UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RAFAEL VIEIRA COELHO

Comparação Analítica dos Esquemas de Autenticação em Sistemas P2P de *Live Streaming*

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Marinho Pilla Barcellos
Orientador

Profa. Dra. Ingrid Jansch-Pôrto
Co-orientador

Porto Alegre, maio de 2011

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Coelho, Rafael Vieira

Comparação Analítica dos Esquemas de Autenticação em Sistemas P2P de *Live Streaming* / Rafael Vieira Coelho. – Porto Alegre: PPGC da UFRGS, 2011.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Orientador: Marinho Pilla Barcellos; Co-orientador: Ingrid Jansch-Pôrto.

1. Live Streaming. 2. High Definition. 3. Poluição de Conteúdo. 4. Autenticação. I. Barcellos, Marinho Pilla. II. Jansch-Pôrto, Ingrid. III. Título.

“O conhecimento é o processo de acumular dados;
a sabedoria reside na sua simplificação.”
— Martin H. Fischer

AGRADECIMENTOS

Meus sinceros votos de agradecimento,

- ao meu excelente amigo e orientador Professor Dr Marinho Pilla Barcellos, por sempre acreditar no meu potencial e pela paciência para me ajudar no desenvolvimento deste trabalho, fazendo com que eu me tornasse um profissional mais capacitado;
- à minha co-orientadora Professora Dra Ingrid Jansch-Pôrto e ao Professor Dr Luciano Paschoal Gaspary pela constante disponibilidade para contribuir no meu mestrado e pelos ótimos conselhos dados durante o mesmo;
- ao Rodolfo Stoffel Antunes e Jonata Pastro pela ajuda e suporte durante a fase final do trabalho que culminaram nesta dissertação;
- à minha mãe, Maria da Graça Vieira Coelho, irmã, Raquel Vieira Coelho, e em memória do meu pai, Carlos Henrique Coelho, por fornecer a base familiar necessária para que eu tivesse estrutura suficiente para me tornar o profissional e a pessoa que sou hoje;
- e um especial agradecimento à minha amada companheira Vanessa Bavaresco por todo carinho e apoio necessário para que eu pudesse sobrepor todas as barreiras encontradas nesta empreitada.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ALPS	Authenticating Live Peer-to-Peer Streams
BiBa	Bins and Balls Signature
DONet	Data-driven Overlay Network
DoS	Denial of Service
DSA	Digital Signature Algorithm
ECC	Error Correcting Code
EMSS	Efficient Multi-chained Stream Signature
FEC	Forward Error Correction
GBG	Generalized Butterfly Graph
HD	High Definition
HORS	Hash to Obtain Random Subset
IDA	Information Dispersal Algorithm
MAC	Message Authentication Codes
MD5	Message-Digest Algorithm 5
MTU	Maximum Transmission Unit
P2P	Peer-to-Peer
PARM	Pollution-Attack Resistant Multicast Authentication
StRepS	Streaming Reputation System
TFDP	Tree-Based Forward Digest Protocol
TTL	Time-To-Live
VoD	Video on Demand

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

As aplicações de *live streaming* em redes P2P têm grande potencial de crescimento, em popularidade e qualidade, mas são potencialmente vulneráveis ao ataque de poluição de conteúdo. Tal ataque corresponde a qualquer adulteração de áudio e/ou vídeo na mídia transmitida buscando ludibriar espectadores. A autenticação de blocos do fluxo de dados contínuo (*stream*) permite a detecção de conteúdo poluído e a possível identificação de pares maliciosos na rede P2P. A literatura contém diversas propostas de autenticação baseadas em assinatura digital leve e em amortização de assinatura digital. O presente trabalho, como nenhum anterior, compara tanto os esquemas de assinatura leve quanto os de amortização sob diferentes perspectivas de qualidade de transmissão (vídeos convencionais e de alta definição) em redes P2P de *live streaming* através da formularização analítica dos mesmos. Para isto, é feita uma comparação quantitativa de sobrecargas e nível de segurança, observando fenômenos e identificando desafios de pesquisa que precisarão ser vencidos. Particularmente, para avaliar a viabilidade dos esquemas, foi definido um coeficiente que leva em consideração o tempo total de transmissão e o custo computacional proporcionado pelo esquema estudado. Como uma das conclusões inéditas dessa análise, o esquema PARM2 obteve o melhor desempenho dentre os esquemas leves de assinatura. Já em relação aos esquemas de amortização, o Linear Digests se sobressaiu entre os demais. Por fim, foi analisada uma solução baseada na amortização de um esquema de assinatura leve, algo também inédito. A partir dessa análise em particular, foi possível observar que a autenticação é viável apenas com baixos e médios níveis de segurança em transmissões convencionais. Além disso, a partir dos resultados foi possível observar que o esquema de amortização Linear Digests deve ser utilizado isoladamente apenas em transmissões convencionais. Por fim, o esquema PARM2, se utilizado individualmente, não apresenta valores de coeficiente aceitáveis para os cenários estudados.

Palavras-chave: Live Streaming, High Definition, Poluição de Conteúdo, Autenticação.

Comparative Analisys of Authentication Schemes in P2P Live Streaming

ABSTRACT

P2P live streaming applications have great potential for improvements in terms of popularity as well as quality. However, they are subject to pollution attacks, in which a malicious user tampers with audio/video of streams in order to trick other users. The authentication of blocks in a stream allows the detection of pollution in received content and can lead to the identification of malicious users. The literature includes several proposals of light digital signature schemes and amortization schemes. Our work, unlike previous ones, compares both amortization schemes and light signature schemes under different perspectives of transmission quality (convencional and high definition videos) in P2P live streaming by formulating equations for them. For such, it quantitatively compares overheads and security levels during P2P streaming sessions, identifying research challenges to be faced. In particular, to assess the viability of such schemes, we defined a coefficient that takes into account the total transmission time and computational cost provided by such scheme. As one of the novel findings of our analysis, the scheme PARM2 showed the best performance among light signature schemes. With respect to amortization, Linear Digests was the most efficient. Finally, we analyzed a solution based on the amortization of a light signature. From this particular analysis, we observed that the authentication is only feasible in conventional transmissions with low and medium security leves. In addition, the amortization scheme Linear Digests must be used in isolation only in conventional transmissions. Finally, the scheme PARM2, if used alone, has no acceptable coefficient values for the studied scenarios.

Keywords: Live Streaming, High Definition, Authentication.

1 INTRODUÇÃO

As aplicações de *live streaming* em redes P2P visam a transmissão de áudio e vídeo em tempo real (*soft*) de uma fonte de dados a vários receptores, para exibição instantânea nos mesmos. Ultimamente, esse tipo de aplicação vem sendo cada vez mais utilizado, chegando a atrair milhões de usuários (SILVA et al., 2011). Além disso, estima-se que uma nova geração de aplicações de transmissão em alta definição na Internet dominará o mercado nos próximos quatro anos, sendo que países como, Japão e Holanda já apresentam infra-estrutura suficiente para sustentar tal tecnologia (FU; JAIN; VICENTE, 2009).

Assim como em outros sistemas P2P de larga escala na Internet, segurança é um aspecto chave para aplicações de *live streaming*. As mesmas estão sujeitas a uma série de ataques, sendo um dos mais relevantes, senão o mais, a poluição de conteúdo. Recentemente, trabalhos na literatura investigaram esse ataque no contexto de *live streaming* (DHUNGEL et al., 2007; YANG et al., 2008; BORGES; ALMEIDA; CAMPOS, 2008). A criação de mecanismos que reduzam os níveis de poluição é um desafio bastante relevante aos olhos da comunidade científica. Embora a prevenção contra a poluição de conteúdo tenha avançado no compartilhamento de arquivos, tais estratégias não podem ser diretamente aplicadas em sistemas de *live streaming*. Há duas razões para tal: as restrições temporais impostas por aplicações de *live streaming* e a falta de conhecimento prévio sobre o conteúdo a ser enviado (o conteúdo é enviado no momento de sua geração, impossibilitando assim o cálculo de todos os *hashes* do arquivo antes da transmissão).

Dhungel et al. (2007) referiram-se a um ataque de poluição em sistemas de *live streaming* como algo devastador, porque *chunks* adulterados são rapidamente distribuídos pela rede. Para evitar, ou pelo menos diminuir, o impacto desse ataque, é necessária a detecção de conteúdo poluído através da autenticação dos dados. A estratégia clássica para autenticação de dados é o uso de assinaturas digitais (CHALLAL; BETTAHAR; BOUABDALLAH, 2004). Em teoria, a fonte da *stream* pode assinar os dados com uma chave privada e possibilitar aos destinatários que verifiquem a autenticidade dos dados recebidos através da chave pública da fonte, que deve ser obtida de forma segura pelos receptores. Desta forma, os pares receptores podem descartar dados falsos ou corrompidos (enviados por pares maliciosos). Como consequência, torna-se possível a identificação de pares poluidores e potencialmente o isolamento dos mesmos. Dados os custos computacionais associados a assinaturas digitais, não existe certeza sobre a viabilidade prática do uso das mesmas para autenticação de uma *stream* de dados enviada em tempo real através de uma rede P2P.

Além disso, a crescente demanda mundial por transmissões em alta definição leva a desafios científicos que ainda não foram apropriadamente identificados, muito menos resolvidos (FU; JAIN; VICENTE, 2009). Quanto maior o volume de dados que precisam ser transmitidos em cenários de alta definição, maior serão as sobrecargas impostas pelo

mecanismo de autenticação utilizado para proteger a autenticidade dos dados transmitidos. Até hoje, pelo que sabemos, nenhum trabalho analisou a questão de transmissões de alta definição em redes P2P de *live streaming*. Mais especificamente, é inédita a investigação sobre formas de autenticar o conteúdo dessas transmissões que não sejam demasiadamente onerosas e que proporcionem um nível de segurança aceitável. Hefeeda e Mokhatarian (2010) comparam esquemas de autenticação durante transmissões de *streaming*. No entanto, a avaliação é restrita a esquemas de amortização e não investiga o comportamento dos esquemas em transmissões com vídeos de alta definição.

Tendo como base essa lacuna existente na literatura, o presente trabalho busca avaliar, em igualdade de condições, o nível de segurança e as sobrecargas dos esquemas de assinatura digital e de amortização de assinatura mais relevantes para P2P *streaming* em uma ampla gama de cenários, incluindo transmissão em alta definição. Para isto, é feita uma comparação quantitativa de custos, sobrecargas e nível de segurança durante transmissões convencionais e em alta definição.

O restante do trabalho é organizado como segue. No Capítulo 2, são apresentadas as características dos sistemas de *live streaming* em redes P2P. Na sequência, apresenta-se no Capítulo 3 uma definição para o ataque de poluição de conteúdo em sistemas de *live streaming* a partir de trabalhos encontrados na literatura. No Capítulo 4, são descritas as principais alternativas para autenticar conteúdo nesses sistemas. No Capítulo 5, os esquemas mais relevantes e atuais são modelados em função do nível de segurança e sobrecargas dos mesmos através de um conjunto de equações. A avaliação do comportamento dos esquemas modelados e os resultados sob a perspectiva de transmissões convencionais e em alta definição compõem o Capítulo 6. Por fim, no Capítulo 7, são apresentadas as conclusões obtidas e perspectivas de trabalhos futuros.

2 *LIVE STREAMING*

O presente capítulo busca fornecer a base para um maior entendimento sobre a disseminação de áudio/vídeo ao vivo através da Internet. Primeiramente, apresenta-se informações relativas à codificação de vídeo e como esses dados conseguem ser enviados e reproduzidos. Na sequência, o funcionamento dos sistemas de P2P *Live Streaming* mais conhecidos é discutido em maiores detalhes. Finalmente, discute-se quais ameaças podem causar danos a este tipo de aplicação.

2.1 Codificação de Vídeo

A transmissão de vídeo pela Internet é um problema desafiador no que tange a criação de novos protocolos e aplicações. Isto acontece devido à necessidade de garantir a qualidade de serviço. Este tipo de aplicação requer uma quantidade maior de largura de banda se comparado com aplicações que utilizam outros tipos de mídia como, por exemplo, dados e voz. Além disso, o vídeo, assim como voz, é sensível ao atraso e a variação do mesmo (*jitter*). Sendo assim, todo protocolo de *live streaming* deve se preocupar com as restrições temporais impostas por esse tipo de aplicação e com a taxa de perda de pacotes.

As aplicações de transmissão de vídeo podem ser dividas em duas categorias: vídeo sob demanda (*video on demand*) e vídeo ao vivo (*live streaming*). Na primeira, os usuários podem escolher quando irão assitir um vídeo que foi gerado e armazenado previamente. Já na segunda, os usuários assistem o vídeo que está sendo gerado *"on-the-fly"* e enviado pela rede. O foco do presente trabalho é estudar apenas sistemas P2P de *live streaming*.

Os vídeos digitais são formados por seqüências de fotos estáticas da área filmada (quadros). Essas imagens devem ser obtidas a uma taxa de 25 a 30 quadros por segundo (REDIESS et al., 2006). Caso contrário, o sistema visual humano não é capaz de identificar o movimento na seqüência de imagens estáticas. Considerando que o vídeo capturado não possua nenhuma compressão, seria necessária uma enorme quantidade de memória para que este vídeo fosse armazenado e uma elevada taxa de transmissão para que ele pudesse ser transmitido em tempo real. Isso implica custos muito elevados em termos de armazenamento e transmissão.

Apesar de seqüências de vídeo digitalizadas precisarem de uma grande quantidade de informação para serem representadas, elas possuem, em geral, uma outra importante propriedade intrínseca: apresentam elevado grau de redundância. Isto significa que a maior parte dos dados necessários para representar o vídeo digitalizado apenas se repete (de um quadro para outro ou dentro do mesmo quadro) ou, ainda, parte da informação representada no vídeo simplesmente não é perceptível para o sistema visual humano (altas freqüências, por exemplo). Esta informação redundante pode ser eliminada da seqüência de vídeo sem prejudicar a sua qualidade. O principal objetivo da compressão de vídeo é,

justamente, o desenvolvimento de técnicas que possibilitem a máxima eliminação possível desta informação redundante e, deste modo, consigam representar o vídeo digital com um número de bits muito menor que o original. Por isso, há vários anos, a indústria vem desenvolvendo padrões para realizar a compressão de vídeos, como o MPEG-2 ou H.262 e o H.264 ou MPEG-4 Parte 10 (STANDARDISATION, 2000, 2002).

O padrão H.264 foi desenvolvido com o objetivo principal de dobrar a taxa de compressão de seu predecessor, o padrão MPEG-2. Este objetivo foi conquistado através do uso conjunto de diversas técnicas avançadas e inovadoras de compressão. Porém acarretou em uma maior complexidade computacional, cerca de quatro vezes maior que o MPEG-2 (RICHARDSON, 2003). A taxa de compressão obtida com o H.264 é 300 vezes superior que o MPEG-2 e o mesmo vídeo SDTV de 10 minutos pode ser representado com cerca de 0,06 GB, ao invés dos 18 GB originais.

Recentemente, foram desenvolvidos formatos de alta definição (*HDTV - High Definition Television*) que são caracterizados por vídeos de alta resolução e qualidade, tais como 1280x720 ou 1920x1080 e áudio com seis canais (POYNTON, 2003). Conteúdo de alta definição normalmente é gerado com base no CODEC H.264 e no seu perfil correspondente à alta definição. Maiores informações sobre H.264 podem ser encontradas em Topiwala et al. (2009).

