotação. É necessário enfatizar que os testes usam o código íris θ_{ref} e θ_{sam} em vez do respectivo I_{ref} e I_{sam}, como indicado no diagrama de blocos da Figura 3.1.

3.2.1 BUSCA PADRÃO

Para poder testar um sistema de regeneração de chave baseado na íris, ambas I_{ref} e I_{sam} são usadas para cada usuário. Além disto, para cada imagem armazenada, seja uma imagem de referência ou uma imagem de teste, as bases de dados armazenam 20 versões

rotacionadas de cada imagem, ou seja, um total de 21 imagens. Será denotado por $I_{\text{ref}}(r, i, u)$ a r-ésima versão rotacionada da imagem de referência de número i, pertencente ao usuário u, para $1 \leq r \leq 21$, $1 \leq i \leq N$, e $1 \leq u \leq U$, onde N é o número máximo de imagens de referência e U é o número máximo de usuários. Similarmente, será escrito $I_{\text{sam}}(r, j, u)$, $1 \leq j \leq M$ para o número das imagens de teste, onde M é o número máximo de imagens de teste. Cada comparação entre duas imagens será chamada de teste, e o conjunto de testes para todos os usuários, será chamado de experimento.

Para os sistemas em [2], [3], os testes são realizados para cada usuário u escolhendo uma imagem de referência para $r = 11$, ou seja $I_{\text{ref}}(11, i, u)$, e comparando-a com até 21 versões de uma imagem correspondente de teste $I_{\text{sam}}(r, j, u)$, $1 \leq r \leq 21$. Se uma identificação positiva ocorrer quando estiver testando a imagem $I_{\text{sam}}(r, j, u)$, então o teste com a imagem j é interrompido e é computado um acerto. Por outro lado, se nenhuma identificação positiva for obtida para $1 \leq r \leq 21$, então um erro de identificação é computado para a imagem de teste j; e, se $j < M$, então a imagem de teste $I_{\text{sam}}(r, j+1, u)$ será a próxima a ser comparada com $I_{\text{ref}}(11, i, u)$. Quando o valor $j = M$ for alcançado, o teste com a imagem $I_{\text{ref}}(11, i, u)$ foi concluído, a imagem de referência $I_{\text{ref}}(11, i+1, u)$ é, então selecionada para poder continuar os testes, o que acontece de maneira similar ao que foi feito com a imagem de referência i. Quando todas as imagens de referência para o usuário u tiverem sido selecionadas, ou seja, quando $i = N$, e não houver mais nenhuma imagem de teste para o usuário u, ou seja, quando $j = M$, os testes com o usuário u estarão concluídos; então o usuário $u + 1$ será selecionado, juntamente com as suas imagens de referência e imagens de teste para que o experimento possa continuar. O experimento terminará quando os testes com o usuário $u = U$ forem completados. A partir deste ponto, este procedimento será referido como busca padrão o qual é ilustrado na Figura 3.2.

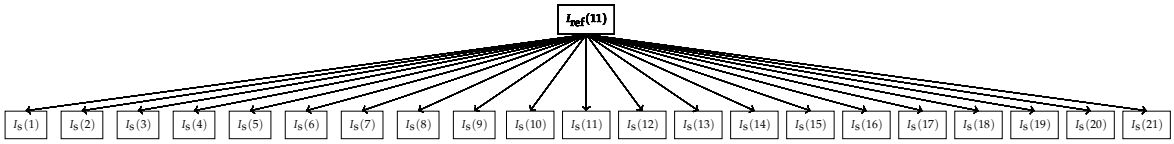


Figura 3.2: *Técnica de busca padrão, em que* $I_s(r) = I_{\text{sam}}(r)$, $1 \leq r \leq 21$.

É importante destacar que, no procedimento de busca padrão até 21 rotações de cada imagem de teste $I_{\text{sam}}(r, j, u)$, $1 \leq r \leq 21, 1 \leq j \leq M$, são realizadas para cada imagem de referência $I_{\text{ref}}(11, i, u)$, $1 \leq i \leq N$, e nenhuma rotação das imagens de referência é realizada.

Resumindo, na busca padrão, 21 rotações são realizadas para cada uma das M imagens de teste, para N imagens de referência e para U usuários.

3.2.2 BUSCA POR ROTAÇÃO

A técnica de busca por rotação realiza a busca para identificar um usuário realizando tanto as rotações das imagens de referência $I_{\text{ref}}(r, i, u)$ como das imagens de teste $I_{\text{sam}}(r, j, u)$. Um esboço do procedimento realizado pela busca por rotação será descrito a seguir. A Figura 3.3 ilustra, para os usuários escolhidos u^*, alguns testes que usam as imagens de referência i, $1 \leq r \leq 21$, ou seja, $I_{\text{ref}}(r, i, u^*)$, para $r \in \{1, 6, 11, 16, 21\}$. Em geral, até 441 testes para cada imagem de teste podem ser executados para verificar autenticidade, com o uso de até 21 versões rotacionadas de cada imagem de referência e até 21 versões rotacionadas de cada imagem de teste. Claramente, muito mais situações são consideradas quando é usada a busca por rotação em comparação com a busca padrão e, como consequência, há um acréscimo no tempo de processamento necessário para execução do experimento. É importante enfatizar que todas as bases de dados utilizadas aqui, contêm todas as rotações das imagens de teste e, também, das imagens de referência necessárias para executar a busca por rotação, ou seja, nenhum dado novo precisou ser criado para conseguir executar o novo sistema proposto.

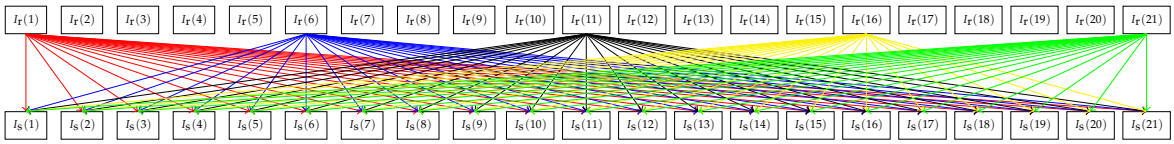


Figura 3.3: *Exemplo, em que* $I_r(r) = I_{\text{ref}}(r)$, $1 \leq r \leq 21$, $I_s(r) = I_{\text{sam}}(r)$, $1 \leq r \leq 21$, *para* $I_{\text{ref}}(r) : r = 1; r = 6; r = 11; r = 16; r = 21$, *para o sistema proposto busca por rotação.*

A busca por rotação consiste em, sistematicamente, comparar um par de imagens onde a primeira é uma imagem de referência ou uma de suas versões rotacionadas $I_{\text{ref}}(r, i, u)$, e, a outra imagem do par, é uma imagem de teste ou uma de suas versões rotacionadas $I_{\text{sam}}(r, j, u)$, onde $1 \leq r \leq 21$, $1 \leq i \leq N$, $1 \leq j \leq M$ e $1 \leq u \leq U$. Por exemplo, para as bases de dados BIOSECURE ou CASIA, um experimento emprega $N = 10$ imagens de referência por usuário, $M = 10$ imagens de teste por usuário e $U = 60$ usuários distintos. Então, um total de $10 \times 10 \times 60 = 6.000$ testes são realizados, e para cada teste, no pior caso, um máximo de $r \times r = 21 \times 21 = 441$ verificações são realizadas quando todas as

versões das rotações de ambas das imagens de referência e das imagens de teste são necessárias. Deste modo, o tempo de simulação necessário para a busca por rotação é aumentado quando as versões rotacionadas das imagens de referência precisam ser testadas. Quando ambas as íris são usadas, para as bases de dados BIOSECURE ou CASIA um experimento que emprega $N = 10$ imagens de referência por usuário, $M = 10$ imagens de teste por usuário e $U = 30$ usuários distintos. Então, um total de $10 \times 10 \times 30 = 3.000$ testes são realizados, e, para cada teste, no pior caso, um máximo de $r \times r = 21 \times 21 = 441$ verificações são realizadas quando todas as versões rotacionadas de ambas das imagens de referência e das imagens de teste são necessárias.

3.3 EXPERIMENTOS E RESULTADOS USANDO A BUSCA POR ROTAÇÃO

Nesta seção, será comparada a busca padrão com a busca por rotação, em termos da eficiência de implementação. Além disto, será analisado o desempenho do sistema de regeneração de chave proposto, em termos dos valores da FAR e da FRR, e compará-los com os resultados obtidos em [2].

3.3.1 EFICIÊNCIA DA IMPLEMENTAÇÃO

Este Capítulo segue a pesquisa desenvolvida em [46], [3] e [2] com novas contribuições. O *software* empregado em [2] e [3] foi desenvolvido na linguagem de programação proprietária do MATLAB®, a qual já vem com uma série de rotinas e operações matemáticas pré-programadas. O *software* empregado nesta tese foi desenvolvido em linguagem de programação C++. Os dados disponíveis nas bases de dados estão no formato de tabelas compactadas do MATLAB®, as quais não permitem acesso de modo rápido por um *software* desenvolvido externamente ao MATLAB®. Por esta razão, o código íris que está nas bases de dados precisou ser convertido para um novo formato, que possibilitasse acesso mais eficiente usando C++. Deste modo, 21 arquivos foram convertidos para cada base de dados, mais três arquivos de controle usados na base de dados NIST-ICE. Estes arquivos de controle definem quem são os usuários utilizados nos testes, quais e quantas são as imagens que pertencem a cada usuário.

É possível afirmar que o tempo de processamento aumenta, consideravelmente, de acordo com o número de falhas de decodificação, que se torna maior quando se tem baixa quali-

dade nas imagens da íris dos usuários. Para uma imagem de referência $I_{\text{ref}}(r_1, i, u)$, o procedimento de busca é interrompido logo que uma identificação positiva é obtida entre esta e uma imagem de teste ou uma de suas rotações $I_{\text{ref}}(r_2, j, u)$, ou quando todas as comparações tenham sido testadas sem nenhum sucesso. A busca por rotação, então, seleciona outra versão rotacionada da mesma imagem de referência $I_{\text{ref}}(r'_1, i, u)$ para continuar o processo de comparação com as imagens de teste rotacionadas. Após todas as versões de uma dada imagem de referência tiverem sido testadas e não for obtida uma identificação positiva, então um erro é declarado. A busca por rotação, então, seleciona outra imagem de referência para continuar com os testes. Quando são realizados os testes com a busca por rotação, observa-se que o menor número de erros de identificação ocorre na base de dados NIST-ICE(exp1), seguida pela base de dados CASIA, BIOSECURE e NIST-ICE(exp2). A base de dados BIOSECURE é a base regular que apresentou o pior resultado para a busca por rotação dentro das bases de dados regulares disponíveis. Por esta razão, foi escolhida a base de dados BIOSECURE para mensurar a eficiência da implementação em C++. A razão pela qual a base de dados NIST-ICE não foi escolhida é devido ao fato de ser uma base de dados irregular.

Para uma única íris, o resultado obtido com a implementação usando a busca padrão é apresentado na Tabela 3.1, a qual mostra o tempo necessário em segundos para realizar os experimentos, assim como o número de testes para 60 usuários, com 10 imagens de referência e 10 imagens de teste, para $1 \leq t_{\text{RS}} \leq 22$. Foi considerado o tempo médio gasto pela repetição de três experimentos em cada implementação.

Tabela 3.1: *Resultados da simulação usando a busca padrão para a base de dados BIOSECURE implementado em C++.*

	C++
Tempo(s)	598
Testes	132.000
Testes/s	220,7

A maneira como o *software* foi implementado para realizar a busca por rotação permitiu ter acesso a diversos dados intermediários de vários estágios do experimento. Deste modo foi possível verificar quais imagens de usuários apresentavam mais erros e, consequentemente, saber quais são os usuários que são mais difíceis de serem identificados. Foi observado, nos testes realizados, que a maior quantidade de identificações positivas ocorria

quando as imagens de referência $I_{\text{ref}}(r,i,u) = I_{\text{ref}}(11,i,u)$ foram comparadas com as imagens de teste mais centrais, que possuam r próximo ou igual a 11, tais como $I_{\text{sam}}(10,j,u)$, $I_{\text{sam}}(11,j,u)$ e $I_{\text{sam}}(12,j,u)$, não necessariamente nesta ordem. Esta observação foi levada em consideração na implementação usando a busca por rotação, realizando cada teste primeiramente com as imagens centrais e, se necessário, continuar o teste usando as imagens mais distantes do centro ($r = 11$). Como resultado, foi produzida a Tabela 3.2, onde é possível verificar os resultados obtidos para a busca padrão e para a busca por rotação.

Tabela 3.2: *Resultados comparativos usando a base de dados BIOSECURE, onde a marca (*) significa que a implementação empregou a busca centralizada.*

	Padrão* (C++)	Rotação* (C++)
Tempo(s)	253	1.936
Testes	132.000	132.000
Testes/s	521,7	68,2

A busca centralizada, ou seja, a busca por rotação no qual compara-se primeiro as imagens mais centrais e segue até as imagens mais rotacionadas obedece a seguinte ordem: $I_{\text{sam}}(11,j,u)$, $I_{\text{sam}}(11-l,j,u)$, $I_{\text{sam}}(11+l,j,u)$, $1 \leq l \leq 10$. A mesma sequência de busca empregada em $I_{\text{sam}}(r,j,u)$ é também empregada nas imagens de referência $I_{\text{ref}}(r,i,u)$, $1 \leq r \leq 21$. Esta modificação na ordem de busca reduziu o tempo gasto em todo o experimento por um fator maior do que dois. Por outro lado, a busca padrão implementada no MATLAB® seleciona as imagens de teste $I_{\text{sam}}(r,j,u)$, $1 \leq r \leq 21$, em ordem crescente, iniciando na imagem de teste $I_{\text{sam}}(1,j,u)$ e terminando na imagem de teste $I_{\text{sam}}(21,j,u)$.

3.3.2 Desempenho do sistema proposto para uma única íris

A Tabela 3.3 apresenta três colunas para cada base de dados. A primeira coluna de cada base de dados contém o resultado obtido com o *software* em C++ da taxa FRR, executando a busca padrão como implementado em [3], e serve como referência para comparação entre o resultado obtido aqui e aquele obtido em [2]. A segunda coluna de cada base de dados contém os resultados obtidos em [2], e a terceira coluna de cada base de dados contém os resultados obtidos com a nova proposta de busca por rotação implementada em (C++).

A Tabela 3.3 mostra que em todos os experimentos realizados, para vários valores de t_{RS} e para todas as bases de dados consideradas, a busca por rotação consistentemente mostrou

Tabela 3.3: *Percentual da taxa FRR para a busca padrão (C++), Kanade et al. [2] e busca por rotação (C++). A taxa FAR é sempre igual a zero para a implementação com busca por rotação.*

t_{RS}	BIOSECURE V1			CASIA V2			ICE-exp1			ICE-exp2		
	Padrão	Kanade	**Rotação**	Padrão	Kanade	**Rotação**	Padrão	Kanade	**Rotação**	Padrão	Kanade	**Rotação**
1	30,7900	30,5300	**15,1500**	50,1900	49,7000	**23,1100**	48,7900	49,3900	**21,3700**	52,0100	52,9900	**24,3400**
2	22,1500	22,1200	**11,0400**	36,1000	35,7800	**15,0900**	34,3000	33,2600	**13,7500**	37,3700	37,7400	**16,1700**
3	16,5200	16,3700	**8,6300**	26,0800	26,2700	**10,8200**	23,9700	24,2600	**9,5300**	27,1200	25,7800	**11,4400**
4	13,1200	12,8800	**7,4900**	18,9900	19,2500	**7,8100**	17,1200	16,5000	**6,9700**	19,9500	20,1000	**8,7300**
5	10,7500	10,6500	**6,6300**	14,5500	14,8200	**5,8500**	12,6000	12,6700	**5,2300**	15,1000	16,2500	**6,5900**
6	9,3200	8,9800	**5,9900**	11,4600	11,7000	**4,4200**	9,3400	10,3100	**3,8300**	11,6500	11,8100	**4,8900**
7	8,3400	8,3500	**5,3900**	9,1300	9,5200	**3,2900**	7,1300	7,2900	**2,7900**	9,2700	9,4200	**3,6900**
8	7,4600	7,2700	**4,7400**	7,3800	7,3200	**2,4900**	5,5000	5,9300	**2,1500**	7,1900	7,7700	**2,6900**
9	6,7100	6,6000	**4,0600**	5,7900	5,9700	**1,6900**	4,2700	4,6100	**1,5600**	5,7300	6,2600	**1,9300**
10	6,0800	5,8700	**3,4000**	4,6500	4,8500	**1,0900**	3,3200	3,6300	**1,1600**	4,4200	4,5400	**1,3800**
11	5,3500	5,2800	**2,7400**	3,7500	3,7700	**0,6400**	2,4800	2,4800	**0,8300**	3,4000	3,4900	**0,9100**
12	4,6800	4,5700	**2,0000**	2,8300	3,1300	**0,2800**	1,8200	2,1300	**0,6200**	2,5700	3,0500	**0,6300**
13	4,0400	3,9700	**1,5900**	2,1300	2,1200	**0,1400**	1,3800	1,4600	**0,3900**	1,8800	2,1200	**0,4600**
14	3,2500	3,2500	**0,9500**	1,4900	1,5700	**0,0300**	1,0600	1,0400	**0,2400**	1,4100	1,4100	**0,3500**
15	2,5400	2,6700	**0,6000**	1,0000	1,0700	**0,0100**	0,8000	0,7600	**0,1600**	1,0800	1,0900	**0,2600**
16	1,9800	2,0000	**0,3600**	0,6000	0,6300	**0**	0,5800	0,6900	**0,1200**	0,7700	0,9400	**0,1800**
17	1,4100	1,4300	**0,1900**	0,3900	0,3000	**0**	0,4600	0,4700	**0,0800**	0,5500	0,6100	**0,1000**
18	0,9700	1,0000	**0,0600**	0,2000	0,2500	**0**	0,2900	0,3800	**0,0500**	0,3900	0,4600	**0,0800**
19	0,6100	0,6300	**0,0400**	0,1200	0,1500	**0**	0,2200	0,2600	**0,0400**	0,3100	0,3900	**0,0300**
20	0,2700	0,4200	**0**	0,0500	0,0500	**0**	0,1600	0,1500	**0,0300**	0,2500	0,2900	**0,0100**
21	0,1700	0,2300	**0**	0,0300	0,0300	**0**	0,1100	0,1300	**0,0200**	0,1900	0,2000	**0,0100**
22	0,0700	0,1300	**0**	0,0100	0	**0**	0,0800	0,1100	**0,0100**	0,1300	0,1300	**0,0100**

melhores resultados do que a busca padrão. Para $t_{RS} = 15$ na base de dados CASIA, o esquema proposto obteve o seu melhor desempenho, de cerca de 100 vezes melhor do que o obtido em [2]. É importante enfatizar que para todos os valores de t_{RS}, a FAR é sempre zero usando a busca por rotação enquanto que para o sistema usado em [2], a FAR é maior do que zero para $t_{RS} \geq 10$.

Na Figura 3.4, é possível ver o gráfico em barras que ilustra um resumo do resultado obtido na Tabela 3.3. A barra titulada busca por rotação representa a média aritmética de todos os resultados da %FRR, considerando todas as bases de dados para cada t_{RS}. A barra titulada "Kanade [2]" representa a média aritmética de todos os resultados da %FRR, considerando todas as bases de dados para cada t_{RS}. Finalmente, a barra titulada Autenticação com busca por rotação representa a média aritmética de todos os resultados da %FRR, considerando todas as bases de dados para cada t_{RS}. Quando t_{RS} aumenta, a %FRR diminui, e em alguns casos é zero ou quase zero.

A Figura 3.5 ilustra a situação na qual nenhuma técnica de separação de usuários é empregada e a Figura 3.6 ilustra a separação de usuários resultante da aplicação da chave de

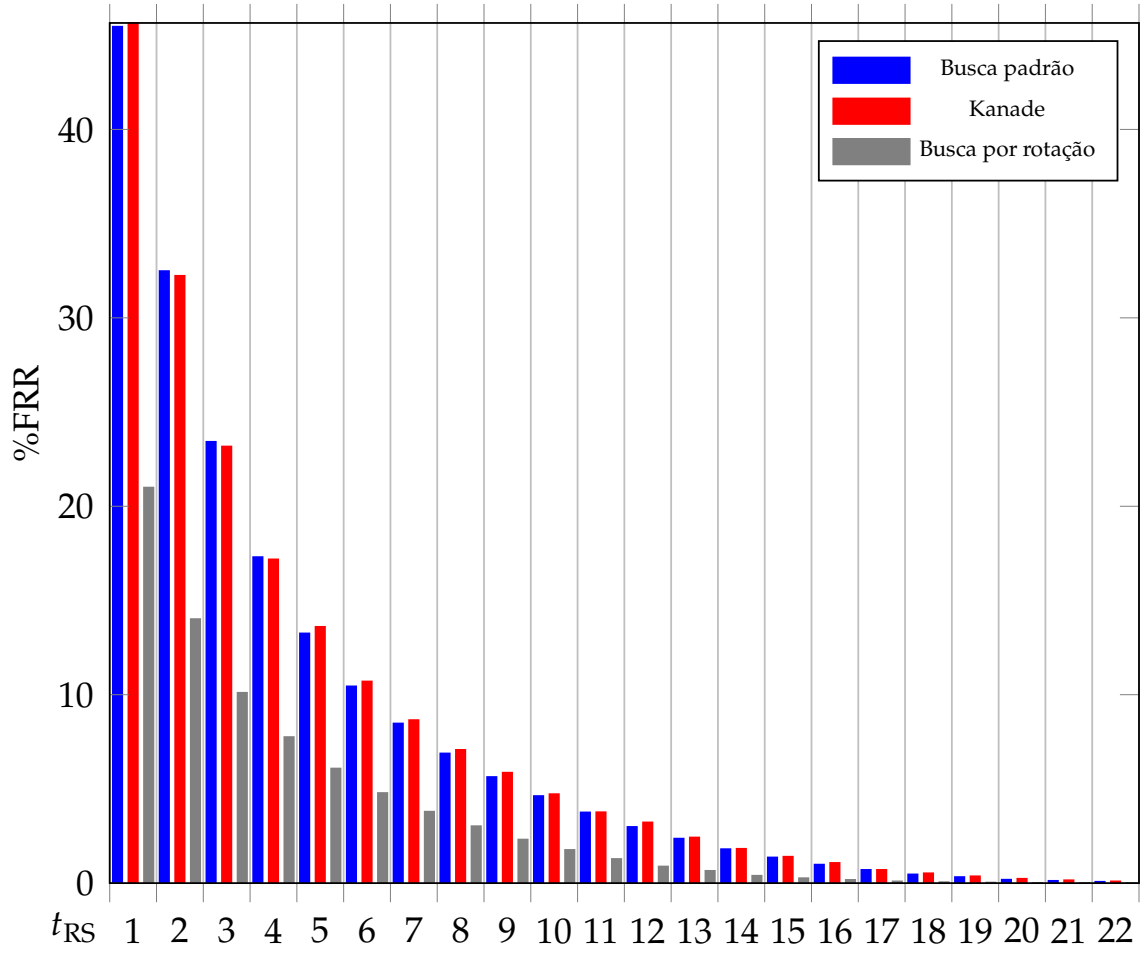


Figura 3.4: *Diagrama de Barras ilustrando o percentual médio da FRR da Tabela 3.3 versus t_{RS}, para a busca padrão, Kanade [2] e busca por rotação.*

embaralhamento (k_{shuf}), dos números aleatórios e da busca por rotação. Nas Figuras 3.5 e 3.6, é possível observar no eixo das abscissas a distância de Hamming normalizada. A distância de Hamming normalizada, entre duas palavras do código íris, é calculada fazendo-se a divisão da distância de Hamming entre duas palavras do código íris, e depois dividindo o valor encontrado pela quantidade total de bits da sequência analisada. É possível afirmar que a Figura 3.6 é muito similar àquela obtida em [3], o que significa que a inclusão da chave de embaralhamento (k_{shuf} ou *shuffling key*) no esquema de regeneração de chave criptográfica não contribuiu significativamente para a separação de usuários. Vale ressaltar que a função principal da chave de embaralhamento é separar a sequência original do código de íris, buscando difundir os erros e melhorar a capacidade de correção de erros do sistema. É importante enfatizar que a redução significativa da FRR obtida aqui, em comparação com o

sistema em [2], é devida essencialmente à busca por rotação em combinação com as técnicas de correções de erros.

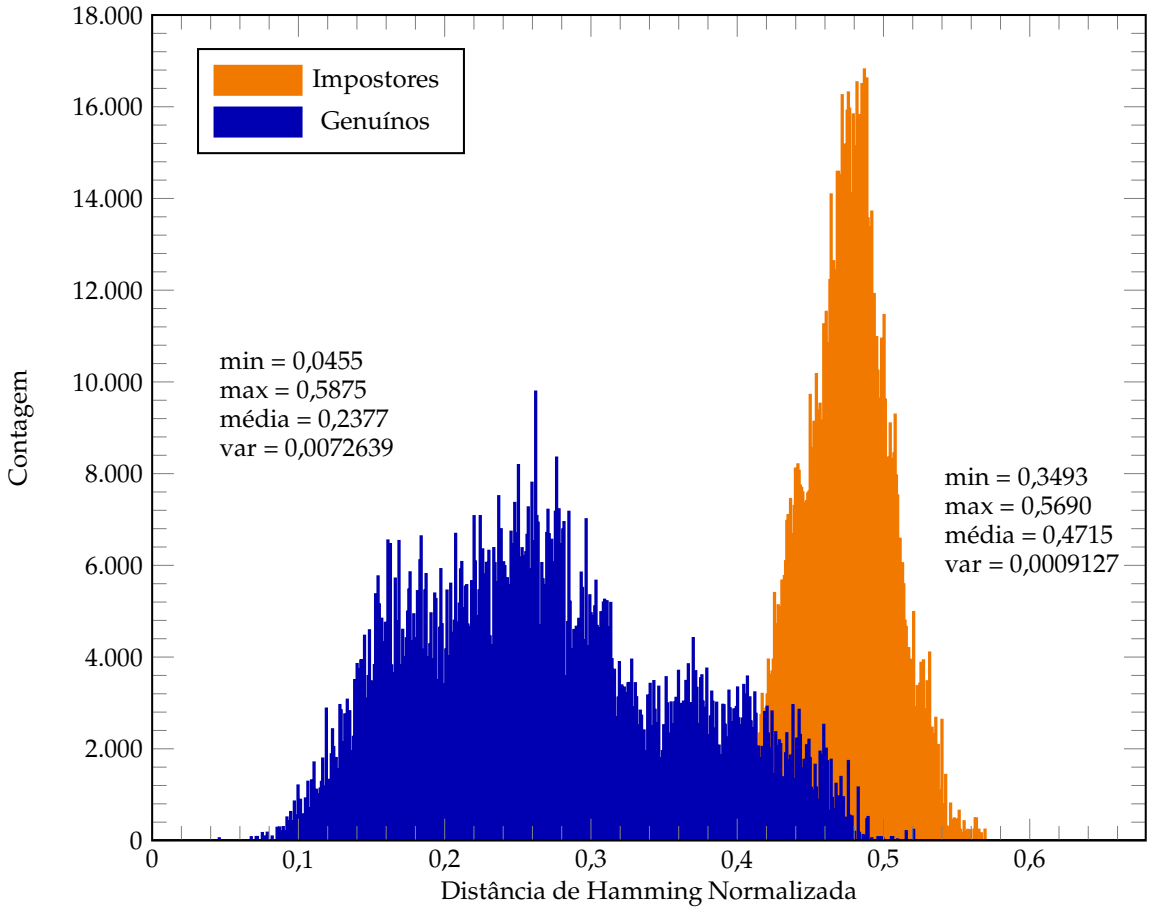


Figura 3.5: *Distância de Hamming normalizada para usuários genuínos e impostores para a base de dados BIOSECURE usando apenas uma íris.*

3.3.3 Desempenho do sistema proposto para ambas as íris

Para ambas as íris, o bloco titulado "Concatenação do código íris" realiza a concatenação de ambos os códigos das íris, o código da íris esquerda e o código da íris direita. Então, θ_{ref}, uma sequência de 2.376 *bits* é enviada para o próximo bloco do sistema proposto, ilustrado na Figura 3.1. Alguns ajustes foram necessários para que o sistema funcionasse como desejado. Então, os valores $m = 7$ e $k_{\text{shuf}} = 396$ *bits* foram empregados, e será utilizado o código encurtado RS [55] de comprimento de bloco 61 sobre GF(2^7) e o código de Hadamard $(64, 7, 32)$ [55]. Os valores de t_{RS} maiores que 22 são evitados porque eles aumentam a FAR, ou seja, para $t_{\text{RS}} > 22$ o esquema começa a, erroneamente, considerar um impostor

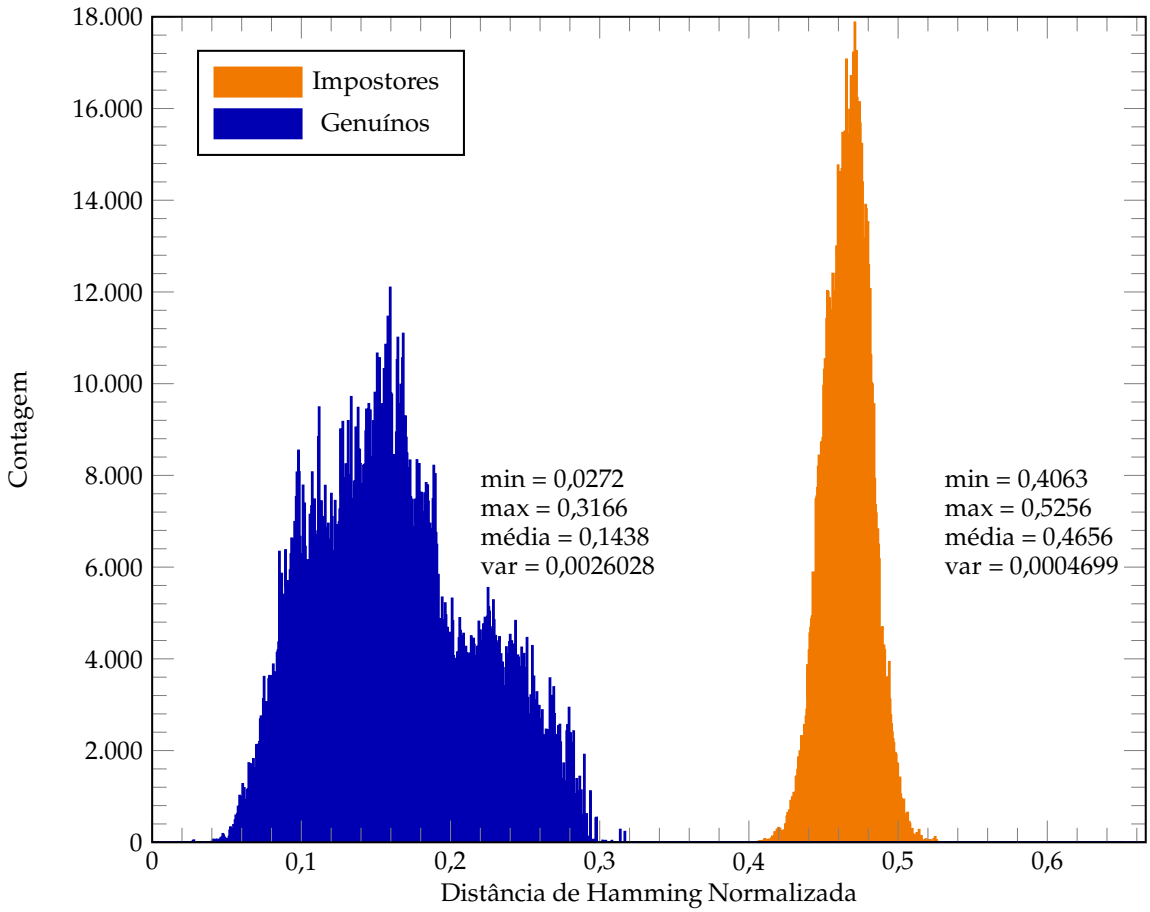


Figura 3.6: *Distância de Hamming normalizada para usuários genuínos e impostores para a base de dados BIOSECURE usando apenas uma íris e empregando a chave de embaralhamento (k_{shuf}), os números aleatórios e a busca por rotação.*

como usuário genuíno [3].