2.2 Sistemas P2P de *Live Streaming*

Uma rede P2P é composta por participantes, também chamados de pares, e suas interconexões. Este tipo de rede é independente da rede física e é implementada dispondo os pares em uma estrutura lógica (também chamada de rede de sobreposição). Sistemas P2P são caracterizados pelo compartilhamento de recursos computacionais (largura de banda, processamento e armazenamento) entre os participantes da rede, sem a necessidade de um servidor centralizado. Como consequência, eles acabam apresentando baixo custo e alto grau de escalabilidade.

No presente trabalho, são estudados apenas os sistemas P2P de *Live Streaming*. Nesse tipo de sistema, a fonte de dados gera o conteúdo *"on-the-fly"*, codifica e transmite aos receptores. O conteúdo gerado é dividido em *chunks* sequenciais e identificados, de forma que cada receptor saiba que *chunks* possui e quais necessita. As topologias mais comuns para a construção da rede de sobreposição são: em árvore ou em malha. Cada uma tem o próprio modo pelo qual os pares encaminham o fluxo de vídeo. Na topologia em árvore, é formada uma árvore na qual a fonte é a raiz e apenas os pares pais encaminham *chunks* para seus filhos. Já na arquitetura em malha, forma-se uma rede distribuída de compartilhamento de *chunks* entre os pares. Como não existe uma estrutura fixa na rede, os pares podem tanto receber quanto enviar *chunks* aos demais participantes da rede.

A topologia em malha apresenta desempenho superior à topologia em árvore (MAGHAREI; REJAIE; GUO, 2007). Sendo assim, ela acabou se popularizando mais que a topologia em árvore. Os sistemas de *live streaming* mais relevantes, CoolStreaming (ZHANG et al., 2005), PPLive (HEI et al., 2007) e AnySee (LIAO et al., 2006), adotam a abordagem de topologia em malha. Os mesmos são descritos a seguir.

O **CoolStreaming** (*Cooperative Overlay Streaming*) é uma implementação do DONet (*Data-driven Overlay Network*). Neste sistema, cada par troca informações sobre dados disponíveis com um conjunto de vizinhos e busca por dados não disponíveis de um ou mais vizinhos através da comparação entre seus *buffer maps* (representação de quais *chunks* o par quer reproduzir no momento). Se houver mais de um vizinho com o *chunk*

desejado, escolhe-se aquele com maior banda passante e que possua maior disponibilidade. A rede de sobreposição não possui uma estrutura fixa ou pré-definida (malha). A entrada dos pares é feita a partir de nós que já façam parte da rede. Quando um par quer entrar no sistema, ele deve contatar a fonte que, por sua vez, escolhe aleatoriamente um nó da rede e envia o identificador do mesmo ao par requisitante. Em seguida, o par entrante requisita ao nó informado pela fonte uma lista de possíveis vizinhos. Além disso, cada nó envia periodicamente mensagens a seus vizinhos para mostrar que está ativo. Desta forma, a lista de vizinhos permanece atualizada no decorrer do tempo.

O **PPLive**, por sua vez, é um dos sistemas de P2P *live streaming* mais difundidos em todo o mundo. Segundo Piatek et al. (2010), ele é utilizado por mais de vinte milhões de usuários atualmente. Ele usa o mesmo conceito de enxames (*swarms*) de pares do sistema de compartilhamento de arquivos BitTorrent. Para que um par entre na rede, ele deve seguir os seguintes passos:

1. obter uma lista de canais de um servidor *tracker*;
2. a partir do canal escolhido, enviar uma série de mensagens requisitando uma lista de pares a outros servidores (usados especificamente para informar a existência de pares da rede P2P);
3. utilizar a lista de pares recebida para sondar possíveis novos vizinhos.

Além disso, para superar possíveis atrasos, são usados dois *buffers* para armazenar os *chunks* recebidos. Segundo Hei et al. (2006), existem duas razões para tal utilização: (1) pré-armazenar dados para combater variações na taxa de download; (2) distribuir eficientemente o conteúdo entre os pares. Um dos *buffers* é gerenciado pelo próprio sistema e o outro pelo aplicativo usado para exibir o vídeo. No entanto, a utilização de dois *buffers* aumenta o atraso inicial da aplicação (*start-up delay*).

Por fim, o sistema **AnySee** também utiliza a arquitetura em malha como estrutura da rede de sobreposição. Porém a idéia é utilizar múltiplas redes de sobreposição possíveis durante a transmissão. Desta forma, os pares podem escolher em quantas redes estarão presentes, o que pode reduzir a latência entre a fonte e seus receptores. Outra característica do AnySee é a utilização de *buffers* menores em relação aos outros sistemas. Isto foi feito como uma medida para diminuir o atraso do início da transmissão.

2.3 Segurança em P2P *Live Streaming*

As aplicações P2P vêm crescendo bastante em popularidade por parte de usuários da Internet. No entanto, devido à necessidade de colaboração entre os participantes da rede P2P, surgem muitas vulnerabilidades (descritas posteriormente nesta seção). Além disso, em aplicações com restrições temporais, como *live streaming*, torna-se impraticável o uso de técnicas tradicionais de segurança (como, por exemplo, as soluções criadas para compartilhamento de arquivos) (BARCELLOS; GASPARY, 2006). Sendo assim, torna-se necessário criar novas técnicas ou adaptações a antigas para tentar resolver os problemas de segurança em sistemas de *live streaming*.

Segundo Moraes et al. (2008), existem basicamente quatro vulnerabilidades que são exploradas por atacantes em sistemas P2P de *live streaming* devido à ausência dos seguintes controles: acesso à rede de sobreposição; comportamento dos pares; incentivo à cooperação entre os pares; autenticidade e integridade dos *chunks*. A seguir, os mesmos são comentados.

Caso não exista algum tipo de controle no acesso à rede de sobreposição, um par malicioso pode interferir no funcionamento de construção da rede sobreposta e no gerenciamento dos participantes da rede. Uma ameaça proveniente dessa vulnerabilidade é o ataque **eclipse**. O nome deste ataque é proveniente do fato de que um ataque bem sucedido é capaz de esconder um segmento de rede do outro segmento. Nesse tipo de ataque, a construção da rede sobreposta é comprometida, mas para que tenha algum efeito considerável ele deve ser realizado em conluio com outros pares (MORAES et al., 2008).

Outra vulnerabilidade é a falta de controle do comportamento dos pares. Quando não existe este tipo de controle, ataques de **negação de serviço** são facilitados já que pares não-maliciosos podem ser sobrecarregados com dados inválidos ou com pedidos excessivos de *chunks* (embora o atacante precise comprometer recursos próprios no ataque). Como consequência, a disponibilidade dos dados pode ser comprometida e a qualidade do vídeo recebido pode se tornar muito ruim.

Além disso, não existindo incentivo à cooperação entre os pares, é possível que pares decidam não contribuir mais com os demais participantes da rede. Um ataque que explora esta vulnerabilidade é o ataque de **omissão** (HARIDASAN; RENESSE, 2008). Como consequência do mesmo, a disponibilidade dos dados pode ficar comprometida. A maior dificuldade em tratar este ataque é provar que um par está praticando o mesmo.

Por fim, a última vulnerabilidade é a ausência de autenticação e integridade dos *chunks* que são gerados e transmitidos originalmente pela fonte de dados. Isto proporciona brechas de segurança necessárias para implantar ataques de **poluição de conteúdo**. No presente trabalho, além de avaliar sistemas P2P de *live streaming* em cenários com qualidades de transmissão distintas, busca-se verificar qual a efetividade dos atuais esquemas de autenticação para combater ataques de poluição de conteúdo. No próximo capítulo, o ataque de poluição de conteúdo e suas implicações são explicados detalhadamente.

3 POLUIÇÃO EM *LIVE STREAMING*

Poluição de conteúdo é um ataque que pode ter um efeito devastador em um sistema P2P de *live streaming* (DHUNGEL et al., 2007). Sendo assim, foram realizados diversos trabalhos que procuram entender esta ameaça e como combatê-la. Neste capítulo, com base em estudos encontrados na literatura, procura-se definir esse tipo de ataque (Seção 3.1) e descrever os principais trabalhos envolvendo o ataque de poluição em *live streaming* (Seção 3.2).

3.1 Definição de Poluição de Conteúdo

Esta seção está organizada da seguinte forma: primeiramente, define-se categorias de participantes da rede P2P. Em seguida, define-se o ataque de poluição e são apresentadas as formas pelas quais podem ser implantados tal ataque. Por fim, é feita uma discussão sobre possíveis motivações para realizar o ataque de poluição de conteúdo em um sistema P2P de *live streaming*.

Neste trabalho, os **participantes da rede P2P** de *live streaming* são considerados como sendo de três tipos:

1. fonte: entidade única que codifica o vídeo e o transmite originalmente aos demais pares. Neste trabalho, parte-se da premissa que a fonte de dados é considerada sempre confiável;

2. receptores: pares altruístas (executam o protocolo de forma correta), interessados em receber/repassar os dados transmitidos;

3. poluidores ou atacantes: pares bizantinos (desviam da especificação do protocolo de forma arbitrária) ou racionais (desviam da especificação do protocolo em busca de benefícios próprios) que buscam distribuir conteúdo poluído.

Poluição de conteúdo é definida de diferentes formas na literatura (DHUNGEL et al., 2007; BORGES; ALMEIDA; CAMPOS, 2008; HARIDASAN; RENESSE, 2008). Segundo Borges et al. (2008), a disseminação de conteúdo poluído ocorre quando um par altera o conteúdo da mídia, degradando assim a qualidade percebida pelos demais pares. Naquele trabalho, os autores não consideram a remoção de dados. Já Haridasan e Van Renesse (2008) definem a poluição de conteúdo como qualquer fabricação ou adulteração dos dados sendo transmitidos no sistema. Ou seja, consideram também a criação de dados poluídos por atacantes.

Em contraste, Dhungel et al. (2007) defendem que este tipo de ataque acontece apenas quando o atacante insere *chunks* poluídos na distribuição de dados da rede P2P, mas não

consideram a alteração de dados gerados pela fonte. Eles afirmam que um atacante pode se conectar a uma sessão de *live streaming*, estabelecer relacionamento com vizinhos e oferecer a eles um grande número de *chunks* do vídeo. Quando os vizinhos requisitam estes *chunks*, o atacante envia dados poluídos. Desta forma, dados poluídos e dados válidos (originados pela fonte) acabam se misturando. É possível que dados poluídos contenham conteúdo diferente do original (áudio ou imagem alterados) ou que estejam comprometidos (corrompidos ou com uma qualidade inferior à original).

Tendo como base os estudos encontrados na literatura, pode-se definir o ataque de poluição de conteúdo em sistemas P2P de *live streaming* como qualquer modificação da mídia (dados originais) que está sendo transmitida originalmente pela fonte para um grupo de receptores. Desta forma, a percepção dos pares receptores pode ser alterada através da inserção de ruído, alteração da imagem, modificação do áudio, etc. Além disso, a disponibilidade dos dados pode ficar comprometida e/ou dados não desejados (poluídos) podem ser recebidos por participantes da rede P2P.

Além de definir o ataque de poluição de conteúdo em sistemas P2P de *live streaming*, deve-se investigar quais as diferentes maneiras pelas quais ele pode ser implantado. Ele pode ser feito das seguintes formas: (1) injeção de *chunks* falsos; (2) modificação do conteúdo de *chunks* válidos; (3) inundação com informações de autenticação falsas. No presente trabalho, a remoção de dados válidos não é considerada poluição de conteúdo, pois embora os receptores possam ser privados do recebimento dos dados, os mesmos não chegam a ser modificados.

A injeção de *chunks* falsos (inserção de dados) acontece quando um atacante distribui randomicamente ou de alguma forma pré-definida *chunks* mal formatados com a intenção de prejudicar os receptores. Como a fonte de dados é considerada confiável e única, o restante dos pares somente pode repassar dados recebidos (jamais criam dados). Caso não exista nenhum mecanismo de autenticação no sistema, os pares continuarão repassando os dados poluídos para seus vizinhos e assim por diante. Desta forma, os dados poluídos são distribuídos rapidamente (com a mesma velocidade dos dados não poluídos) pela rede, o que diminui drasticamente a qualidade de transmissão.

Já na modificação de *chunks*, o atacante precisa receber um *chunk* válido enviado pela fonte, alterar seu conteúdo e retransmitir aos outros receptores. O objetivo do atacante pode ser disponibilizar um conteúdo diferente ao que está sendo proposto a transmissão. Ou seja, os receptores acabam recebendo conteúdo falsificado ou indesejado, o que pode diminuir a atratividade do sistema aos usuários fazendo com que desistam de ver o vídeo em questão.

O objetivo do ataque de inundação com informações de autenticação falsas é exaurir os recursos computacionais de um receptor com múltiplos *chunks* assinados com a própria chave privada do atacante, caso seja utilizado um esquema de assinatura digital. Caso este tipo de ataque seja feito em conluio com outros atacantes, o serviço de transmissão do vídeo será negado a uma porção de usuários, o que pode levá-los a saírem do sistema.

Finalmente, é possível conjeturar sobre **motivações** para atacar sistemas de *live streaming*. Segundo Dhungel et al., este comportamento malicioso pode ser encorajado pela concorrência entre canais de televisão pela audiência. Ou seja, um canal pode tentar poluir a transmissão do outro canal para que usuários desistam de ver o programa em questão devido à degradação da qualidade da imagem, modificação de áudio, inserção de ruído, etc. Outro possível ataque seria a troca do áudio (ou da imagem) visando, por exemplo, a inserção de mensagens terroristas através da substituição de *chunks* legítimos por outros. Da mesma forma, *spammers* poderiam valer-se de instrumentos semelhantes para

divulgar publicidade não autorizada.

3.2 Estudos sobre o Ataque de Poluição em P2P *Streaming*

Nesta seção, são apresentados os trabalhos que buscam estudar o comportamento do ataque de poluição de conteúdo e formas de combatê-lo no contexto de transmissões em redes P2P de *live streaming*. Dhungel et al. (2007) e Yang et al. (2008) tentam entender a poluição de conteúdo e quais as medidas preventivas possíveis. Já Borges, Almeida e Campos (2008) e Jin et al. (2006) criam mecanismos para combater este tipo de ataque através da análise do comportamento dos participantes da rede P2P. A seguir, os referidos trabalhos são comentados em maiores detalhes.

Dhungel et al. (2007) apresentam um estudo sobre o ataque de poluição de conteúdo em sistemas P2P de *live streaming*. Eles conduziram alguns experimentos de coleta de dados no sistema PPLive, que demonstram os efeitos devastadores que um ataque de poluição pode causar, reduzindo em até 74% o número de usuários em comparação com condições normais de uso. Este estudo também aponta quatro possíveis contramedidas para proteger sistemas P2P de *live streaming* da poluição de conteúdo: lista-negra, tráfego criptografado, verificação de *hash* e amortização de assinatura digital. Como conclusão, Dhungel et al. (2007) indicam que a amortização de assinatura digital oferece a melhor proteção contra o ataque de poluição de conteúdo.

Já os autores **Yang et al. (2008)** fazem uma análise formal sobre poluição de conteúdo e discutem as possíveis implicações de tal ataque em sistemas P2P de *live streaming*. Além disso, eles criaram um modelo probabilístico para capturar o progresso da disseminação de conteúdo poluído, que foi implementado no sistema Anysee. Como conseqüência da análise formal e do modelo probabilístico, foram tiradas as seguintes conclusões:

1. o número de poluidores pode aumentar exponencialmente;

2. com apenas 1% de poluidores no sistema, é possível comprometê-lo como um todo em alguns minutos;

3. os principais fatores que influenciam na disseminação de conteúdo poluído é a banda disponível para os poluidores e o grau de vizinhança dos mesmos.

Yang et al. (2008) também examinam diversos mecanismos de combate à poluição de conteúdo e, diferentemente de Dhungel et al. (2007), apontam esquemas de assinatura baseados em funções unidirecionais de *hash* como a solução mais eficiente na identificação de poluidores, principalmente quando utilizados desde o início da transmissão.

Por outro lado, existem trabalhos que visam analisar o comportamento dos pares da rede P2P para tentar identificar os possíveis poluidores. Borges, Almeida e Campos (2008) consideram a grande dificuldade existente na determinação da contaminação de um dado, pois a qualidade percebida pelo usuário final também pode ser afetada por pedidos de retransmissão, atrasos e ocupação indevida da banda. Baseados nisto, eles afirmam que não basta que a aplicação verifique se os dados estão poluídos, é necessário um controle complementar sobre o comportamento dos pares. As principais formas de ter este tipo de controle em sistemas de *live streaming* em redes P2P são: (1) sistemas de reputação; (2) sistemas de lista-negra. Os mesmos são explicados na sequência.

Um sistema de reputação coleta, distribui e agrega avaliações sobre o comportamento dos participantes da rede P2P. Através destas avaliações, os pares descobrem quais participantes podem ser considerados como não confiáveis. **Borges, Almeida e Campos (2008)** estudaram os ataques de poluição de conteúdo e definiram o StRepS (*Streaming Reputation System*): uma extensão do sistema de reputação Scrubber para suportar o seu uso em aplicações de *live streaming* (COSTA et al., 2007). Eles consideram a resistência a ataques de conluio de participantes que estejam tentando aumentar a reputação de pares poluidores. A maneira encontrada por Borges, Almeida e Campos para incentivar a não distribuição de dados poluídos foi através da inflação na penalização de pares poluidores. Desta forma, um par é recompensado linearmente quando repassa um dado válido e é punido exponencialmente quando repassa conteúdo falso.

Os sistemas de lista-negra utilizam uma lista de pares mal comportados para excluir pares maliciosos durante a seleção de vizinhos para a troca de *chunks*. Neste tipo de abordagem, os pares monitoram o comportamento de seus vizinhos. Caso um participante da rede atinja determinada reputação, ele é adicionado à lista de pares maliciosos e a interação com ele não é indicada como segura. Normalmente, este tipo de solução necessita de um controle centralizado sobre a lista de pares maliciosos para evitar que pares não-maliciosos sejam adicionados a lista-negra.

Jin et al. (2006) criaram um sistema centralizado que faz uso de lista-negra para tentar identificar os pares que estão distribuindo dados falsos pela rede de sobreposição em um sistema de *streaming* P2P. Os pares analisam o comportamento dos vizinhos e reportam esses dados ao servidor. A partir desses dados, o servidor centralizado calcula a reputação para cada nó participante da rede de sobreposição. Eles consideraram o problema de computar a reputação na presença de pares mentirosos como um problema de minimização e o resolveram através do algoritmo de Levenberg-Marquardt (MARQUARDT, 1963). O algoritmo provê uma solução numérica para o problema de minimização da soma dos quadrados de múltiplas funções. Para isto, é necessário estimar valores para variáveis não conhecidas. Segundo Jin et al., o algoritmo pode achar a solução rapidamente mesmo quando as estimativas iniciais para as variáveis não conhecidas estiverem longe do resultado ótimo.

Tendo como base os estudos apresentados nesta seção, nota-se que é necessário fazer a detecção do conteúdo poluído para evitar a disseminação massiva desses dados entre os participantes da rede P2P. Isto pode ser feito através da autenticação dos dados pela fonte. A assinatura digital dos dados permite a criação do *digest* (assinatura) para que os receptores possam verificar se foi realmente a fonte que enviou o conteúdo em questão (DHUNGEL et al., 2007). Desta forma, os pares receptores podem descartar dados enviados por pares poluidores ou dados que foram adulterados pelos mesmos. Mesmo os trabalhos que apontam a necessidade de um controle dos pares indicam que é necessário fazer a detecção de conteúdo poluído previamente. Sendo assim, o presente trabalho foca na modelagem e avaliação das propostas de autenticação de dados (Capítulos 5 e 6) como uma solução para o problema de poluição de conteúdo em P2P *live streaming*. No Capítulo 4, é explicado em maiores detalhes o funcionamento dos principais esquemas de autenticação de dados encontrados na literatura.

4 AUTENTICAÇÃO DE FLUXOS DE *STREAMING*

Neste capítulo, são indicadas as principais formas de autenticação de dados encontradas na literatura. Na Seção 4.1, são expostas as abordagens possíveis e as razões pelas quais foram selecionados os esquemas de amortização de assinatura e os esquemas leves de assinatura como foco de estudo do presente trabalho. Os esquemas de amortização mais relevantes são explicados em maiores detalhes na Seção 4.2 e o funcionamento dos esquemas leves de assinatura é apresentado na Seção 4.3.

4.1 Abordagens para a Autenticação de Dados

Caso não seja empregada alguma forma de autenticação de dados, pares maliciosos podem adulterar os *chunks* recebidos antes de repassá-los aos demais pares da rede P2P. Existem na literatura diversas abordagens que podem solucionar este problema: (1) utilização de algoritmos clássicos de criptografia assimétrica; (2) esquemas baseados em MAC; (3) amortização de assinatura ao longo do tempo; e (4) esquemas leves de assinatura. As mesmas são explicadas a seguir.

Os esquemas de assinatura digital baseados em **algoritmos clássicos** de criptografia assimétrica buscam garantir a autenticidade dos dados recebidos através da distribuição de uma chave pública entre os receptores, reservando uma chave privada na fonte. Os mais conhecidos e utilizados até o momento são o RSA, DSA e ECDSA. Embora proporcionem uma forma segura de autenticação de dados através do uso de complexidade matemática, as sobrecargas impostas por este tipo de solução não permitem seu uso direto em sistemas de *live streaming* devido às restrições temporais.

Já os esquemas baseados em **MAC** (*Message Authentication Codes*) (CANETTI et al., 1999) permitem a autenticação de dados entre uma fonte e um receptor a partir de uma chave privada compartilhada entre os mesmos. Por usarem algoritmos de criptografia simétrica, são mais leves. No entanto, esse tipo de solução permite que receptores também possam assinar *chunks*, o que permitiria a um atacante adulterar um *chunk* e assinar o mesmo com a chave compartilhada. Sendo assim, MACs não podem ser utilizados para disseminação de dados em sistemas nos quais os receptores não sejam confiáveis, como por exemplo, sistemas P2P de *live streaming*.

Perrig et al. (2002) propôs o esquema TESLA (*Timed Efficient Stream Loss-tolerant Authentication*) que visa evitar o uso de esquemas de assinaturas mais complexos e que receptores possam assinar dados. Nesse esquema, um MAC autentica um bloco de dados, mas a chave secreta correspondente só é revelada após a distribuição dos dados do bloco. Mesmo sendo uma boa solução, ela não é adequada para *live streaming* devido ao fato de que tanto a fonte quanto os receptores precisam compartilhar uma chave secreta. Para poder usar este tipo de solução, deve-se partir do pressuposto de que um atacante não

pode ter acesso à chave secreta, o que não é verdade em P2P *live streaming*. Caso um atacante tenha acesso à mesma, ele pode gerar um MAC para um *chunk* poluído.

Além das soluções descritas anteriormente, existem estratégias que visam ser mais eficientes que esquemas tradicionais através da diminuição das sobrecargas proporcionadas (atraso, tamanho de assinatura, custo computacional, etc.). Sendo assim, elas podem ser utilizadas por sistemas com requisitos diferenciados (baixa latência, alta largura de banda, etc.), como por exemplo *live streaming* em redes P2P. As duas principais abordagens nesse sentido são os **esquemas de amortização de assinatura** e os **esquemas leves de assinatura**. A primeira consiste na amortização das sobrecargas ao longo do tempo. Para isto, gera-se uma única assinatura para um bloco de *chunks*, ao invés de gerar uma assinatura para cada *chunk*, pois os custos de geração e conferência de assinaturas crescem de forma sub-linear em relação à quantidade de dados. Já a segunda visa reduzir o custo computacional através da utilização de funções unidirecionais de *hash* para a criação e verificação de assinaturas. Neste trabalho, são consideradas para análise estas duas últimas abordagens por serem mais atuais e apropriadas para *live streaming* (ambas são apresentadas nas próximas seções).

4.2 Esquemas de Amortização de Assinatura

Para evitar a manipulação por pares intermediários dos *chunks*, é possível fazer a assinatura de cada *chunk* diretamente com a chave privada da fonte. No entanto, este tipo de solução proporciona altas sobrecargas e custos. Como tentativa de melhoria, foi desenvolvido um esquema bem simples de assinatura no qual deve ser aplicada uma função de *hash* sobre cada *chunk* que será enviado e o resultado deste *hash* deve ser codificado com a chave privada da fonte, gerando assim o respectivo *digest*. O *digest* deve ser enviado juntamente com o *chunk* correspondente. Desta forma, os receptores podem verificar a autenticidade dos dados que estão recebendo através da comparação entre o *hash* sobre os dados e a decodificação do *digest*. A coincidência entre eles indica que os dados recebidos são válidos. Caso contrário, os dados não devem ser repassados ou reproduzidos, pois foram comprometidos ou estão corrompidos (este esquema simples é chamado de Hash Chaining) (GENNARO; ROHATGI, 1997).

Em busca de eficiência, surgiram diversos trabalhos que buscam amortizar os custos da assinatura ao longo do tempo, incluindo os baseados em **árvores** (WONG; LAM, 1999; HABIB et al., 2005), em **grafos** (PERRIG; CANETTI; WATSON, 2000; MINER; STADDON, 2001), em **códigos de correção de erros (ECC)** (LYSYANSKAYA; TAMASSIA; TRIANDOPOULOS, 2004, 2010) e em **códigos de correção antecipada de erros (FEC)** (para resistir à perda de *chunks*) (PARK; CHONG; SIEGEL, 2003; PANNETRAT; MOLVA, 2003; PARK; CHO, 2004). Logo em seguida, é descrito com maiores detalhes o funcionamento dos esquemas de amortização de assinatura digital de maior relevância científica: (1) Tree Chaining; (2) Graph Chaining; (3) Butterfly Graph Chaining; (4) Linear Digests; (5) AuthECC; (6) SAIDA.

4.2.1 Tree Chaining

Wong e Lam (1999) propuseram um esquema baseado em árvore no qual é utilizado o conceito de blocos de *chunks* (uma árvore binária por bloco). As folhas da árvore contêm os *hashes* dos *chunks* do bloco. Como se trata de uma árvore binária, a cada dois nós folha é feito o *hash* da concatenação de seus *hashes* para formar os nós pais. Este agrupamento é feito das folhas em direção à raiz, formando assim a árvore de assinatura do bloco.

A assinatura de um *chunk* consiste no *hash* raiz (assinatura do bloco), posição do *chunk* no bloco e seus irmãos (no caminho até a raiz) na árvore necessários para verificar a assinatura. Para verificar a autenticidade de um *chunk*, o receptor precisa checar o caminho do nó correspondente até a raiz da árvore. Na Figura 4.1 estão exemplificados quais os valores de *hash* necessários para fazer a verificação do terceiro *chunk* (a árvore apresentada na Figura 4.1 serve puramente para exemplificar o funcionamento do esquema).

Primeiramente, deve-se computar o próprio *hash* e o de seus sucessores na árvore para que seja possível calcular o valor de *hash* da raiz (H(C18)). Ou seja, deve-se calcular H(C3), H(C4), H(C12) e H(C58). Em seguida, o valor H(C18) é comparado com o valor da raiz (decodificado com a chave pública da fonte) contido no *chunk* recebido. Caso estes valores sejam iguais, o terceiro *chunk* pode ser considerado autêntico.

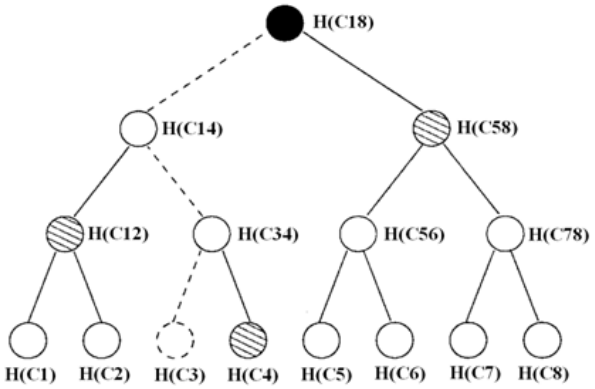


Figura 4.1: Exemplo do esquema de autenticação Tree Chaining com um bloco de oito *chunks*

Em comparação com o esquema Hash Chaining, o Tree Chaining é superior em até duas ordens de grandeza no que tange o processo de autenticação pois amortiza-se o custo de uma assinatura/verificação sobre *n chunks* (WONG; LAM, 1999). No entanto, segundo Park, Chong e Siegel (2002), o esquema baseado em árvore sofre de um problema de sobrecarga de comunicação (largura de banda). Neste esquema, cada *chunk* tem que carregar 200 bytes extras, considerando uma assinatura RSA de 1024 *bits* e um bloco com 16 *chunks* cada.

4.2.2 Graph Chaining

Graph Chaining corresponde a um tipo de solução que utiliza um grafo por bloco de *chunks*, ou seja, um grafo é criado para cada bloco de *chunks* (PERRIG; CANETTI; WATSON, 2000; MINER; STADDON, 2001). Para cada grafo, existirá um pacote S (assinatura do bloco) que é assinado com um esquema clássico de assinatura de chave pública (por exemplo, RSA). Um *chunk* C_i só pode ser autenticado caso exista um caminho entre ele e S no grafo.

A Figura 4.2 apresenta um exemplo do esquema de autenticação baseado em grafos no qual cada bloco contém quatro *chunks* ($C1$, $C2$, $C3$ e $C4$). Como pode ser visto no exemplo, são criados quatro *chunks* e as ligações entre eles (este valor não é fixo, é usado meramente com intuito de exemplificar o funcionamento do esquema estudado). Quando é feita a ligação entre o *chunk* $C4$ e o *chunk* $C3$, concatena-se o valor de *hash* de $C4$ a $C3$. É feito um processo similar até o término dos *chunks* do bloco, criando assim a assinatura

do mesmo (pacote S). O pacote S é assinado e torna-se responsável por armazenar a informação que autenticará todos os *chunks* do bloco.