A Tabela 3.4 exibe os resultados dos experimentos que empregam uma única íris e ambas as íris. Contrário à expectativa, o tempo de processamento t_T para os experimentos que empregam ambas as íris é menor do que o tempo correspondente de processamento t_S para uma única íris. No início da pesquisa, foi imaginado, erroneamente, que t_T seria maior do que t_S, uma vez que o comprimento da sequência do código íris para duas íris é duas vezes maior do que para uma única íris. No entanto, os experimentos mostraram que t_T é quase duas vezes menor do que t_S para a busca padrão e quase cinco vezes menor do que t_S quando a busca por rotação é empregada. Analisando todo o processo, é possível observar que o tempo de processamento aumenta, proporcionalmente, a quantidade de erros ocorridos. Quando ambas as íris são empregadas, o código de Hadamard usado é mais

poderoso, $(64, 7, 32)$, e, como consequência, o número total de falhas na decodificação decresce, reduzindo o tempo para execução do experimento correspondente. Se for observado o número total de comparações necessárias em cada experimento, é possível verificar que este número é duas vezes menor para duas íris do que para uma única íris.

Tabela 3.4: *Resultados comparativos dos experimentos usando a base de dados BIOSECURE para uma única íris e para ambas as íris (C++).*

	Uma íris		Duas íris	
Opção de busca	Padrão	Rotação	Padrão	Rotação
Tempo (s)	253	1.936	171	406
Comparações	132.000	132.000	66.000	66.000
Comparações/s	521	68	385	162

A Tabela 3.5 apresenta a comparação do percentual da FRR entre o sistema proposto aqui, o qual é baseado na busca por rotação, e o esquema proposto em [3], o qual é baseado na busca padrão. Conforme pode ser verificado na Tabela 3.5, o uso da busca por rotação alcança um percentual melhor para a FRR em todos os casos considerados. Uma vez que em [3] o valor percentual para a FRR só cobre a faixa $10 \leq t_{\text{RS}} \leq 14$, foi colocado "N/D" (Não Disponível) para os outros valores de t_{RS}. Para a base de dados NIST-ICE, só é possível fazer uma comparação direta do percentual da FRR, apenas quando $t_{\text{RS}} = 10$. Neste caso, o sistema proposto alcançou um resultado cerca de 85 vezes melhor do que o obtido em [3]. Porém, para uma comparação na qual seja considerado o percentual da FRR e o valor do t_{RS}, o esquema proposto aqui alcançou um resultado duas vezes melhor para a base de dados NIST-ICE e três vezes melhor para as bases de dados BIOSECURE e CASIA. Como consequência, é possível recuperar uma chave criptográfica com 371 *bits* para a FRR de 0,4700% e FAR igual a 0% para a base NIST-ICE, empregando ambas as íris.

A Figura 3.7 ilustra a situação onde nenhuma técnica de separação dos usuários é empregada e a Figura 3.8 ilustra a separação entre usuários genuínos e impostores, resultado da aplicação da chave de embaralhamento (k_{shuf}), dos números pseudo aleatórios e da busca por rotação para ambas as íris.

Uma comparação entre a Figura 3.7 e a Figura 3.5, ou entre a Figura 3.8 e a Figura 3.6, indica uma similaridade considerável. Seja para uma íris ou para duas íris, o uso da busca por rotação sempre alcançou resultados pelo menos tão bons quanto aqueles obtidos pela busca padrão. Esta conclusão vem do fato que a busca por rotação só é ativada quando a

Tabela 3.5: *Percentual FRR de [3] e da busca por rotação para ambas as íris. FAR é sempre igual a zero para ambos os casos.* $\|\mathbf{K}\|$ *é o comprimento da chave. N/D = Não Disponível.*

t_{RS}	$\|\mathbf{K}\|$	BIOSECURE V1		CASIA V2		NIST-ICE	
		[3]	**Rotação**	[3]	**Rotação**	[3]	**Rotação**
1	413	N/D	**2,6650**	N/D	**2,7270**	N/D	**2,9010**
2	399	N/D	**1,7480**	N/D	**0,9130**	N/D	**1,5140**
3	385	N/D	**1,1070**	N/D	**0,1900**	N/D	**0,8600**
4	371	N/D	**0,6540**	N/D	**0,0290**	N/D	**0,4710**
5	357	N/D	**0,3350**	N/D	**0**	N/D	**0,2120**
6	343	N/D	**0,1480**	N/D	**0**	N/D	**0,0980**
7	329	N/D	**0,0550**	N/D	**0**	N/D	**0,0550**
8	315	N/D	**0,0090**	N/D	**0**	N/D	**0,0290**
9	301	N/D	**0,0020**	N/D	**0**	N/D	**0,0080**
10	287	1,0300	**0**	0,6700	**0**	0,3400	**0,0040**
11	273	0,6000	**0**	0,2300	**0**	0,1600	**0**
12	259	0,1700	**0**	0,1300	**0**	0,1100	**0**
13	245	0,1300	**0**	0,1000	**0**	0,0500	**0**
14	231	0,1000	**0**	0,0700	**0**	0	**0**
15	217	N/D	**0**	N/D	**0**	N/D	**0**
16	203	N/D	**0**	N/D	**0**	N/D	**0**
17	189	N/D	**0**	N/D	**0**	N/D	**0**
18	175	N/D	**0**	N/D	**0**	N/D	**0**
19	161	N/D	**0**	N/D	**0**	N/D	**0**
20	147	N/D	**0**	N/D	**0**	N/D	**0**
21	133	N/D	**0**	N/D	**0**	N/D	**0**
22	119	N/D	**0**	N/D	**0**	N/D	**0**

busca padrão falha na identificação do usuário, e, consequentemente, a busca por rotação emprega mais comparações. Embora haja a possibilidade de executar até 441 comparações, o número requerido, para o sistema de regeneração e identificação dos usuários quando foi empregada a busca por rotação aumenta em média cerca de duas vezes em comparação com a busca padrão, tanto para uma única íris como para ambas as íris.

3.4 Conclusões sobre a autenticação com busca por rotação

Empregando as rotações das imagens de referência do código íris, foi desenvolvida a busca por rotação que é bem mais eficiente do que a abordagem usada em [2], usando apenas uma íris para identificação positiva de usuários e reconstrução da chave criptográfica. Agora, é possível recuperar chaves criptográficas com 198 *bits* com percentual da FRR de $0,2400\%$ e FAR de 0% para a base de dados NIST-ICE(exp1). Quando ambas as íris são

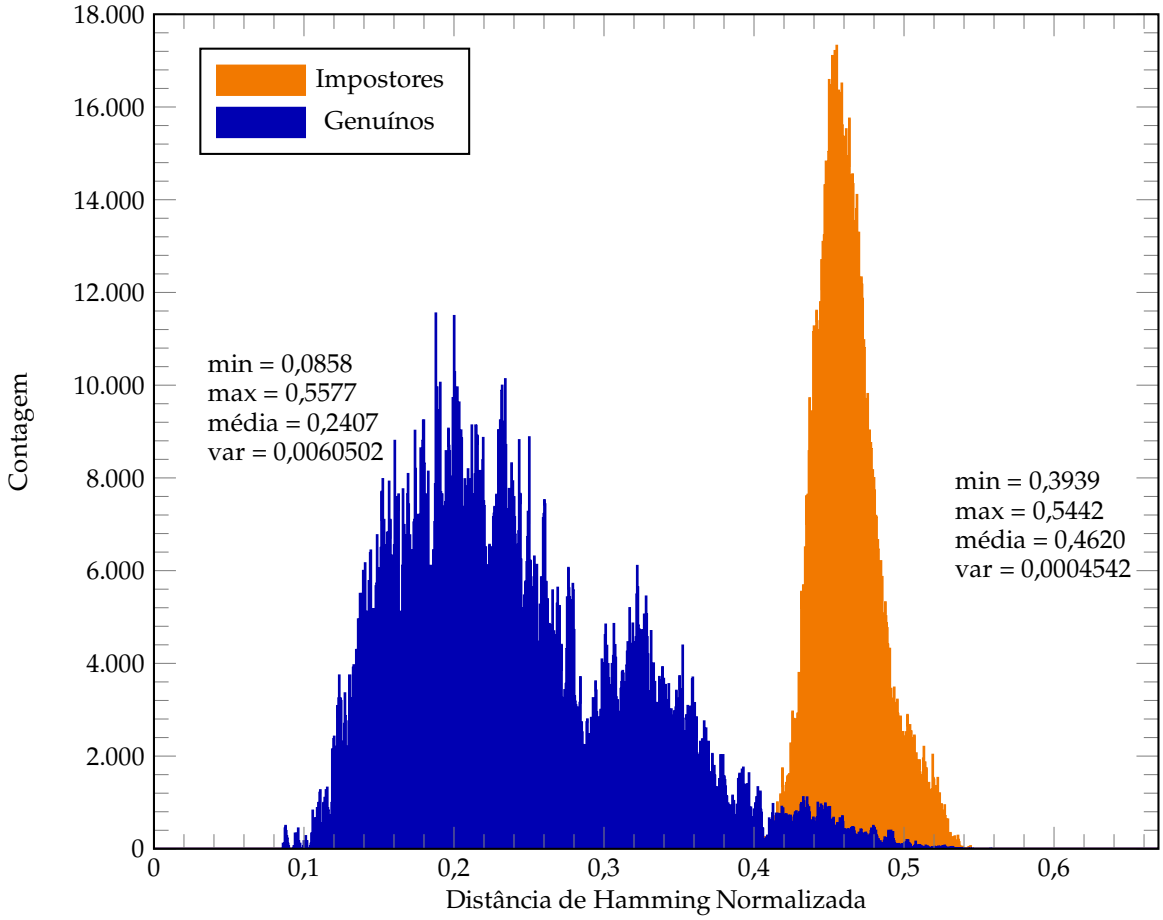


Figura 3.7: *Distância de Hamming normalizada para usuários genuínos e impostores para a base de dados BIOSECURE usando ambas as íris.*

empregadas, a busca por rotação é mais eficiente do que a proposta usada em [3]. Agora, é possível recuperar chaves criptográficas com 371 *bits* com percentual da FRR de $0,4700\%$ e FAR de 0% para a base de dados NIST-ICE. A Tabela 3.6 apresenta uma comparação entre alguns dos principais sistemas propostos na literatura e três diferentes resultados obtidos com a proposta apresentada neste Capítulo.

A Tabela 3.6 contém resultados da FRR em percentual, obtidos por trabalhos anteriores e resultados obtidos usando a busca por rotação. Em alguns trabalhos anteriores, são usados esquemas de códigos corretores de erros diferentes do esquema usado no sistema de identificação biométrico proposto neste Capítulo. Em [46] e [2] é usado o código de Reed-Solomon; em [57] é usado o código produto baseado nos códigos de Reed-Muller; nos demais são usados os código de Reed-Solomon e o código de Hadamard. O comprimento da chave criptográfica $\|\mathbf{K}\| = m_{RS} \cdot k_{RS}$ é calculado conforme a equação 3.1, onde m_{RS} de-

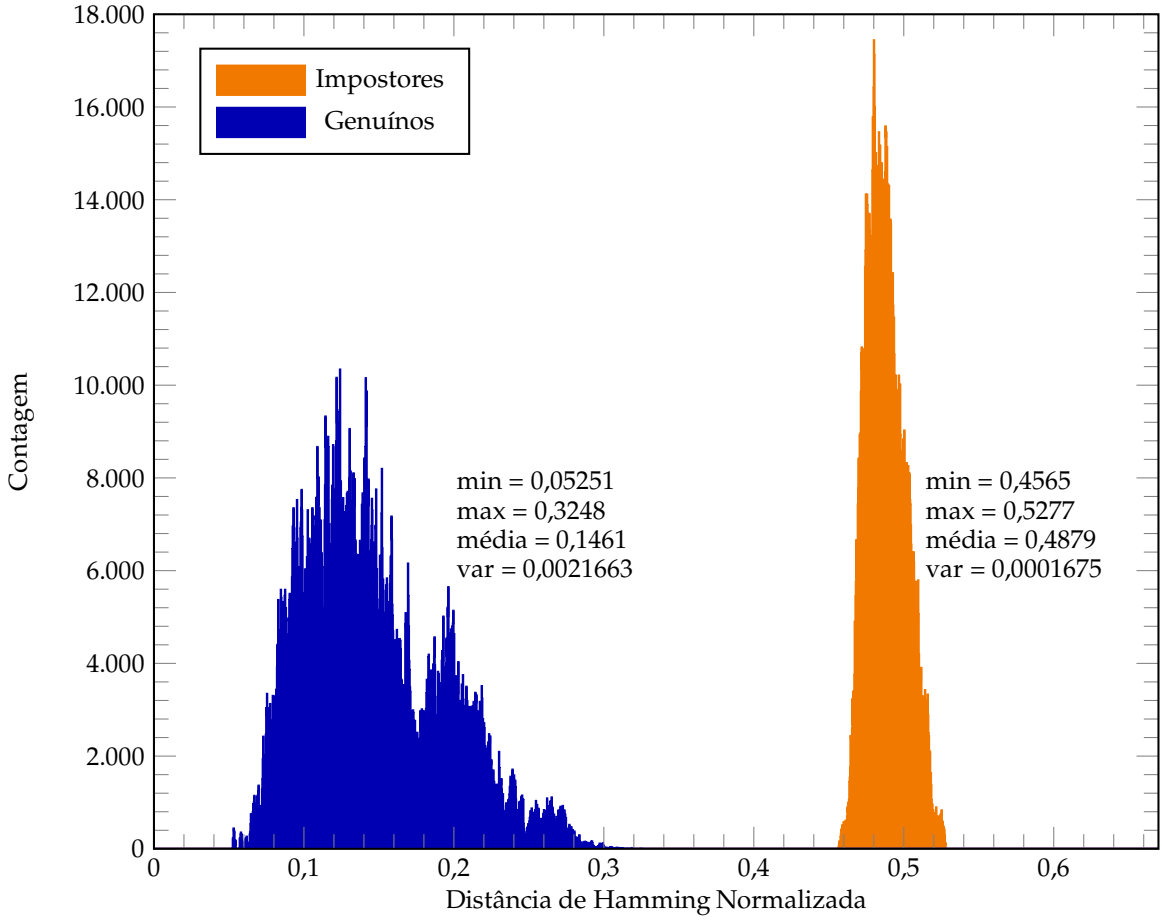


Figura 3.8: *Distância de Hamming normalizada para usuários genuínos e impostores para a base de dados BIOSECURE usando ambas as íris e empregando a chave de embaralhamento (k_{shuf}), os números aleatórios e a busca por rotação.*

fine quais os possíveis símbolos do código RS em $GF(2^{m_{RS}})$, k_{RS} é o número de blocos de entrada do código RS antes da codificação, n_{RS} é o número de blocos de saída do código RS depois da codificação e t_{RS} é a capacidade de correção do código RS [3]:

$$\|\mathbf{K}\| \quad = \quad m_{RS} \cdot (n_{RS} - 2t_{RS}) = m_{RS} \cdot \left(\frac{\| \boldsymbol{\theta}'_{\text{ref}} \|}{2^{k_{RS}}} - 2t_{RS} \right), \tag{3.1}$$

por exemplo, para uma única íris é possível utilizar: $m_{RS} = 6$, $n_{RS} = 61$ e $t_{RS} = 14$:

$$\|\mathbf{K}\| \quad = \quad m_{RS} \cdot (n_{RS} - 2t_{RS}) = 6 \cdot (61 - 2 \cdot 14) = 198. \tag{3.2}$$

Para os experimentos que empregam ambas as íris, é possível observar que o tempo necessário para a realização de um experimento é menor do que o tempo necessário quando

Tabela 3.6: *Resultado percentual da FRR para diversos algoritmos biométricos. O algoritmo proposto emprega a busca por rotação.*

Esquema	ECC	Comprimento da chave $\|\|\mathbf{K}\|\|$ (em *bits*)	FRR (%)	FAR (%)	Base de Dados
Referência [46]	RS	282	8,4200	0	NIST-ICE (Íris direita)
Referência [2]	RS	128/256	0,7600	0,1000	NIST-ICE (Íris direita)
Referência [57]	RMP	42	0,4700	0	NIST-ICE (Íris direita)
Referência [58]*	RSH	147	0,1800	0	NIST-ICE (Ambas as íris)
Referência [3]*	RSH	231	0	0	NIST-ICE (Ambas as íris)
Referência [3]*	RSH	287	0,3400	0	NIST-ICE (Ambas as íris)
Rotação	**RSH**	**198**	**0,2400**	**0**	**NIST-ICE (Íris direita)**
Rotação*	**RSH**	**273**	**0**	**0**	**NIST-ICE (Ambas as íris)**
Rotação*	**RSH**	**371**	**0,4700**	**0**	**NIST-ICE (Ambas as íris)**
Obs: (*) denota os sistemas multi biométricos; ECC: código corretor de erros; RSH: código Reed Solomon e Hadamard; RMP: códigos produto baseados nos códigos de Reed Muller. O percentual FAR só é diferente de zero em [2].					

uma única íris é empregada. Os resultados obtidos na proposta apresentada aqui, neste Capítulo, é, pelo menos, duas vezes melhor do que o melhor resultado alcançado pelos artigos usados para comparação, tanto para uma única íris como para ambas as íris.

Quanto a quantidade de testes realizados com o emprego da busca por rotação, embora exista a possibilidade de serem realizados até 441 testes com cada par de imagens de referência/imagem de teste, na média, a quantidade se testes com cada par foi apenas duas vezes maior do que com o emprego da busca padrão.

É possível afirmar que a busca por rotação pode ser implementada como um *upgrade* na grande maioria dos sistemas de identificação que usam o código íris, com poucas modificações. A sugestão proposta é, primeiramente, executar a busca padrão, que pode ser seguida pela busca por rotação, caso necessário. A busca por rotação será necessária quando a busca padrão não obtiver sucesso na identificação do usuário. Em outras palavras, os resultados obtidos pela busca por rotação são sempre, pelo menos, tão bons quanto aqueles baseados na busca padrão. Nos experimentos realizados nas bases de dados já indicadas, a busca por rotação sempre obteve os melhores resultados.

Capítulo 4

Autenticação com busca por rotação e voto de maioria

"Os bem-educados contradizem outras pessoas. Os sábios contradizem a si mesmos."

— Oscar Wilde

O desempenho de sistemas biométricos para identificação de usuário baseados em código de íris melhora quando se consegue reduzir a distância de Hamming entre o código de íris de referência e o código de íris de teste empregados. O objetivo deste Capítulo é propor e analisar, por meio de simulação em computador, a técnica de decisão por voto de maioria aplicada à identificação biométrica por meio do código de íris.

Este Capítulo aborda a técnica de voto de maioria, sendo realizada uma descrição da mesma, sua aplicação no código íris e os resultados obtidos. Ao término deste Capítulo será feita uma comparação entre os dados obtidos no Capítulo 3 e os dados obtidos com a técnica proposta.

Trabalhos recentes na área de biometria têm focado, principalmente, em novos métodos para obter códigos de íris [9], [11] e [18]. Não obstante, aspectos de segurança de sistemas biométricos também têm sido alvos de interesse de trabalhos recentes [10], [59] e [60]. Neste

Capítulo, o foco é no desempenho de sistemas biométricos que, relativamente, têm recebido pouca atenção nos últimos anos [3], [61].

4.1 DESCRIÇÃO DA TÉCNICA DE VOTO DE MAIORIA

A técnica de voto de maioria é bastante conhecida no contexto de códigos corretores de erros [54, p. 273]. Neste Capítulo, a técnica de voto de maioria consiste em analisar duas ou mais palavras do código de íris, comparando-as coordenada a coordenada, e produzindo como resultado a sequência denominada de *sequência majoritária*, na qual o valor binário de cada coordenada corresponde ao valor do voto de maioria calculado naquela coordenada dentre as sequências examinadas. Em outras palavras, observa-se o valor de uma determinada coordenada em todas as sequências consideradas e realiza-se o voto de maioria, contando a quantidade de 0's e de 1's, no caso de sequências binárias, e atribui-se como valor final aquele que for a maioria na contagem realizada.

Para exemplificar a técnica de voto de maioria, considere três sequências binárias de comprimento $n = 8$ cada uma, de modo que: $\mathbf{x} = (0,1,0,1,1,0,1,0)$, $\mathbf{y} = (0,1,0,0,0,0,1,0)$ e $\mathbf{z} = (0,1,0,1,0,1,1,1)$. A sequência majoritária $\mathbf{m}$ é obtida aplicando a técnica de voto de maioria às sequências $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$ da seguinte forma. Cada coordenada m_i de $\mathbf{m}$ é obtida pelo voto de maioria das respectivas coordenadas x_i, y_i e z_i, em que $0 \leq i \leq 7$. Para $i = 3$, obtém-se $m_3 = 1$, pois $x_3 = 1$, $y_3 = 0$ e $z_3 = 1$. Assim, procedendo para os demais valores de i resulta $\mathbf{m} = (0,1,0,1,0,0,1,0)$.

Na primeira coluna da Tabela 4.1, são exibidas as sequências $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{z}$, aqui consideradas como sequências de referência, a sequência majoritária $\mathbf{m}$ e a sequência de teste $\mathbf{s}$. A Tabela 4.1 apresenta valores da distância de Hamming entre pares de sequências. Analisando os valores da distância de Hamming, verifica-se que $d(\mathbf{m},\mathbf{s})$ entre as sequências $\mathbf{m}$ e $\mathbf{s}$ é igual a 1. Comparando $d(\mathbf{m},\mathbf{s})$ com $d(\mathbf{x},\mathbf{s}) = 2$, $d(\mathbf{y},\mathbf{s}) = 2$ e $d(\mathbf{z},\mathbf{s}) = 1$, percebe-se que usar $\mathbf{m}$ como sequência de referência reduz o valor de $d(\mathbf{m},\mathbf{s})$.

O exemplo discutido com o auxílio da Tabela 4.1 indica que utilizar a técnica de voto de maioria para obter uma sequência de referência parece promissora. Assim, na sequência, utilizam-se as bases de dados BIOSECURE [8], CASIA [8] e NIST-ICE [1], que possuem códigos de íris de teste e códigos de íris de referência, com o objetivo de gerar códigos de íris de referência por meio da técnica de voto de maioria e avaliar o impacto desta técnica no cálculo da distância de Hamming.

Tabela 4.1: *Exemplo do cálculo da HD usando voto de maioria.*

Sequência	**x** (HD)	**y** (HD)	**z** (HD)	**s** (HD)	**m** (HD)
x = 01011010	00000000 (0)	00011000 (2)	00001101 (3)	00001100 (2)	00001000 (1)
y = 01000010	00011000 (2)	00000000 (0)	00010101 (3)	00010100 (2)	00010000 (1)
z = 01010111	00001101 (3)	00010101 (3)	00000000 (0)	00000001 (1)	00000101 (2)
s = 01010110	00001100 (2)	00010100 (2)	00000001 (1)	00000000 (0)	00001000 (1)
m = 01010010	00001000 (1)	00010000 (1)	00000101 (2)	00000100 (1)	00000000 (0)

4.2 APLICAÇÃO NO CÓDIGO ÍRIS

A técnica de voto de maioria aplicada ao código íris consiste em analisar *bit* a *bit* o código íris, que possui 1188 *bits* no total, e buscar minimizar a quantidade de erros que possam acontecer durante a captura das imagens de referência da íris. Analisando o código íris, *bit* a *bit*, é possível perceber que existem muitas diferenças entre duas imagens de referência quaisquer. De acordo com o teste realizado em todas as bases de dados disponíveis, foi encontrada uma distância de Hamming mínima de 54 *bits*, e máxima de 645 *bits* entre usuários genuínos, conforme pode ser visto na Tabela 4.2. Quando for analisada uma posição (*bit*) em uma palavra do código íris, esta será comparada com a mesma posição em outras palavras do código íris, fazendo o voto da maioria, ou seja, ao comparar três palavras, por exemplo, será contada a quantidade de zeros e a quantidade de uns, atribuindo o valor para esta posição (*bit*) aquele que for a maioria na contagem realizada. Esta técnica será aplicada apenas nas palavras do código íris responsáveis pelas imagens de referência, pois em um caso real de identificação, só é necessário obter uma imagem de teste. É possível exigir mais de uma imagem de teste quando a identificação de um usuário for necessária, porém esta abordagem não foi implementada, pois este procedimento não é comum em um caso real de identificação.

Na Tabela 4.2, é possível observar as distâncias de Hamming mínimas, as distâncias de Hamming máximas e as distâncias de Hamming médias, para a aplicação da técnica de voto majoritário para 3, 5, 7 e 9 imagens. Para cada imagem de um usuário, nas bases de dados tem-se um código binário de comprimento 1.188 *bits*, que a representa. É possível observar que, ao aumentar a quantidade de imagens usadas na técnica proposta, a distância de Hamming média diminui. Este fato implica na redução de erros entre a imagem de referência e a imagem de teste, que serão utilizadas nos experimentos. Ainda na Tabela 4.2,

é possível observar que a redução da distância de Hamming média entre o uso da técnica proposta para $i = 9$ imagens e o não uso da técnica ($i = 1$), é de, respectivamente, $12,73\%$, $6,77\%$, $19,79\%$ e $18,99\%$, para as bases de dados BIOSECURE, CASIA, NIST-ICE(exp1) e NIST-ICE(exp2).

Este ganho percentual significa que o código íris de referência, que será usado para comparação com o código íris de teste, tem estas reduções percentuais, em média para as bases de dados respectivas, na distância de Hamming, o que resultará em uma redução de erros de identificação, quando for utilizado o sistema de comparação proposto.

Os resultados obtidos na Tabela 4.2 para uma íris, e na Tabela 4.3 para ambas íris, mostram que a aplicação da técnica de voto majoritário nas bases de dados BIOSECURE, CASIA e NIST-ICE, comprovam o que ocorre no exemplo dado na Tabela 4.1, ou seja, é possível reduzir as distâncias de Hamming médias para todas as bases de dados. Na seção 4.4 é possível ver o ganho obtido ao aplicar esta técnica no pré-processamento das imagens de referência (θ_{ref}) que serão usadas no sistema de identificação biométrico proposto.

Na Tabela 4.3, é possível observar as distâncias de Hamming mínimas, as distâncias de Hamming máximas e as distâncias de Hamming médias, respectivamente, para $i = 1$, $i = 3$, $i = 5$, $i = 7$ e $i = 9$ imagens, para as bases de dados BIOSECURE, CASIA e NIST-ICE, quando são empregadas ambas as íris para compor os códigos íris de referência e os códigos íris de teste.

Ainda na Tabela 4.3, é possível observar um resumo para a aplicação da técnica de voto majoritário, quando é empregado o uso de ambas as íris, desde uma imagem, até nove imagens. É possível observar, também, que quando a quantidade de imagens é aumentada, a distância de Hamming média é reduzida.

4.3 SISTEMA DE IDENTIFICAÇÃO UTILIZANDO O VOTO DE MAIORIA

Exibe-se na Figura 4.1, por meio de um diagrama de blocos, o sistema biométrico de regeneração de chave criptográfica empregando a técnica de voto de maioria. O sistema proposto é semelhante ao apresentado em [61], com o acréscimo do bloco de pré-processamento denotado por "Manipulação do código de íris por decisão majoritária". Nesse bloco, aplica-se a técnica de voto de maioria, para gerar o código de íris de referência, denotado por θ_{ref}. Desse ponto em diante, o funcionamento do sistema da Figura 4.1 é similar ao do sistema

Tabela 4.2: *Distância de Hamming para usuários genuínos nas bases de dados BIOSECURE, CASIA e NIST-ICE, com $i \in \{1,3,5,7,9\}$ códigos de íris. (Obs.: N/D - Não Disponível).*

	Base de Dados		BIOSECURE	CASIA	NIST-ICE (exp1)	NIST-ICE (exp2)
1 código	Numeração dos códigos combinados	Melhor	1	1	N/D	N/D
		Pior	1	1	N/D	N/D
		Total de testes	1	1	N/D	N/D
	Distância de Hamming	Min	54	155	114	123
		Max	617	645	641	636
		Média	289,99	378,65	340,07	346,56
3 códigos	Numeração dos códigos combinados	Melhor	(5,8,9)	(4,9,10)	N/D	N/D
		Pior	(2,3,4)	(2,3,4)	N/D	N/D
		Total de testes	120	120	N/D	N/D
	Distância de Hamming	Min	55	122	118	137
		Max	602	647	603	598
		Média	267,51	362,77	297,62	304,67
5 códigos	Numeração dos códigos combinados	Melhor	(2,5,8,9,10)	(1,2,4,6,8)	N/D	N/D
		Pior	(1,3,4,5,8)	(2,4,5,6,9)	N/D	N/D
		Total de testes	252	252	N/D	N/D
	Distância de Hamming	Min	52	119	112	133
		Max	599	645	581	594
		Média	259,55	357,25	285,29	291,05
7 códigos	Numeração dos códigos combinados	Melhor	(2,5,6,7,8,9,10)	(1,2,4,6,7,8,10)	N/D	N/D
		Pior	(1,2,3,4,5,7,10)	(2,3,4,5,6,8,9)	N/D	N/D
		Total de testes	120	120	N/D	N/D
	Distância de Hamming	Min	61	133	116	135
		Max	590	641	565	608
		Média	255,48	354,52	278,08	285,89
9 códigos	Numeração dos códigos combinados	Melhor	(1,2,4,5,6,7,8,9,10)	(1,2,3,4,5,6,7,8,10)	N/D	N/D
		Pior	(1,2,3,4,5,7,8,9,10)	(2,3,4,5,6,7,8,9,10)	N/D	N/D
		Total de testes	10	10	N/D	N/D
	Distância de Hamming	Min	70	139	116	129
		Max	576	633	581	596
		Média	253,09	353,03	272,78	280,74
Redução da HD (1 cód x 9 cód)			12,73%	6,77%	**19,79**%	18,99%

proposto em [61].

A inclusão do bloco de manipulação não afeta o funcionamento do sistema de identificação biométrico proposto por [62], no Capítulo 3, na verdade este novo bloco realiza um pre-processamento das imagens de referência, antes de fornecer ao próximo bloco a imagem de referência (θ_{ref}) que será usada nos testes. Para o correto funcionamento do bloco de manipulação é necessário fornecer a quantidade de imagens i desejada para a execução da técnica de voto majoritário.

Tabela 4.3: *Distância de Hamming para usuários genuínos para ambas as íris, nas bases de dados BIOSECURE, CASIA e NIST-ICE, com 1, 3, 5, 7 e 9 imagens. N/D = Não Disponível*

	Base de Dados		BIOSECURE	CASIA	NIST-ICE
1 código	Numeração dos códigos combinados	Melhor	1	1	N/D
		Pior	1	1	N/D
		Total de testes	1	1	N/D
	Distância de Hamming	Min	212	370	303
		Max	1.119	1.237	1.151
		Média	579,99	757,30	706,36
3 códigos	Numeração dos códigos combinados	Melhor	(4,9,10)	(1,6,9)	N/D
		Pior	(1,7,8)	(1,2,4)	N/D
		Total de testes	120	120	N/D
	Distância de Hamming	Min	195	335	314
		Max	1.119	1.238	1.207
		Média	535,02	725,54	652,08
5 códigos	Numeração dos códigos combinados	Melhor	(1,3,5,9,10)	(1,2,6,9,10)	N/D
		Pior	(1,2,5,7,8)	(1,2,3,4,5)	N/D
		Total de testes	252	252	N/D
	Distância de Hamming	Min	188	343	319
		Max	1.105	1.237	1.201
		Média	519,09	714,51	624,52
7 códigos	Numeração dos códigos combinados	Melhor	(1,2,3,5,8,9,10)	(1,2,3,6,8,9,10)	N/D
		Pior	(2,3,4,5,6,7,8)	(2,3,4,5,6,8,9)	N/D
		Total de testes	120	120	N/D
	Distância de Hamming	Min	187	338	319
		Max	1.100	1.216	1.207
		Média	510,97	709,04	613,70
9 códigos	Numeração dos códigos combinados	Melhor	(1,2,3,4,6,7,8,9,10)	(2,3,4,5,6,7,8,9,10)	N/D
		Pior	(1,2,3,4,5,6,7,8,10)	(1,2,3,4,5,6,8,9,10)	N/D
		Total de testes	10	10	N/D
	Distância de Hamming	Min	192	357	315
		Max	1.085	1.214	1.201
		Média	506,17	706,06	604,99
Redução da HD, (1 cód x 9 cód)			12,73%	6,77%	14,35%

Nas bases de dados regulares (BIOSECURE e CASIA), que foram definidas deste modo no Capítulo 3 [62], devido ao fato de terem as mesmas quantidades de imagens de referência e imagens de teste por usuário, tem-se 10 imagens de referência para cada íris, deste modo, pode ser escolhido ($i = 3$, 5, 7 ou 9) para garantir a maioria. Na base de dados irregular NIST-ICE, que também foi definida deste modo por [62], devido ao fato de não ter as

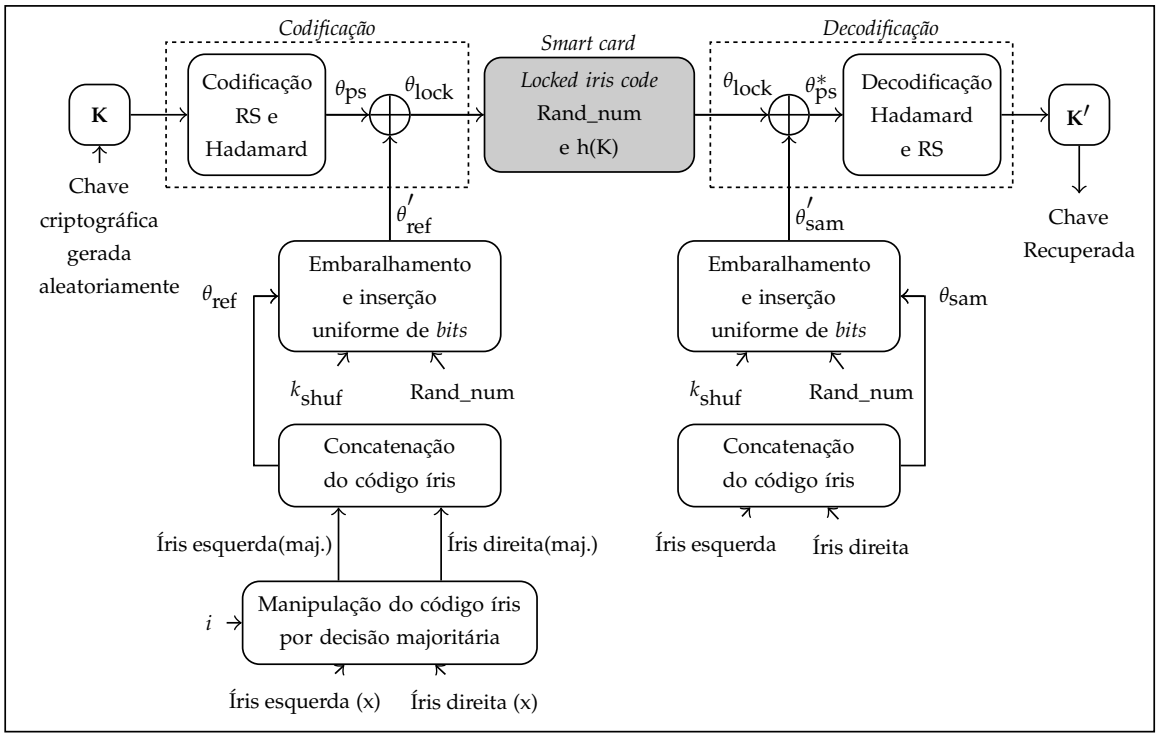


Figura 4.1: *Sistema de regeneração de chave uni ou multi biométrico com manipulação majoritária do código íris, empregando smart card e senha.*

mesmas quantidades de imagens de referência e imagens de teste por usuário, a quantidade de imagens por íris não é igual, variando desde uma imagem até 31 imagens para cada íris, por usuário. Os testes com todas as bases de dados foram realizados com as mesmas opções para i, ou seja ($i = 3$, 5, 7 ou 9), porém foi necessário, na base de dados NIST-ICE, limitar o máximo de imagens que poderiam ser utilizadas, por conta da quantidade máxima de imagens disponíveis, por usuário, nesta base de dados.

Para as bases de dados BIOSECURE e CASIA, considerando a quantidade de códigos de íris usados para gerar θ_{ref}, foram realizadas simulações com todas as possíveis combinações. Para $i = 3$ são possíveis 120 combinações, i.e., C_{10}^{3} combinações. Nesse caso foram realizadas duas rodadas de 120 experimentos, totalizando 240 experimentos. Para $i = 5$ resulta $C_{10}^{5} = 252$ e foram realizados 252 experimentos. Para $i = 7$ resulta $C_{10}^{7} = 120$ e foram realizados 240 experimentos (duas rodadas de 120). Para $i = 9$ resulta $C_{10}^{9} = 10$ e foram realizados 240 experimentos (vinte e quatro rodadas de 10 experimentos cada). Desse modo, a quantidade de experimentos manteve-se em 240 para $i = 3$, $i = 7$ e $i = 9$, enquanto que para $i = 5$ foram realizados 252 experimentos. O total de experimentos realizados para cada

base de dados foi de 972. Na Tabela 4.2, é possível observar a quantidade de combinações máximas para cada i, assim como a combinação de imagens que resultou na menor distância de Hamming e a combinação de imagens que resultou na maior distância de Hamming. O tempo médio de cada experimento foi, aproximadamente, 31 minutos e o tempo total, para ambas as bases de dados foi de, aproximadamente, 41 dias, 22 horas e 33 minutos. O objetivo foi manter o mais uniforme possível a quantidade de experimentos para os diferentes valores de i. Os resultados percentuais para a FRR, apresentados nas Tabelas 4.8 e 4.9 na sequência deste Capítulo, são as médias obtidas a partir de todas as rodadas de experimentos realizadas.

Todos os testes realizados foram executados empregando a busca por rotação [61]. Na Tabela 4.4, pode-se observar o valor médio da FRR(%), obtido para $1 \leq t_{\mathrm{RS}} \leq 10$, em que t_{RS} é a capacidade de correção de erros do código Reed-Solomon [54, p. 81]. Ainda na Tabela 4.4, observa-se que para $1 \leq t_{\mathrm{RS}} \leq 10$, os piores resultados obtidos com $i = 9$ são melhores que os resultados obtidos com $i = 1$. Assim, pode-se utilizar qualquer combinação aleatória permitida de $i = 9$ códigos de iris, para tomar a decisão por voto de maioria, que o resultado obtido será melhor do que no caso em que $i = 1$.

Tabela 4.4: *Percentual de erros para experimentos, usando* 1 *imagem e voto da maioria com* 9 *imagens (em* 240 *experimentos), para a base de dados BIOSECURE.*

t_{RS}	1 imagem	9 imagens			
		média	mínimo	máximo	desvio padrão
1	**15,1500**	**10,3804**	9,4833	11,2667	0,3075
2	**11,0400**	**8,3610**	7,7667	8,9000	0,2362
3	**8,6300**	**7,3429**	6,8167	7,8167	0,1727
4	**7,4900**	**6,7167**	6,2833	7,1333	0,1533
5	**6,6300**	**6,1051**	5,6167	6,5500	0,1793
6	**5,9900**	**5,4638**	4,9167	5,9167	0,2268
7	**5,3900**	**4,7697**	3,9667	5,3833	0,2553
8	**4,7400**	**4,0755**	3,4667	4,6500	0,2137
9	**4,0600**	**3,4415**	2,8833	3,8500	0,1842
10	**3,4000**	**2,7778**	2,2000	3,2833	0,1949

Para a base de dados NIST-ICE, devido a variação da quantidade de imagens por usuário, não foi possível fazer experimentos com todas as combinações possíveis, porém o método escolhido para utilização das imagens de referência, para esta base de dados, permite testar diversas combinações diferentes.

Para a base de dados NIST-ICE, a quantidade máxima M de códigos de íris disponíveis para um determinado usuário é variável, e não existe separação entre códigos de referência e códigos de teste. O *software* utilizado é programado para identificar o número mais adequado de códigos de íris e, se necessário, reduzir automaticamente para o maior valor possível de i em cada teste específico. Por exemplo, se para um determinado usuário $M = 7$, então o *software* utiliza, no máximo, $i = 5$, pois não há, na base de dados, códigos de íris de teste específicos e, portanto, pelo menos um dos 7, deve ser usado como θ_{sam}. Como restam 6 códigos de íris para aplicar o voto de maioria e a quantidade par pode resultar em empate, opta-se por usar 5 códigos de íris.

A implementação também verifica quanto à "contaminação" de códigos de íris de referência em relação ao código de íris de teste, não permitindo que o código de íris de teste faça parte da composição dos códigos de íris de referência usados na decisão por voto de maioria. A escolha dos códigos de referência, para a base de dados do NIST-ICE, é então feita numa lista $\mathcal{L}$ de códigos numerados de 1 a M. A regra adotada neste trabalho para a escolha de até nove códigos é descrita a seguir.

Para um dado valor de i fixado, é necessário selecionar i códigos de íris c_j, $1 \leq j \leq i$, cuja posição $l(c_j)$, $1 \leq l(c_j) \leq M$, na lista $\mathcal{L}$ é indicada na Tabela 4.5, na qual $1 \leq i \leq 9$. Após a escolha dos i códigos c_j, que são usados para gerar θ_{ref}, cada c_j é comparado com todas as escolhas anteriores e com θ_{sam}. Caso o c_j atual já tenha sido escolhido entre os códigos c_j anteriores, o código que representa o c_j atual é substituído pelo código na lista $\mathcal{L}$ associado à posição $(l(c_j) \mod M) + 1$, que passa a compor o novo código de íris de referência θ_{ref}. Desse modo, garante-se que $j = j_1 \neq j_2$, o que implica $c_{j_1} \neq c_{j_2}$ e $c_j \neq \theta_{\text{sam}}$, $1 \leq j \leq M$.

Cada uma das oito linhas na Tabela 4.6, constituem exemplos de escolha dos códigos de íris de referência para compor θ_{ref}. A partir da terceira coluna, os números indicados na Tabela 4.6 são as posições de cada código na lista $\mathcal{L}$ correspondente, tanto para os códigos de referência quanto para os códigos de teste. Por exemplo, na terceira linha, $l(c_5) = 13$ significa que c_5 ocupa a décima terceira posição na lista $\mathcal{L}$ na base de dados, e é escolhido como um dos códigos de íris, que irão compor o código de referência. Essa posição, pela fórmula usada para a localização a priori de c_5 seria a 12, porém a posição 12 já é usada para o código de teste, por isso faz-se c_5 assumir o código na posição $(l(c_5) \mod 15) + 1$, na lista $\mathcal{L}$, i.e., a posição de número 13. Para este exemplo, o código de íris de referência θ_{ref}, que será usado no teste, será composto por: $l(c_1)$, $l(c_2)$, $l(c_3)$, $l(c_4)$ e $l(c_5)$, ou seja, pelas

Tabela 4.5: *Lista para escolha a priori da base de dados NIST-CIE, dos códigos de referência a serem usados para gerar* θ_{ref}.

Número de ordem	Código	Posição na lista $\mathcal{L}$
primeiro	c_1	$l(c_1) = 1$
segundo	c_2	$l(c_2) = \lceil M/2 \rceil$
terceiro	c_3	$l(c_3) = M$
quarto	c_4	$l(c_4) = \lceil M/4 \rceil$
quinto	c_5	$l(c_5) = \lceil 3M/4 \rceil$
sexto	c_6	$l(c_6) = \lceil 3M/8 \rceil$
sétimo	c_7	$l(c_7) = \lceil 5M/8 \rceil$
oitavo	c_8	$l(c_8) = \lceil M/8 \rceil$
nono	c_9	$l(c_9) = \lceil 7M/8 \rceil$

posições, respectivamente: 11, 8, 15, 4 e 13.

Tabela 4.6: *Exemplo de escolha dos códigos usados nos testes da base de dados NIST-ICE. (*) indica que é necessário pelo menos uma redução módulo M do índice localizador na lista* $\mathcal{L}$.

M	i	θ_{sam}	θ_{ref}	$l(c_1) = 1$ $c_1 = \theta_{\text{ref}}$	$l(c_2) =$ $\lceil$M/2$\rceil$	$l(c_3) =$ M	$l(c_4) =$ $\lceil$M/4$\rceil$	$l(c_5) =$ $\lceil$3M/4$\rceil$	$l(c_6) =$ $\lceil$3M/8$\rceil$	$l(c_7) =$ $\lceil$5M/8$\rceil$	$l(c_8) =$ $\lceil$M/8$\rceil$	$l(c_9) =$ $\lceil$7M/8$\rceil$
7	3	5	6	6	4	7	-	-	-	-	-	-
7	9	3	1	1	4	7	2	6	-	-	-	-
15	5	12	11	11	8	15	4	13*	-	-	-	-
15	7	15	3	3	8	1*	4	12	6	10	-	-
25	9	1	7	7	13	25	8*	19	10	16	4	22
25	3	3	19	19	13	25	-	-	-	-	-	-
31	7	6	25	25	16	31	8	24	12	20	-	-
31	9	27	2	2	16	31	8	24	12	20	4	28

Na Tabela 4.7 é possível ver um exemplo das escolhas das imagens de referência e das imagens de teste para um determinado usuário que possui um máximo de 12 imagens no banco de dados. Não é necessário realizar nenhuma comparação quando a última imagem, de um determinado usuário, é considerada como sendo imagem de referência, pois esta imagem já foi comparada com todas as anteriores. As comparações são únicas, ou seja, não há nenhuma duplicação de comparações entre quaisquer imagens, em outras palavras, a comparação entre a imagem de referência 2 e a imagem de teste 10 é semelhante a comparação entre a imagem de referência 10 e a imagem de teste 2.

Tabela 4.7: *Exemplo de escolha das imagens de referência e de teste usadas nos testes da base de dados NIST-ICE, para um usuário que possui 12 imagens, no máximo.*

img. de ref.	img.1 de teste	img.2 de teste	img.3 de teste	img.4 de teste	img.5 de teste	img.6 de teste	img.7 de teste	img.8 de teste	img.9 de teste	img.10 de teste	img.11 de teste
1	2	3	4	5	6	7	8	9	10	11	12
2	3	4	5	6	7	8	9	10	11	12	
3	4	5	6	7	8	9	10	11	12		
4	5	6	7	8	9	10	11	12			
5	6	7	8	9	10	11	12				
6	7	8	9	10	11	12					
7	8	9	10	11	12						
8	9	10	11	12							
9	10	11	12								
10	11	12									
11	12										

4.4 Resultados obtidos com a busca por rotação e voto de maioria

Todos os testes realizados, foram executados com a busca por rotação [62] sempre ativada. Na Tabela 4.8 são apresentados os resultados obtidos para a base de dados BIOSECURE, para uma íris. Na coluna onde o i é igual a 1, estão os resultados obtidos por [62], para uma íris, conforme indicado na própria tabela. Os resultados desta coluna podem ser comparados com as colunas onde $i = 3$, ou $i = 5$, ou $i = 7$, ou $i = 9$, também, respectivamente, para uma íris. É possível observar na Tabela 4.8 que os percentuais, na sua maioria, tendem a diminuir quanto maior for o valor do i, para o mesmo valor de t_{RS}.

Para realizar a comparação entre o método usado no Capítulo 3 e o método proposto neste Capítulo, serão analisados os percentuais de erro para t_{RS} e para a soma total de erros, que está na última linha da Tabela 4.8. Para $t_{\mathrm{RS}} = 1$, para uma íris, a FRR é igual a 15,1500% para $i = 1$ e uma FRR igual a 10,3826% para $i = 9$, isto resulta em um ganho de 31,4800%. Analisando o total de erros, tem-se uma FRR igual a 81,0500% para $i = 1$ e uma FRR igual a 65,4853% para $i = 9$, isto resulta em um ganho de 19,2000%. Para $t_{\mathrm{RS}} = 1$, para duas íris, a FRR é igual a 2,6650% para $i = 1$ e uma FRR igual a 2,5297% para $i = 9$, isto resulta em um ganho de cerca de 5,1%. Analisando o total de erros, tem-se uma FRR igual a 6,7230% para $i = 1$ e uma FRR igual a 5,9578% para $i = 9$, isto resulta em um ganho de cerca de 11,4%. Para a base BIOSECURE, o melhor resultado foi obtido nos experimentos em que $i = 9$.

Na Tabela 4.9, são apresentados os resultados obtidos com os experimentos realizados com a base de dados CASIA. Nestes experimentos a busca por rotação é sempre empregada.

Tabela 4.8: *Percentual da FRR para a base de dados BIOSECURE, empregando a busca por rotação, $i = 1$, 3, 5, 7 ou 9, para uma íris e para ambas as íris. Obs: N/D = Não Disponível*

BIOSECURE	uma íris					ambas as íris				
C_{10}^{i}	1	120	252	120	10	1	120	252	120	10
Rounds	N/D	2	1	2	24	N/D	2	1	2	24
total	N/D	240	252	240	240	N/D	240	252	240	240
i / t_{RS}	1 img	3 img	5 img	7 img	9 img	1 img	3 img	5 img	7 img	9 img
1	**15,1500**	11,4974	10,7935	10,5343	**10,3826**	**2,6650**	2,7019	2,5798	2,5201	**2,5297**
2	**11,0400**	8,7883	8,5284	8,4058	**8,3798**	**1,7480**	1,8347	1,7849	1,7579	**1,7431**
3	**8,6300**	7,5006	7,4091	7,3546	**7,3545**	**1,1070**	1,1754	1,1052	1,1163	**1,0786**
4	**7,4900**	6,7391	6,6846	6,6790	**6,7281**	**0,6540**	0,6126	0,5267	0,5311	**0,4822**
5	**6,6300**	6,1269	6,0638	6,0633	**6,1252**	**0,3350**	0,2225	0,1603	0,1460	**0,1142**
6	**5,9900**	5,5499	5,4673	5,4565	**5,4940**	**0,1480**	0,0431	0,0218	0,0164	**0,0100**
7	**5,3900**	4,9506	4,8394	4,8019	**4,7990**	**0,0550**	0,0053	0,0004	0,0004	**0**
8	**4,7400**	4,3464	4,2107	4,1458	**4,1174**	**0,0090**	0,0003	0	0	**0**
9	**4,0600**	3,7286	3,5666	3,4954	**3,4843**	**0,0020**	0	0	0	**0**
10	**3,4000**	3,0410	2,8677	2,7953	**2,8140**	**0**	0	0	0	**0**
11	**2,7400**	2,3312	2,1584	2,1157	**2,1224**	**0**	0	0	0	**0**
12	**2,0000**	1,6918	1,5580	1,5158	**1,5312**	**0**	0	0	0	**0**
13	**1,5900**	1,1653	1,0935	1,0713	**1,0771**	**0**	0	0	0	**0**
14	**0,9500**	0,7738	0,7391	0,7214	**0,7140**	**0**	0	0	0	**0**
15	**0,6000**	0,4834	0,4506	0,4253	**0,4036**	**0**	0	0	0	**0**
16	**0,3600**	0,2508	0,2306	0,1940	**0,1755**	**0**	0	0	0	**0**
17	**0,1900**	0,1124	0,0885	0,0626	**0,0426**	**0**	0	0	0	**0**
18	**0,0600**	0,0333	0,0211	0,0106	**0,0048**	**0**	0	0	0	**0**
19	**0,0400**	0,0086	0,0035	0,0014	**0,0002**	**0**	0	0	0	**0**
20	**0**	0,0007	0,0002	0,0001	**0**	**0**	0	0	0	**0**
21	**0**	0	0	0	**0**	**0**	0	0	0	**0**
22	**0**	0	0	0	**0**	**0**	0	0	0	**0**
Total	**81,0500**	69,1203	66,7746	65,8501	**65,7505**	**6,7230**	6,5958	6,1791	6,0882	**5,9578**

Para $t_{RS} = 1$, $i = 1$ e para uma íris, a FRR é igual a 23,1100%; para $i = 9$ a FRR é igual a 12,3849%, o que resulta em um ganho de 46,41%. Para o total de erros, $i = 1$ e para uma íris, tem-se uma FRR igual a 76,7600%; para $i = 9$ a FRR é igual a 38,8990%, o que resulta em um ganho de 49,32%. Para $t_{RS} = 1$, $i = 1$ e para duas íris, a FRR é igual a 2,7270%; para $i = 9$ a FRR é igual a 1,9117%, o que resulta em um ganho de cerca de 29,9%. Para o total de erros, $i = 1$ e para uma íris, a FRR é igual a 3,8590%; para $i = 9$ a FRR é igual a 2,1725%, o que resulta em um ganho de cerca de 43,7%.

Na Tabela 4.10, é possível observar os resultados obtidos com os experimentos realizados com a base de dados NIST-ICE. Nestes experimentos a busca por rotação [62] está sempre ativada. Para $t_{RS} = 1$, $i = 1$, para uma íris e para o olho direito (exp1), a FRR é igual a 21,3700%; para $i = 9$ a FRR é igual a 6,7570%, o que resulta em um ganho de cerca de 68,4%. Para o total de erros, $i = 1$ e para uma íris, a FRR é igual a 70,9300%; para $i = 9$

Tabela 4.9: *Percentual da FRR para a base de dados CASIA, empregando a busca por rotação, $i = 1, 3, 5, 7$ ou 9, para uma íris e para ambas as íris. Obs: N/D = Não Disponível*

CASIA	uma íris					ambas as íris				
C_{10}^{i}	1	120	252	120	10	1	120	252	120	10
Rounds	N/D	2	1	2	24	N/D	2	1	2	24
total	N/D	240	252	240	240	N/D	240	252	240	240
i / t_{RS}	1 img	3 img	5 img	7 img	9 img	1 img	3 img	5 img	7 img	9 img
1	**23,1100**	15,8923	13,8890	12,9478	**12,4000**	**2,7270**	2,7579	2,1853	1,9806	**1,9117**
2	**15,0900**	10,1897	8,8208	8,2403	**7,9310**	**0,9130**	0,4867	0,3104	0,2593	**0,2539**
3	**10,8200**	7,1367	6,0853	5,6397	**5,3260**	**0,1900**	0,0196	0,0079	0,0060	**0,0069**
4	**7,8100**	5,0909	4,2380	3,8541	**3,5781**	**0,0290**	0,0006	0	0	**0**
5	**5,8500**	3,6596	2,9825	2,7036	**2,5426**	**0**	0	0	0	**0**
6	**4,4200**	2,6952	2,2171	2,0435	**1,9564**	**0**	0	0	0	**0**
7	**3,2900**	2,0555	1,7478	1,6483	**1,5907**	**0**	0	0	0	**0**
8	**2,4900**	1,5549	1,3821	1,3400	**1,3048**	**0**	0	0	0	**0**
9	**1,6900**	1,1210	1,0265	1,0176	**0,9952**	**0**	0	0	0	**0**
10	**1,0900**	0,7285	0,6855	0,6940	**0,6771**	**0**	0	0	0	**0**
11	**0,6400**	0,4312	0,3964	0,4006	**0,3879**	**0**	0	0	0	**0**
12	**0,2800**	0,2225	0,1899	0,1965	**0,1893**	**0**	0	0	0	**0**
13	**0,1400**	0,0952	0,0768	0,0786	**0,0738**	**0**	0	0	0	**0**
14	**0,0300**	0,0308	0,0264	0,0253	**0,0245**	**0**	0	0	0	**0**
15	**0,0100**	0,0050	0,0057	0,0049	**0,0062**	**0**	0	0	0	**0**
16	**0**	0,0001	0,0004	0,0003	**0,0007**	**0**	0	0	0	**0**
17	**0**	0,0001	0	0,0001	**0**	**0**	0	0	0	**0**
18	**0**	0	0	0	**0**	**0**	0	0	0	**0**
19	**0**	0	0	0	**0**	**0**	0	0	0	**0**
20	**0**	0	0	0	**0**	**0**	0	0	0	**0**
21	**0**	0	0	0	**0**	**0**	0	0	0	**0**
22	**0**	0	0	0	**0**	**0**	0	0	0	**0**
Total	**76,7600**	50,9090	43,7702	40,8351	**38,9843**	**3,8590**	3,2647	2,5037	2,2458	**2,1725**

a FRR é igual a 21,8900%, o que resulta em um ganho de cerca de 69,1%. Para $t_{RS} = 1$, $i = 1$, para uma íris e para o olho esquerdo (exp2), a FRR é igual a 24,3400%; para $i = 9$ a FRR é igual a 7,6320%, o que resulta em um ganho de cerca de 68,6%. Para o total de erros, $i = 1$ e para uma íris, a FRR é igual a 84,8700%; para $i = 9$ a FRR é igual a 24,2020%, o que resulta em um ganho de cerca de 71,5%. Para $t_{RS} = 1$, $i = 1$ e para duas íris, a FRR é igual a 2,9010%; para $i = 9$ a FRR é igual a 0,2510%, o que resulta em um ganho de cerca de 91,3%. Para o total de erros, $i = 1$ e para uma íris, a FRR é igual a 6,1520%; para $i = 9$ a FRR é igual a 0,4890%, o que resulta em um ganho de cerca de 92,1%.

Tabela 4.10: *Percentual da FRR para a base de dados NIST-ICE, empregando a busca por rotação, $i = 1, 3, 5, 7$ ou 9, para uma íris e para duas íris.*

NIST-ICE	exp1 - olho direito					exp2 - olho esquerdo					Ambos				
t_{RS} \ i	1	3	5	7	9	1	3	5	7	9	1	3	5	7	9
1	**21,3700**	10,3240	8,5690	7,8540	**6,7570**	**24,3400**	11,0830	9,5380	8,1330	**7,6320**	**2,9010**	0,4710	0,2770	0,3320	**0,2510**
2	**13,7500**	6,2710	5,1140	4,5520	**4,1700**	**16,1700**	6,7470	5,7210	5,0710	**4,7130**	**1,5140**	0,1340	0,0910	0,1340	**0,1210**
3	**9,5300**	4,0960	3,3650	3,0180	**2,7240**	**11,4400**	4,3680	3,8190	3,4080	**3,2300**	**0,8600**	0,0320	0,0320	0,0910	**0,0650**
4	**6,9700**	3,0070	2,3250	2,0900	**2,0000**	**8,7300**	3,0730	2,6930	2,3320	**2,4270**	**0,4710**	0,0160	0,0140	0,0370	**0,0380**
5	**5,2300**	2,1970	1,7770	1,5910	**1,5260**	**6,5900**	2,1980	1,8310	1,6720	**1,7790**	**0,2120**	0,0160	0,0040	0,0160	**0,0120**
6	**3,8300**	1,6540	1,3780	1,2090	**1,1930**	**4,8900**	1,5700	1,2920	1,2310	**1,2560**	**0,0980**	0	0	0,0050	**0,0020**
7	**2,7900**	1,1950	1,0150	0,8920	**0,8870**	**3,6900**	1,1560	0,8920	0,8780	**0,8230**	**0,0550**	0	0	0,0050	**0**
8	**2,1500**	0,8980	0,7830	0,6740	**0,6960**	**2,6900**	0,8420	0,6100	0,5910	**0,6260**	**0,0290**	0	0	0	**0**
9	**1,5600**	0,7070	0,5810	0,5080	**0,5460**	**1,9300**	0,5980	0,4190	0,4210	**0,4250**	**0,0080**	0	0	0	**0**
10	**1,1600**	0,5210	0,4450	0,3740	**0,4340**	**1,3800**	0,4530	0,3320	0,3120	**0,3250**	**0,0040**	0	0	0	**0**
11	**0,8300**	0,3600	0,3360	0,2760	**0,3270**	**0,9100**	0,3550	0,2660	0,2500	**0,2460**	**0**	0	0	0	**0**
12	**0,6200**	0,1940	0,2290	0,2020	**0,2370**	**0,6300**	0,2590	0,2180	0,2070	**0,2090**	**0**	0	0	0	**0**
13	**0,3900**	0,1090	0,1500	0,0980	**0,1880**	**0,4600**	0,2000	0,1730	0,1660	**0,1730**	**0**	0	0	0	**0**
14	**0,2400**	0,0520	0,1040	0,0630	**0,1090**	**0,3500**	0,1500	0,1390	0,1300	**0,1340**	**0**	0	0	0	**0**
15	**0,1600**	0,0270	0,0440	0,0410	**0,0520**	**0,2500**	0,0980	0,1160	0,0960	**0,1000**	**0**	0	0	0	**0**
16	**0,1200**	0,0030	0,0190	0,0300	**0,0250**	**0,1800**	0,0640	0,0890	0,0710	**0,0640**	**0**	0	0	0	**0**
17	**0,0800**	0,0030	0,0110	0,0220	**0,0110**	**0,1000**	0,0340	0,0550	0,0410	**0,0320**	**0**	0	0	0	**0**
18	**0,0500**	0	0,0050	0,0250	**0,0050**	**0,0800**	0,0090	0,0300	0,0200	**0,0070**	**0**	0	0	0	**0**
19	**0,0400**	0,0030	0,0030	0,0140	**0,0030**	**0,0300**	0	0,0090	0,0090	**0**	**0**	0	0	0	**0**
20	**0,0300**	0	0	0,0080	**0**	**0,0100**	0	0,0050	0,0050	**0**	**0**	0	0	0	**0**
21	**0,0200**	0	0,0030	0,0030	**0**	**0,0100**	0	0	0	**0**	**0**	0	0	0	**0**
22	**0,0100**	0	0	0	**0**	**0,0100**	0	0	0	**0**	**0**	0	0	0	**0**
total	**70,9300**	31,6220	26,2570	23,5440	**21,8900**	**84,8700**	33,2560	28,2470	25,0420	**24,2020**	**6,1520**	0,6690	0,4180	0,6210	**0,4890**

4.5 CONCLUSÕES SOBRE A AUTENTICAÇÃO USANDO VOTO MAJORITÁRIO

Os resultados encontrados, a partir das simulações usando a técnica de voto majoritário, neste Capítulo, foram obtidos sempre com o emprego da busca por rotação, de modo que a comparação natural é feita entre o resultado obtido aqui, neste Capítulo, e o resultado obtido no Capítulo 3. Estes resultados são apresentados na Figura 4.2 para uma única íris, e na Figura 4.3 para ambas as íris.