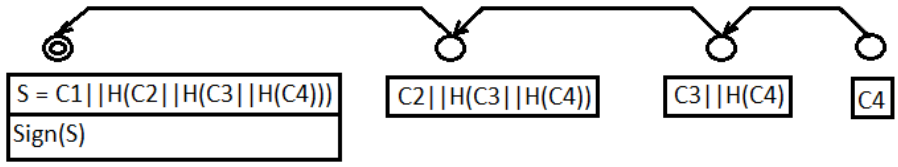


Figura 4.2: Exemplo do esquema Graph-based com um bloco de quatro *chunks*

Para evitar perda de *chunks* e possíveis inconsistências no grafo, Miner e Staddon utilizam probabilidade para priorizar alguns *chunks* através de redundância. Além disso, é possível criar mais arestas entre os nós para aumentar a redundância dos valores de *hash* já que cada ligação no grafo indica que o apontado armazena o valor de *hash* do outro nó. O pacote contendo a assinatura do bloco (S) provavelmente terá a prioridade mais alta para garantir o seu recebimento.

4.2.3 Buterfly Graph Chaining

O esquema de amortização Butterfly Graph Chaining se baseia na construção de um grafo acíclico dirigido, tendo um pacote S contendo a assinatura, n *chunks* por bloco e Q linhas no grafo (ZHANG; SUN; WONG, 2005; ZHANG et al., 2007, 2009). Os n *chunks* são divididos em $(log_2 Q + 1)$ colunas, onde cada coluna contém Q *chunks*. Um *chunk* é denotado como $C(k, j)$, onde $k \in 0, 1, \ldots, log_2 Q$, indicando a coluna e $j \in 0, 1, \ldots, Q - 1$, indicando o *chunk* da coluna.

No presente trabalho, utilizamos o conceito de grafo *Butterfly* generalizado (GBG - *Generalized Butterfly Graph*) (ZHANG et al., 2007) pois este tipo de grafo é mais flexível. Isto acontece porque o número de linhas (Q) é um parâmetro de entrada do sistema e é independente do número de *chunks* por bloco (n). Como consequência, o tamanho da assinatura não aumenta proporcionalmente ao aumento do número de *chunks* por bloco. O número de colunas é determinado por n/Q. Caso n seja igual a $Q \times (log_2 Q + 1)$, o grafo generalizado corresponde a um grafo *butterfly* comum.

A criação das arestas do grafo é feita a partir do algoritmo round-robin. Além disso, existe uma aresta entre todos os *chunks* da coluna zero e o pacote S (contém a assinatura). Nesse tipo de grafo, cada aresta $(C(k_1, j_1), C(k_2, j_2))$ é criada através da concatenação do *hash* do *chunk* $C(k_1, j_1)$ com o do *chunk* $C(k_2, j_2)$. Um exemplo de um grafo *butterfly* pode ser observado na Figura 4.3, com quatro colunas e oito *chunks* em cada coluna (este valor não é fixo). O pacote S contém a assinatura e os *hashes* de todos os *chunks* da coluna zero. A probabilidade de verificação decai linearmente da primeira coluna até a última.

Embora o esquema de amortização Butterfly Graph Chaining apresente características interessantes, ele também apresenta algumas limitações, como pode ser observado na sequência:

- o pacote S, contendo a assinatura, deve ser enviado diversas vezes (Rep determina o número de vezes que a assinatura será replicada) para evitar perda de *chunks* pois parte-se da premissa de que eles são transmitidos por um meio não-confiável;
- para que a assinatura de um *chunk* não fique muito grande (não cabendo em uma MTU), deve-se ajustar o número de linhas do grafo (Q);

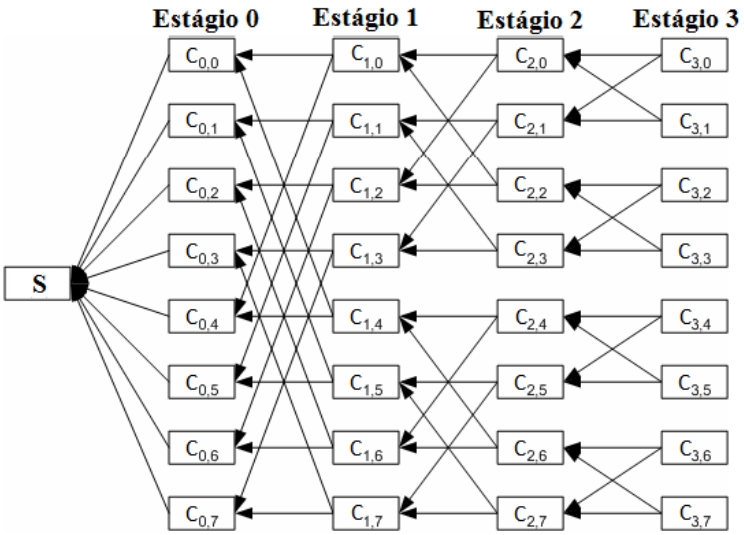


Figura 4.3: Exemplo de Grafo Butterfly

- Q (número de linhas do grafo) e n (número de *chunks* por bloco) são enviados juntamente com a assinatura;
- todos os *chunks* de uma coluna tem a mesma probabilidade de serem verificados com sucesso.

4.2.4 Linear Digests

Uma das propostas mais recentes de amortização é o esquema Linear Digests (Haridasan and van Renesse 2008). Ele é utilizado no SecureStream, uma aplicação de *multicast* na camada de aplicação para transmissão de *live streaming*. No referido trabalho, Haridasan e Van Renesse fazem uma comparação entre alguns esquemas mais antigos de amortização de assinatura digital com base em simulações. Resultados preliminares apresentados pelos autores dão indícios de eficiência. Por essa razão, foi um dos esquemas adotados para avaliação no presente trabalho. No entanto, ainda é necessário fazer uma comparação analítica com os esquemas mais recentes de amortização.

Na Figura 4.4, apresenta-se como é realizada a assinatura pela fonte através do esquema Linear Digests para blocos com *n chunks*. Os *n chunks* do bloco precisam estar disponíveis para que seja possível criar a assinatura do respectivo bloco. Primeiramente, é necessário fazer o *hash* de cada *chunk* e, posteriormente, a concatenação dos mesmos. O próximo passo é a codificação (função *Sign()* na figura) do resultado da concatenação com a chave privada (SK - *Secret Key*) da fonte, resultando assim no *digest* S que deve ser enviado em separado, previamente ao envio dos *n chunks*.

Para fazer a verificação da autenticidade dos *chunks* de um bloco, é necessário que os receptores recebam o pacote S. A Figura 4.5 demonstra o que um receptor deve fazer quando receber um *chunk* (C_i). Ele deve verificar a assinatura para decidir se o *chunk* C_i é válido. Para isto, deve-se fazer o *hash* do *chunk* C_i recebido e fazer a decodificação (função *Verify()* na figura) do *digest* (S) previamente recebido com a chave pública (PK - *Public Key*) da fonte. O valor de *hash* de C_i (chamado de Y na figura) deve ser extraído do resultado da decodificação de S para ser comparado com Z (valor de *hash* do *chunk* C_i recebido). Caso sejam iguais, o *chunk* C_i é autêntico e pode ser reproduzido. Caso contrário, o mesmo deve ser descartado.

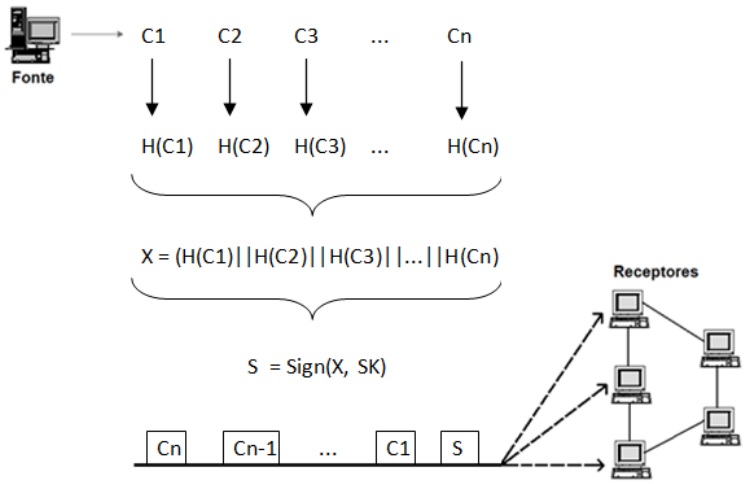


Figura 4.4: Assinatura dos *chunks* baseado no esquema Linear Digests

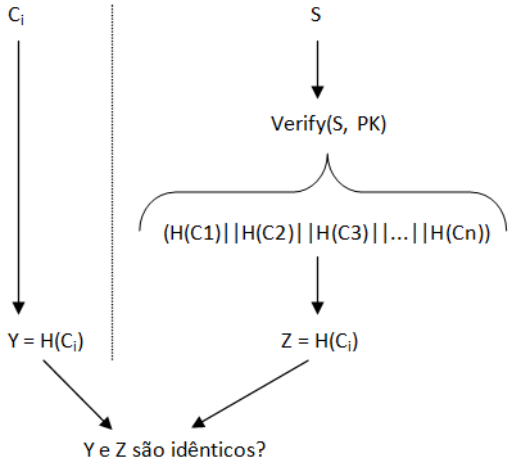


Figura 4.5: Verificação da assinatura pelos receptores baseado no esquema Linear Digests

O esquema Linear Digests é bem simples e eficiente, mas incorre na utilização de *buffer* na fonte e em atrasos adicionais caso a assinatura não tenha sido recebida porque deve ser feita uma requisição para novo envio da mesma, o que pode ser um problema para um sistema de *streaming* devido às suas restrições temporais.

4.2.5 AuthECC

Lysyanskaya; Tamassia e Triandopoulos propuseram um esquema de amortização de assinatura que utiliza funções de *hash* e o código de correção de erro Reed-Solomon (REED; SOLOMON, 1960). O funcionamento do AuthECC pode ser descrito através dos seguintes passos:

1. inicialmente, a fonte de dados precisa fazer o *hash* de cada um dos *chunks* de um bloco e assinar a concatenação desses *hashes*;

2. os *hashes* computados e a assinatura gerada no passo 1 devem ser agrupados e corrigidos utilizando o código de correção de erro Reed-Solomon com a seguinte taxa de correção: $\sigma = \gamma^2/((1+\epsilon)\times\phi)$, sendo que γ corresponde a taxa de sobrevivência, ϵ é o índice de tolerância do decodificador e ϕ diz respeito a taxa de inundação do Reed-Solomon (note $\sigma < 1$ desde que $0 < \gamma \leq 1$, $0 < \epsilon < 1$ e $\phi \geq 1$). Caso um bloco de *n chunks* seja enviado, o receptor deve obter pelo menos $\gamma \times n$ *chunks* e um atacante pode inundar a rede com até $\phi \times n$ *chunks*;

3. na sequência, a fonte envia os *n chunk* juntamente com as *n* informações de correção de erro geradas no segundo passo através do codificador Reed-Solomon. Caso o valor σ seja mantido abaixo de 1, são incluídos $1/\sigma$ valores de *hash* em cada *chunk*;

4. o receptor então pode reconstruir a assinatura criada pela fonte utilizando apenas $\gamma \times n$ *chunks* válidos, tendo recebido *m chunks* ($\gamma \times n \leq m \leq \phi \times n$);

5. através da assinatura reconstruída, o receptor pode verificar a autenticidade dos *n chunks* do bloco.

Embora o esquema de amortização AuthECC (LYSYANSKAYA; TAMASSIA; TRIANDOPOULOS, 2004) tenha tido uma atualização recente em seu trabalho (LYSYANSKAYA; TAMASSIA; TRIANDOPOULOS, 2010), as avaliações preliminares dele não foram muito encorajadoras. Segundo os próprios autores, o desempenho (sobrecarga de comunicação e custo computacional) de tal esquema (baseado em códigos de correção de erro ou ECC) é inferior ao de esquemas baseados em códigos de correção antecipada de erros (FECs), como por exemplo SAIDA. Segundo Dhungel et al. (2007), os atrasos proporcionados pelo esquema AuthECC são ligeiramente maiores do que o esquema Tree chaining (caso seja utilizada uma árvore Merkle). Por estas razões apontadas, o mesmo não será considerado para análise.

4.2.6 SAIDA

Um possível problema proveniente do envio de dados em um meio não-confiável pode ser a perda de *chunks*. SAIDA (PARK; CHONG; SIEGEL, 2003) é um esquema de amortização de assinatura digital que visa prover autenticação mesmo na presença de perdas de *chunks* através da técnica de FEC (*Forward Error Correction*) chamada IDA (*Information Dispersal Algorithm*) (RABIN, 1989). Nesta solução, as informações de autenticação de um bloco de *chunks* são geradas e então submetidas ao algoritmo de

FEC (chamado IDA), codificando-as com certa redundância e dispersando o resultado ao longo dos *n chunks* do bloco. dessa forma, mesmo que um número de *chunks* seja perdido, os *chunks* restantes do bloco ainda podem ser verificados. O funcionamento do SAIDA (*Signature Amortization using IDA*) para autenticar blocos com *n chunks* pode ser descrito pelos seguintes passos:

1. faz o *hash* de cada um dos *n chunks* do bloco gerado pela fonte;
2. concatena os *hashes* dos *chunks* do passo (1) e assina digitalmente com sua chave privada esta concatenação, resultando em um dado de tamanho W;
3. para que o esquema funcione mesmo na presença de perda de *chunks*, o algoritmo IDA processa a assinatura, introduzindo redundância e dividindo a assinatura em X pedaços de tamanho W/m, sendo $X/m \approx 1$;
4. para reconstruir a assinatura, é necessário apenas receber m pedaços (assumindo que até $(X - m)$ pedaços podem ser perdidos durante a transmissão, sendo $m \leq X$).

O esquema de amortização de assinatura **cSAIDA** é um aprimoramento do esquema SAIDA (PANNETRAT; MOLVA, 2003). Existem algumas diferenças pontuais entre eles. Na sequência, elas são apresentadas e suas consequências são discutidas. No cSAIDA, os valores de *hash* são obtidos a partir dos *chunks* recebidos. Enquanto isto, no SAIDA, apenas utiliza-se a assinatura enviada nas mensagens para recuperar os valores de *hash*.

Além disso, o cSAIDA reduz a sobrecarga de comunicação em relação ao SAIDA. No esquema SAIDA, a concatenação dos *hashes* dos *chunks* (X) e a assinatura são codificadas com FECs (algoritmo IDA) e distribuídas junstamente aos *n chunks* do bloco. No entanto, uma fração considerável destes *hashes* pode ser computado a partir dos *chunks* recebidos. Sendo assim, não é necessário o envio de todo X. Para alcançar este objetivo, no cSAIDA, é utilizado dois códigos de correção antecipada de erro, enquanto que no SAIDA utiliza-se apenas um.