Na Figura 4.2, o t_{RS} varia de 1 até 10 e para cada t_{RS} são apresentadas oito barras, para as quatro opções de teste que foram realizados empregando uma única íris. É possível visualizar as barras como um conjunto de quatro pares, um par para cada base de dados, respectivamente, BIOSECURE, CASIA, NIST-ICE(exp1) e NIST-ICE(exp2).

A barra mais à esquerda de cada par é a representação gráfica dos dados referentes a uma única íris, para $i = 1$, das Tabelas 4.8, 4.9 e 4.10, respectivamente. A barra mais à direita de cada par é a representação gráfica dos dados referentes a uma única íris, para $i = 9$, das Tabelas 4.8, 4.9 e 4.10, respectivamente.

É possível observar graficamente, que os resultados empregando o voto majoritário com nove imagens é melhor que o resultado obtido quando apenas uma imagem é usada nos experimentos, respectivamente, para cada base de dados.

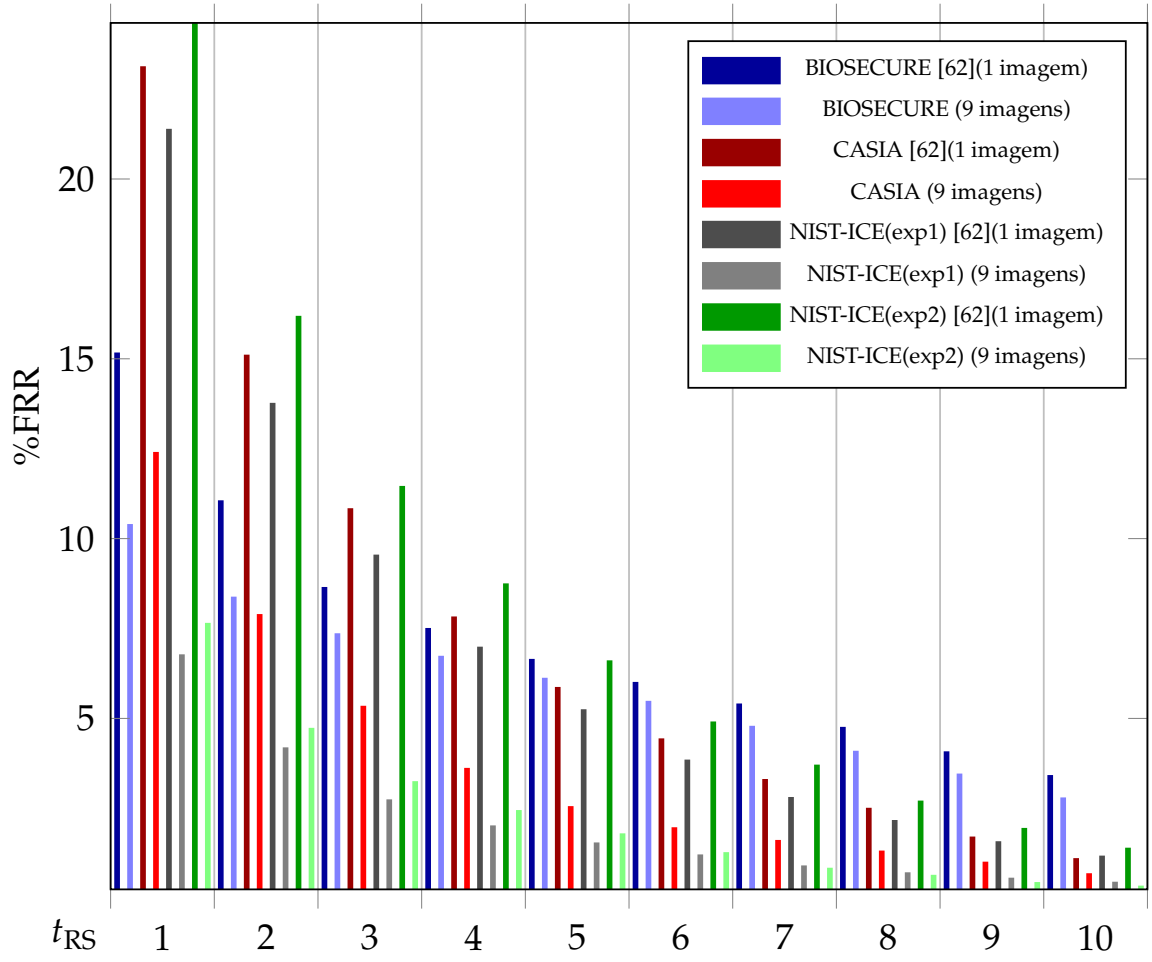


Figura 4.2: *Diagrama de Barras ilustrando o percentual médio da FRR das Tabelas 4.8, 4.9 e 4.10, para uma única íris.*

Na Figura 4.3 são apresentados os resultados obtidos empregando ambas as íris, para as bases de dados BIOSECURE, CASIA e NIST-ICE, respectivamente. É possível observar um conjunto de seis barras para cada t_{RS}, variando de 1 até 10. As seis barras, podem ser vistas como sendo três pares, de modo que cada barra mais à esquerda representa o resultado obtido para a respectiva base de dados quando $i = 1$ e a barra mais à direita representa o resultado obtido para a respectiva base de dados quando $i = 9$.

Mais uma vez é possível observar que todos os resultado obtidos, quando é empregado o voto majoritário composto por nove imagens, é melhor que o resultado obtido quando

apenas uma imagem é empregada nos experimentos. Vale ressaltar que para $t_{RS} \geq 4$, muitos percentuais obtidos da FRR já são iguais a zero e que para $t_{RS} \geq 11$, todos os percentuais da FRR são iguais a zero.

Para todos os experimentos realizados neste Capítulo, a FAR sempre foi igual a zero para qualquer t_{RS} e para todas as bases de dados utilizadas.

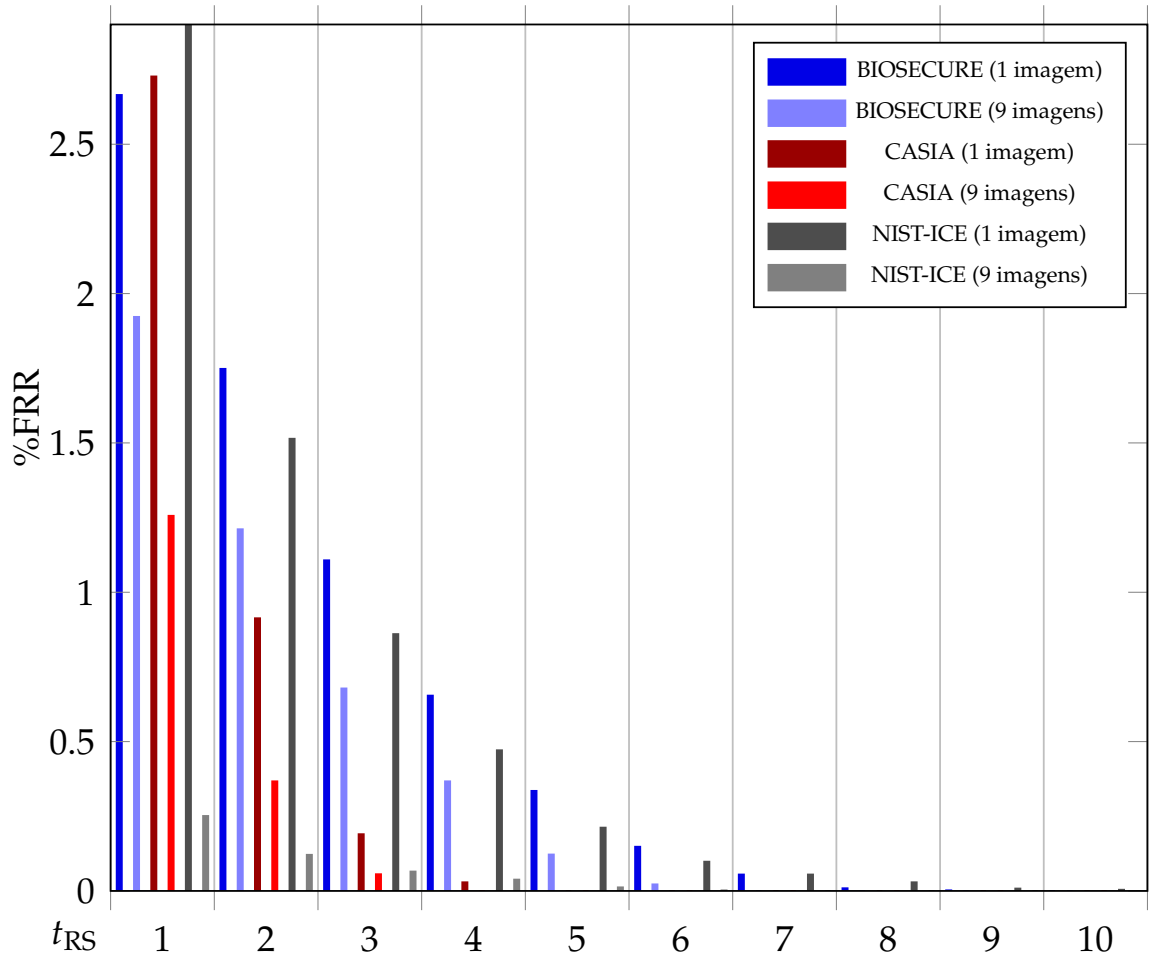


Figura 4.3: *Diagrama de Barras ilustrando o percentual médio da FRR das Tabelas 4.8, 4.9 e 4.10, para ambas as íris.*

O ganho obtido pela técnica proposta, aqui, neste Capítulo, em comparação com a técnica proposta no Capítulo 3 depende da base de dados, da quantidade de íris usada nos experimentos e do t_{RS}. Para estabelecer um parâmetro de comparação justo, na medida do possível, foi construída a Tabela 4.11, na qual são colocados os erros totais aproximados da técnica utilizada, aqui, neste Capítulo, e os erros totais aproximados obtidos no Capítulo 3. É possível verificar que o ganho médio aproximado para uma íris é de cerca de 59,1%, já

para ambas as íris, o ganho é de cerca de 60,9%.

Tabela 4.11: *Quantidade de erros (em média) empregando a busca por rotação versus voto majoritário (com 9 imagens).*

Proposta	Uma íris	Ambas as íris
Busca por rotação	30.568	701
Voto de maioria	12.504	274
Ganho	59,1%	60,9%

Na Tabela 4.12, é possível verificar o desempenho do esquema proposto, neste Capítulo, em relação aos melhores resultados obtidos até o momento, usando as mesmas bases de dados. Também foi incluído o resultado obtido no Capítulo 3. Agora é possível recuperar chaves criptográficas com 222 *bits* com percentual da FRR de $0,2370\%$ e FAR de 0% para a base de dados NIST-ICE(exp1), usando uma única íris; e chaves criptográficas com 413 *bits* com percentual da FRR de $0,2510\%$ e FAR de 0% para a base de dados NIST-ICE, usando ambas as íris. É provável que a técnica de voto majoritário, apresentada aqui para o código de íris, possa ser implementada como um pré-processamento em outros sistemas de identificação biométrica, contribuindo para uma melhoria na identificação dos usuários.

Tabela 4.12: *Resultado percentual da FRR para diversos algoritmos biométricos. O algoritmo proposto emprega o voto da majoritário.*

Esquema	ECC	Comprimento da chave $\|\|K\|\|$ (em *bits*)	FRR (%)	FAR (%)	Base de Dados
Referência [46]	RS	282	8,4200	0	NIST-ICE (íris direita)
Referência [2]	RS	128/256	0,7600	0,1000	NIST-ICE (íris direita)
Referência [57]	RMP	42	0,4700	0	NIST-ICE (íris direita)
Referência [58]*	RSH	147	0,1800	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	231	0	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	287	0,3400	0	NIST-ICE (ambas as íris)
Busca por rotação	RSH	198	0,2400	0	NIST-ICE (íris direita)
Busca por rotação*	RSH	273	0	0	NIST-ICE (ambas as íris)
Busca por rotação*	RSH	371	0,4700	0	NIST-ICE (ambas as íris)
Voto de maioria	**RSH**	**222**	**0,2370**	**0**	**NIST-ICE (íris direita)**
Voto de maioria*	**RSH**	**329**	**0**	**0**	**NIST-ICE (ambas as íris)**
Voto de maioria*	**RSH**	**413**	**0,2510**	**0**	**NIST-ICE (ambas as íris)**

Obs: (*) denota os sistemas multi biométricos; ECC: código corretor de erros; RSH: código Reed Solomon e Hadamard; RMP: códigos produto baseados nos códigos de Reed Muller.
O percentual FAR só é diferente de zero em [2].

Capítulo 5

Autenticação adaptativa com busca por rotação

"Precisamos analisar o todo para depois, compreendermos as partes..."

— Aristóteles

Este Capítulo aborda a técnica de alteração da proporção de *bits* de dados e *bits* inseridos no pré-processamento das palavras do código íris. O padrão da proporção utilizado até o momento é o de três *bits* do código íris para dois *bits* inseridos. Este padrão foi o utilizado nos Capítulos 3 e 4, porém é possível alterar a proporção entre a quantidade de *bits* do código e a quantidade de *bits* a serem inseridos. Neste Capítulo, são abordadas quatro novas proporções, que são 4:3, 5:4, 6:5 e 7:6, onde o primeiro número representa a quantidade de *bits* do código e o segundo número representa a quantidade de *bits* a serem inseridos. Ao alterar a proporção de *bits*, é alterada a quantidade de *bits* adicionados ao final do processo de inserção, muitas vezes extrapolando o limite máximo disponível para codificação/decodificação pelo código de Hadamard. Devido a este motivo é preciso truncar a palavra-código resultante, perdendo alguns *bits* do código íris, por isso o código íris foi analisado para buscar uma alternativa de diminuição da quantidade de *bits* truncados

e, consequentemente, perdidos. Esse método foi chamado de remoção da "redundância" e está diretamente relacionado com a alteração da proporção de *bits* inseridos, fato que fez com que ambas as técnicas fossem apresentadas em conjunto, embora uma não dependa da outra para ser aplicada. As duas técnicas combinadas foram chamadas de autenticação adaptativa.

5.1 ANÁLISE DO CÓDIGO ÍRIS

O código íris usado nos experimentos segue a pesquisa desenvolvida por [3], o qual foi obtido usando o *software* OSIRIS [63] a partir da codificação das imagens disponibilizadas pelas bases BIOSECURE, CASIA e NIST-ICE. Este código resultante do *software* é composto por sequências de comprimento igual a 1.188 *bits* para cada imagem e, por enquanto, não há nenhuma pesquisa que tenha conseguido encontrar qual a dependência que existe entre os *bits* desta sequência resultante. As bases de dados usadas em toda esta tese são provenientes do estudo iniciado por [46] e continuado por [3], por este motivo as comparações iniciais foram feitas entre os resultados obtidos aqui, e os resultados obtidos pelos dois estudos iniciais.

Para diminuir a quantidade de *bits* truncados, foi feita uma análise dos dados do código íris, para verificar se seria possível escolher os *bits* a serem removidos do código íris intacto, ou seja, sem nenhuma modificação. A ideia inicial foi buscar partes do código nas quais havia, no máximo 10% de divergência, entre os *bits* referentes a cada uma das imagens de referência e a cada uma das imagens de teste de um determinado usuário. Em caso de obtenção destes posicionamentos dos *bits*, seria possível removê-los, sem alterar significativamente a distância de Hamming entre o código íris inalterado e a distância de Hamming do código com os *bits* removidos nas posições escolhidas.

Na Tabela 5.1 é possível observar um exemplo montado para explicar a ideia da remoção de "redundância". Neste exemplo, são removidos 2 *bits*, os que estão nas posições 5 e 8 respectivamente. Estas posições são iguais em todas as três imagens de referência e em todas as três imagens de teste, o que irá resultar na manutenção das distâncias de Hamming entre quaisquer duas imagens escolhidas, antes ou depois da remoção da "redundância", conforme pode ser visto nas colunas três e seis da Tabela 5.1.

A princípio, não é possível garantir que serão encontradas posições nas quais todos os *bits* sejam iguais, porém se houver poucas imagens com *bits* diferentes em uma mesma

Tabela 5.1: *Exemplo da técnica de remoção de "redundância" com sequências de* 8 bits*, removendo* 2 bits*.*

Imagem	código inalterado	Distância de Hamming para Img. de ref. 1	posição dos 2 *bits* removidos	código com redundância removida	Distância de Hamming para Img. de ref. 1
Img. de ref. 1	11101001	0	5 e 8	111000	0
Img. de ref. 2	10101001	1	5 e 8	101000	1
Img. de ref. 3	11101101	1	5 e 8	111010	1
Img. de teste. 1	10101001	1	5 e 8	101000	1
Img. de teste. 2	00111011	4	5 e 8	001101	4
Img. de teste. 3	10011001	3	5 e 8	100100	3

posição, ainda assim, será possível utilizar a técnica, pois na realidade, serão removidos erros das imagens, o que resultará em provável melhoria na identificação deste usuário. A remoção de erros acontecerá, pois se todos os *bits* de uma determinada posição forem iguais, com exceção de uma posição, é muito provável que esta posição diferente, que é minoria, está com o *bit* errado, e ao utilizar a técnica proposta, esta posição será removida, reduzindo, portanto, a distância de Hamming entre esta imagem e qualquer outra imagem deste mesmo usuário.

5.2 SISTEMA EMPREGANDO A AUTENTICAÇÃO ADAPTATIVA COM BUSCA POR ROTAÇÃO

Na Figura 5.1 é possível observar o diagrama de blocos do sistema proposto para implementação da remoção da "redundância" e da alteração na proporção de *bits* inseridos no código íris.

Este sistema é similar ao sistema proposto no Capítulo 3, com alteração no bloco que faz o "embaralhamento e inserção uniforme de *bits*", para fazer também a remoção da redundância. Este bloco precisa ser carregado com as informações do número de "*data bits*" e do número de "*insert* bits", que são respectivamente a quantidade de *bits* do código que serão usados para compor o θ'_{ref} ou o θ'_{sam} que serão entregues aos blocos de codificação ou decodificação, respectivamente.

Os demais blocos do sistema são os mesmos usados no Capítulo 3 sem nenhuma alteração. Os resultados obtidos neste Capítulo são provenientes apenas das alterações na proporção de *bits* de dados versus *bits* inseridos e na remoção da "redundância". Este sistema permite realizar experimentos com apenas uma íris ou com ambas as íris.

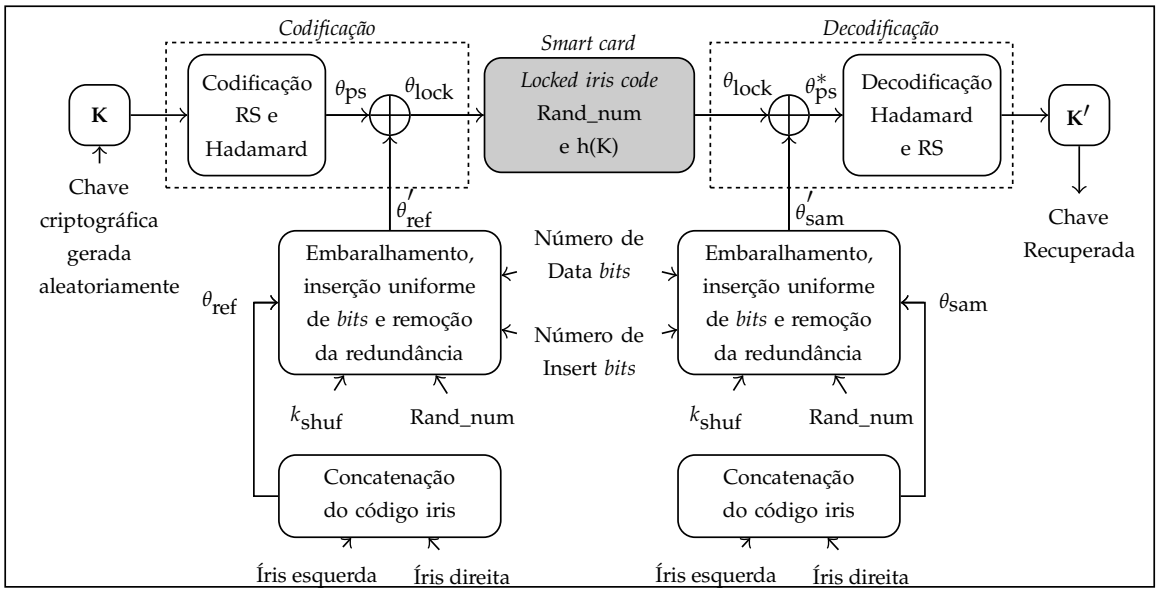


Figura 5.1: *Sistema de regeneração de chave uni ou multi biométrico com alteração da proporção de* bits *inseridos e remoção da redundância, empregando smart card, íris e senha.*

5.3 REMOÇÃO DE REDUNDÂNCIA NAS BASES DE DADOS

Para poder usar as bases de dados no *software* desenvolvido em C++, foi necessário converter as mesmas para o formato de texto simples .txt. Ao realizar a conversão, naturalmente, foi usada a representação em binário dos códigos, uma vez que esta representação é a mesma usada no *software* usado anteriormente, desenvolvido em MATLAB®. Ao concluir a conversão, foi verificado que os arquivos em texto simples eram relativamente grandes em termos de espaço gasto em *Mega Bytes* (MB).

Os arquivos originais foram, então, convertidos para texto simples em hexadecimal, tais como o que é mostrado na Figura 5.2. A representação das imagens é feita por linha, com um identificador de início, composto pelo caractere "l"(de linha), o número da linha com quatro caracteres e o caractere "i"(de início) "l0001i", seguidos dos caracteres que representam o código íris em hexadecimal e terminando cada linha, que representa uma imagem, com o caractere "f"(de fim).

Ao rolar a barra de rolagem, para verificar se a conversão havia sido feita de maneira correta, é possível observar que alguns caracteres praticamente não eram alterados desde o início até o final do arquivo. Curiosamente, esta observação contribuiu para fundamentar a ideia de que havia uma “redundância” em cada código binário, gerado a partir das imagens

do código íris, usando o OSIRIS [63], e que poderia ser possível aumentar a quantidade de *bits* inseridos, sem perder *bits* originais do código.

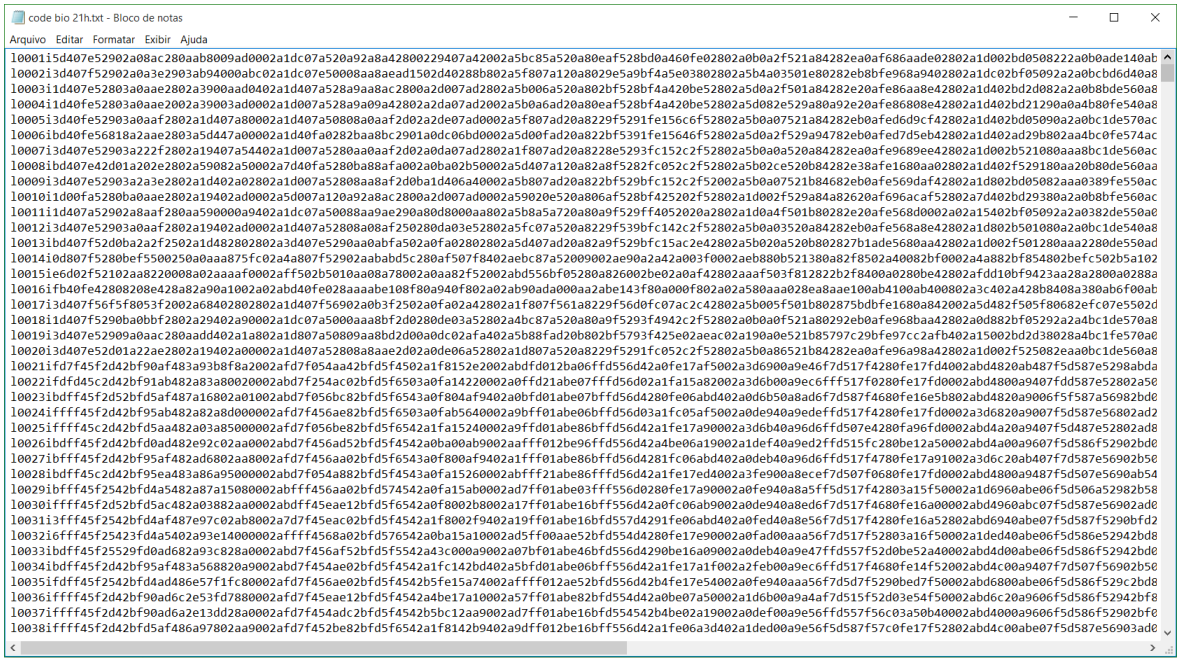


Figura 5.2: *Imagem do arquivo em texto claro do código íris da base de dados BIOSECURE rotação 10 para direita em hexadecimal.*

Na Figura 5.3 foram acrescentados espaços entre os *bits* identificados, em hexadecimal, que praticamente não são alterados desde a primeira linha até a última linha. Existem ainda vários casos em que os dois *bits* em binário adjacentes a estes em hexadecimal são repetidos para muitos usuários, porém nos experimentos realizados neste Capítulo, estes *bits* adicionais não foram alterados.

Ao observar as sequências da Figura 5.3 é possível identificar que a quantidade de caracteres entre cada conjunto de dois caracteres em destaque é de 31, que se representados em binário, são 124 *bits*. Deste modo, a cada 124 *bits* de informação, existem 8 *bits* de "controle", ou algo semelhante, que aparentemente não são relevantes para contribuir na identificação de um determinado usuário. Estes 8 *bits* a cada 124 *bits* foram os escolhidos para serem removidos do código íris, permitindo que quando fosse feita a inserção dos *bits* pseudo aleatórios provenientes da rotina que gera os *Random_num*, seja possível reduzir a quantidade de *bits* de dados perdidos ao realizar o truncamento do código, que resultará no θ'_{ref} para a etapa de codificação ou no θ'_{sam} para a etapa de decodificação. No total, são removidos 72 *bits* de "redundância" de cada linha (palavra) do código íris, para uma íris e 144 *bits* para

duas íris.

Para todas as bases de dados usadas nos experimentos, é possível observar que existem as mesmas repetições, nas mesmas posições do exemplo da Figura 5.3, tornando possível a utilização da mesma rotina de remoção de *bits* para qualquer usuário.

Figura 5.3: *Imagem do arquivo em texto claro do código íris da base de dados BIOSECURE rotação 10 para direita em hexadecimal, com espaços.*

Na Tabela 5.2, é possível observar o comprimento em *bits* original e o comprimento reduzido do código íris, após a remoção dos *bits* "redundantes".

Tabela 5.2: *Comprimento em* bits *do código íris usando a técnica de remoção de "redundância".*

Comprimento	uma íris	duas íris
Código original	1.188 *bits*	2.376 *bits*
Código reduzido	1.116 *bits*	2.232 *bits*

5.4 PROPORÇÃO DE *bits* INSERIDOS

A ideia de inserção de *bits* foi introduzida por Kanade [46], e tem como propósito aumentar a capacidade de correção de erros do código de Hadamard. Originalmente são inseridos 2 *bits* iguais a zero após três *bits* do código íris. A partir do estudo realizado por

Câmara iniciado em 2012 e publicado em [3], a inserção de zeros foi substituída pela inserção de números pseudo aleatórios. Ambos os estudos mantiveram a proporção de 3:2, ou seja, três *bits* de dados para dois *bits* de números pseudo aleatórios.

Um problema inicial na proporção de *bits* de dados versus *bits* inseridos é que o número de *bits* de dados escolhidos deve, a princípio, ser divisor da quantidade de *bits* do código íris, para evitar perda de informação. No caso padrão, 3 divide 1.188, então tem-se exatamente 396 conjuntos de 3 *bits* cada, os quais serão acrescidos de 2 *bits*, pseudo aleatórios, formando 396 conjuntos de 5 *bits*, resultando em 1.980 *bits*.

Analisando o processo de inserção e depois de truncamento, é possível verificar que a alteração na proporção de *bits* de dados versus *bits* inseridos não gera perda na capacidade de identificação de usuários, desde que a quantidade de *bits* de dados truncados e, consequentemente, perdidos não seja maior que 35 para uma íris, e não seja maior que 70 para ambas as íris; valores obtidos experimentalmente. Nos experimentos realizados, foi estimado que a perda de *bits* de dados de até cerca de 3% não compromete o resultado dos experimentos de identificação de usuários. No caso padrão a quantidade máxima de *bits* usados no código de Hadamard é de 1.952 (32×61), o que resulta em uma perda de 28 *bits*, dos quais 16 *bits* são de dados, representando cerca de $1,35\%$ de perda. O limite estabelecido de no máximo 3% é uma sugestão extraída diretamente dos experimentos realizados, nos quais, perdas acima deste valor nos *bits* de dados, resultaram em obtenção de maiores taxas de erros, na identificação de usuários genuínos.

Na Tabela 5.3, são apresentadas as informações sobre as proporções de *bits*, os comprimentos do código íris, os comprimentos após a inserção dos *bits* pseudo aleatórios, os comprimentos efetivos, os *bits* truncados, os *bits* de dados perdidos e o percentual de perda para cada proporção de *bits* testada. Nos experimentos, a partir da proporção 8:7 os erros de identificação começaram a aumentar significativamente para $t_{\mathrm{RS}} \leq 5$, ou seja o %FRR aumentou, motivo pelo qual os testes com esta proporção foram descartados. A explicação, até este momento, para este comportamento é que o percentual de *bits* de dados perdidos começa a aumentar, resultando em maiores dificuldades de identificação de usuários genuínos.