O funcionamento da criação das informações de autenticação (assinatura) no cSAIDA está representado na Figura 4.6. Primeiramente, deve-se computar os *hashes* de cada um dos *chunks* do bloco ($C1$, $C2$, ..., C_n), resultando no vetor $X = H(C1), H(C2), ..., H(C_n)$. Em seguida, utiliza-se o primeiro código de correção antecipada de erro, tendo como entrada o vetor X e a taxa máxima estimada de perda de *chunks* por bloco (ρ), gerando $n \times \rho + n$ *chunks* de código da forma descrita pela Equação 4.1. O símbolo $\overline{X}$ representa os *chunks* de redudância criados. Para cada n chunks, são gerados $n + \rho \times n$ *chunks* de redundância através da função geradora de códigos de correção antecipada de erros.

$$\{X, \overline{X}\} \leftarrow C_{(n,[\rho \times n])}(X) \tag{4.1}$$

Na sequência, deve-se assinar o conteúdo de X, gerando a assinatura S, e concatenar a $\overline{X}$. A saída desta operação deve ser a entrada para a segunda operação de código de correção antecipada de erro. A segunda operação é computada a partir da Equação 4.2, onde $Y = \overline{X}||S$.

$$\{Y, \overline{Y}\} \leftarrow C_{([n \times (1-\rho)],[\rho \times n])}(\overline{X}||S) \tag{4.2}$$

A assinatura é gerada então através da concatenação entre S, $\overline{X}$ e $\overline{Y}$, usa-se o resultado da concatenação como um vetor de evidências $\tau_1, \tau_2, \ldots, \tau_n$. A assinatura é enviada juntamente aos *chunks* do bloco. Segundo Pannetrat e Molva (2003), desde que a taxa de perda

de *chunks* seja menor ou igual que ρ, qualquer subconjunto de pelo menos $n\times(1-\rho)$ *chunks* do bloco pode ser autenticado utilizando qualquer subconjunto de pelo menos $n \times (1-\rho)$ das evidências recebidas.

Para verificar a autenticidade de um *chunk* C_i, o par receptor deve possuir os *hashes* do bloco e a assinatura do bloco. Caso a verificação da assinatura seja válida e o *hash* de C_i seja igual a $H(C_i)$, o *chunk* C_i é considerado autêntico.

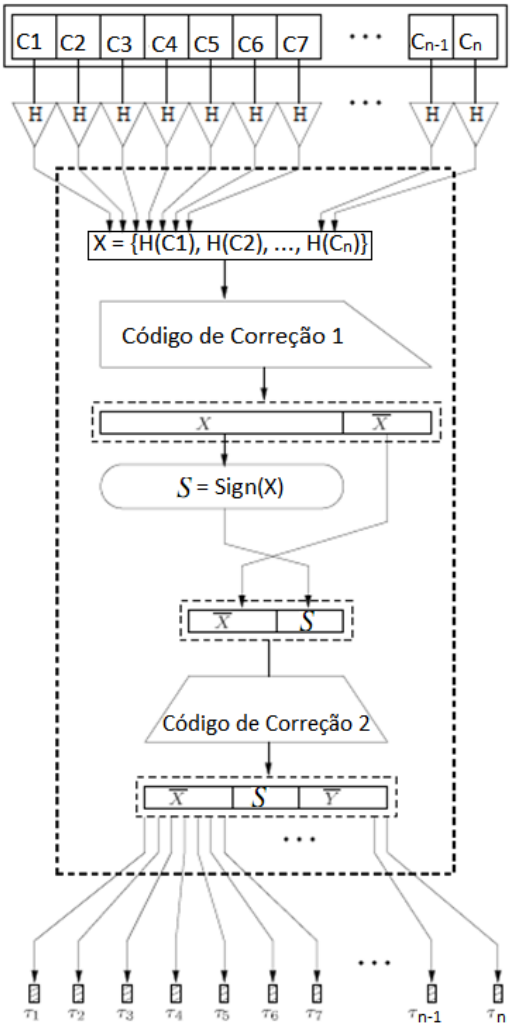


Figura 4.6: Geração de evidências no cSAIDA

Uma segunda proposta de melhoria do esquema SAIDA é apresentado por Park e Cho (2004) e chama-se **eSAIDA**. No eSAIDA, um *hash* é designado para cada par de *chunks* adjacentes, ao invés de apenas um por *chunk* como no SAIDA. Isto faz com que as sobrecargas sejam reduzidas, mas também obriga o recebimento dos dois *chunks* irmãos para que os mesmos possam ser autenticados. A fração de *chunks* contendo o *hash* de seu *chunk* adjacente é parametrizado pela variável de entrada $s(0 \leq s < 1)$. Esta variável define o *tradeoff* entre a taxa de sucesso na verificação de autenticidade e a sobrecarga de comunicação.

4.3 Esquemas Leves de Assinatura Digital

Os esquemas apresentados na seção anterior amortizam a sobrecarga de funções "convencionais"de assinatura digital. Uma abordagem distinta, explorada nesta seção, é o uso de funções de *hash* de menor custo do que as funções convencionais. A segurança nessa abordagem baseia-se no fato de que as funções de *hash*, utilizadas para criar as assinaturas e a chave pública, são unidirecionais e não podem ser revertidas por possíveis atacantes. No decorrer desta seção, a assinatura de um *chunk* será gerada com base na composição de um conjunto de evidências selecionadas a partir da cadeia de *hashes*.

No decorrer desta seção, são descritos os esquemas leves de assinatura mais relevantes encontrados na literatura: (1) **BiBa** (*Bins and Balls Signature*) (Perrig 2001); (2) **HORS** (*Hash to Obtain Random Subset*) (Reyzin and Reyzin 2002); (3) **PEAC** (Cirulli and Di Pietro 2008); (4) **PARM** (*Pollution-Attack Resistant Multicast Authentication*) (Lin et al. 2006); (5) **PARM2** (Lin et al. 2008); e (6) **ALPS** (*Authenticating Live Peer-to-Peer Streams*) (Meier and Wattenhofer 2008).

4.3.1 BiBa

BiBa (*bins and balls signature*) é um protocolo de autenticação de *broadcast* que utiliza um esquema de assinatura digital de mesmo nome para fazer a verificação da origem de uma seqüência de *chunks* (PERRIG, 2001). Ele se baseia na tentativa de encontrar colisões para criar assinaturas. Uma colisão corresponde a encontrar duas evidências (ou *balls*) que correspondam ao mesmo valor (ou *bin*) através de uma função randômica de *hash*. Os termos *bins* e *balls* são provenientes do paradoxo do aniversário da teoria das probabilidades.

Perrig (2001) utiliza dois geradores de números pseudo-randômicos (Goldreich et al. 1986) e uma função *hash* da família de funções do modelo *random oracle* (Bellare and Rogaway 1993) para gerar os valores que são utilizados na criação das evidências (também chamadas de SEALs no referido trabalho, *SElf Authenticating vaLues*). O funcionamento do esquema de assinatura BiBa pode ser divido em três fases:

1. criação das evidências pela fonte;
2. busca por K colisões (criação da assinatura);
3. verificação da assinatura no receptor.

Primeiramente, devem ser criadas t evidências $(s_1, s_2, ..., s_t)$ através da função geradora de números pseudo-randômicos $F : \{0,1\}^{m2} \times \{0,1\}^{m1} \rightarrow \{0,1\}^{m2}$, sendo $m1$ e $m2$ o tamanho em *bits*. Para cada evidência s_i, a chave pública é $f_{s_i} = F_{s_i}(0)$, sendo que $F_{s_i}(.)$ representa a função geradora de número pseudo-randômicos para a evidência de índice i. Sendo assim, um receptor pode verificar a autenticidade de s_i, verificando se $F_{s_i}(0) = f_{s_i}$. Obviamente, este processo é feito para todas as evidências criadas.

Para assinar um *chunk* C, a fonte deve encontrar K colisões entre as t evidências. Para isto, a fonte deve computar o *hash* $h = H(C, z)$, sendo z um contador que é incrementado quando uma colisão não é encontrada. Em seguida, deve ser computada uma instância de uma família de funções randômicas de *hash* $G : \{0,1\}^{m2} \rightarrow [0, nb-1]$, para todas as evidências $(S_1, S_2, ..., S_t)$. O valor nb é o tamanho da função G, ou seja, o número de *bins*.

Na sequência, é necessário fazer a seleção das evidências, criando assim a assinatura. O BiBa apresenta três maneiras para esta etapa: (1) forma básica, através de uma

dupla colisão; (2) múltiplas colisões duplas; e (3) colisões múltiplas. A seguir, elas são explicadas.

Na forma mais básica, a fonte de dados precisa encontrar uma dupla colisão: $G_h(S_i) = G_h(S_j), com S_i \neq S_j$. O par $< S_i, S_j >$ forma assim a assinatura. Como pode ser observado no exemplo da Figura 4.7, é encontrada uma colisão entre S_3 e S_4, formando assim a assinatura $< S_3, S_4 >$.

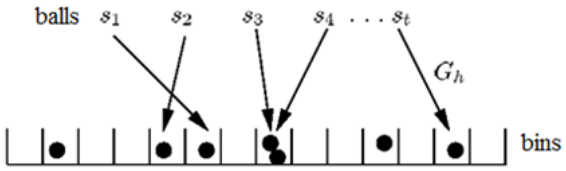


Figura 4.7: Bins and Balls (BiBa)

Pretendendo proporcionar uma maior segurança na fase de criação da assinatura, é possível fazer uma extensão do BiBa, buscando enontrar múltiplas colisões duplas ao invés de apenas uma colisão dupla. Neste caso, a assinatura deve ser composta por z pares (lembrando que z é o contador de segurança explicado previamente) de evidências: $< S_{a_1}, S_{b_1} >, ..., < S_{a_z}, S_{b_z} >$, sendo que todas devem ser distintas ($S_1 \neq S_2 \neq ... \neq S_k$) e $G_h(S_{a_i}) = G_h(S_{b_i})(1 \leq i \leq z)$.

Por fim, a última alternativa para gerar as assinaturas seria encontrar colisões múltiplas. Para isto, deve-se encontrar K colisões entre todas as evidências: $G_h(S_1) = G_h(S_2) = ... = G_h(S_k)$, sendo $S_1 \neq S_2 \neq ... \neq S_K$. Desta forma, as evidências $< S_1, ..., S_K >$ formam a assinatura do *chunk* C. Após ter criado a assinatura, esta deve ser enviada juntamente ao *chunk* C e o contador z.

A forma como é feita a verificação da assinatura por parte do receptor irá depender da forma como foi realizada a geração da assinatura por parte da fonte de dados. Caso tenha sido utilizada a forma mais simples e menos segura de seleção de evidências (busca por uma dupla-colisão), o receptor irá receber o *chunk* C e a assinatura $< S_i, S_j, z >$. A partir deste momento, ele precisa fazer o cômputo $h = H(C, z)$ e verificar se cada evidência recebida é autêntica e distinta, ou seja, se $S_i \neq S_j$ e $G_h(S_i) = G_h(S_j)$. Caso seja utilizada a alternativa mais segura na geração de assinatura (colisões múltiplas), o receptor irá receber o *chunk* C e a assinatura $< S_1, ..., S_K, z >$. Para verificar a autenticidade das evidências, ele deve computar $h = H(C, z)$, verificar se $S_1 \neq S_2 \neq ... \neq S_K$ e se $G_x(S_1) = G_x(S_2) = ... = G_x(S_K)$.

À medida que as evidências são reveladas aos receptores, aumenta a probabilidade de um par malicioso criar uma assinatura falsa. Porém se as chaves forem atualizadas periodicamente, esta limitação deixa de existir já que as evidências comprometidas (utilizadas em excesso) são substituídas. O protocolo BiBa utiliza cadeias de *hash* para fazer a renovação das evidências. Para isto, o tempo é divido em períodos de mesma duração e a cadeia de *hash* é formada através de funções pseudo-randômicas. Conforme o tempo passa, apenas as evidências do período ativo são consideradas como válidas.

O esquema de assinatura BiBa tem como metas proporcionar assinaturas menores e um processo de verificação mais rápido que esquemas propostos anteriormente. No entanto, as chaves públicas e o tempo de geração da assinatura do BiBa são maiores que de esquemas criados previamente.

4.3.2 HORS

L. Reyzin e N. Reyzin (2002) definem um esquema de assinatura que pode ser visto como uma tentativa de aperfeiçoamento do esquema de assinatura BiBa. Eles visaram diminuir o tempo de assinatura do BiBa. Além disso, o esquema HORS (*Hash to Obtain Random Subset*) acabou diminuindo ligeiramente o tamanho tanto das chaves quanto das assinaturas, se comparados com os tamanhos provenientes do esquema BiBa.

HORS parte do princípio da utilização de uma função (chamada na presente dissertação de HRS) cuja segurança baseia-se na dificuldade de encontrar dois *chunks* (C_1 e C_2), sendo $HRS(C_1) \subseteq HRS(C_2)$. Caso a fonte gere N assinaturas, deve ser computacionalmente impossível encontrar $N + 1$ *chunks* $(C_1, C_2, ..., C_{n+1})$ nas quais seja mantida a seguinte relação: $HRS(C_{n+1}) \subseteq HRS(C_1) \cup HRS(C_2)... \cup HRS(C_n)$.

A função HRS do HORS deve ser construída através de uma função *hash* (por exemplo, SHA-1) seguindo os passos a seguir:

1. dividir a saída do *hash* do *chunk* C em segmentos de bits de comprimento $logK$ cada, onde K corresponde ao número de evidências por assinatura (uma assinatura é representada por um conjunto de evidências);

2. interpretar cada segmento criado como um inteiro na forma binária;

3. combinar estes inteiros para formar um vetor de no máximo K posições.

HORS necessita de uma operação de *hash* para assinar. Para verificar a assinatura, por sua vez, é necessário um *hash* e Y usos da função HRS. O funcionamento do esquema de assinatura pode ser dividido em três fases:

1. geração das chaves privada e pública pela fonte de dados;

2. criação das evidências (constituindo assim a assinatura) pela fonte de dados;

3. verificação da assinatura pelo receptor.

Primeiramente, são criados k valores randômicos $(s_1, s_2, ..., s_k)$. Cada valor criado é utilizado como parâmetro na função de *hash* $(v_i = H(s_i))$, resultando em k valores como saída. Cria-se um par de chaves, pública e privada, com base nos k valores randômicos e seus k *hashes* correspondentes. Mais precisamente, a chave privada SK é gerada pela concatenação do valor de k com os k valores randômicos $(k, s1, s2, ..., sk)$. A chave pública, por sua vez, corresponde à concatenação do valor de k com os k *hashes* $(k, v1, v2, ..., vk)$.

A criação de evidências (formando assim a assinatura) de um *chunk* C é feita utilizando a chave privada (SK) criada anteriormente. Deve-se fazer o *hash* do *chunk* e o resultado desta operação deve ser subdividido em k segmentos $(h_1, h_2, ..., h_k)$ de tamanho log_2X *bits* cada. Em seguida, cada h_j é interpretado como um inteiro i_j, sendo $1 \leq j \leq k$. Isto resultará na assinatura do *chunk* C, representado pelo conjunto de k evidências: $(s_{i1}, s_{i2}, ..., s_{ik})$.

Por fim, para verificar a autenticidade de um *chunk* C, com assinatura $(s_1, s_2, ..., s_k)$, deve-se fazer o *hash* da *chunk* C e subdividir o resultado em k segmentos $(h_1, h_2, ..., h_k)$ de log_2K *bits* cada. A partir deste momento, cada segmento é interpretado como um inteiro (da mesma maneira que é feito na assinatura). Se, para cada $j(1 \leq j \leq K)$, o resultado de $H(s_j)$ for igual a v_j (valor da chave pública), o *chunk* pode ser considerado como autêntico. Caso contrário, deve ser descartado.

O esquema HORS apresenta tempos de verificação e assinatura menores que o esquema de assinatura BiBa. No entanto, HORS mantém o uso de chaves públicas grandes (na ordem de megabytes) e alta sobrecarga na fonte devido à geração de assinaturas. Tanto HORS quanto BiBa necessitam de um número muito grande de cadeias de *hash*, o que é indesejável tanto para a distribuição da chave pública quanto para a manutenção das evidências.

4.3.3 PEAC

PEAC é um protocolo probabilístico de autenticação baseado em funções unidirecionais de *hash* para *broadcast* em redes sem fio, comunicação via satélite ou IP Multicast (CIRULLI; DI PIETRO, 2008). O funcionamento do PEAC é dividido em quatro passos:

1. inicialização pela fonte;
2. seleção das evidências por *chunk* pela fonte;
3. verificação da autenticidade das evidências;
4. renovação temporal de chaves.

Para a criação da chave pública (PK) e da chave privada (representada por q vetores), é necessário criar k números randômicos ($R_1, R_2, R_3, ..., R_k$), o que constituirá o primeiro vetor de chave privada (SK_0). Em seguida, é criada uma cadeia de *hashes* a partir destes valores através do *hash* do valor anterior, como pode ser observado na Figura 4.8. Após feitos q *hashes*, estão construídos os $q+1$ vetores de chave privada ($SK_0, SK_1, SK_2, ..., SK_q$) e o vetor de chave pública (PK) que é proveniente do *hash* dos valores do último vetor de chave privada (SK_q).

O tempo é divido em q fatias temporais de igual duração e apenas um vetor de chave privada será considerado válido por fatia de tempo. Mais precisamente, a cadeia de *hashes* é usada do fim para o início a medida que o tempo avança. Desta forma, não é possível descobrir os próximos valores de chave privada porque a função de *hash* utilizada para criar os vetores de chave privada é unidirecional. A partir de um valor do vetor de evidências de tempo t_q não é possível descobrir nenhum valor do vetor de evidências de tempo t_{q+1}. Cirulli e Di Pietro (2008) afirmam que o vetor de chave pública (PK) deve ser assinado digitalmente e sua distribuição deve ser feita em um canal seguro para todos os receptores.

No passo seleção das evidências, a fonte de dados precisa utilizar o vetor SK_i (sendo i o índice da fatia temporal) para selecionar os elementos que serão utilizados pelos receptores para verificar a autenticidade do *chunk* recebido. O conjunto de evidências consiste em p elementos ($E_0, E_1, E_2, ..., E_{p-1}$) do vetor SK. Juntamente ao *chunk* (C), é concatenado o *timestamp* (T), que também é assinado digitalmente. Este valor pode ser utilizado pelos receptores para verificar se o *chunk* recebido foi criado na fatia de tempo correta. Para fazer esta seleção das evidências para um chunk ($C||T$) a ser enviado na fatia de tempo t_j, a fonte utiliza uma tabela para manter um histórico de quais elementos do vetor $SK_{(q-1)-j}$ foram utilizados até o momento. A fonte zera um contador z e faz, por p vezes, o seguinte computo $i_0 = hash(C||z) \times (mod(k))$, onde k é o tamanho do vetor $SK_{(q-1)-j}$, sempre incrementando z antes de cada computo (formando p contadores z).

Desta forma, cria-se o vetor $i_1, i_2, ..., i_p$ que são os índices dos elementos (colunas) selecionados do vetor $SK_{(q-1)-j}$ para serem enviados junto ao *chunk*. É importante ressaltar que colisões podem acontecer neste processo. Neste caso, o valor i_x não é aceito e o

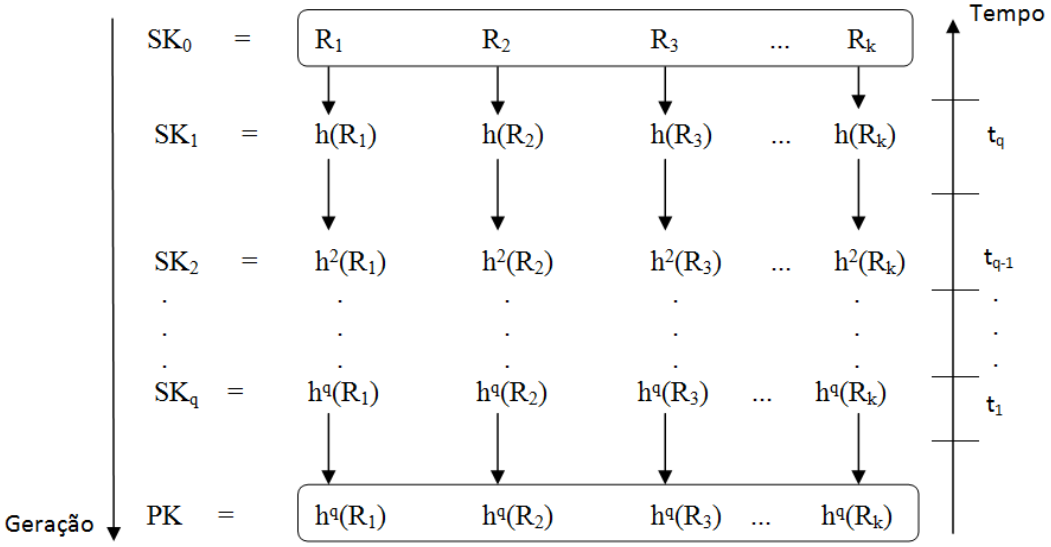


Figura 4.8: Matriz de evidências no PEAC

processo continua. Após o término deste processo, são enviados aos receptores o próprio *chunk* C, o conjunto de evidências e o vetor de contadores $z_1, z_2, ..., z_p$.

Quando um *chunk* é recebido pelo par, este pode verificar a autenticidade do mesmo através da verificação da autenticidade das evidências comparando-as com o vetor de chave pública PK. Primeiramente, é necessário verificar se o *timestamp* contido no *chunk* é novo, ou seja, não pertença a algum *chunk* anterior. O restante do processo de validação pode ser descrito pelos seguintes passos:

1. calcular o vetor de elementos $i_1, i_2, ..., i_p$ através do seguinte computo: $i_b = H(C||z_b) \times (mod(k))$, sendo $b = 1, 2, ..., p$;

2. para cada índice $i_b \in i_1, i_2, ..., i_p$, o receptor verifica se o resultado de $H_j(E_b)$ é igual ao valor de índice i_b do vetor de chave pública (PK), sendo E_b a evidência recebida de índice b. Caso todos as p evidências sejam consideradas válidas, o *chunk* é autêntico. Caso contrário, o *chunk* deve ser descartado.

O último passo do funcionamento do esquema de assinatura PEAC é a renovação de chaves. Cirulli e Di Pietro (2008) apontam duas estratégias como opção para fazer a renovação das chaves privadas ($SK_1, SK_2, ..., SK_q$) e da chave pública (PK). A primeira estratégia é repetir o passo de inicialização após o término dos q períodos de tempo (sendo q o número de vetores SK). A segunda abordagem seria fazer esta renovação antes do término e enviar a nova chave pública aos participantes. Em ambos os casos, o novo vetor PK deve ser assinado pelo esquema de assinatura PEAC para prover a autenticação do mesmo. Diferentemente do PARM (descrito na sequência), que utiliza esquemas clássicos de assinatura digital para esta função, a utilização do próprio esquema acarreta menor sobrecarga computacional.

4.3.4 PARM

PARM é um esquema de autenticação de conteúdo baseado em funções unidirecionais de *hash* para aplicações *multicast* que visa ser rápido, resistente a ataques de poluição e a ataques de negação de serviço (LIN; SHIEH; LIN, 2006). Pode ser utilizado em conjunto

com o SAIDA (Park et al. 2003), um sistema de amortização de assinatura projetado para uso em ambientes sujeitos a altas perdas de *chunks* através de esquemas de codificação (vide Subseção 4.2.6). O esquema baseia-se na criação de evidências que são verificadas *chunk* a *chunk* pelos receptores. O funcionamento do PARM pode ser representado da seguinte forma:

1. inicialização na fonte de dados;
2. geração de evidências por *chunk* pela fonte;
3. validação de evidências por *chunk* no receptor;
4. renovação parcial de chaves.

Na fase inicial, deve-se criar k valores randômicos de X *bits* cada $(R_1, R_2, ...R_k)$ para formar a primeira cadeia da chave privada SK_0, como pode ser observado na Figura 4.9. O próximo passo da fase de inicialização é formar as demais cadeias SK através do *hash* dos valores anteriores. Ou seja, $SK_0[0] = R_1$ e $SK_1[0] = h(R_1)$. Este processo de utilizar a saída do *hash* de uma cadeia na cadeia seguinte é feito q vezes. A cadeia q é a chave pública PK. O último passo desta fase é assinar digitalmente com um esquema clássico a chave pública (PK) e disponibilizá-la aos receptores.

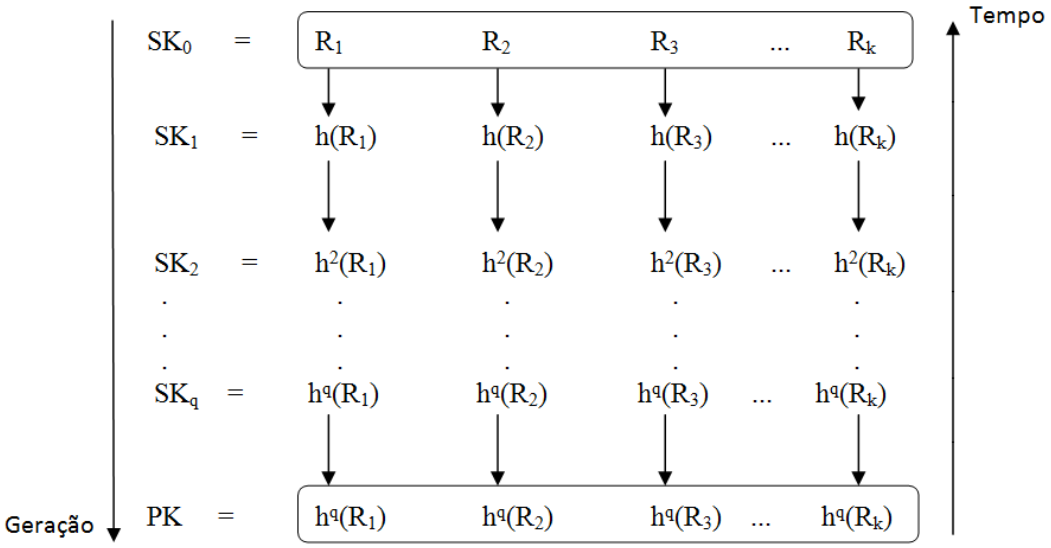


Figura 4.9: Matriz de evidências no PARM

A geração de evidências de um *chunk* deve ser feita previamente à transmissão do mesmo. O funcionamento da geração de evidências está representado pela Figura 4.11. Nela, podemos observar a matriz criada na fase de inicialização e a utilização de uma tabela auxiliar (tabela de uso) para ajudar na escolha dos valores que serão enviados como evidências. Um exemplo deste tipo de tabela está representado na Figura 4.10. Existem dois campos na tabela: índice da tabela e número de vezes utilizado.

Para criar as evidências, é necessário fazer um *hash* de X bits do *chunk* C a ser enviado. O resultado deste *hash* é divido em p segmentos de Y *bits* cada, sendo o conjunto denotado como $I = i_1, i_2, ..., i_p$. Cada segmento em I indica uma coluna da matriz. Para cada índice i, a fonte seleciona a linha de índice $(q - 1) - a_i$ da matriz, sendo a_i o número de vezes que foi utilizada aquela coluna. Obtém-se a evidência que se encontra na coluna de índice

Índice	1	2	3	4	···	k-1	k
Utilizado	1	5	2	0	···	8	2

Figura 4.10: Tabela de Utilização (tabela de uso)

i na referida linha da matriz. Desta forma, são escolhidas as evidências (na Figura 4.11 estão circulados) a serem enviadas juntamente ao *chunk*.

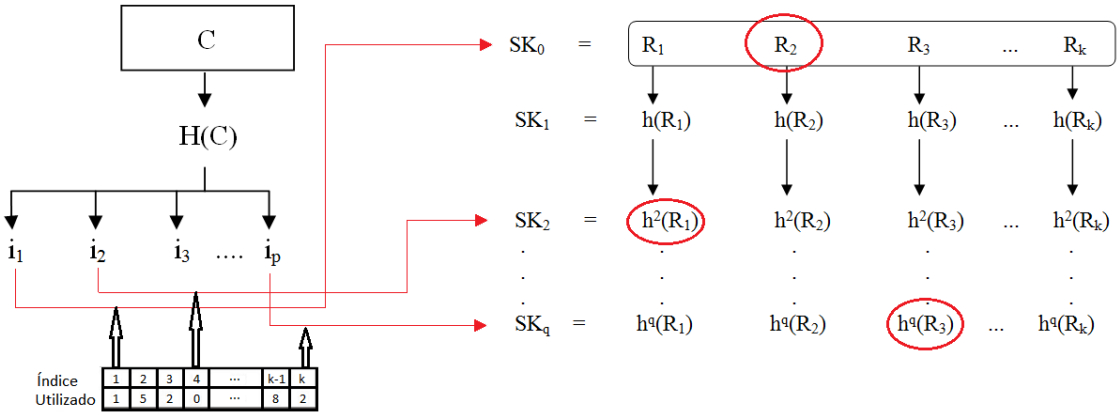


Figura 4.11: Geração de evidências no PARM

Para verificar a autenticidade do *chunk*, o receptor deve fazer o *hash* (com a mesma função utilizada pela fonte) do *chunk C* recebido, que será dividido em *p* segmentos. Baseado na tabela de uso (cada receptor deve manter uma tabela de uso), é possível saber qual a posição da evidência na matriz e saber quantos *hashes* são necessários para se encontrar o valor correspondente na chave pública. Caso os valores da chave pública sejam iguais aos calculados através das evidências, o *chunk C* é valido. Caso contrário, deve ser descartado.

Um procedimento essencial do funcionamento do PARM é a renovação parcial de chaves. Y. Lin, Shieh e W. Lin (2006) afirmam que este passo é essencial para que a comunicação entre a fonte e os receptores continue segura. Para efetuar periodicamente a renovação parcial de chaves, PARM emprega um valor limite T (*threshold*), que representa o número máximo de elementos que podem ser usados em SK_0. Quando este valor é excedido, deve-se criar novos números randômicos nos elementos que ultrapassaram T. Em seguida, para cada elemento renovado de SK_0, efetua-se sucessivas operações de *hash* para preencher todos as posições da coluna correspondente, até chegar-se à linha correspondente à chave pública (como foi feito na fase de inicialização). Os novos valores que formam a *PK* (chave pública) devem ser enviados aos receptores. Sendo assim, esta chave pública parcial deve ser assinada digitalmente por um esquema clássico, corrigida com códigos de correção antecipada de erros e enviada aos receptores para atualização local.

Segundo W. Lin, Shieh e Y. Lin (2008), no PARM é utilizada a premissa de que existe sincronização entre a fonte e os receptores, o que pode sobrecarregar o *buffer* dos receptores. Além disso, a utilização da renovação parcial de chaves, embora seja uma técnica interessante, acaba sendo muito custosa para o sistema devido a utilização de esquemas clássicos de assinatura (ex: DSA) para poder distribuir as chaves parcialmente criadas aos receptores para atualização local.