Ainda na Tabela 5.3 é possível observar que a partir da proporção 6:5 o valor do "n" foi alterado para 63. Esta alteração foi necessária para aumentar o comprimento efetivo máximo permitido, fazendo com que a quantidade de *bits* truncados e, consequentemente,

Tabela 5.3: *Perda de dados com a alteração da proporção de* bits *inseridos.*

Proporção	m	n	2^k	Comprimento do código íris	Comprimento com *bits* inseridos	Comprimento efetivo	*bits* truncados	*bits* de dados perdidos	percentual de perda
3:2 uma íris	6	61	32	1.188	1.980	1.952	28	16	1,35%
4:3 uma íris	6	61	32	1.116	1.953	1.952	1	0	0%
5:4 uma íris	6	61	32	1.116	2.009	1.952	57	31	2,78%
6:5 uma íris	6	63	32	1.116	2.046	2.016	30	15	1,34%
7:6 uma íris	6	63	32	1.116	2.073	2.016	57	30	2,69%
8:7 uma íris	6	63	32	1.116	2.093	2.016	77	38	3,41%
3:2 duas íris	7	61	64	2.376	3.960	3.904	56	33	1,39%
4:3 duas íris	7	61	64	2.232	3.906	3.904	2	0	0%
5:4 duas íris	7	61	64	2.232	4.018	3.904	114	62	2,78%
6:5 duas íris	7	63	64	2.232	4.092	4.032	60	30	1,34%
7:6 duas íris	7	63	64	2.232	4.146	4.032	114	60	2,69%
8:7 duas íris	7	63	64	2.232	4.185	4.032	153	80	3,58%

a quantidade de *bits* de dados perdidos se tornasse menor. O valor de "n" foi ajustado para 61 no teste padrão, justamente para se adequar à proporção 3:2.

5.5 Resultados obtidos para a autenticação adaptativa com busca por rotação

Os resultados obtidos neste Capítulo serão comparados com os resultados obtidos no Capítulo 3, pois a busca por rotação está sempre ativada em todos os experimentos realizados neste Capítulo, tornando esta, a comparação mais justa.

Na Tabela 5.4, são apresentados os percentuais da FRR para a base de dados BIOSECURE, para uma íris e para duas íris, com as alterações na proporção de *bits* de dados versus *bits* inseridos, desde o experimento padrão (3:2) até o melhor resultado obtido nos experimentos realizados aqui neste Capítulo, que foi com a proporção 7:6. O t_{RS} varia de 1 até 22.

Analisando a Tabela 5.4, é possível verificar que para uma íris e para $t_{\mathrm{RS}} = 1$, o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 31,7%, se for considerado o percentual total de erros, o ganho é de cerca de 38%. Para duas íris e para $t_{\mathrm{RS}} = 1$ o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 55%, se for considerado o percentual total de erros, o ganho é de cerca de 65%.

Na Tabela 5.5, são apresentados os percentuais da FRR para a base de dados CASIA, para uma íris e para duas íris, com as alterações na proporção de *bits* de dados versus *bits* inseridos, desde o experimento padrão (3:2) até o melhor resultado obtido nos experimentos

Tabela 5.4: *Percentual da FRR para a base de dados BIOSECURE, empregando a alteração na proporção de* bits *inseridos, para uma íris e para duas íris.*

BIOSECURE	uma íris					duas íris				
db x ib / t_{RS}	3:2	4:3	5:4	6:5	7:6	3:2	4:3	5:4	6:5	7:6
1	**15,1500**	12,5390	11,7440	10,4890	**10,3440**	**2,6650**	2,5330	2,4890	2,1780	**1,2000**
2	**11,0400**	9,3670	8,9890	8,0780	**7,9500**	**1,7480**	1,6780	1,5000	1,3780	**0,6560**
3	**8,6300**	7,7940	7,2670	6,5390	**6,5000**	**1,1070**	1,0000	0,9670	0,8110	**0,3670**
4	**7,4900**	6,6170	6,4220	5,3720	**5,4780**	**0,6540**	0,4890	0,4330	0,3670	**0,1220**
5	**6,6300**	5,8390	5,7670	4,4890	**4,5390**	**0,3350**	0,2110	0,1440	0,1330	**0,0110**
6	**5,9900**	5,1390	4,8390	3,7330	**3,7280**	**0,1480**	0,0110	0,0560	0,0440	**0**
7	**5,3900**	4,4780	3,9280	3,1280	**3,1110**	**0,0550**	0	0,0110	0	**0**
8	**4,7400**	3,7670	3,2500	2,6330	**2,4670**	**0,0090**	0	0	0	**0**
9	**4,0600**	3,2940	2,5670	2,1060	**1,9610**	**0,0020**	0	0	0	**0**
10	**3,4000**	2,5780	1,9890	1,7330	**1,4440**	**0**	0	0	0	**0**
11	**2,7400**	2,0720	1,4610	1,2830	**1,0280**	**0**	0	0	0	**0**
12	**2,0000**	1,6220	0,9670	1,0060	**0,6560**	**0**	0	0	0	**0**
13	**1,5900**	1,1280	0,5720	0,7000	**0,4610**	**0**	0	0	0	**0**
14	**0,9500**	0,7110	0,2890	0,4720	**0,2440**	**0**	0	0	0	**0**
15	**0,6000**	0,3890	0,1720	0,3000	**0,1500**	**0**	0	0	0	**0**
16	**0,3600**	0,2060	0,0890	0,1830	**0,0720**	**0**	0	0	0	**0**
17	**0,1900**	0,1000	0,0390	0,1170	**0,0560**	**0**	0	0	0	**0**
18	**0,0600**	0,0720	0,0110	0,0780	**0,0170**	**0**	0	0	0	**0**
19	**0,0400**	0,0170	0,0060	0,0500	**0,0060**	**0**	0	0	0	**0**
20	**0**	0,0060	0	0,0220	**0**	**0**	0	0	0	**0**
21	**0**	0	0	0,0110	**0**	**0**	0	0	0	**0**
22	**0**	0	0	0	**0**	**0**	0	0	0	**0**
total	**81,0500**	67,7330	60,3670	52,5220	**50,2110**	**6,7230**	5,9220	5,6000	4,9110	**2,3560**

realizados aqui neste Capítulo, que foi com a proporção 7:6. O t_{RS} varia de 1 até 22.

Analisando a Tabela 5.5, é possível verificar que para uma íris e para $t_{RS} = 1$, o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 49,6%, se for considerado o percentual total de erros, o ganho é de cerca de 55,1%. Para duas íris e para $t_{RS} = 1$ o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 78,4%, se for considerado o percentual total de erros, o ganho é de cerca de 83%.

Na Tabela 5.6 são apresentados os percentuais da FRR para a base de dados NIST-ICE, para uma íris e para duas íris, com as alterações na proporção de *bits* de dados versus *bits* inseridos, desde o experimento padrão (3:2) até o melhor resultado obtido nos experimentos realizados aqui neste Capítulo, que foi com a proporção 7:6. O t_{RS} varia de 1 até 22.

Tabela 5.5: *Percentual da FRR para a base de dados CASIA, empregando a alteração na proporção de* bits *inseridos, para uma íris e para duas íris.*

CASIA	uma íris					duas íris				
db x ib / t_{RS}	3:2	4:3	5:4	6:5	7:6	3:2	4:3	5:4	6:5	7:6
1	**23,1100**	16,9720	14,9170	12,5390	**11,6440**	**2,7270**	1,9670	1,7780	1,6440	**0,5890**
2	**15,0900**	10,9440	9,7890	7,9610	**7,5060**	**0,9130**	0,4890	0,3670	0,3780	**0,0670**
3	**10,8200**	7,7390	6,7220	5,5670	**4,9560**	**0,1900**	0,0560	0	0,0330	**0**
4	**7,8100**	5,5830	4,8500	3,8280	**3,4830**	**0,0290**	0,0110	0	0	**0**
5	**5,8500**	4,1720	3,6440	2,7940	**2,4220**	**0**	0	0	0	**0**
6	**4,4200**	3,0440	2,6060	1,9720	**1,6610**	**0**	0	0	0	**0**
7	**3,2900**	2,1390	1,7500	1,4440	**1,1610**	**0**	0	0	0	**0**
8	**2,4900**	1,5440	1,1670	0,9940	**0,7280**	**0**	0	0	0	**0**
9	**1,6900**	1,0890	0,7280	0,6670	**0,4780**	**0**	0	0	0	**0**
10	**1,0900**	0,7220	0,3830	0,4000	**0,2280**	**0**	0	0	0	**0**
11	**0,6400**	0,4280	0,2330	0,2500	**0,1170**	**0**	0	0	0	**0**
12	**0,2800**	0,2170	0,0890	0,1500	**0,0390**	**0**	0	0	0	**0**
13	**0,1400**	0,1000	0,0560	0,0610	**0,0170**	**0**	0	0	0	**0**
14	**0,0300**	0,0220	0,0110	0,0390	**0**	**0**	0	0	0	**0**
15	**0,0100**	0	0	0,0110	**0**	**0**	0	0	0	**0**
16	**0**	0	0	0	**0**	**0**	0	0	0	**0**
17	**0**	0	0	0	**0**	**0**	0	0	0	**0**
18	**0**	0	0	0	**0**	**0**	0	0	0	**0**
19	**0**	0	0	0	**0**	**0**	0	0	0	**0**
20	**0**	0	0	0	**0**	**0**	0	0	0	**0**
21	**0**	0	0	0	**0**	**0**	0	0	0	**0**
22	**0**	0	0	0	**0**	**0**	0	0	0	**0**
total	**76,7600**	54,7170	46,9440	38,6780	**34,4390**	**3,8590**	2,5220	2,1440	2,0560	**0,6560**

Analisando a Tabela 5.6, é possível verificar que para uma íris usando o olho direito (exp1) e para $t_{RS} = 1$, o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 40,3%, se for considerado o percentual total de erros, o ganho é de cerca de 43,9%. para uma íris usando o olho esquerdo (exp2) e para $t_{RS} = 1$, o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 45,6%, se for considerado o percentual total de erros, o ganho é de cerca de 53,6%. Para duas íris e para $t_{RS} = 1$ o ganho entre a proporção 7:6 e a proporção padrão 3:2 é de cerca de 44,7%, se for considerado o percentual total de erros, o ganho é de cerca de 55,7%.

Tabela 5.6: *Percentual da FRR para a base de dados NIST-ICE, empregando a alteração na proporção de* bits *inseridos, para uma íris e para duas íris.*

NIST-ICE	exp1 - olho direito					exp2 - olho esquerdo					Ambos				
db x ib / t_{RS}	3:2	4:3	5:4	6:5	7:6	3:2	4:3	5:4	6:5	7:6	3:2	4:3	5:4	6:5	7:6
1	**21,3700**	17,3540	15,2180	13,7410	**12,7610**	**24,3400**	18,1510	16,5060	13,9810	**13,2460**	**2,9010**	2,9910	2,8580	2,3920	**1,6050**
2	**13,7500**	11,4920	9,8960	8,9270	**8,3270**	**16,1700**	11,3200	10,2660	8,6830	**8,0980**	**1,5140**	1,6540	1,5200	1,2900	**0,7280**
3	**9,5300**	8,1820	7,1560	6,2250	**5,6870**	**11,4400**	7,5820	6,9880	5,9510	**5,3780**	**0,8600**	1,0010	0,8880	0,7600	**0,2190**
4	**6,9700**	5,9990	5,3570	4,5170	**3,9240**	**8,7300**	5,5140	4,8430	4,0920	**3,7170**	**0,4710**	0,5140	0,4070	0,3910	**0,1120**
5	**5,2300**	4,3990	3,7580	3,2200	**2,7020**	**6,5900**	4,1290	3,4870	2,9370	**2,6520**	**0,2120**	0,2300	0,2190	0,2250	**0,0430**
6	**3,8300**	3,2290	2,7730	2,3010	**1,9210**	**4,8900**	3,1320	2,5210	2,1130	**1,8700**	**0,0980**	0,0860	0,1230	0,1020	**0,0160**
7	**2,7900**	2,4700	1,9080	1,6430	**1,4190**	**3,6900**	2,2700	1,8180	1,5990	**1,3190**	**0,0550**	0,0320	0,0480	0,0540	**0**
8	**2,1500**	1,7550	1,3350	1,1740	**1,0180**	**2,6900**	1,6970	1,3330	1,1280	**0,9530**	**0,0290**	0,0210	0,0370	0,0110	**0**
9	**1,5600**	1,2010	0,9310	0,7890	**0,6900**	**1,9300**	1,2670	0,9330	0,7940	**0,6620**	**0,0080**	0	0,0050	0	**0**
10	**1,1600**	0,8510	0,6520	0,5240	**0,4800**	**1,3800**	0,8710	0,6440	0,6120	**0,4440**	**0,0040**	0	0	0	**0**
11	**0,8300**	0,5760	0,4200	0,3880	**0,3060**	**0,9100**	0,5960	0,4710	0,4620	**0,3210**	**0**	0	0	0	**0**
12	**0,6200**	0,3550	0,2920	0,2270	**0,2070**	**0,6300**	0,4190	0,3640	0,3250	**0,2430**	**0**	0	0	0	**0**
13	**0,3900**	0,2460	0,1610	0,1690	**0,1470**	**0,4600**	0,3210	0,2550	0,2640	**0,1730**	**0**	0	0	0	**0**
14	**0,2400**	0,1530	0,0870	0,1230	**0,0960**	**0,3500**	0,2480	0,1980	0,2160	**0,1110**	**0**	0	0	0	**0**
15	**0,1600**	0,1280	0,0550	0,0900	**0,0650**	**0,2500**	0,1890	0,1320	0,1640	**0,0710**	**0**	0	0	0	**0**
16	**0,1200**	0,0790	0,0410	0,0710	**0,0330**	**0,1800**	0,1520	0,0980	0,1140	**0,0610**	**0**	0	0	0	**0**
17	**0,0800**	0,0760	0,0300	0,0440	**0,0110**	**0,1000**	0,0930	0,0550	0,0930	**0,0360**	**0**	0	0	0	**0**
18	**0,0500**	0,0630	0,0220	0,0160	**0,0030**	**0,0800**	0,0820	0,0090	0,0640	**0,0270**	**0**	0	0	0	**0**
19	**0,0400**	0,0270	0	0,0110	**0,0030**	**0,0300**	0,0360	0,0020	0,0480	**0,0090**	**0**	0	0	0	**0**
20	**0,0300**	0,0080	0	0,0110	**0**	**0,0100**	0,0110	0	0	**0**	**0**	0	0	0	**0**
21	**0,0200**	0	0	0,0080	**0**	**0,0100**	0	0	0	**0**	**0**	0	0	0	**0**
22	**0,0100**	0	0	0,0050	**0**	**0,0100**	0	0	0	**0**	**0**	0	0	0	**0**
total	**70,9300**	58,6430	50,0900	44,2220	**39,8010**	**84,8700**	58,0810	50,9220	43,6410	**39,3940**	**6,1520**	6,5290	6,1060	5,2230	**2,7240**

5.6 CONCLUSÕES SOBRE A AUTENTICAÇÃO ADAPTATIVA COM BUSCA POR ROTAÇÃO

Na Figura 5.4, são apresentados os percentuais médios da FRR para uma única íris, para as bases de dados BIOSECURE, CASIA e NIST-ICE. Todos os experimentos realizados neste Capítulo foram feitos com a busca por rotação sempre ativada, por este motivo a comparação dos resultados obtidos, aqui, são feitos em relação aos resultados obtidos no Capítulo 3. Ainda se referindo à Figura 5.4, nela o t_{RS} varia desde 1 até 10, e para cada t_{RS} existem oito barras, duas para cada base de dados, respectivamente para BIOSECURE, CASIA, NIST-ICE(exp1) e NIST-ICE(exp2), sendo que cada par de barra, para cada base de dados, representa os percentuais da FRR obtidos para a proporção 3:2, para cada barra mais à esquerda de cada par, e a proporção 7:6, para cada barra mais à direita de cada par. Para qualquer uma das bases de dados utilizada, quando $20 \leq t_{RS} \leq 22$, usando uma única íris e para a proporção 7:6, todos os percentuais da FRR são iguais a zero. Vale lembrar que todos os percentuais da FAR são sempre iguais a zero, para todas as bases de dados usadas nos experimentos e para todos os $1 \leq t_{RS} \leq 22$.

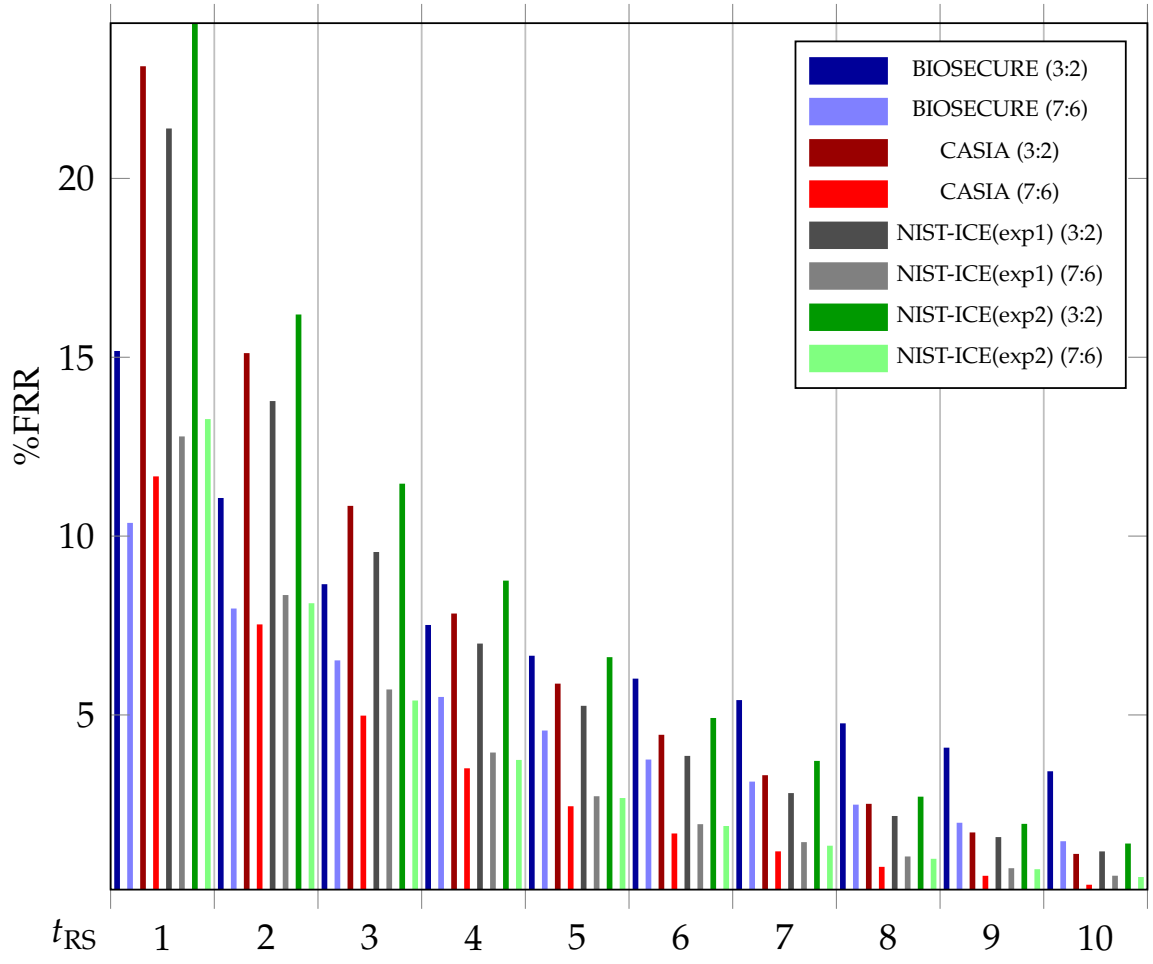


Figura 5.4: *Diagrama de Barras ilustrando o percentual médio da FRR das Tabelas 5.4, 5.5 e 5.6, para uma única íris.*

Na Figura 5.5 são apresentados os percentuais médios da FRR para duas íris, para as bases de dados BIOSECURE, CASIA e NIST-ICE. Ainda na Figura 5.5, o t_{RS} varia desde 1 até 10, e para cada t_{RS} existem seis barras, duas para cada base de dados, respectivamente para BIOSECURE, CASIA e NIST-ICE, sendo que cada par de barra, para cada base de dados, representa os percentuais da FRR obtidos para a proporção 3:2, para cada barra mais à esquerda de cada par, e a proporção 7:6, para cada barra mais à direita de cada par. Para qualquer uma das bases de dados utilizada, quando $7 \leq t_{RS} \leq 22$, usando duas íris e para a proporção 7:6, todos os percentuais da FRR são iguais a zero. Vale lembrar que todos os percentuais da FAR são sempre iguais a zero para todas as bases de dados usadas nos experimentos e para todos os $1 \leq t_{RS} \leq 22$.

Na Tabela 5.7 é apresentado o resumo do ganho obtido com o emprego da técnica de

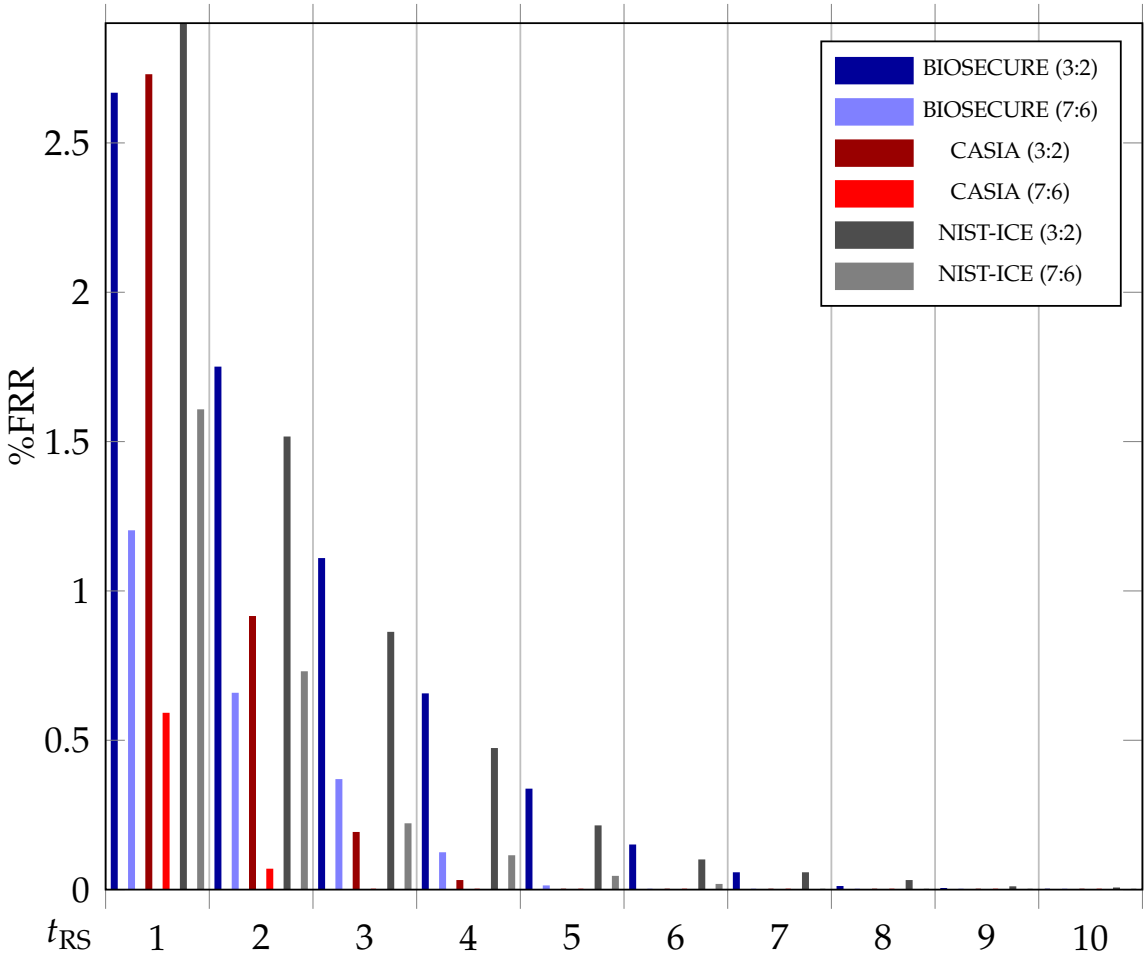


Figura 5.5: *Diagrama de Barras ilustrando o percentual médio da FRR das Tabelas 5.4, 5.5 e 5.6, para ambas as íris.*

remoção de redundância e alteração na proporção de *bits* de dados versus *bits* inseridos, para a proporção 7:6, que é a proporção na qual foram obtidos os melhores resultados. O ganho percentual é de aproximadamente 48,6% para os experimentos realizados com uma única íris e de aproximadamente 62,9% para os experimentos realizados com ambas as íris.

Tabela 5.7: *Quantidade de erros (em média) empregando a busca por rotação versus Remoção de Redundância e Proporção de* bits *(7:6).*

Proposta	Uma íris	Ambas as íris
Rotation	30.568	701
7:6	15.713	260
Ganho	48,6%	62,9%

Na Tabela 5.8 é possível verificar o desempenho do esquema proposto neste Capítulo em relação aos melhores resultados obtidos até o momento, usando as mesmas bases de dados. Também foi incluído o resultado obtido no Capítulo 3. Agora é possível recuperar chaves criptográficas com 234 *bits* com percentual da FRR de $0,2070\%$ e FAR de 0% para a base de dados NIST-ICE(exp1), usando uma única íris; e chaves criptográficas com 399 *bits* com percentual da FRR de $0,2190\%$ e FAR de 0% para a base de dados NIST-ICE, usando ambas as íris.

Tabela 5.8: *Resultado percentual da FRR para diversos algoritmos biométricos. O algoritmo proposto emprega a Remoção da Redundância e a alteração na proporção de* bits *inseridos.*

Esquema	ECC	Comprimento da chave $\|\|\mathbf{K}\|\|$ (em *bits*)	FRR (%)	FAR (%)	Base de Dados
Referência [46]	RS	282	8,4200	0	NIST-ICE (íris direita)
Referência [2]	RS	128/256	0,7600	0,1000	NIST-ICE (íris direita)
Referência [57]	RMP	42	0,4700	0	NIST-ICE (íris direita)
Referência [58]*	RSH	147	0,1800	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	231	0	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	287	0,3400	0	NIST-ICE (ambas as íris)
Busca por rotação	RSH	198	0,2400	0	NIST-ICE (íris direita)
Busca por rotação*	RSH	273	0	0	NIST-ICE (ambas as íris)
Busca por rotação*	RSH	371	0,4700	0	NIST-ICE (ambas as íris)
Proposto	**RSH**	**234**	**0,2070**	**0**	**NIST-ICE (íris direita)**
Proposto*	**RSH**	**294**	**0**	**0**	**NIST-ICE (ambas as íris)**
Proposto*	**RSH**	**399**	**0,2190**	**0**	**NIST-ICE (ambas as íris)**

Obs: (*) denota os sistemas multi biométricos; ECC: código corretor de erros; RSH: código Reed Solomon e Hadamard; RMP: códigos produto baseados nos códigos de Reed Muller.
O percentual FAR só é diferente de zero em [2].

CAPÍTULO 6

SISTEMA GENERALIZADO PARA AUTENTICAÇÃO BIOMÉTRICA

"Não encontre um defeito, encontre uma solução."

— Henry Ford

A proposta deste Capítulo é reunir as técnicas busca por rotação, voto da maioria, remoção da "redundância" e alteração na proporção de *bits*, em um único sistema, de modo a verificar o quanto é possível melhorar o percentual da FRR, em relação aos trabalhos publicados, até o momento, que utilizam as mesmas bases de dados. Também é feito um pequeno ajuste no k_{shuf} para verificar o impacto nos percentuais da taxa de falsa aceitação FAR.

6.1 TÉCNICAS UTILIZADAS

Basicamente serão integradas quatro alterações na proposta de [3], que motivou este trabalho de pesquisa, que são:

- A implementação da busca por rotação (Capítulo 3);

- A implementação do voto de maioria (Capítulo 4);
- A implementação da remoção da redundância (Capítulo 5);
- A implementação da alteração na proporção de *bits* inseridos (Capítulo 5).

6.1.1 BUSCA POR ROTAÇÃO

A busca por rotação é a técnica de usar todas as imagens de referência e todas as suas rotações (21 rotações de cada imagem), assim como usar todas as imagens de teste e todas as suas rotações (21 rotações de cada imagem), já disponibilizadas nas bases de dados utilizadas por [3], para fazer até: $21 \times 21 = 441$, comparações para cada teste de identificação de um usuário e consequente recuperação da chave criptográfica K em um determinado experimento.

Devido ao fato de que não estão disponíveis todos os percentuais para ambas as íris, foram realizados alguns experimentos para encontrar o ganho entre a busca por rotação e a busca padrão, os resultados podem ser vistos na Tabela 6.1. Nesta tabela é possível verificar que o ganho para uma íris é de 54,8% e para ambas as íris é de 77,3%. Estes dados indicam um ganho de aproximadamente 2,2 vezes para uma íris e de aproximadamente 4,4 vezes para ambas as íris.

Tabela 6.1: *Quantidade de erros (em média) empregando a busca padrão versus a busca por rotação.*

Proposta	Uma íris	Ambas as íris
Busca padrão	67.679	3093
Busca por rotação	30.568	701
Ganho	54,8%	77,3%

6.1.2 VOTO DE MAIORIA

A técnica do voto de maioria consiste em mesclar várias imagens de referência "auxiliares" de um determinado usuário, para compor uma imagem "média" que será considerada como sendo a imagem de referência e será usada em um teste, de modo que se uma imagem de referência possuir um ou mais erros, estes poderão vir a ser corrigidos pelas outras imagens de referência "auxiliares". Não é possível afirmar que haverá correção de erros em todos os casos, pois esta possibilidade só se tornará realidade se as imagens de referência "auxiliares", usadas para compor a imagem de referência que será usada no teste, não

possuírem erros nas mesmas posições que a imagem de referência.

Resumidamente, se a imagem de referência a ser comparada fosse hipoteticamente a imagem de número 1, é possível utilizar, por exemplo, as imagens de número 2, 3, 4 e 5, junto com a de número 1, para compor a imagem que será utilizada nos testes comparativos. Deste modo, cada *bit* de uma determinada posição do código íris, seria a composição média entre o *bit* desta posição em cada uma das imagens 1, 2, 3, 4 e 5. A conta que é realizada no *software* desenvolvido é dividir a soma dos *bits*, em cada posição, de todas as imagens utilizadas, pela metade inteira do montante de imagens de referência utilizadas mais um. No exemplo dado, com cinco imagens de referência, cada soma *bit* a *bit* de cada posição é dividida por 3, e o resultado é o inteiro resultante da divisão. Ainda sobre o exemplo citado, cada posição será igual a 1, se a soma for maior ou igual a 3, em caso contrário, será atribuído o valor 0 para o *bit* que não atender a esta condição, caracterizando, portanto, o voto da maioria.

A fórmula geral pode ser vista na equação 6.1. A variável *Length* é o número máximo de *bits* que possui a sequência do código íris truncado, após os *bits* inseridos e a redundância removida. Os valores possíveis para *Length* são: 1.116, 1.188, 2.232 ou 2.376.

$$bit_{ref}(y) = \left\lfloor \frac{\sum_{x=1}^{i} \left(bit_{ref}(x) \right)}{(i+1)/2} \right\rfloor \text{ para } \begin{cases} i = 1,3,5,7 \text{ ou } 9; \\ y = 1,2,3,\ldots,Length. \end{cases} \tag{6.1}$$

Conforme pode ser observado na Tabela 6.2, o ganho médio obtido com esta técnica, em comparação com a busca por rotação foi de cerca de 59,1% para uma única íris e de cerca de 60,9% para ambas as íris. Estes resultados representam um ganho de cerca de 2,4 vezes para uma íris e de cerca de 2,6 vezes para ambas as íris. Se for feita uma comparação com a busca padrão o ganho é de cerca de 81,5% para uma íris e de cerca de 91,1% para ambas as íris. Estes resultados representam um ganho de cerca de 5,4 vezes para uma íris e de cerca de 11,3 vezes para ambas as íris.

6.1.3 AUTENTICAÇÃO ADAPTATIVA COM BUSCA POR ROTAÇÃO

A técnica da autenticação adaptativa com busca por rotação foi explicada no Capítulo 5. Nesta subseção é apresentado um resumo desta técnica, assim como os resultados comparativos entre a aplicação desta, em relação à busca padrão e em relação à busca por rotação.

Resumidamente, a técnica de remoção de redundância avalia uma posição y em todos os códigos íris de um determinado usuário, de uma base de dados e decide se esta posição

Tabela 6.2: *Quantidade de erros (em média) empregando a técnica de busca padrão ou a técnica de busca por rotação comparadas com a técnica de voto majoritário (com 9 imagens).*

Proposta	Uma íris	Ambas as íris
Busca padrão	67.679	3093
Busca por rotação	30.568	701
Voto majoritário	12.504	274
Ganho (Voto de maioria × padrão)	81,5%	91,1%
Ganho (Voto de maioria × rotação)	59,1%	60,9%

pode ser removida sem que haja prejuízo para a identificação deste usuário. A posição só é considerada "redundante" quando pelo menos 90% dos *bits* da posição y são iguais, não afetando significativamente, a distância de Hamming entre as palavras do código íris deste usuário. Estas posições removidas permitem que uma maior quantidade de *bits* pseudo aleatórios possam ser inseridos no código íris, aumentando a capacidade de correção do código de Hadamard usado no sistema que realiza a codificação e a decodificação.

A técnica de alteração na proporção de *bits* pseudo aleatórios inseridos, reflete diretamente na capacidade de correção do código de Hadamard, pois quanto mais *bits* inseridos no código íris, melhor será a capacidade de correção. As proporções utilizadas nos experimentos são 3:2, 4:3, 5:4, 6:5 e 7:6. Os experimentos mostram que a proporção 7:6 é a que resulta em um melhor resultado, por este motivo na Tabela 6.3 são comparados os resultados desta proporção com a busca padrão e com a busca por rotação.

Tabela 6.3: *Quantidade de erros (em média) empregando a busca padrão ou a busca por rotação versus Remoção de Redundância e Proporção de* bits *(7:6).*

Proposta	Uma íris	Ambas as íris
Busca padrão	67.679	3093
Busca por rotação	30.568	701
7:6	15.713	260
Ganho (7:6 × padrão)	76,8%	91,6%
Ganho (7:6 × rotação)	48,6%	62,9%

Ainda sobre a Tabela 6.3, é possível observar que o ganho entre a técnica 7:6 e a busca padrão é de cerca de 76,8% para uma íris e de cerca de 91,6% para ambas as íris. Estes ganhos representam um resultado cerca de 4,3 vezes melhor para uma íris e cerca de 11,9 vezes melhor para ambas as íris. Quando comparados com a busca por rotação, este resultado é

de cerca de 48,6% para uma íris e de cerca de 62,9% para ambas as íris. Estes ganhos representam um resultado cerca de 1,9 vezes melhor para uma íris e cerca de 2,7 vezes melhor para ambas as íris.

6.2 MODELO GENERALIZADO

Na Figura 6.1 é possível observar o modelo generalizado proposto, no qual as técnicas busca por rotação, voto da maioria, remoção da redundância e alteração na proporção de *bits* inseridos são integradas, o que permite serem aplicadas a experimentos para encontrar o ganho em conjunto.

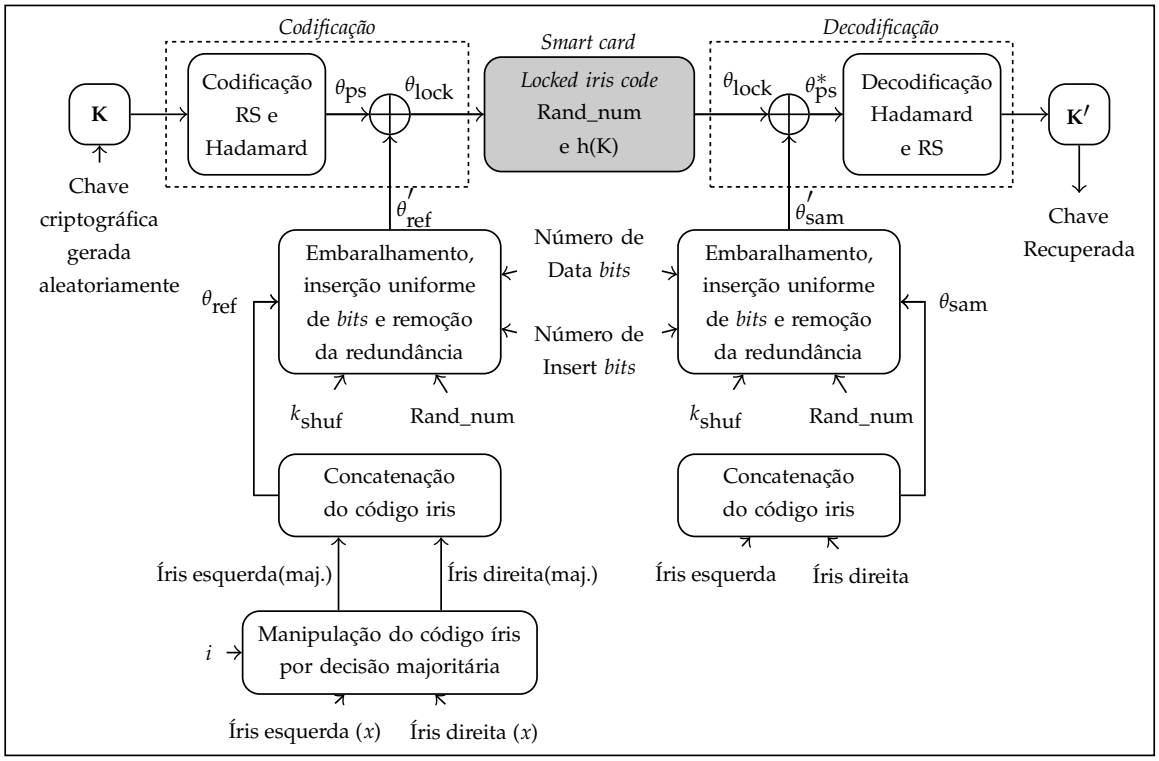


Figura 6.1: *Sistema de regeneração de chave uni ou multi biométrico com a busca por rotação, voto da maioria, remoção da redundância e alteração da proporção de* bits *inseridos, empregando smart card, íris e senha.*

6.3 RESULTADOS OBTIDOS COM O MODELO GENERALIZADO

Para realizar os experimentos com o modelo generalizado, foram escolhidas apenas as melhores configurações, aquelas nas quais os resultados obtidos por cada uma das quatro técnicas propostas foi o melhor possível. A busca por rotação foi empregada, para o voto

da maioria, foi escolhida a opção com 9 imagens para compor a imagem de referência a ser usada nos testes. A opção de remoção da redundância também foi ativada em todos os experimentos. A proporção de *bits* inseridos utilizada foi a 7:6.

Na Tabela 6.4 são apresentados os resultados dos percentuais da FRR para os experimentos realizados com uma única íris, para as bases de dados BIOSECURE, CASIA e NIST-ICE(exp1) e NIST-ICE(exp2), assim como os dados obtidos por Kanade [2]. É possível observar que para todas as bases de dados e para qualquer t_{RS}, o resultado obtido pelo modelo generalizado é melhor para todos os percentuais da FRR diferentes de zero.

Tabela 6.4: *Comparativo Kanade [2] versus modelo generalizado (M. G.) para uma única íris.*

t_{RS}	BIOSECURE		CASIA		ICE exp1		ICE exp2	
	Kanade [2]	M. G.	Kanade [2]	M. G.	Kanade [2]	M. G.	Kanade [2]	M. G.
1	30,5300	**8,7533**	49,7000	**7,6750**	49,3900	**3,9570**	52,9900	**4,7840**
2	22,1200	**7,1533**	35,7800	**4,4858**	33,2600	**2,3580**	37,7400	**2,8090**
3	16,3700	**6,2683**	26,2700	**2,7392**	24,2600	**1,5990**	25,7800	**1,8880**
4	12,8800	**5,5258**	19,2500	**1,8883**	16,5000	**1,1820**	20,1000	**1,2630**
5	10,6500	**4,6733**	14,8200	**1,4950**	12,6700	**0,8920**	16,2500	**0,8550**
6	8,9800	**3,7850**	11,7000	**1,1633**	10,3100	**0,6660**	11,8100	**0,6050**
7	8,3500	**2,9367**	9,5200	**0,7025**	7,2900	**0,5080**	9,4200	**0,4210**
8	7,2700	**2,1033**	7,3200	**0,2858**	5,9300	**0,3660**	7,7700	**0,2820**
9	6,6000	**1,4517**	5,9700	**0,0875**	4,6100	**0,2840**	6,2600	**0,2050**
10	5,8700	**0,9300**	4,8500	**0,0242**	3,6300	**0,1960**	4,5400	**0,1390**
11	5,2800	**0,5383**	3,7700	**0,0092**	2,4800	**0,1310**	3,4900	**0,0960**
12	4,5700	**0,2392**	3,1300	**0,0008**	2,1300	**0,0490**	3,0500	**0,0680**
13	3,9700	**0,0833**	2,1200	**0**	1,4600	**0,0250**	2,1200	**0,0390**
14	3,2500	**0,0142**	1,5700	**0**	1,0400	**0,0110**	1,4100	**0,0200**
15	2,6700	**0,0025**	1,0700	**0**	0,7600	**0**	1,0900	**0,0020**
16	2,0000	**0**	0,6300	**0**	0,6900	**0**	0,9400	**0**
17	1,4300	**0**	0,3000	**0**	0,4700	**0**	0,6100	**0**
18	1,0000	**0**	0,2500	**0**	0,3800	**0**	0,4600	**0**
19	0,6300	**0**	0,1500	**0**	0,2600	**0**	0,3900	**0**
20	0,4200	**0**	0,0500	**0**	0,1500	**0**	0,2900	**0**
21	0,2300	**0**	0,0300	**0**	0,1300	**0**	0,2000	**0**
22	0,1300	**0**	0	**0**	0,1100	**0**	0,1300	**0**

Na Tabela 6.5 são apresentados os resultados dos percentuais da FRR para os experimentos realizados com ambas as íris, para as bases de dados BIOSECURE, CASIA e NIST-ICE,

assim como os dados obtidos por Câmara [3]. Os dados obtidos por [3] só possuem %FRR para $10 \leq t_{RS} \leq 14$, por este motivo foram feitos experimentos simulando a proposta de Câmara [3] para obter todos os percentuais aproximados. O "N/D" significa que os dados não estão disponíveis.

Tabela 6.5: *Comparativo Câmara [3], [3] simulado e modelo generalizado (M. G.) para ambas as íris.*

t_{RS}	BIOSECURE			CASIA			ICE		
	Câmara [3]	[3] Simulado	M. G.	Câmara [3]	[3] Simulado	M. G.	Câmara [3]	[3] Simulado	M. G.
1	N/D	11,8670	**1,3817**	N/D	23,6440	**0,0500**	N/D	16,8620	**0,1500**
2	N/D	7,7560	**0,4833**	N/D	14,9890	**0**	N/D	9,4180	**0,0750**
3	N/D	5,1110	**0,0783**	N/D	9,9000	**0**	N/D	6,2880	**0,0320**
4	N/D	3,7890	**0**	N/D	5,7670	**0**	N/D	3,8580	**0,0160**
5	N/D	2,8780	**0**	N/D	3,1780	**0**	N/D	2,4620	**0**
6	N/D	2,1890	**0**	N/D	1,2440	**0**	N/D	1,6050	**0**
7	N/D	1,6220	**0**	N/D	0,4560	**0**	N/D	0,7490	**0**
8	N/D	1,0560	**0**	N/D	0,1000	**0**	N/D	0,4230	**0**
9	N/D	0,6560	**0**	N/D	0,0220	**0**	N/D	0,1610	**0**
10	1,0300	0,2890	**0**	0,6700	0,0110	**0**	0,3400	0,1180	**0**
11	0,6000	0,1440	**0**	0,2300	0	**0**	0,1600	0,1120	**0**
12	0,1700	0,0440	**0**	0,1300	0	**0**	0,1100	0,0960	**0**
13	0,1300	0,0220	**0**	0,1000	0	**0**	0,0500	0,0640	**0**
14	0,1000	0	**0**	0,0700	0	**0**	0	0,0320	**0**
15	N/D	0	**0**	N/D	0	**0**	N/D	0,0050	**0**
16	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
17	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
18	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
19	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
20	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
21	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**
22	N/D	0	**0**	N/D	0	**0**	N/D	0	**0**

Vale explicar a diferença obtida pelos experimentos realizados para simular a proposta de Câmara [3], pois os resultado obtidos nestas simulações são um pouco melhores. Existe uma pequena diferença entre a simulação realizada e a proposta de Câmara [3], que consiste no método de inserção de *bits* adotado. A proposta de Câmara [3] prevê a inserção de *bits* não uniformes, enquanto que a proposta desta tese segue a ideia inicial sugerida por Kanade [2], na qual a inserção de *bits* é feita de modo uniforme. Existe um ganho entre a proposta de Kanade [2] e a proposta de Câmara [3], por isto a inserção de *bits* uniformes foi escolhida.

A proposta de Câmara [3], que é para ambas as íris, insere *bits* não uniformes, que, resumidamente, é feita da seguinte forma: é gerada uma sequência aleatória de comprimento

igual a 1.528 *bits*, que é o "Rand_num"; então, dois *bits* desta sequência são inseridos a cada três *bits* do código íris de referência, até os primeiros 2.208 *bits*, e um *bit* desta sequência é inserido a cada três *bits*, dos 168 *bits* restantes, resultando em uma sequência modificada do código íris de 3.904 *bits*; o comprimento do código íris sem modificações, para ambas as íris é 2.376 *bits*; $(2.208/3) \times 5 + (168/3) \times 4 = 3.904$.

A Tabela 6.6 contém as quantidades de erros totais para as propostas de Kanade [2] para uma única íris e de Câmara [3] para ambas as íris, comparadas com os erros totais da proposta desta tese com o modelo generalizado. É possível observar que o ganho percentual para uma íris é de 90% ou em valores absolutos, cerca de 9,9 vezes. Para ambas as íris, o ganho percentual é de 98,7% ou em valores absolutos, cerca de 80,3 vezes.

Tabela 6.6: *Quantidade de erros totais obtidos por Kanade [2] e por Câmara [3] versus modelo generalizado (M. G.). (*) significa que não estão disponíveis todos os percentuais da FRR, não permitindo calcular precisamente o ganho.*

Proposta	Uma íris	Ambas as íris
Kanade [2]	73.245	-
Câmara [3]	-	116*
Câmara [3] (simulado)	-	5.533
M. G.	7368	68
Ganho (M. G. $\times$ [2])	90%	-
Ganho (M. G. $\times$ [3] simulado)	-	98,7%

6.4 CONCLUSÕES SOBRE O MODELO GENERALIZADO

Na Figura 6.2, é apresentado o gráfico de barras com os percentuais médios da Tabela 6.4, para todas as bases de dados. Para cada $1 \leq t_{RS} \leq 22$, a barra da esquerda representa os percentuais obtidos por Kanade [2] e a barra da direita representa os percentuais obtidos pela proposta que emprega o uso do modelo generalizado no mesmo experimento.

Na Figura 6.3, é apresentado o gráfico de barras com os percentuais médios da Tabela 6.4, para todas as bases de dados. Para cada $1 \leq t_{RS} \leq 22$, a barra da esquerda representa os percentuais obtidos por Câmara [2] e a barra da direita representa os percentuais obtidos pela proposta que emprega o uso do modelo generalizado no mesmo experimento.

Na Tabela 6.7 é apresentado o resumo da quantidade total de erros obtidos por Kanade [2], da quantidade total de erros obtidos pela proposta de Câmara [3] e da quantidade

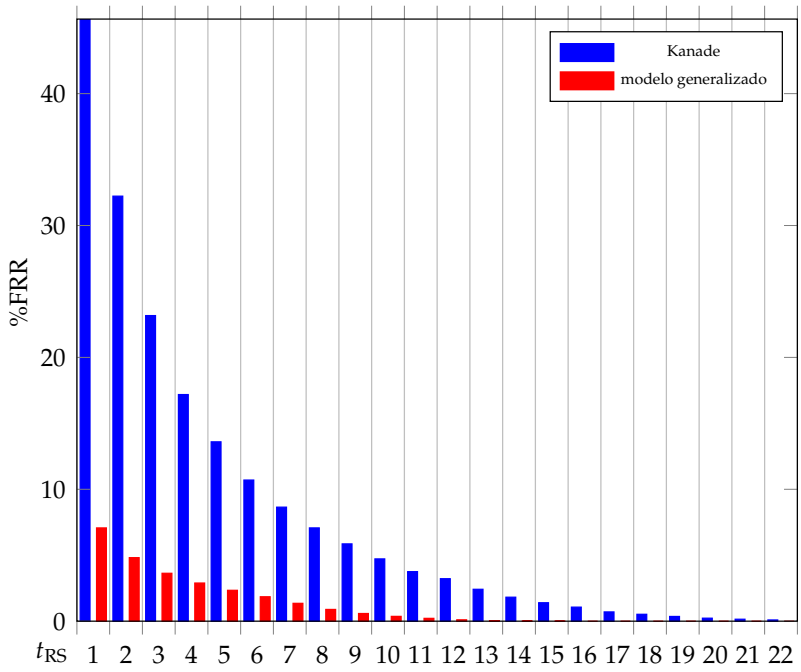


Figura 6.2: *Diagrama de Barras ilustrando o percentual médio para uma íris, da FRR da Tabela 6.2 versus t_{RS}, para Kanade [2] e o modelo generalizado.*

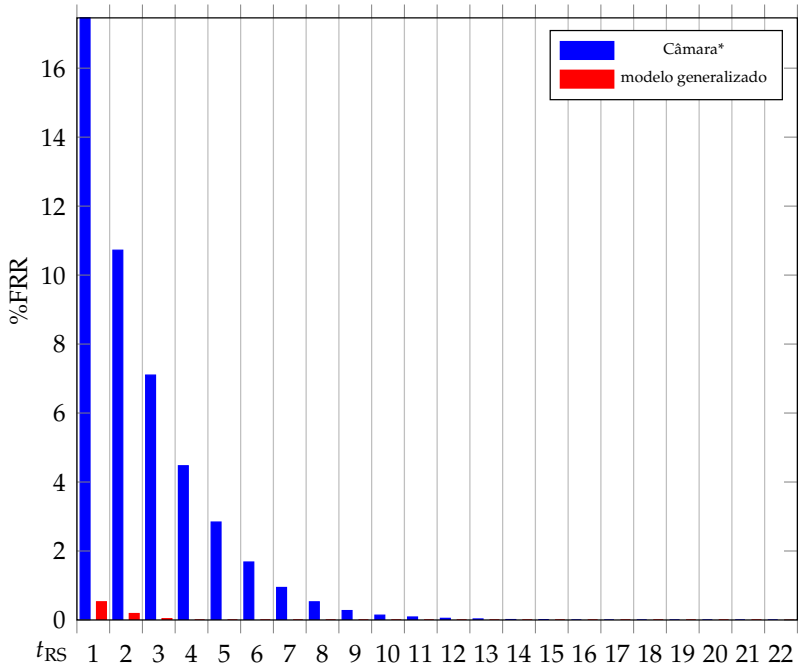


Figura 6.3: *Diagrama de Barras ilustrando o percentual médio para ambas as íris, da FRR da Tabela 6.3 versus t_{RS}, para Câmara [3] e o modelo generalizado. (*) Resultado obtido com experimentos próprios*

total de erros obtidos pela proposta do modelo generalizado, todas para as bases de dados BIOSECURE, CASIA e NIST-ICE.

Tabela 6.7: *Resumo da quantidade de erros totais obtidos por Kanade [2] e por Câmara [3] versus modelo generalizado (M. G.). (*) Resultado obtido com experimentos próprios.*

Proposta	Uma íris	Ambas as íris
Kanade [2] ou Câmara [3]*	73.245	5.533*
M. G.	7368	68
Ganho (M. G. $\times$ [2] ou M. G. $\times$ [3]*)	90%	98,7%

Ainda de acordo com a Tabela 6.7, para uma única íris, o ganho é de 90% entre a proposta do modelo generalizado e a proposta de Kanade [2]. Para ambas as íris, o ganho é de 98,7% entre a proposta do modelo generalizado e a proposta de Câmara [3].

Com o modelo generalizado proposto, é possível recuperar chaves criptográficas de 246 *bits* com percentual da FRR de $0,1960\%$ e FAR de 0% para a base de dados NIST-ICE(exp1), usando uma única íris; e chaves criptográficas com 427 *bits* com percentual da FRR de $0,1500\%$ e FAR de 0% para a base de dados NIST-ICE, usando ambas as íris. Estes dados podem ser observados na Tabela 6.8.

Tabela 6.8: *Resultado percentual FRR para diversos algoritmos biométricos. O algoritmo proposto emprega o modelo generalizado.*

Esquema	**ECC**	**Comprimento da chave $\|\|\mathbf{K}\|\|$ (em *bits*)**	**FRR (%)**	**FAR (%)**	**Base de Dados**
Referência [46]	RS	282	8,4200	0	NIST-ICE (íris direita)
Referência [2]	RS	128/256	0,7600	0,1000	NIST-ICE (íris direita)
Referência [57]	RMP	42	0,4700	0	NIST-ICE (íris direita)
Referência [58]*	RSH	147	0,1800	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	231	0	0	NIST-ICE (ambas as íris)
Referência [3]*	RSH	287	0,3400	0	NIST-ICE (ambas as íris)
Busca por rotação	RSH	198	0,2400	0	NIST-ICE (íris direita)
Busca por rotação*	RSH	273	0	0	NIST-ICE (ambas as íris)
Busca por rotação*	RSH	371	0,4700	0	NIST-ICE (ambas as íris)
modelo generalizado	**RSH**	**246**	**0,1960**	**0**	**NIST-ICE (íris direita)**
modelo generalizado*	**RSH**	**371**	**0**	**0**	**NIST-ICE (ambas as íris)**
modelo generalizado*	**RSH**	**427**	**0,1500**	**0**	**NIST-ICE (ambas as íris)**
Obs: (*) denota os sistemas multi biométricos; ECC: código corretor de erros; RSH: código Reed Solomon e Hadamard; RMP: códigos produto baseados nos códigos de Reed Muller. O percentual FAR só é diferente de zero em [2].					

CAPÍTULO 7

CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

"Não existem grandes talentos sem grandes vontades."

— Honoré de Balzac

NESTE Capítulo são feitas as considerações finais sobre toda a pesquisa realizada e são apresentadas sugestões para futuros trabalhos de pesquisa.

7.1 CONSIDERAÇÕES FINAIS

Nesta tese são propostos quatro sistemas para regeneração de chaves criptográficas. O primeiro é o de busca por rotação, que consiste, resumidamente, em aumentar o número de testes realizados para identificação de um usuário, dos 21 realizados nos artigos publicados até o início desta pesquisa, para 441. Este acréscimo na quantidade de testes realizados proporciona uma melhora de pelo menos 54,8%, quando apenas uma íris é empregada nos experimentos e, de pelo menos 77,3%, quando ambas as íris são empregadas nos experimentos. Esta técnica, para uma única íris, foi publicada em [62], e para ambas as íris, foi publicada em [61].

Seguindo a busca por rotação, foi proposta a segunda técnica, chamada de voto de maioria, ou voto majoritário, que consiste em verificar *bit* a *bit* a sequência binária que representa o código íris, comparando a posição de um determinado *bit* com a mesma posição de outras sequências que também representam o código íris, sempre para o mesmo usuário e sempre apenas para as imagens de referência. Esta técnica não foi aplicada às imagens de teste, pois em um caso de identificação real de um usuário, pode não ser possível obter mais que uma imagem do mesmo. A quantidade de sequências i comparadas foi escolhida como sendo um número ímpar entre 1 e 10, de modo que sempre a maioria pudesse definir qual o *bit* mais adequado para a posição verificada. Deste modo, é possível diminuir a distância de Hamming entre o código íris usado como referência e o código íris usado como teste. Esta redução permitiu que a quantidade de erros nos códigos íris de referência, usados em um teste, fosse reduzida, proporcionando uma redução na quantidade de erros de identificação encontrado nos experimentos realizados. Este ganho é significativo, mesmo se for levado em consideração o ganho que a técnica de busca por rotação conseguiu obter. O ganho obtido com o voto de maioria em relação à busca por rotação foi de cerca de 59% para uma única íris e de cerca de 61% para ambas as íris. Esta técnica pode ser aplicada com ou sem a busca por rotação. Foi dada preferência à realização de experimentos em conjunto, devido ao fato de que, deste modo, é possível avaliar a melhoria alcançada além do que já havia sido obtido com a busca por rotação isoladamente. Foi aceito para publicação um artigo descrevendo esta técnica, no Simpósio Brasileiro de Telecomunicações 2016 [64]. Após a introdução destas duas técnicas, foram propostas mais duas novas técnicas, que são aplicadas em conjunto para a obtenção de um melhor resultado, que são a remoção de redundância e a alteração na proporção de *bits* inseridos. A remoção da redundância consiste em verificar todas as imagens de referência do código íris *bit* a *bit*, para tentar encontrar quais posições possuem pelo menos 90% dos *bits* iguais. Deste modo, esta posição pode ser considerada uma posição com informação redundante, pois ao ser removida, não altera significativamente a distância de Hamming entre quaisquer duas imagens de referência. A remoção deste *bit* altera a quantidade de *bits* disponível no código de íris, sendo necessário haver a reposição do mesmo por uma informação, que contribua para um aumento na taxa de identificação do usuário. A reposição acontece com a alteração na proporção de *bits* inseridos, gerando uma nova sequência, que possui um comprimento normalmente maior que o comprimento do código íris. Como o comprimento da sequência com *bits* inseridos é maior

do que o comprimento da palavra do código de Hadamard usado no processo de codificação/decodificação, é necessário realizar um truncamento no final desta sequência. Quanto maior for a quantidade de *bits* truncados, maior será a quantidade de *bits* do código de íris perdidos. Deste modo, as duas técnicas se complementam, pois enquanto uma remove redundância, a outra acrescenta *bits* de uma sequência pseudo-aleatória conhecida, que irão contribuir na capacidade de correção de erros do código de Hadamard. A ideia de inserção de *bits*, originalmente, foi proposta por Kanade [2], porém a proporção sugerida foi de 3:2, ou seja, três *bits* de dados do código íris para dois *bits* inseridos. Na nova proposta, foi aumentada a proporção para 7:6, ou seja, sete *bits* de dados do código íris para 6 *bits* inseridos. Quanto à remoção da redundância, foi possível encontrar 72 posições redundantes, que foram removidas para a realização dos experimentos. Estas duas técnicas podem ser usadas em conjunto, com ou sem a busca por rotação. Nos experimentos realizados, a busca por rotação foi sempre empregada, deste modo foi obtido o ganho entre estas duas técnicas em relação à busca por rotação isolada. Para uma íris, o ganho obtido foi de cerca de 48,6% e para duas íris o ganho foi de cerca de 62,9%.

Na sequência, foram aplicadas as quatro técnicas propostas em um mesmo experimento. Os resultados foram, então, comparados com os resultados das publicações mais recentes encontradas. Para os experimentos com apenas uma íris, o ganho obtido foi de cerca de 90% e, para ambas as íris, o ganho obtido foi de cerca de 98,7%. Com o sistema generalizado proposto, é possível recuperar chaves criptográficas de 246 *bits* com percentual da FRR de $0,196\%$ e FAR de 0% para a base de dados NIST-ICE(exp1), usando uma única íris; e chaves criptográficas com 427 *bits* com percentual da FRR de $0,150\%$ e FAR de 0% para a base de dados NIST-ICE, usando ambas as íris. Estes dados são os melhores obtidos até o momento.

Para cada nova técnica sugerida nesta tese foi proposto um diagrama para o sistema de regeneração de chaves criptográficas. Todas as operações necessárias para o correto funcionamento dos sistemas foram explicadas com detalhes, de modo a tornar possível tanto para nós, quanto para novos pesquisadores continuarem desenvolvendo este trabalho de pesquisa. De modo resumido, é possível ajustar os parâmetros fornecidos ao sistema generalizado, para que o mesmo realize experimentos com qualquer uma das técnicas apresentadas nesta tese.

O sistema de identificação biométrica generalizado, proposto nesta tese, proporciona uma FRR, para uma única correção de erros do código de Reed Solomon ($t_{RS} = 1$), em torno

de 8,7% para a base de dados BIOSECURE; cerca de 7,7% para a base de dados CASIA; cerca de 4% para a base de dados NIST-ICE(exp1); e cerca de 4,8% para a base de dados NIST-ICE(exp2), quando é empregado o uso de uma única íris. Quando ambas as íris são usadas, os percentuais para o FRR são cerca de 1,4% para a base de dados BIOSECURE; cerca de 0,05% para a base de dados CASIA; e cerca de 0,15% para a base de dados NIST-ICE.