4.3.5 PARM2

W. Lin, Shieh e Y. Lin (2008) propuseram diversas melhorias no esquema de assinatura PARM, resultando em uma nova versão (doravante denominada PARM2). As principais alterações são as seguintes:

- para permitir o uso de uma camada de transporte subjacente sem entrega ordenada de pacotes, os receptores não utilizam tabela de uso. Para isto, a fonte inclui em cada *chunk* o número de *hashes* para chegar na chave pública (equivalente aos valores da tabela de uso) e um número de seqüência;
- os receptores utilizam *w buffers* contendo evidências recentemente recebidas para tentar diminuir o tempo de verificação da assinatura;
- a fonte renova uma coluna de *hashes* quando a metade de suas evidências tenham sido enviadas.

4.3.6 ALPS

Trata-se de um esquema de assinatura digital que foi desenvolvido para a autenticação da *stream* em aplicações P2P de *live streaming* (MEIER; WATTENHOFER, 2008). A idéia desta proposta é superar algumas limitações (atrasos, tamanho da assinatura e complexidade computacional) impostas por abordagens baseadas em esquemas clássicos de assinatura digital (por exemplo, RSA) e por esquemas de assinaturas propostos anteriormente (por exemplo, BiBa). A seguir, o esquema ALPS é descrito detalhadamente. Os passos do funcionamento do ALPS são:

1. inicialização pela fonte (criação da chave privada, matriz de evidências e chave pública);
2. seleção das colunas das evidências na matriz pela fonte;
3. seleção das linhas das evidências na matriz pela fonte;
4. atualização dos índices das sub-matrizes tendo como base as colunas selecionadas no passo 1 (pela fonte);
5. criação da assinatura pela fonte (conjunto de evidências selecionadas com base nos passos 2, 3 e 4);
6. verificação da assinatura no receptor.

Para a criação da chave pública e privada, é utilizada uma função de *hash* segura $H{:}0{,}1^a \rightarrow 0{,}1^a$, com um parâmetro de segurança a. Primeiramente, são criados k números randômicos $R_{1,1,1}, ..., R_{k,1,1}$ (chamados de evidências no presente trabalho) que formarão a chave pública da fonte e devem pertencer ao intervalo [0, 1]. Os índices de um valor $R_{i,u,j}$ representam o identificador da coluna (i), o identificador da sub-matriz (u) e a linha da sub-matriz (j).[1]

Cada valor criado inicialmente para formar a chave pública será o valor de entrada para construir as cadeias de *hash* e, conseqüentemente, a chave privada (últimos valores

[1] Meier e Wattenhofer utilizam o termo bloco para designar as sub-matrizes, mas não utilizaremos esse termo para evitar confusão com os grupos de chunks.

das cadeias). Tendo como base o exemplo da Figura 4.12, os valores que formam a chave privada SK são $R_{1,1,1}; R_{2,1,1}; R_{3,1,1}; ...; R_{k,1,1}$. A função de *hash* é aplicada em cada um destes valores, formando assim uma nova linha na matriz. Isto é feito novamente nos valores da nova linha e assim por diante. Desta forma, cria-se a chave pública PK da fonte $(R_{1,q+1,1}; R_{2,q+1,1}; R_{3,q+1,1}; ...; R_{k,q+1,1})$.

No ALPS, a matriz é dividida em U sub-matrizes de tamanho b (ou seja, cada sub-matriz corresponde à uma matriz de b linhas e k colunas). Durante a criação da assinatura, para cada evidência, é necessário selecionar qual das sub-matrizes está sendo utilizada no momento (indicado por U). Inicialmente, estes índices $[U_1, U_2, ..., U_k]$ são inicializados com o valor da última sub-matriz. Conforme uma assinatura é criada, caso uma evidência da coluna i tenha sido escolhida, o índice da sub-matriz U_i correspondente é decrementado. Caso nenhuma sub-matriz esteja disponível, deve-se fazer a troca completa das cadeias de *hash* (criando assim novas chaves pública e privada).

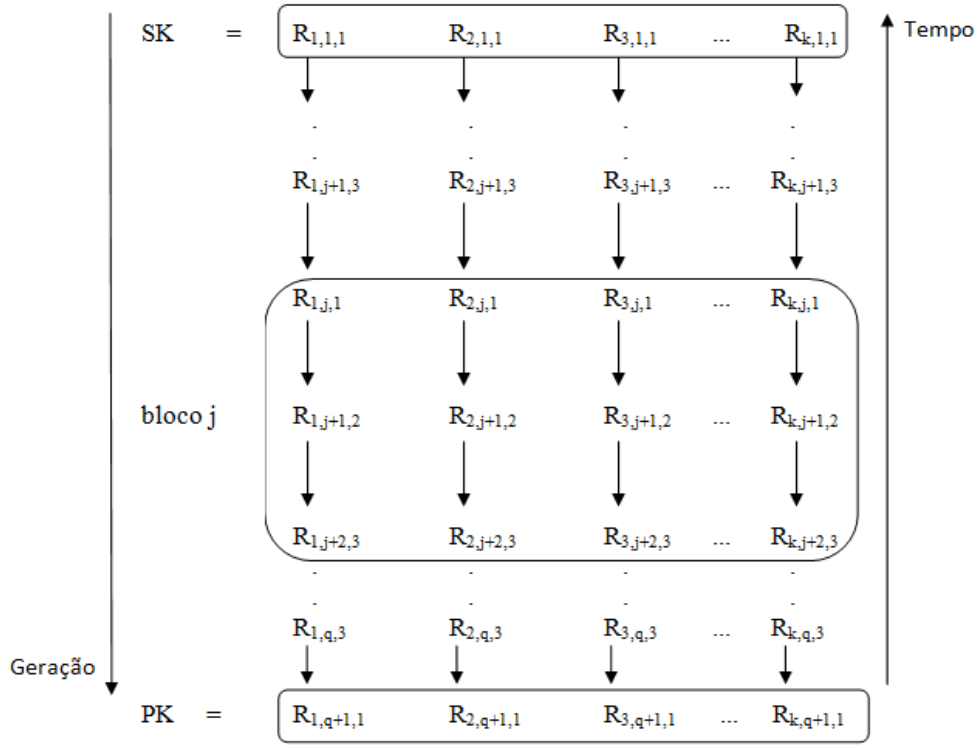


Figura 4.12: Criação da chave pública e privada no ALPS

Antes de enviar um *chunk* C, um *hash* de C e de um contador é computado para selecionar diferentes colunas $k_1, k_2, ..., k_p$. Para cada coluna selecionada, uma evidência será incluída na assinatura. Como medida de segurança, a soma das linhas precisa chegar a uma constante $Y = p \times (1 + b)/2$ para maximizar o número de possíveis combinações, sendo a cadeia k_p escolhida para tentar chegar ao valor esperado. Caso isso não seja possível, devem ser feitas novas tentativas até que a soma seja igual a Y.

Considerando as p colunas k_i escolhidas, seleciona-se p linhas no intervalo $[1, 2, ..., b]$, sendo b o tamanho de uma sub-matriz. No exemplo da Figura 4.12, as sub-matrizes são aninhadas de três em três ($b = 3$). O conjunto de evidências nas posições da matriz indicadas pelas linhas e colunas selecionadas forma então a assinatura do *chunk*. Esta assinatura, o contador utilizado e os índices das sub-matrizes são anexados ao *chunk*, e o resultado enviado aos receptores.

A verificação da assinatura é realizada de uma maneira parecida com o procedimento de criação da assinatura. O chunk C e o contador incluídos na mensagem recebida determinam como foi feita a seleção das evidências da matriz. Para cada evidência recebida, um receptor deve seguir a cadeia de *hash* até que seja encontrado um valor igual à uma evidência pré-armazenada ou contido na chave pública. Caso não seja encontrado após $2 \times b$ passos, considera-se como uma assinatura inválida. Caso contrário, deve-se comparar o número de passos para encontrar este valor com o índice da linha da sub-matriz obtido a partir do chunk C e do contador recebidos.

4.4 Análise Qualitativa dos Esquemas

Um estudo comparativo entre os esquemas de amortização de assinatura digital é apresentado por Hefeeda e Mokhatarian (2010). Eles definiram fórmulas analíticas e fizeram simulações baseado nessas fórmulas. No entanto, não consideraram esquemas leves de assinatura em seu estudo. Na sequência, explica-se quais foram os esquemas selecionados para análise e quais as principais diferenças entre o presente trabalho e o realizado por Hefeeda e Mokhatarian.

Os esquemas de amortização de assinatura selecionados para análise na presente dissertação são o **Linear Digests**, **Butterfly Graph Chaining** e **cSAIDA**. O esquema Linear Digests foi escolhido por ser atual e por não ter sido avaliado no trabalho realizado por Hefeeda e Mokhatarian. Já os outros dois esquemas de amortização selecionados foram apontados pelos referidos autores como os esquemas de melhor desempenho em seu estudo. Os demais esquemas de amortização apresentados na Seção 4.2 não são considerados na análise da presente dissertação por estarem defasados ou por não apresentarem um bom desempenho em relação aos selecionados.

O presente trabalho estende a avaliação ao incluir os esquemas leves de assinatura, adotando o mesmo conjunto de cenários empregado ao se comparar esquemas de amortização. Mais precisamente, estende-se a avaliação feita por Hefeeda e Mokhatarian, incluindo os três esquemas de assinatura leves mais relevantes e atuais (**PARM**, **PARM2** e **ALPS**). O PEAC, embora seja recente, não é utilizado na modelagem e nas simulações por se tratar de um esquema projetado para sistemas de *broadcast*.

Na próxima seção, avalia-se com um mesmo conjunto de cenários o comportamento dos principais esquemas de autenticação e de uma **solução conjunta** (algo inédito, até onde sabemos), envolvendo os esquemas de amortização e de assinatura leve que obtiverem melhor desempenho na análise comparativa. Desta forma, utiliza-se um esquema mais leve que esquemas clássicos e adicionalmente o custo dessa assinatura é amortizado ao longo do tempo. Nesta avaliação, são considerados os custos e sobrecargas tendo como base sessões de *live streaming* com uma ampla gama de requisitos: são utilizados cenários com transmissões convencionais e com transmissões de vídeo em alta definição.

5 ANÁLISE QUANTITATIVA

Neste capítulo, reune-se um conjunto de equações que expressam a eficiência de cada um dos esquemas de assinatura e de amortização relevantes ao contexto de transmissão de *streaming*. Os esquemas de amortização de assinatura analisados são os seguintes: **Linear Digests**; **Butterfly Graph Chaining**; **cSAIDA**. Já os esquemas de assinatura leve selecionados são o **PARM**, **PARM2** e **ALPS**.

Primeiramente, são apresentadas as métricas de desempenho consideradas. Na sequência, é explicado como foi feita a formulação das sobrecargas de desempenho dos esquemas de amortização de assinatura, tendo como base essas métricas. Por fim, demonstra-se como foi feito o desenvolvimento das fórmulas relativas às sobrecargas e custos proporcionados pelos esquemas leves de assinatura estudados, e quando as mesmas foram obtidas com base na literatura. Para um melhor entendimento das equações, favor verificar os símbolos contidos nas Tabelas 5.1, 5.2, 5.3 e 5.4 que são apresentadas a seguir. Nestas estão contidos os símbolos dos parâmetros de entrada e variáveis internas dos esquemas, variáveis utilizadas nas métricas de desempenho, parâmetros do ambiente (taxas, tempos de verificação, etc.) e funções utilizadas nos cômputos dos esquemas.

Tabela 5.1: Parâmetros de entrada e variáveis internas dos esquemas de autenticação

Símbolo	**Significado**
p	número de evidências por assinatura
k	número de colunas da matriz de evidências
q	número de linhas da matriz de evidências
T	limiar para a renovação parcial de chave pública
n	número de *chunks* por bloco
b	número de linhas em cada sub-matriz no ALPS
u	número de sub-matrizes no ALPS
w	número de *buffers* no PARM2 e ALPS
Q	número de linhas do butterfly graph
$C(k, j)$	*chunk* j na coluna k no Butterfly Graph Chaining
V_{min}	número mínimo de *chunks* que devem ser verificados
m	número de *chunks* que podem ser perdidos no SAIDA
τ	evidências utilizadas no SAIDA
nb	número de *bins* no BiBa
s_i	evidência de índice i no BiBa
ϵ	índice de tolerância do decodificador do Reed-Solomon
S	pacote contendo a assinatura de um bloco de *chunks*

Tabela 5.2: Variáveis utilizadas nas métricas de desempenho

Símbolo	**Significado**
d	contador de segurança do ALPS (em bytes)
P_f	probabilidade de falha na geração de assinatura no ALPS
i	número de sequência do PARM2 (em bytes)
c	número de bytes por evidência
h	número de bytes na saída de um *hash*
R	número de evidências já reveladas aos receptores
P_k	tamanho da chave pública (em bytes)
L	tamanho de um *chunk* (em bytes)
Rep	número de assinaturas duplicadas em um bloco
W	tamanho da saída de uma assinatura digital (bytes)
z	contador de não-colisões no BiBa

Tabela 5.3: Variáveis do ambiente

Símbolo	**Significado**
t_{EC}	tempo gasto com códigos de correção de erro
t_S	tempo de verificação no SAIDA
t_H	tempo para computar um *hash*
t_V	tempo de verificação de uma assinatura digital
α	taxa de envio de *chunks*
ρ	taxa de perda de *chunks*
σ	taxa no código de correção de erro Reed-Solomon
γ	taxa de sobrevivência do esquema AuthECC
ϕ	taxa de inundação do esquema AuthECC

Tabela 5.4: Funções utilizadas pelos esquemas de autenticação

Símbolo	**Significado**
$H(.)$	função de *hash*
$Sign(.)$	função de assinatura digital
$Verify(.)$	função de verificação de assinatura digital
$F(.)$	função geradora de número pseudo-randômicos
$HRS(.)$	função derivada utilizada no HORS
$C(.)$	geração de código de correção antecipada de erro
$G(.)$	função randômica de *hash*

5.1 Métricas de Desempenho

A avaliação dos esquemas de autenticação é realizada a partir da análise de diversos aspectos, como segurança, sobrecargas de processamento e transmissão e atrasos (associados à impossibilidade de verificar a autenticidade de um *chunk*). Para isto, são utilizadas as seguintes métricas na análise como uma forma de quantificar o desempenho: número de possíveis assinaturas na fonte de dados; nível mínimo de segurança propiciado; custo computacional no receptor (parte-se da premissa de que a fonte de dados é confiável e tem alto poder computacional, por isso não é considerado o custo computacional proporcionado por ela); sobrecarga de comunicação; sobrecarga de armazenamento no receptor (estruturas de dados necessárias); tamanho do *buffer* necessário no receptor (número de *chunks* armazenados); atrasos proporcionados na fonte e no receptor (atraso total). A seguir, as mesmas são explicadas detalhadamente.

Tendo como foco a análise da segurança proporcionada pelos esquemas, foram usados o **número de possíveis assinaturas** e o **nível de segurança**. O primeiro termo representa o número total de combinações de evidências (gerando assim assianturas) que a fonte de dados pode fazer. Ou seja, a partir das evidências da matriz, como elas podem ser arranjadas com o intuito de formar assinaturas. Quanto maior for esta capacidade, mais seguro será o esquema. Já o segundo termo expressa o inverso da probabilidade de um par malicioso conseguir produzir uma assinatura válida para um *chunk* poluído (ou seja, um *chunk* não criado pela fonte), tendo como base evidências reveladas por assinaturas enviadas previamente. Este valor tende a diminuir à medida que assinaturas são recebidas da fonte. Com um "alto nível de segurança", torna-se "muito difícil" que um poluidor consiga alterar dados gerados pela fonte de dados (considerando o tempo disponível que um atacante possui na janela de tempo em que um *chunk* é válido/interessante a seus vizinhos). Estas duas métricas não chegam a ter uma relação direta pois uma enfoca na segurança proporcionada antes do início da transmissão (número de possíveis assinaturas) enquanto que a outra busca indicar o nível de segurança durante a transmissão, indicando qual a chance que um atacante tem de criar uma assinatura válida.