7.2 TRABALHOS FUTUROS

A seguir, são indicadas algumas sugestões de trabalhos futuros:

- Verificar o ganho obtido com a implementação do sistema generalizado para autenticação biométrica em outras bases de dados;
- Investigar o possível ganho com alteração da quantidade de *bits* usados na rotina de embaralhamento, pois seria possível dividir a sequência em partes menores, talvez resultando em melhoria nos resultados obtidos;
- Pesquisar uma alternativa para o processo de codificação usado atualmente. Em outras palavras, realizar experimentos com codificações diferentes da combinação Hadamard-Reed Solomon, pois os resultados obtidos para ambas as íris para a FRR são muito próximos de zero, para a correção de apenas um símbolo com o código RS, o que pode indicar que uma codificação mais simples, possa atingir bons resultados;
- Implementar um procedimento de baixo custo para captura de imagens e utilização do OSIRIS [63] para conversão das imagens em uma base de dados própria.
- Analisar os tipos de erros na geração do código de íris para identificação de códigos mais adequados à aplicação.

REFERÊNCIAS

[1] NIST, "Iris challenge evaluation," *Disponível em http://iris.nist.gov/ice*, 2005, acessado em 19 de fevereiro de 2016.

[2] S. Kanade, D. Camara, D. Petrovska-Delacrtaz, and B. Dorizzi, "Application of biometrics to obtain high entropy cryptographic keys," *World Acad. Sci. Eng. Tech*, vol. 52, p. 330, 2009.

[3] D. Camara, J. S. de Lemos-Neto, and V. C. da Rocha Jr., "Multi-instance based cryptographic key regeneration system," *Journal of Communication and Information Systems*, vol. 29, no. 1, pp. 46–55, 2014.

[4] A. Jain, P. Flynn, and A. A. Ross, *Handbook of biometrics*. Springer Science & Business Media, 2007.

[5] A. Jain, D. Maltoni, D. Maio, and J. Wayman, "Biometric systems technology, design and performance evaluation," 2005.

[6] J. Ashbourn, "Practical biometrics," *From Aspiration to Implementation, Berlin et al: Springer*, 2004.

[7] D. Petrovska-Delacrétaz, G. Chollet, and B. Dorizzi, *Guide to biometric reference systems and performance evaluation*. Springer, 2009.

[8] BIOSECURE, "Biosecure network of excellence," *Disponível em http://www.biosecure.info*, 2007, acessado em 19 de fevereiro de 2016.

[9] S. Barra, A. Casanova, F. Narducci, and S. Ricciardi, "Ubiquitous iris recognition by means of mobile devices," *Pattern Recognition Letters*, vol. 57, pp. 66–73, 2015.

[10] A. Mallikarjuna and S. Madhuri, "Biometric security techniques for iris recognition system," *IJRCCT*, vol. 2, no. 8, pp. 589–593, 2013.

[11] M. F. Zafar, Z. Zaheer, and J. Khurshid, "Novel iris segmentation and recognition system for human identification," in *Applied Sciences and Technology (IBCAST), 2013 10th International Bhurban Conference on*. IEEE, 2013, pp. 128–131.

[12] H. Betaouaf and A. Bessaid, "A biometric identification algorithm based on retinal blood vessels segmentation using watershed transformation," in *Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on*. IEEE, 2013, pp. 256–261.

[13] K. I. Hasan and M. A. Amin, "Dual iris matching for biometric identification," *Signal, Image and Video Processing*, vol. 8, no. 8, pp. 1605–1611, 2014.

[14] J. Bringer, H. Chabanne, and A. Patey, "Practical identification with encrypted biometric data using oblivious ram," in *Biometrics (ICB), 2013 International Conference on*. IEEE, 2013, pp. 1–8.

[15] A. Darwish and M. Pasquier, "Biometric identification using the dynamic features of the eyes," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 1–6.

[16] I. Rigas and O. V. Komogortsev, "Biometric recognition via probabilistic spatial projection of eye movement trajectories in dynamic visual environments," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 10, pp. 1743–1754, 2014.

[17] F. Alonso-Fernandez and J. Bigun, "Eye detection by complex filtering for periocular recognition," in *Biometrics and Forensics (IWBF), 2014 International Workshop on*. IEEE, 2014, pp. 1–6.

[18] C. Narmatha and S. Manimurugan, "A new approach for iris image identification using modified contour segmentation," in *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*. IEEE, 2014, pp. 1–7.

[19] M. Abo-Zahhad, S. M. Ahmed, and S. N. Abbas, "A novel biometric approach for human identification and verification using eye blinking signal," *Signal Processing Letters, IEEE*, vol. 22, no. 7, pp. 876–880, 2015.

[20] H. Rai and A. Yadav, "Iris recognition using combined support vector machine and Hamming distance approach," *Expert Systems with Applications*, vol. 41, no. 2, pp. 588–593, 2014.

[21] V. G. Garagad and N. C. Iyer, "A novel technique of iris identification for biometric systems," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 973–978.

[22] D. Camara, "Soluções de segurança através do uso combinado de técnicas biométricas e criptográficas." 2009.

[23] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

[24] W. Mao, *Modern cryptography: theory and practice*. Prentice Hall Professional Technical Reference, 2003.

[25] N. Ferguson and B. Schneier, *Practical cryptography*. Wiley New York, 2003, vol. 23.

[26] D. E. Robling Denning, *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc., 1982.

[27] S. A. Vanstone and P. C. van Oorschot, *Handbook of applied cryptography*. CRC press, 1997.

[28] R. Keon, "Rsa laboratories frequently asked questions about today's cryptography, version 4.1," *Published online: http://www. rsa. com/rsalabs/faq/February*, 2001.

[29] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 29, no. 4, pp. 561–572, 2007.

[30] T. E. Boult, W. J. Scheirer, and R. Woodworth, "Revocable fingerprint biotokens: Accuracy and security analysis," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[31] A. Lumini and L. Nanni, "An improved biohashing for human authentication," *Pattern recognition*, vol. 40, no. 3, pp. 1057–1065, 2007.

[32] M. Savvides, B. V. Kumar, and P. K. Khosla, "Cancelable biometric filters for face recognition," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3. IEEE, 2004, pp. 922–925.

[33] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM journal on computing*, vol. 38, no. 1, pp. 97–139, 2008.

[34] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," *International Journal of Information Security*, vol. 1, no. 2, pp. 69–83, 2002.

[35] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel, "Cryptographic key generation from voice," in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 2001, pp. 202–213.

[36] A. Goh and D. C. Ngo, "Computation of cryptographic keys from face biometrics," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2003, pp. 1–13.

[37] A. B. Teoh, D. C. Ngo, and A. Goh, "Personalised cryptographic key generation based on facehashing," *Computers & Security*, vol. 23, no. 7, pp. 606–614, 2004.

[38] Y.-J. Chang, W. Zhang, and T. Chen, "Biometrics-based cryptographic key generation," in *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2203–2206.

[39] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE transactions on computers*, vol. 55, no. 9, pp. 1081–1088, 2006.

[40] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM conference on Computer and communications security*. ACM, 1999, pp. 28–36.

[41] G. I. Davida, Y. Frankel, B. Matt, and R. Peralta, "On the relation of error correction and cryptography to an o ine biometric based identification scheme," in *Proceedin s of the Workshop on Codes and Cryptography 1999*. Citeseer, 1998.

[42] T. C. Clancy, N. Kiyavash, and D. J. Lin, "Secure smartcardbased fingerprint authentication," in *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*. ACM, 2003, pp. 45–52.

[43] U. Uludag and A. Jain, "Securing fingerprint template: Fuzzy vault with helper data," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. IEEE, 2006, pp. 163–163.

[44] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[45] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zémor, "Optimal iris fuzzy sketches," in *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*. IEEE, 2007, pp. 1–6.

[46] S. Kanade, D. Camara, E. Krichen, D. Petrovska-Delacrétaz, and B. Dorizzi, "Three factor scheme for biometric-based cryptographic key regeneration using iris," in *Biometrics Symposium, 2008. BSYM'08*. IEEE, 2008, pp. 59–64.

[47] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, and B. V. Kumar, "Biometric encryption using image processing," in *Photonics West'98 Electronic Imaging*. International Society for Optics and Photonics, 1998, pp. 178–188.

[48] I. Tomeo-Reyes, J. Liu-Jimenez, I. Rubio-Polo, J. Redondo-Justo, and R. Sanchez-Reillo, "Input images in iris recognition systems: A case study," in *Systems Conference (SysCon), 2011 IEEE International*. IEEE, 2011, pp. 501–505.

[49] J. Daugman, "New methods in iris recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 5, pp. 1167–1175, 2007.

[50] ——, "The importance of being random: statistical principles of iris recognition," *Pattern recognition*, vol. 36, no. 2, pp. 279–291, 2003.

[51] S. M. Kay, "Fundamentals of statistical signal processing, vol. ii: Detection theory," *Signal Processing. Upper Saddle River, NJ: Prentice Hall*, 1998.

[52] W. B. Davenport, W. L. Root *et al.*, *An introduction to the theory of random signals and noise*. McGraw-Hill New York, 1958, vol. 159.

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

[54] S. Lin and D. J. Costello, *Error control coding: fundamentals and applications*. Pearson Education, 2004.

[55] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge University Press, 2010.

[56] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. Elsevier, 1988, vol. 16.

[57] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zémor, "Theoretical and practical boundaries of binary secure sketches," *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 4, pp. 673–683, 2008.

[58] S. Kanade, D. Petrovska-Delacrétaz, and B. Dorizzi, "Multi-biometrics based cryptographic key regeneration scheme," in *Biometrics: Theory, Applications, and Systems, 2009. BTAS'09. IEEE 3rd International Conference on*. IEEE, 2009, pp. 1–7.

[59] J. Bringer, C. Morel, and C. Rathgeb, "Security analysis of bloom filter-based iris biometric template protection," in *Biometrics (ICB), 2015 International Conference on*. IEEE, 2015, pp. 527–534.

[60] K. Nandakumar and A. K. Jain, "Biometric template protection: Bridging the performance gap between theory and practice," *Signal Processing Magazine, IEEE*, vol. 32, no. 5, pp. 88–100, 2015.

[61] G. N. Melo, V. C. da Rocha Jr., and J. S. de Lemos-Neto, "User identification and key regeneration system employing rotated reference images of the iris," *Journal of Communication and Information Systems*, vol. 31, no. 1, pp. 60–68, 2016.

[62] ——, "User identification and key regeneration system employing rotated reference images of the iris," *Simpósio Brasileiro de Telecomunicações*, pp. 1–5, 2015.

[63] BIOSECURE, "Open source for iris - biosecure project," *disponível em: http://biosecure.it-sudparis.eu/AB/*, 2007, acessado em 20 de novembro de 2015.

[64] G. N. Melo, V. C. da Rocha Jr., and J. S. de Lemos-Neto, "Autenticação do código de iris usando busca por rotação e voto de maioria," *Simpósio Brasileiro de Telecomunicações*, pp. 1–5, 2016.

[65] Mathworks, "Explore new ideas," *http://www.mathworks.com/products/matlab/, acessado em maio de 2015*, 2015.

[66] B. Stroustrup, *The design and evolution of C++*. Pearson Education India, 1994.

APÊNDICE A

BASES DE DADOS E CÓDIGOS CORRETORES DE ERROS

"Quando não podemos mais sonhar, morremos."

— Emma Goldman

O MATLAB® é um *software* de simulação de codificação, realização de operações matemáticas complexas e de muitas aplicações na área da engenharia, porém, para situações particulares, é possível encontrar soluções dedicadas mais adequadas. Neste apêndice será feita uma comparação entre uma versão do *software* desenvolvido em MATLAB® e uma versão do *software* desenvolvido em linguagem de programação C++, que realizam a codificação do código íris.

A.1 MATLAB® VERSUS C++

Segundo a fabricante [65], O MATLAB® é um *software* de linguagem de programação em alto nível e de ambiente interativo usado por milhões de engenheiros e cientistas por todo o mundo. Ele permite que você explore e visualize ideias e colabore em áreas que

incluem processamento de sinais e imagens, comunicações, sistemas de controle e controle financeiro, entre outras.

Nesta tese, foi escolhido trabalhar com a linguagem de programação C++, foi pensado em trabalhar com a linguagem *assembly*, mas esta ideia foi abandonada devido ao fato de que a quantidade de manipulações de dados, usando operações com vetores e matrizes seria grande, provavelmente dificultando muito o desenvolvimento. De acordo com [66], a linguagem de programação C ou C++ tem um tempo de resposta relativamente próximos entre si, e devido ao fato de se dispor de compiladores mais modernos que trabalham com a linguagem de programação C++, foi resolvido optar por esta linguagem de programação. Além deste fato, outros dois motivos reforçaram o opção em trabalhar com a linguagem de programação C++: o de ser possível personalizar totalmente o *software* de acordo com as necessidades da pesquisa; e devido ao domínio do autor na linguagem de programação C++.

De acordo com [66] a linguagem de programação C++ foi desenvolvida com a intenção de facilitar a criação de programas, permitindo que se obtivesse o mesmo desempenho da linguagem de programação C e a flexibilidade necessária exigida pelos programadores. A opção pela linguagem de programação C++ foi estabelecida e se iniciou o desenvolvimento do *software*. Mas não seria possível nem sequer começar o desenvolvimento, devido ao fato que não se tinha acesso direto aos dados armazenados nas tabelas compactadas no formato do MATLAB®, das bases de dados usadas nos experimentos, é preciso convertê-las.

A.2 CONVERSÃO DOS DADOS

As bases de dados BIOSECURE, CASIA e NIST-ICE foram as bases utilizadas para realizar os experimentos e, como todas estão disponíveis em arquivos compactados de tabelas do MATLAB®, é necessário converter estes arquivos para um formato que permita rápido acesso pela linguagem de programação C++. O formato que foi escolhido foi o de arquivo texto sem formatação, pois, além de ser facilmente acessado pelo C++, também permite sua visualização pela maioria dos editores de texto disponíveis nos diversos sistemas operacionais, tanto proprietários, quanto de código livre. A conversão, então, passou a ser um dos focos da pesquisa.

Quando os arquivos das bases de dados foram observados dentro do MATLAB®, pôde-se verificar que são organizados em 21 tabelas contendo cada uma 1.200 linhas por 1.188

colunas para as bases BIOSECURE ou CASIA, e 21 tabelas contendo cada uma 2.953 linhas por 1.188 colunas para a base NIST-ICE. A ideia foi tentar copiar cada tabela, de cada base de dados, para uma planilha eletrônica e depois convertê-la para o formato de texto sem formatação. Isto foi feito, copiando-se cada planilha do MATLAB® para o Microsoft Excel, depois concatenando todas as células de cada código íris (de cada linha) da planilha e, depois estas células foram copiadas da planilha e armazenada em arquivos texto usando-se o bloco de notas.

No total foram gravados 21 arquivos para cada base de dados, onde o arquivo com sufixo 11 significa que código íris corresponde à imagem centralizada de cada usuário. O sufixo 01 contém os códigos íris das imagens rotacionadas mais à esquerda e o sufixo 21 contém os códigos íris das imagens rotacionadas mais à direita. Além destes 63 arquivos, são disponibilizados três arquivos de controle para a base de dados NIST-ICE que também foram convertidos para arquivos texto. Estes três arquivos definem quais são os usuários que serão usados nos testes e quais as imagens que pertencem a cada usuário.

A.3 DETALHES DO SOFTWARE C++

A quantidade de usuários de referência é definida de acordo com a base de dados e de acordo com a quantidade de íris utilizada: ou uma íris (um olho) ou duas íris (dois olhos) de cada vez. Na Tabela A.1 estão a quantidade de usuários e a quantidade de imagens de referência e imagens de teste usadas nos experimentos. A quantidade de imagens por usuário na base de teste NIST-ICE é variável, enquanto que a quantidade de imagens por usuário nas bases de teste BIOSECURE e CASIA é fixa. A letra "(r)" na Tabela A.1 significa referência e a letra "(t)" significa teste.

Tabela A.1: *Quantidade de testes realizados em cada base de dados.*

	BIOSECURE			CASIA			NIST-ICE			
							exp_1		exp_2	
	Usuários	Imagens		Usuários	Imagens		Usuários	Imagens	Usuários	Imagens
Uma íris	60	10(r)	10(t)	60	10(r)	10(t)	120	variável	124	variável
Testes	6.000			6.000			12.214		14.653	
	Usuários	Imagens		Usuários	Imagens		Usuários		Imagens	
Ambas as íris	30	10(r)	10(t)	30	10(r)	10(t)	69		variável	
Testes	3.000			3.000			6.229			

A Figura A.1 contém a estrutura principal do código em C++, desenvolvido para execu-

ção dos experimentos. É possível observar que existem estruturas de repetição do tipo *for* e do tipo *while*. Os quatro primeiros *for* definem, respectivamente: qual o usuário de referência; qual a imagem de referência deste usuário; qual o usuário de teste; e qual a imagem de teste deste usuário, que participarão do teste. O quinto *for* define quantas correções de erros serão aceitas na codificação do código RS. O *while* só é usado quando a busca por rotação é ativada, e realiza a rotação das imagens de referência. A busca por rotação é detalhada no Capítulo 3. O sexto e último *for* realiza a rotação das imagens de teste.

```
for (i_user)// usuário de referência
 for (i_imagens)// imagem de referência do usuário: i_user
  for (i_user_test)// usuário de teste
   for (i_imagens_test)// imagem de teste do usuário de teste: i_user_test
    for (tRS)// número de correções de erros do código RS (t)
     RS_encoding// codificação usando RS
     Hadamard_encoding// codificação usando Hadamard
     while (rotation search)// rotação das imagens de referência
      for (code_index)// rotação das imagens de teste
       Hadamard_decoding// decodificação usando Hadamard
       match_decision// decisão sobre a identificação
      end
     end
    end
   end
  end
 end
end
```

Figura A.1: *Estrutura principal do experimento (C++)*

O *"RS_encoding"* chama a função mostrada na Figura A.2, para realizar a codificação usando o código RS. O *"Hadamard_encoding"* chama a função que realiza a codificação usando o código de Hadamard. O *"Hadamard_decoding"* chama a função que realiza a decodificação usando o código de Hadamard. O *"match_decision"* faz a comparação propriamente dita e define se houve identificação positiva do usuário e de sua respectiva imagem de referência, quando comparado com a sua imagem de teste.

Na Figura A.2 é disponibilizada a função desenvolvida em C++ que realiza a codificação do código de Reed Solomon. Este código é uma versão resumida do *software* desenvolvido para a realização dos testes, para facilitar o entendimento. Nele, vale destacar que: a matriz IN[129][1025] é uma variável em C++, que corresponde à entrada do codificador RS; as ope-

rações "*add*" e "*mult*" correspondem respectivamente à adição ($\oplus$) e à multiplicação ($\otimes$), e são realizadas em $GF(2^m)$; e a matriz gk[j][2*t-j] é uma variável em C++, que corresponde aos coeficientes de $g(x)$. Os coeficientes de $g(x)$ foram gerados pelo *software* de forma recursiva, deste modo são armazenados todos os coeficientes para $0 \leq i \leq n-k$. Todos os coeficientes são armazenados na variável gk[][] na ordem que foram gerados, porém, por conveniência, quando é necessário usá-los na função, é preciso inverter a sua ordem, por isso o índice usado é [2*t-j].

```
void Codificador_RS(int n,int k,int IN[129][1025])
{
 int i,j,t;
 t=(n-k)/2;
 //Início do codificador Reed Solomon usando registradores de deslocamento (
     processo recursivo)
 for(i=1;i<=k;i++)
  for(j=0;j<=2*t-1;j++)
   b[j][i]=add[mult[add[b[j][i-1]][IN[t][i]]][gk[j][2*t-j]]][b[j-1][i-1]];
 //Fim do codificador Reed Solomon
}
```

Figura A.2: *Função para codificação RS, adaptado do software desenvolvido em C++*

A.3.1 O CÓDIGO REED-SOLOMON

Na Figura A.3 [54] é possível observar o esquema usado para codificar uma mensagem com o código Reed-Solomon (RS) [56]. Trata-se de um registrador de deslocamento linear com realimentação (LFSR - *Linear Feedback Shift Register*). Os coeficientes de $g(x)$ são obtidos a partir dos coeficientes de:

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)\cdots(x-\alpha^{n-k}), \tag{A.1}$$

que, após a realização de todas as multiplicações dos fatores, resulta em:

$$g(x) = x^{n-k} + g_{n-k-1}\cdot x^{n-k-1} + g_{n-k-2}\cdot x^{n-k-2} + \cdots + g_3\cdot x^3 + g_2\cdot x^2 + g_1\cdot x + g_0. \tag{A.2}$$

Para o código de Reed Solomon:

$$n-k = 2t, \tag{A.3}$$

então, substituindo na equação (A.2):

$$g(x) = x^{2t} + g_{2t-1}\cdot x^{2t-1} + g_{2t-2}\cdot x^{2t-2} + \cdots + g_3\cdot x^3 + g_2\cdot x^2 + g_1\cdot x + g_0. \tag{A.4}$$

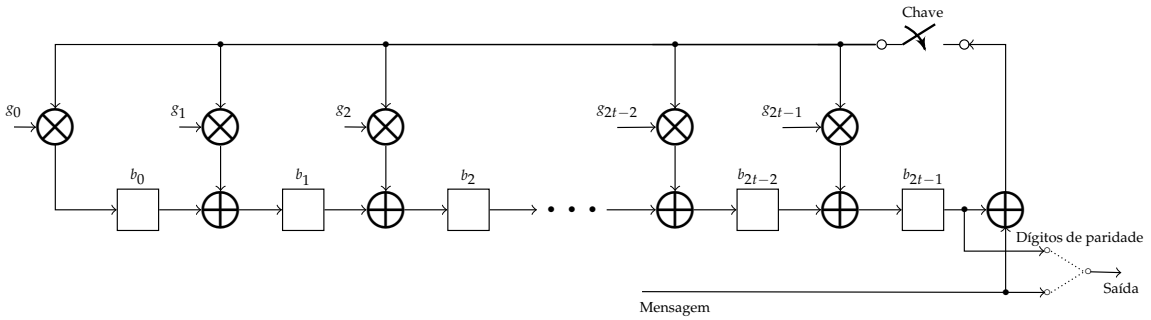


Figura A.3: *Circuito para codificar um código cíclico não binário - LFSR codificador RS.*

A partir da equação A.4 é possível obter os coeficientes de $g(x)$ que serão utilizados para calcular os termos b_j presentes na Figura A.3. O *software* em C++, para obter os coeficientes de $g(x)$, foi implementado com estruturas de repetição, conforme pode ser observado na Figura A.4. A ideia da implementação da multiplicação dos fatores da equação A.1 é multiplicar apenas os dois primeiros fatores de cada vez e após encontrar o resultado, é realizada novamente a multiplicação do resultado obtido na primeira iteração com o próximo fator. Deste modo, é possível realizar a implementação usando estruturas de repetição do tipo *for*.

No Exemplo A.1 é possível observar que os valores de $g(x)$ são deixados em função de: α, α^2, α^3, α^4, α^6, α^7, α^8, α^9 e α^{10}. Para o *software* é necessário encontrar os valores numéricos de cada α, deste modo, é necessário escolher um polinômio $g(x)$ que possa ser usado no Exemplo A.1. Os polinômios $g(x)$ escolhidos para implementar o *software* em C++ podem ser observados na Tabela A.2. Estes polinômios são exatamente os mesmos usados pelo MATLAB®.

Exemplo A.1

Cálculo do $g(x)$ em GF(16), exemplificando o modo como o *software* desenvolvido realiza esta operação.

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4), \tag{A.5}$$

calculando a multiplicação dos dois primeiros fatores:

$$g(x) = (x^2 - x(\alpha + \alpha^2) + \alpha^3)(x - \alpha^3)(x - \alpha^4), \tag{A.6}$$

calculando a multiplicação dos dois novos primeiros fatores:

$$g(x) = (x^3 - x^2(\alpha + \alpha^2 + \alpha^3) + x(\alpha^3 + \alpha^4 + \alpha^5) + \alpha^6)(x - \alpha^4), \tag{A.7}$$

calculando a multiplicação dos dois novos primeiros fatores:

$$g(x) = x^4 - x^3(\alpha + \alpha^2 + \alpha^3 + \alpha^4) + x^2(\alpha^3 + \alpha^4 + \alpha^6 + \alpha^7) - x(\alpha^6 + \alpha^7 + \alpha^8 + \alpha^9) + \alpha^{10}, \tag{A.8}$$

os coeficientes de $g(x)$ pertencem ao $GF(4)$, deste modo é possível reescrever a equação A.8 trocando os sinais dos coeficientes "$-$" por "$+$":

$$g(x) = x^4 + x^3(\alpha + \alpha^2 + \alpha^3 + \alpha^4) + x^2(\alpha^3 + \alpha^4 + \alpha^6 + \alpha^7) + x(\alpha^6 + \alpha^7 + \alpha^8 + \alpha^9) + \alpha^{10}, \tag{A.9}$$

substituindo os coeficientes de $g(x)$:

$$g(x) = x^4 + x^3 \underbrace{(\alpha + \alpha^2 + \alpha^3 + \alpha^4)}_{g_3} + x^2 \underbrace{(\alpha^3 + \alpha^4 + \alpha^6 + \alpha^7)}_{g_2} + x \underbrace{(\alpha^6 + \alpha^7 + \alpha^8 + \alpha^9)}_{g_1} + \underbrace{\alpha^{10}}_{g_0}$$

$$g(x) = x^4 + g_3x^3 + g_2x^2 + g_1x + g_0, \tag{A.10}$$

□

Tabela A.2: *Polinômios $g(x)$ escolhidos para a implementação em C++.*

m	$GF(2^m)$	Polinômio $g(x)$	$g(\alpha)$	α^m
2	$GF(4)$	$x^2 + x + 1$	$\alpha^2 + \alpha + 1 = 0$	$\alpha^2 = \alpha + 1$
3	$GF(8)$	$x^3 + x + 1$	$\alpha^3 + \alpha + 1 = 0$	$\alpha^3 = \alpha + 1$
4	$GF(16)$	$x^4 + x + 1$	$\alpha^4 + \alpha + 1 = 0$	$\alpha^4 = \alpha + 1$
5	$GF(32)$	$x^5 + x^2 + 1$	$\alpha^5 + \alpha^2 + 1 = 0$	$\alpha^5 = \alpha^2 + 1$
6	$GF(64)$	$x^6 + x + 1$	$\alpha^6 + \alpha + 1 = 0$	$\alpha^6 = \alpha + 1$
7	$GF(128)$	$x^7 + x^3 + 1$	$\alpha^7 + \alpha^3 + 1 = 0$	$\alpha^7 = \alpha^3 + 1$
8	$GF(256)$	$x^8 + x^4 + x^3 + x^2 + 1$	$\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1 = 0$	$\alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + 1$
9	$GF(512)$	$x^9 + x^4 + 1$	$\alpha^9 + \alpha^4 + 1 = 0$	$\alpha^9 = \alpha^4 + 1$
10	$GF(1024)$	$x^{10} + x^3 + 1$	$\alpha^{10} + \alpha^3 + 1 = 0$	$\alpha^{10} = \alpha^3 + 1$

O Exemplo A.2 mostra como é feito o cálculo matemático de todas as adições e de todas as multiplicações em $GF(4)$, para o polinômio $g(x)$ dado pela Tabela A.2.

Exemplo A.2

Cálculo matemático dos coeficientes α^i para $g(\alpha) = \alpha^2 + \alpha + 1$ em $GF(4)$:

$g(\alpha) = \alpha^2 + \alpha + 1$

$\text{p}/g(\alpha) = 0 \Rightarrow \alpha^2 = \alpha + 1$

$g(\alpha)$		Binário	Decimal
$0 =$	0	$= 00 =$	0
$1 =$	1	$= 01 =$	1
$\alpha =$	α	$= 10 =$	2
$\alpha^2 =$	$\alpha \oplus 1$	$= 11 =$	3

Cálculo das adições $\oplus$ para $g(\alpha) = \alpha^2 + \alpha + 1$ em $GF(4)$:

$0 \oplus 0 = 0$;

$0 \oplus 1 = 1 \oplus 0 = 1$;

$0 \oplus 2 = 2 \oplus 0 = 2$;

$0 \oplus 3 = 3 \oplus 0 = 3$;

$1 \oplus 1 = 0$;

$1 \oplus 2 = 2 \oplus 1 = \alpha \oplus 1 = 3$;

$1 \oplus 3 = 3 \oplus 1 = \alpha^2 \oplus 1 = \alpha \oplus 1 \oplus 1 = \alpha = 2$;

$2 \oplus 2 = \alpha \oplus \alpha = 0$;

$2 \oplus 3 = 3 \oplus 2 = \alpha^2 \oplus \alpha = \alpha \oplus 1 \oplus \alpha = 1$;

$3 \oplus 3 = \alpha^2 \oplus \alpha^2 = 0$;

Cálculo das multiplicações $\otimes$ para $g(\alpha) = \alpha^2 + \alpha + 1$ em $GF(4)$:

$0 \otimes 0 = 0$;

$0 \otimes 1 = 1 \otimes 0 = 0$;

$0 \otimes 2 = 2 \otimes 0 = 0$;

$0 \otimes 3 = 3 \otimes 0 = 0$;

$1 \otimes 1 = 1$;

$1 \otimes 2 = 2 \otimes 1 = \alpha \otimes 1 = \alpha = 2$;

$1 \otimes 3 = 3 \otimes 1 = \alpha^2 \otimes 1 = \alpha^2 = 3$;

$2 \otimes 2 = \alpha \otimes \alpha = \alpha^2 = 3$;

$2 \otimes 3 = 3 \otimes 2 = \alpha^2 \otimes \alpha = (\alpha \oplus 1) \otimes \alpha = \alpha^2 \oplus \alpha = \alpha \oplus 1 \oplus \alpha = 1$;

$3 \otimes 3 = \alpha^2 \otimes \alpha^2 = (\alpha \oplus 1) \otimes (\alpha \oplus 1) = \alpha^2 \oplus \alpha \oplus \alpha \oplus 1 = \alpha^2 \oplus 1 = \alpha \oplus 1 \oplus 1 = \alpha = 2$;

montando os resultados em formato de tabelas:

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

e

$\otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

□

É possível calcular todos os α^i da equação A.10 usando as informações da Tabela A.2, observando que para o Exemplo A.1: $m = 4$ e para o Exemplo A.2: $m = 2$. O Exemplo A.3 calcula todos os coeficientes de α^i para $g(\alpha) = \alpha^4 + \alpha + 1$ e para $m = 4$.

Exemplo A.3

Cálculo dos coeficientes α^i, pelo método convencional, do Exemplo A.1 para $g(\alpha) = \alpha^4 + \alpha + 1$:

$$
\begin{aligned}
\text{por definição: } 0 = &\quad 0 &&= 0000_{(2)} = 0_{(10)} \\
\text{para } i = 0\text{: } \alpha^0 = &\quad 1 &&= 0001_{(2)} = 1_{(10)} \\
\text{para } i = 1\text{: } \alpha^1 = &\quad \alpha &&= 0010_{(2)} = 2_{(10)} \\
\text{para } i = 2\text{: } \alpha^2 = &\quad \alpha^2 &&= 0100_{(2)} = 4_{(10)} \\
\text{para } i = 3\text{: } \alpha^3 = &\quad \alpha^3 &&= 1000_{(2)} = 8_{(10)} \\
\text{para } i = 4\text{: } \alpha^4 = &\quad \alpha + 1 &&= 0011_{(2)} = 3_{(10)} \\
\text{para } i = 5\text{: } \alpha^5 = &\quad \alpha^4\alpha = \alpha^2 + \alpha &&= 0110_{(2)} = 6_{(10)} \\
\text{para } i = 6\text{: } \alpha^6 = &\quad \alpha^5\alpha = \alpha^3 + \alpha^2 &&= 1100_{(2)} = 12_{(10)} \\
\text{para } i = 7\text{: } \alpha^7 = &\quad \alpha^6\alpha = \alpha^3 + \alpha + 1 &&= 1011_{(2)} = 11_{(10)} \\
\text{para } i = 8\text{: } \alpha^8 = &\quad \alpha^7\alpha = \alpha^2 + 1 &&= 0101_{(2)} = 5_{(10)} \\
\text{para } i = 9\text{: } \alpha^9 = &\quad \alpha^8\alpha = \alpha^3 + \alpha &&= 1010_{(2)} = 10_{(10)}
\end{aligned}
$$

$$\begin{aligned}
\text{para } i = 10\text{: } \alpha^{10} &= \alpha^9\alpha = \alpha^2 + \alpha + 1 &&= 0111_{(2)} = 7_{(10)} \\
\text{para } i = 11\text{: } \alpha^{11} &= \alpha^{10}\alpha = \alpha^3 + \alpha^2 + \alpha &&= 1110_{(2)} = 14_{(10)} \\
\text{para } i = 12\text{: } \alpha^{12} &= \alpha^{11}\alpha = \alpha^3 + \alpha^2 + \alpha + 1 &&= 1111_{(2)} = 15_{(10)} \\
\text{para } i = 13\text{: } \alpha^{13} &= \alpha^{12}\alpha = \alpha^3 + \alpha^2 + 1 &&= 1101_{(2)} = 13_{(10)} \\
\text{para } i = 14\text{: } \alpha^{14} &= \alpha^{13}\alpha = \alpha^3 + 1 &&= 1001_{(2)} = 9_{(10)}.
\end{aligned}$$

Para explicar melhor como o *software* realiza as adições e multiplicações em $GF(2^m)$, será detalhado o procedimento implementado com a ajuda dos Exemplos A.4, A.5, A.6 e A.7:

Exemplo A.4

P/ $m = 4$ e $g(\alpha) = \alpha^4 + \alpha + 1$:

Cálculo das adições:

$x_{(10)} \oplus y_{(10)} = x_{(2)} \oplus y_{(2)}$;

$10_{(10)} \oplus 15_{(10)} = 1010_{(2)} \oplus 1111_{(2)} = 0101_{(2)} = 5_{(10)}$;

$14_{(10)} \oplus 7_{(10)} = 1110_{(2)} \oplus 0111_{(2)} = 1001_{(2)} = 9_{(10)}$;

$11_{(10)} \oplus 3_{(10)} = 1011_{(2)} \oplus 0011_{(2)} = 1000_{(2)} = 8_{(10)}$; □

As adições são independentes do polinômio $g(x)$ e podem ser calculadas convertendo-se os números dados de decimal para binário, realizando a soma ou-exclusivo e depois convertendo o resultado de volta para decimal.

Exemplo A.5

Cálculo das multiplicações:

$$10_{(10)} \otimes 15_{(10)} = 1010_{(2)} \otimes 1111_{(2)} = \begin{array}{r} 1010 \\ \otimes 1111 \\ \hline 1010 \\ \oplus 1010 \\ \oplus 1010 \\ \oplus 1010 \\ \hline 1100110 \end{array}$$

O resultado da multiplicação precisa pertencer ao $GF(2^m)$. Para $m = 4$, o maior valor permitido é $1111_{(2)}$. Para reduzir o resultado da multiplicação, o resultado será per-

corrido da esquerda para a direita, sempre procurando o número 1. Este número será substituído pelo polinômio $g(x)$ usado em $GF(2^m)$, devidamente deslocado à esquerda, começando a contagem de deslocamentos a partir da posição inicial do maior expoente do polinômio $g(x)$. Após esta substituição, será realizada a operação ou-exclusivo do número restante, sem o primeiro 1 encontrado, com o número correspondente do polinômio $g(x)$ deslocado:

$$\begin{array}{rr} 110\dot{:}0110_{(2)} = & 0100110 \\ & \oplus 1100 \\ \hline & 101010 \end{array}$$

O polinômio $g(x)$ possui grau 4, ou seja, a posição inicial para identificação dos deslocamentos é o quinto *bit*, da direita para a esquerda. Como o primeiro 1 está na posição 7, são dois deslocamentos. O mesmo procedimento é repetido até obter um resultado que pertença a $GF(16)$:

$$\begin{array}{rr} 10\dot{:}1010_{(2)} = & 1010 \\ & \oplus 0110 \\ \hline & 1100 \end{array}$$

Neste caso foi apenas um deslocamento. O resultado obtido pertence ao $GF(16)$, então, agora, basta convertê-lo para decimal para obter o resultado da multiplicação inicial:
$1100_{(2)} = 12_{(10)}$, ou seja:
$10_{(10)} \otimes 15_{(10)} = 12_{(10)}$. □

Exemplo A.6

Calculo de $15_{(10)} \otimes 15_{(10)}$:

$$15_{(10)} \otimes 15_{(10)} = 1111_{(2)} \otimes 1111_{(2)} =$$

$$\begin{array}{r} 1111 \\ \otimes 1111 \\ \hline 1111 \\ \oplus 1111 \\ \oplus 1111 \\ \oplus 1111 \\ \hline 1010101 \end{array}$$

Reduzindo em $GF(16)$:

$$\begin{array}{rr} 101 \vdots 0101_{(2)} = & 0101 \\ \text{(nenhum deslocamento)} & \oplus 0011 \\ \text{(dois deslocamentos)} & \oplus 1100 \\ \hline & 1010 \end{array}$$

Convertendo para decimal:

$1010_{(2)} = 10_{(10)}$

ou seja: $15_{(10)} \otimes 15_{(10)} = 1010_{(2)} = 10_{(10)}$ □

Exemplo A.7

Calculo de $31_{(10)} \otimes 31_{(10)}$ em $GF(32)$:

$$31_{(10)} \otimes 31_{(10)} = 11111_{(2)} \otimes 11111_{(2)} =$$

$$\begin{array}{r} 11111 \\ \otimes 11111 \\ \hline 11111 \\ \oplus 11111 \\ \oplus 11111 \\ \oplus 11111 \\ \oplus 11111 \\ \hline 101010101 \end{array}$$

Reduzindo em $GF(32)$, onde $g(\alpha) = \alpha^5 + \alpha^2 + 1$:

$$\begin{array}{rr} 1010 \vdots 10101_{(2)} = & 1010101 \\ \text{(três deslocamentos)} & \oplus 101000 \\ \hline & 1111101 \end{array}$$

$$\begin{array}{rr} 11 \vdots 11101_{(2)} = & 111101 \\ \text{(um deslocamento)} & \oplus 1010 \\ \hline & 110111 \end{array}$$

$$
\begin{array}{rr}
 & 1\dot{:}10111_{(2)} = 10111 \\
\text{(nenhum deslocamento)} & \oplus 00101 \\
\hline
 & 10010
\end{array}
$$

Convertendo para decimal:

$10010_{(2)} = 18_{(10)}$,

ou seja: $31_{(10)} \otimes 31_{(10)} = 10010_{(2)} = 18_{(10)}$ □

A Figura A.4 mostra a rotina que foi implementada em C++, que encontra todos os coeficientes de $g(x)$ da equação A.10 para qualquer $n - k$ válido. No teste padrão de comparação, $n = 61$ e $17 \leq k \leq 59$. Por questões de conveniência, o *software* foi limitado a trabalhar no máximo com $n = 1023$. Para o caso mais complexo que foram realizados experimentos até o momento, o valor máximo foi de $n = 127$. Na Figura A.4 está disponibilizada também a chamada da função "gf_mult2", que é responsável por realizar a multiplicação em $GF(2^m)$, dos dois números decimais que estão dentro dos parêntesis. O símbolo " ^ " em C++ realiza a soma ou-exclusivo de dois números decimais, retornando o resultado também em decimal; a conversão decimal-binário-decimal é interna da própria linguagem de programação.

Todas as multiplicações e as adições em $GF(2^m)$, para $2 \leq m \leq 10$ para os termos α^i foram geradas e armazenadas em arquivos de texto sem formatação, para agilizar o tempo gasto na execução dos experimentos. Na Tabela A.3 é possível observar todas as adições possíveis, na Tabela A.4 todas as multiplicações possíveis e na Tabela A.5 todos os coeficientes de $g(\alpha)$, todos para $m = 4$ em $GF(16)$, para $g(\alpha) = \alpha^4 + \alpha + 1$.

O LFSR da Figura A.3 recebe a mensagem de entrada sequencialmente, um *bit* de cada vez, o que obriga escrever as equações em função do momento i em que ocorreram. Deste modo:

$$b_{0,i} = (b_{2t-1,i-1} \oplus IN_i) \otimes g_0; \tag{A.11}$$

$$b_{1,i} = ((b_{2t-1,i-1} \oplus IN_i) \otimes g_1) \oplus b_{0,i-1}; \tag{A.12}$$

$$b_{2,i} = ((b_{2t-1,i-1} \oplus IN_i) \otimes g_2) \oplus b_{1,i-1}; \tag{A.13}$$

$$\vdots$$

$$b_{2t-2,i} = ((b_{2t-1,i-1} \oplus IN_i) \otimes g_{2t-2}) \oplus b_{2t-3,i-1}; \tag{A.14}$$

$$b_{2t-1,i} = ((b_{2t-1,i-1} \oplus IN_i) \otimes g_{2t-1}) \oplus b_{2t-2,i-1}; \tag{A.15}$$

```
void gerar_coeficientes_gfx(void)
{
 fator1[1022]=1;fator1[1023]=2;//primeiro fator (x-a) onde a=2
 fator2[1022]=1;fator2[1023]=4;//segundo fator (x-a^2) onde a^2=4
 for (i=1;i<=(n-k-1);i++)
 {
  for (i2=0;i2<2048;i2++) Pmul_res[i2]=0;
  fator_mult[1022-i]=1;
  for (i2=(1023-i);i2<(1023);i2++)
   fator_mult[i2]=gf_mult2(fator1[i2],fator2[1023])^fator1[i2+1];
  fator_mult[1023]=gf_mult2(fator1[1023],fator2[1023]);
  for (j=(1023-(n-k));j<=1023;j++)
   fator1[j]=fator_mult[j];
  fator2[1023]=gf_mult2(fator2[1023],2);
  for (j=0;j<=1023;j++)
   if (fator1[j]!=0) fator_tab[i][j]=fator1[j];
 }
j=n-k+1;
for (i=0;i<=1023;i++)
 if (fator_mult[i]!=0)
 {
  j=j-1;
  gn[i][j]=fator_mult[i];
 }
for (i=0;i<=s;i++)
 for (j=0;j<=s;j++)
  addgfx[i][j]=i^j;
for (i=0;i<=s;i++)
 for (j=0;j<=s;j++)
  multgfx[i][j]=gf_mult2(i,j);
}
```

Figura A.4: *Função para calcular* $\alpha^i \oplus \alpha^j$ *e* $\alpha^i \otimes \alpha^j$ *e* $g(x)$ *para* $2 \leq m \leq 10$ *em C++*

Fazendo: $2t - 1 = j$:

$$b_{j,i} = ((b_{j,i-1} \oplus IN_i) \otimes g_j) \oplus b_{j-1,i-1}; \tag{A.16}$$

Generalizando,

$$b_{j,i} = \begin{cases} ((b_{j,i-1} \oplus IN_i) \otimes g_j) \oplus b_{j-1,i-1}, & \text{se } 0 < j \leq 2t-1; 1 \leq i \leq k; \\ 0, & \text{nos outros casos.} \end{cases} \tag{A.17}$$

A equação (A.17) foi organizada com os índices i e j para facilitar a implementação em C++. Ainda é necessário, além de implementar a equação (A.17), foram realizadas as opera-

Tabela A.3: *Adições $\alpha^i \oplus \alpha^j$ em $GF(16)$, para $g(\alpha) = \alpha^4 + \alpha + 1$.*

$\oplus$	$0 \Leftrightarrow 0$	$\alpha^0 \Leftrightarrow 1$	$\alpha^1 \Leftrightarrow 2$	$\alpha^4 \Leftrightarrow 3$	$\alpha^2 \Leftrightarrow 4$	$\alpha^8 \Leftrightarrow 5$	$\alpha^5 \Leftrightarrow 6$	$\alpha^{10} \Leftrightarrow 7$	$\alpha^3 \Leftrightarrow 8$	$\alpha^{14} \Leftrightarrow 9$	$\alpha^9 \Leftrightarrow 10$	$\alpha^7 \Leftrightarrow 11$	$\alpha^6 \Leftrightarrow 12$	$\alpha^{13} \Leftrightarrow 13$	$\alpha^{11} \Leftrightarrow 14$	$\alpha^{12} \Leftrightarrow 15$
$0 \Leftrightarrow 0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\alpha^0 \Leftrightarrow 1$	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
$\alpha^1 \Leftrightarrow 2$	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
$\alpha^4 \Leftrightarrow 3$	3	2	8	0	7	6	5	4	11	10	9	8	15	14	13	12
$\alpha^2 \Leftrightarrow 4$	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
$\alpha^8 \Leftrightarrow 5$	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
$\alpha^5 \Leftrightarrow 6$	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
$\alpha^{10} \Leftrightarrow 7$	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
$\alpha^3 \Leftrightarrow 8$	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
$\alpha^{14} \Leftrightarrow 9$	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
$\alpha^9 \Leftrightarrow 10$	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
$\alpha^7 \Leftrightarrow 11$	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
$\alpha^6 \Leftrightarrow 12$	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
$\alpha^{13} \Leftrightarrow 13$	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
$\alpha^{11} \Leftrightarrow 14$	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
$\alpha^{12} \Leftrightarrow 15$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabela A.4: *Multiplicações $\alpha^i \otimes \alpha^j$ em $GF(16)$, para $g(\alpha) = \alpha^4 + \alpha + 1$.*

$\otimes$	$0 \Leftrightarrow 0$	$\alpha^0 \Leftrightarrow 1$	$\alpha^1 \Leftrightarrow 2$	$\alpha^4 \Leftrightarrow 3$	$\alpha^2 \Leftrightarrow 4$	$\alpha^8 \Leftrightarrow 5$	$\alpha^5 \Leftrightarrow 6$	$\alpha^{10} \Leftrightarrow 7$	$\alpha^3 \Leftrightarrow 8$	$\alpha^{14} \Leftrightarrow 9$	$\alpha^9 \Leftrightarrow 10$	$\alpha^7 \Leftrightarrow 11$	$\alpha^6 \Leftrightarrow 12$	$\alpha^{13} \Leftrightarrow 13$	$\alpha^{11} \Leftrightarrow 14$	$\alpha^{12} \Leftrightarrow 15$
$0 \Leftrightarrow 0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\alpha^0 \Leftrightarrow 1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\alpha^1 \Leftrightarrow 2$	0	2	4	6	8	10	12	14	3	1	7	5	11	9	15	13
$\alpha^4 \Leftrightarrow 3$	0	3	6	5	12	15	10	9	11	8	13	14	7	4	1	2
$\alpha^2 \Leftrightarrow 4$	0	4	8	12	3	7	11	15	6	2	14	10	5	1	13	9
$\alpha^8 \Leftrightarrow 5$	0	5	10	15	7	2	13	8	14	11	4	1	9	12	3	6
$\alpha^5 \Leftrightarrow 6$	0	6	12	10	11	13	7	1	5	3	9	15	14	8	2	4
$\alpha^{10} \Leftrightarrow 7$	0	7	14	9	15	8	1	6	13	10	3	4	2	5	12	11
$\alpha^3 \Leftrightarrow 8$	0	8	3	11	6	14	5	13	12	4	15	7	10	2	9	1
$\alpha^{14} \Leftrightarrow 9$	0	9	1	8	2	11	3	10	4	13	5	12	6	15	7	14
$\alpha^9 \Leftrightarrow 10$	0	10	7	13	14	4	9	3	15	5	8	2	1	11	6	12
$\alpha^7 \Leftrightarrow 11$	0	11	5	14	10	1	15	4	7	12	2	9	13	6	8	3
$\alpha^6 \Leftrightarrow 12$	0	12	11	7	5	9	14	2	10	6	1	13	15	3	4	8
$\alpha^{13} \Leftrightarrow 13$	0	13	9	4	1	12	8	5	2	15	11	6	3	14	10	7
$\alpha^{11} \Leftrightarrow 14$	0	14	15	1	13	3	2	12	9	7	6	8	4	10	11	5
$\alpha^{12} \Leftrightarrow 15$	0	15	13	2	9	6	4	11	1	14	12	3	8	7	5	10

ções de adição em $GF(2^m)$ e de multiplicação em $GF(2^m)$, representadas, respectivamente, pelos símbolos $\oplus$ e $\otimes$. No Exemplo A.2 é possível acompanhar como encontrar as tabelas de adição e de multiplicação em $GF(4)$, para o polinômio $g(x)$ de $GF(4)$ usado no *software* desenvolvido.

A.3.2 O CÓDIGO DE HADAMARD

Uma matriz $\mathbf{H}=\left[h_{ij}\right]_{n\times n}$ é uma matriz de Hadamard se $h_{ij} \in \{+1, -1\}$ para todo i, j e $\mathbf{HH^T} = n\mathbf{I}$, $n \in \mathbb{N}^*$ e $\mathbf{I}$ é a matriz identidade. Visto de forma apropriada, as colunas da

Tabela A.5: *Coeficientes de $g(\alpha)$ para $g(\alpha) = \alpha^4 + \alpha + 1$, em $GF(16)$.*

$n-k$	g_{14}	g_{13}	g_{12}	g_{11}	g_{10}	g_9	g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0
2													1	6	8
3												1	14	13	12
4											1	13	12	8	7
5										1	11	4	6	2	1
6									1	7	9	3	12	10	12
7								1	12	13	15	2	7	14	13
8							1	9	4	3	4	13	6	14	12
9						1	3	1	13	9	3	13	7	10	1
10					1	4	8	10	12	9	4	2	12	2	7
11				1	10	5	3	10	13	3	15	3	6	8	12
12			1	5	9	5	8	1	4	13	9	4	12	13	8
13		1	8	5	10	4	3	9	12	7	11	13	14	6	2
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

matriz de Hadamard n x n formam um código binário de comprimento n, com n palavras código. Para enxergar esta informação, note que cada coluna de $\mathbf{H}$ é um vetor binário, no qual os símbolos são $\{+1, -1\}$ ao invés de $\{0, 1\}$. Um dos aspectos mais interessantes deste código é o fato de que, para todo n par, quaisquer duas palavras código tem distância de Hamming de exatamente $n/2$ entre si [55] e [54].

Dada uma matriz $\mathbf{H}$, $n \times n$, o código de Hadamard de comprimento n, Had_n, é formado pelas palavras código das colunas de $\mathbf{H}$, em que os $+1$'s são substituídos por 0's e os -1's são substituídos por 1's, e pelo complemento das colunas de $\mathbf{H}$. Existem alguns métodos para encontrar a matriz de Hadamard e, consequentemente, o código de Hadamard [55] e [54]. No *software* em C++ é usado o método de Sylvester para construção da matriz de Hadamard. No Exemplo A.8 é possível ver como encontrar as matrizes de Hadamard de ordem 2^k. No *software* desenvolvido, k por questões de simplificação foi limitado em, no máximo 10, ou seja, é possível trabalhar com matrizes de Hadamard de ordem até $\mathbf{H}_{2^{10}} = \mathbf{H}_{1024}$. Nos experimentos realizados até o momento, a maior ordem da matriz de Hadamard necessária foi $\mathbf{H}_{2^7} = \mathbf{H}_{128}$.

As matrizes de Hadamard foram geradas pelo *software* desenvolvido e armazenadas em arquivos de texto para agilizar o desempenho, diminuindo em cerca de dez segundos o tempo gasto na execução de cada experimento. Para realizar a codificação e a decodificação

usando o código de Hadamard, são necessárias respectivamente, a matriz de Hadamard e a matriz de Hadamard transposta, de mesma ordem; ambas foram armazenadas no mesmo arquivo por questões de simplificação.

Exemplo A.8

Método de Sylvester: para $n = 1$, $H_1 = [1]$, então

$$H_2 = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix}, \text{ ou seja, } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Para $n = 4$:

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix}, \text{ ou seja, } H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Generalizando,

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}, \text{ para } 2 \leq k \in \mathbb{N}. \qquad \square$$

A.4 TESTANDO O SOFTWARE

Para testar a confiabilidade do *software* desenvolvido, foi determinado um experimento padrão, que pudesse ser implementado tanto na versão antiga, desenvolvida em MATLAB®, quanto na nova versão, desenvolvida em C++. Devido ao fato de que existem pelo menos quinze variáveis envolvidas nas simulações, foram realizados inicialmente vários experimentos apenas em C++ para identificar quais precisam ser fixadas.

Como previsto, as dificuldades foram muitas e os erros inúmeros, porém, com as dificuldades, surgem as ideias. Uma ideia que surgiu durante a fase de comparações foi a de verificar a facilidade/dificuldade de armazenamento dos dados do código íris em hexadecimal, pois seria possível reduzir o tamanho de cada arquivo que continha o código íris das bases de dados, em praticamente quatro vezes. Esta ideia foi abandonada, pois gastava-se mais tempo para converter os dados de hexadecimal para binário, do que lê-los diretamente em binário. Porém, esta ideia revelou uma característica presente nas palavras do código

íris, de todas as bases de dados disponíveis, que terminou por contribuir, mais adiante nos experimentos. Esta técnica é apresentada no Capítulo 5.

Não foi possível testar, inicialmente, o experimento por completo, devido à quantidade de possibilidades de existência de erros no *software*. Não se sabia quais rotinas estavam corretas e quais estavam erradas. Foi resolvido testá-las separadamente: a rotina de codificação usando o código de Hadamard; a rotina de codificação usando o código de Reed Solomon; e a rotina de decodificação usando o código de Hadamard. E assim foi feito, os erros foram corrigidos um a um, e após este passo, foi possível realizar experimentos de comparação entre as duas versões dos *softwares*. Também, para facilitar, foi reduzida a quantidade de usuários para um, e, gradativamente a mesma, foi aumentada, quando o sucesso foi verificado. Na versão mais recente usada para realizar os experimentos é possível executar os parâmetros do teste padrão, sempre podendo verificar se alguma alteração recente no *software* foi feita erroneamente.

O teste padrão consiste em realizar o experimento da seguinte forma:

- Usar a base de dados BIOSECURE;
- Experimento com uma única íris;
- 60 usuários, 10 imagens de teste e 10 imagens de referência para cada usuário;
- Escolher a opção de controlar a chave criptográfica "K";
- Escolher inserir zeros em vez de Rand_num;
- Escolher k_{shuf} como uma sequência de zeros;
- Manter a redundância;
- Realizar a busca padrão;
- Compor o código íris com apenas uma imagem de referência por vez;
- Escolher 3 como a quantidade de *bits* de dados e 2 como a quantidade de *bits* a serem inseridos no código íris;
- Escolher a variação do t_{RS} (quantidade de correções de erros do código RS) entre 1 e 22;
- Escolher o experimento para usuários genuínos.

Embora sejam muitas as exigências para realização do teste padrão, é possível escolher esta opção, quando o *software* é executado, que o mesmo automaticamente realizará o teste

solicitado. Quanto às demais opções, todas, menos a escolha da quantidade de usuários, imagens de referência e imagens de teste, precisam ser escolhidas em cada experimento realizado. A proposta para criação deste *software* foi justamente esta: poder realizar quaisquer testes em um mesmo arquivo executável. A versão antiga em MATLAB® possui um arquivo executável específico para cada base de dados; um arquivo executável para cada quantidade de íris escolhida por usuário (uma íris ou duas íris); um arquivo executável para genuínos e outro para impostores; e todas as opções disponíveis para personalização precisam ser informadas antes de cada execução.

Na Tabela A.6 está disponibilizada a saída do experimento padrão, realizado para estabelecer a confiabilidade no *software* desenvolvido em C++. Vale ressaltar que a chave criptográfica "K" não pode ser nula, pois afetaria a codificação do código de Reed Solomon. A opção escolhida foi definir um vetor de entrada como sendo a chave criptográfica "K". Este vetor contém na posição i o valor i, é uma ideia simples, pois seria fácil de implementar em ambos os *softwares*, além de atender às necessidades do código de RS. No Exemplo A.9 é possível observar o vetor de entrada $IN\,[i]$, que é uma variável da implementação em C++ e em MATLAB®. Para o experimento padrão, foi escolhida a opção de testes com uma única íris; a sequência usada no embaralhamento k_{shuf}, foi escolhida como sendo a sequência de 198 *bits* todos iguais a zeros; para o Rand_num foi escolhida a sequência de 792 *bits*, toda de zeros; como o teste em MATLAB® já estava implementado com inserção de dois *bits* pseudo aleatórios a cada três *bits* de dados, com busca padrão, com apenas uma imagem compondo a imagem de referência, estas opções foram as escolhidas; quanto ao teste para genuínos, esta opção foi escolhida pois o tempo de execução deste experimento é, pelo menos, doze vezes menor do que o tempo de execução do experimento para usuários impostores.

Exemplo A.9

Vetor de entrada IN$[i]$, para $1 \leq i \leq 32$:

IN$[i]$=1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32; □

A saída do experimento definido como padrão, foi comparada com a saída do experimento realizado no MATLAB®, quando constatou-se igualdade dos resultados, obtidos em ambos os testes, passou-se a realizar outros experimentos em busca de resultados nas outras bases de dados disponíveis. Realizou-se, durante aproximadamente dois meses, experimentos para tornar o *software* mais rápido, pois quanto mais rápido fosse a duração do

Tabela A.6: *Saídas do teste padrão, para a base de dados BIOSECURE, para uma única íris, com k_{shuf}=zeros, Rand_num=zeros, bits de dados = 3, bits inseridos = 2, com redundância, busca padrão, com $i = 1$ imagem, para usuários genuínos.*

t_{RS}	|K|	Acertos	Erros
1	354	3.804=63,40%	2.196=36,60%
2	342	4.496=74,93%	1.504=25,07%
3	330	4.843=80,72%	1.157=19,28%
4	318	5.135=85,58%	865=14,42%
5	306	5.288=88,13%	712=11,87%
6	294	5.397=89,95%	603=10,05%
7	282	5.442=90,70%	558=9,30%
8	270	5.528=92,13%	472=7,87%
9	258	5.571=92,85%	429=7,15%
10	246	5.613=93,55%	387=6,45%
11	234	5.685=94,75%	315=5,25%
12	222	5.732=95,53%	268=4,47%
13	210	5.768=96,13%	232=3,87%
14	198	5.805=96,75%	195=3,25%
15	186	5.832=97,20%	168=2,80%
16	174	5.876=97,93%	124=2,07%
17	162	5.910=98,50%	90=1,50%
18	150	5.946=99,10%	54=0,90%
19	138	5.961=99,35%	39=0,65%
20	126	5.984=99,73%	16=0,27%
21	114	5.990=99,83%	10=0,17%
22	102	5.996=99,93%	4=0,07%

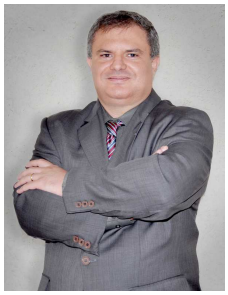
experimento, mais testes poderiam ser idealizados e executados. Dentre muitos pequenos detalhes, tais como substituir somas ou multiplicações recorrentes por variáveis, reduzindo quantidade de operações matemáticas, foram realizadas alterações estruturais em várias funções e várias rotinas, até que surgiu uma nova ideia.

Esta ideia tinha o objetivo de alterar o mecanismo de busca, de modo a torná-lo mais simples e, consequentemente, mais rápido. A busca por redução no tempo gasto por experimento, consistiu em analisar passo a passo todo o *software* desenvolvido, para tentar identificar quais rotinas ou funções eram as responsáveis pelo maior gasto de tempo da execução do experimento. A principal constatação foi que, mais da metade do tempo gasto em todo o experimento se dava na rotina de decodificação do código de Hadamard. Esta rotina foi exaustivamente analisada e alterada para que fosse obtido o seu melhor desem-

penho com o *software* desenvolvido.

Um outro item importante foi a constatação de que a busca conseguia obter mais identificações positivas, quando a rotação das imagens de teste, em relação às imagens de referência, eram de até quatro rotações. No experimento do MATLAB® a imagem de teste era rotacionada, desde a rotação 1 até a rotação 21 sequencialmente, enquanto que a imagem de referência era mantida fixa na rotação 11. Após analisar quais rotações das imagens de teste obtinham mais acerto, foi verificado que estes aconteciam, na grande maioria, entre as rotações mais centrais, normalmente entre as rotações 7 e 15, sendo que o maior número acontecia nas rotações mais próximas da 11. Foi implementada a busca centralizada que resultou em um ganho de tempo de, aproximadamente, duas vezes, apenas com esta alteração.

SOBRE O AUTOR



O autor nasceu em Recife, Pernambuco, no dia 11 de agosto de 1975. Formou-se em Engenharia Elétrica, modalidade Eletrônica, pela Universidade Federal de Pernambuco em 1998 e concluiu o Mestrado em Engenharia Elétrica em 2010, também pela Universidade Federal de Pernambuco. Seus interesses de pesquisa incluem Teoria da Informação, Códigos Corretores de Erro, Sistemas de Comunicação Digital, Processamento Digital de Sinais e Eletrônica Básica e Sistemas Embarcados.

Endereço: Universidade Federal de Pernambuco - UFPE
Centro de Tecnologia e Geociências - CTG
Escola de Engenharia
Departamento de Eletrônica e Sistemas - DES
Grupo de Pesquisa em Comunicações
Avenida da Arquitetura, s/n - Bloco A, 4º andar
Cidade Universitária, CEP: 50740-550 - Recife -
Pernambuco - Brasil, Fone: + 55 81 2126-8217

e-mail: `guilherme.nmelo@ufpe.br`

Esta tese foi diagramada usando LaTeX 2_ε[1] pelo autor.

[1] LaTeX 2_ε é uma extensão do LaTeX. LaTeX é uma coleção de macros criadas por Leslie Lamport para o sistema TeX, que foi desenvolvido por Donald E. Knuth. TeX é uma marca registrada da Sociedade Americana de Matemática ($\mathcal{AMS}$). O estilo usado na formatação desta tese foi escrito por Dinesh Das, Universidade do Texas. Modificado por Renato José de Sobral Cintra (2001), por André Leite Wanderley (2005), por José Sampaio de Lemos Neto (2010) e por Guilherme Nunes Melo (2016), todos da Universidade Federal de Pernambuco.