Para avaliar a eficiência e a viabilidade dos esquemas, é analisado o **custo computacional** dos mesmos. Trata-se do custo total de processamento nos receptores necessário para verificar a autenticidade dos dados recebidos. Esta métrica é especialmente relevante quando o receptor tem baixa capacidade de processamento (por exemplo, um PDA). Caso essa métrica apresente um valor alto, alguns receptores podem não ser capazes de suportar tamanha demanda.

Além disso, avalia-se as sobrecargas impostas pelos esquemas para que os dados possam ser verificados de forma correta. Para isto, foram formuladas equações para a **sobrecarga de comunicação** e para a **sobrecarga de armazenamento**, reaproveitando sempre que possível equações estabelecidas em trabalhos anteriores. A sobrecarga de comunicação é calculada como a quantidade adicional de bytes a ser enviada para que os receptores possam verificar a autenticidade dos dados, normalizada pelo número de *chunks* transmitidos (considera-se o efeito de bytes enviados separadamente de *chunks*, por exemplo antes da transmissão iniciar ou periodicamente). Por sua vez, a sobrecarga de armazenamento é calculada como a quantidade total de bytes que deve ser dedicada ao armazenamento de estruturas de dados devido ao mecanismo de assinatura no receptor.

Finalmente, são consideradas duas métricas para avaliar os atrasos proporcionados pelos esquemas de autenticação: **tamanho necessário do *buffer* nos receptores**; e **atraso na fonte e no receptor (atraso total)**. A primeira métrica diz respeito à quantidade de *chunks* que precisa ser armazenada pelos receptores antes que seja possível a verificação

dos dados recebidos. Já a segunda corresponde ao somatório dos tempos necessários para criar uma assinatura (fonte de dados) e para verificar a autenticidade da mesma (receptor). É importante ressaltar que atrasos relativos à transmissão na rede ou devido ao processo de codificação de vídeo não são considerados no presente trabalho.

5.2 Esquemas de Amortização de Assinatura

As métricas de desempenho utilizadas para avaliar os esquemas de amortização são as seguintes: (1) custo computacional por bloco de *chunks* no receptor; (2) sobrecarga de comunicação por *chunk*; (3) tamanho do *buffer* exigido no receptor; (4) atraso total (na fonte e no receptor). As Equações 5.1, 5.2, 5.4, 5.5, 5.7 e 5.8 foram desenvolvidas por Hefeeda e Mokhtarian . As equações restantes deste capítulo, referentes aos esquemas de amortização, foram formuladas no presente trabalho. A seguir, as equações citadas são explicadas em maiores detalhes.

As equações 5.1 e 5.2 se referem ao **custo computacional** para verificar a autenticidade de um bloco de *chunks*. Todos os esquemas de amortização (Linear Digests, Butterfly e cSAIDA) precisam efetuar a verificação de uma assinatura digital (cujo tempo é representado pelo símbolo t_V) e realizar múltiplas operações de *hash* para verificar a autenticidade de um bloco de *n chunks* (assume-se que *chunks* possuem L bytes). A Equação 5.1 representa o custo computacional dos esquemas de amortização Linear Digests e Butterfly Graph Chaining para verificar a autenticidade de um bloco de *n chunks*, denotando como t_H o tempo para computar um *hash*. Como cada *chunk* tem L bytes, torna-se necessário dividir este valor por 64 já que o *hash* só é computado sobre dados de 64 bytes.

O custo necessário para o cSAIDA verificar um bloco de *chunks* é dado pela Equação 5.2. No presente trabalho, é ignorado o custo da codificação de FEC por ser baixo. Neste esquema, também é necessário levar em consideração o tempo necessário para verificar uma assinatura digital (t_V). Além disso, torna-se necessário computar *n hashes* (sendo que o tempo para esta operação é dado pelo símbolo t_H) sobre dados de $L+h$ bytes de 64 em 64 bytes (L representa o tamanho de um chunk e h representa o tamanho de uma saída de *hash*).

$$CC_{Butterfly} = CC_{LDigests} = t_V + t_H \times n \times L/64 \tag{5.1}$$

$$CC_{cSAIDA} = t_V + t_H \times n \times (L + h)/64 \tag{5.2}$$

Já as **sobrecargas de comunicação** dos esquemas de amortização são expressas pelas Equações 5.3, 5.4 e 5.5 (em bytes). A sobrecarga de comunicação imposta pelo esquema Linear Digests é diretamente proporcional ao esquema de assinatura digital utilizado pelo mesmo pois implica no tamanho da chave pública proporcionado. Por exemplo, caso seja utilizado o esquema clássico de assinatura DSA, a chave pública teria 128 ou 256 bytes, e no caso do ECDSA, 32 bytes (Meier and Wattenhofer 2008). No Linear Digests, o receptor precisa obter uma chave pública de tamanho P_k previamente a transmissão de modo seguro. Durante a transmissão, para cada bloco de *chunks* enviado pela fonte, uma assinatura deve ser recebida (pacote s) (no presente trabalho, não é considerado o tamanho do pacote s). Sendo assim, para efeitos de comparação, o tamanho da chave pública pode ser dividido entre os *n chunks* de um bloco (Equação 5.3).

As Equações 5.4 e 5.5 representam, respectivamente, a sobrecarga de comunicação do esquema Butterfly Graph Chaining e do esquema cSAIDA. Além do tamanho da chave

pública, os principais parâmetros que influenciam a sobrecarga de comunicação do esquema Butterfly Graph Chaining são o número de assinaturas duplicadas (Rep) e o número de linhas do grafo (Q).

O número de linhas (Q) é um parâmetro de entrada do sistema que deve ser ajustado para que a assinatura não exceda uma MTU (*Maximum Transmission Unit*). Além de ser enviado juntamente ao *chunk*, ele define o tamanho de todo o grafo já que o número de colunas também é definido com base neste parâmetro: $(log_2 Q + 1)$. Quanto maior for o valor de Q, maior serão o grafo e o tamanho do pacote S (assinatura) que contém os *hashes* concatenados dos *chunks* do bloco. Além disso, o número de vezes que a assinatura precisa ser enviada (Rep) para evitar a perda de *chunks* também influencia na sobrecarga de comunicação pois quanto maior for este valor, maior será a largura de banda necessária para transmiti-la. Já no cSAIDA, o maior impacto na sobrecarga de comunicação deve-se ao número mínimo de *chunks* que precisam ser recebidos antes que qualquer *chunk* do presente bloco possa ser verificado, representado pelo símbolo V_{min} (maiores informações sobre essas fórmulas podem ser encontradas no trabalho realizado por Hefeeda e Mokhtarian).

$$CO_{LDigests} = P_k/n \tag{5.3}$$

$$CO_{Butterfly} = (Rep \times (P_k + Q \times h) + h \times (2 \times n - Q))/n \tag{5.4}$$

$$CO_{cSAIDA} = (P_k + (n - V_{min}) \times h)/V_{min} \tag{5.5}$$

Além disso, deve-se avaliar o **tamanho do buffer** que deve ser usado nos receptores para cada esquema. O número mínimo de *chunks* que precisa ser armazenado no receptor para verificar a autenticidade de um bloco de *chunks* impacta diretamente nos atrasos impostos pelos esquemas de amortização. No Linear Digests, não é necessário usar um *buffer*, pois quando um receptor recebe um *chunk*, ele pode imediatamente verificar sua autenticidade (lembrando que no Linear Digests a fonte envia um pacote assinado contendo os *hashes* de cada *chunk* do bloco). Por outro lado, Butterfly Graph Chaining e cSAIDA precisam armazenar todos os n *chunks* do bloco antes da verificação de autenticidade de qualquer *chunk*.

Por fim, os **atrasos totais** (na fonte e no receptor) dos esquemas de amortização são calculados a partir das equações 5.6, 5.7 e 5.8. Parte-se da premissa de que *chunks* são recebidos a uma taxa constante. Considerando o pior caso, no receptor, o esquema Linear Digests precisa armazenar n *chunks* e a assinatura do bloco (pacote s), resultando em um atraso total de $(n+1)/\alpha$ segundos (Equação 5.6), sendo que α representa a taxa de *chunks* por segundo. Butterfly Graph Chaining apresenta um atraso similar no receptor, mas nesse esquema não é necessário receber o pacote s contendo a assinatura (Equação 5.7). Finalmente, no cSAIDA, o receptor não precisa receber todos os n *chunks* no *buffer* devido a utilização de codificação de FEC. Segundo Hefeeda e Mokhtarian, esta fração de n é bem similar ao valor original. Portanto, este valor foi arredondado para n. Além disso, eles definem que o atraso por parte da fonte também é de n/α segundos, resultando assim em $(2 \times n)/\alpha$ segundos no total (Equação 5.8).

$$D_{LDigests} = (n + 1)/\alpha \tag{5.6}$$

$$D_{Butterfly} = n/\alpha \tag{5.7}$$

$$D_{cSAIDA} = (2 \times n)/\alpha \tag{5.8}$$

5.3 Esquemas de Assinatura Leve

A avaliação dos esquemas leves de assinatura é feita a partir das seguintes métricas (conforme definidas na Seção 5.1): (1) número de possíveis assinaturas na fonte de dados; (2) nível mínimo de segurança propriciado; (3) custo computacional por *chunk* no receptor; (4) sobrecarga de comunicação; (5) sobrecarga de armazenamento no receptor. As equações 5.11 e 5.14 foram obtidas de Lin et al. (2006); 5.10, 5.16, e 5.19, de Meier e Wattenhofer (2008); e 5.15, 5.18, e 5.21, de Lin et al. (2008). As demais equações dos esquemas leves foram desenvolvidas no presente trabalho (Equações 5.9, 5.12, 5.13, 5.17, 5.20 e 5.22).

Para calcular o **número de possíveis assinaturas** que a fonte de dados pode gerar antes do início da transmissão, deve-se considerar o número de possíveis combinações na seleção das linhas e colunas da matriz, durante a seleção de evidências para um *chunk*. Sendo assim, deve-se multiplicar o número de possibilidades para índices de linha com o número de possibilidades para índices de coluna. No caso do PARM e PARM2 (Equação 5.9), o número de possíveis índices para as linhas da matriz de evidências é calculado através do arranjo com repetição de q elementos escolhidos p a p: $P^q_{p(rep)}$ (utiliza-se arranjo pois a ordem de escolha é importante, lembrando que q e p representam, respectivamente, o número de linhas da matriz e o número de evidências por assinatura de chunk). O número de possíveis índices para as colunas da matriz, por sua vez, é calculado pela combinação de k colunas p a p com repetição: $C^k_{p(rep)}$ (são selecionadas p evidências por assinaturas, por isso a seleção é feita de p em p valores).

Já no ALPS, o número de possíveis índices para uma linha de uma sub-matriz é definido pelo arranjo com repetição de b elementos escolhidos $p-1$ a $p-1$ (utiliza-se arranjo por ser essencial a ordem na escolha) e o número de possíveis índices para as colunas da matriz é dado pela combinação de k colunas escolhidas p a p sem repetição (Equação 5.10). Utiliza-se $p-1$ pois o último valor é selecionado com base na seguinte restrição para tentar maximizar o número de combinações distintas: a soma dos p elementos deve resultar em uma constante igual a $t = (1 \times p \times (1+b))/2$. Segundo Meier e Wattenhofer (2008), esta constante t corresponde ao valor esperado de uma soma de valores (escolhidos uniformemente de forma randômica) para as linhas da sub-matriz, o que maximiza o número de possíveis combinações para estes valores. Note que estes valores escolhidos para as linhas não precisam ser distintos, diferentemente dos valores relativos às colunas.

$$\begin{aligned} N_{PARM} &= P^q_{p(rep)} \times C^k_{p(rep)} \\ &= q^p \times C^p_{k+p-1} \\ &= q^p \times (k+p-1)!/(p! \times ((k+p-1)-p)!) \\ &= q^p \times (k+p-1)!/(p! \times (k-1)!) \end{aligned} \tag{5.9}$$

$$\begin{aligned} N_{ALPS} &= (1-P_f) \times P^b_{p-1(rep)} \times C^k_p \\ &= (1-P_f) \times b^p \times P^k_p/p! \\ &= (1-P_f) \times b^p \times k!/(p! \times (k-p)!) \end{aligned} \tag{5.10}$$

As Equações 5.11, 5.12 e 5.13 representam uma estimativa do **nível de segurança** proporcionado por cada esquema analisado: PARM, PARM2 e ALPS. O nível de segurança, para determinado instante de tempo, é definido através da quantidade de evidências reveladas, do número de linhas e do número de colunas adotados na matriz de evidências. O número de elementos na matriz de evidências corresponde à quantidade total de escolhas que a fonte tem durante a seleção das p evidências para criar uma assinatura. No entanto, evidências reveladas previamente (R) aumentam a chance que um atacante tem de criar uma assinatura válida. Sendo assim, o número total de evidências na matriz deve ser dividido pelo número de evidências reveladas e elevado à quantidade de evidências que precisam ser selecionadas por assinatura, p.

No PARM e PARM2, o número total de evidências na matriz é dado pelo produto entre as linhas e as colunas da matriz de evidências: $q \times k$ evidências. Já no ALPS, o número total de evidências para um determinado instante de tempo é dado pelo número de elementos de uma sub-matriz de evidências, representado pelo produto entre o número de linhas de uma sub-matriz pelo número de colunas da matriz ($b \times k$ evidências).

Uma das principais diferenças entre PARM e PARM2 é a renovação de chaves. O PARM renova sua chave após serem utilizadas de forma repetida as evidências da primeira linha da matriz mais de T vezes (Equação 5.11 representa o nível de segurança proporcionado pelo esquema PARM). Já no caso do PARM2, renovam-se as colunas de *hash* nas quais foram utilizadas mais da metade de suas evidências, como pode ser observado na segunda parte da Equação 5.12. Por fim, no caso do ALPS, não existe renovação parcial de chaves, e sim completa. Portanto, só é feita a renovação quando são utilizadas todas as sub-matrizes da matriz de evidências, ou seja, quando o número de evidências reveladas (R) for maior que o número de elementos na matriz ($u \times b \times k$).

$$S_{PARM} = (q \times k/R)^p, R/q \leq T \tag{5.11}$$

$$S_{PARM2} = (q \times k/R)^p, (2 \times R)/k \leq T \tag{5.12}$$

$$S_{ALPS} = ((b \times k \times u - R \times u)/R)^p, R \leq (u \times k) \tag{5.13}$$

Além disso, para avaliar a eficiência dos esquemas leves de assinatura, foram modeladas as equações relativas ao **custo computacional** para o receptor verificar a autenticidade dos *chunks*. Como o PARM pode ser usado em conjunto com o esquema de amortização SAIDA, os tempos das ações de amortização e de codificação visando correção de erros (t_S e t_{EC}) devem ser computados e são adicionados à equação que faz o cômputo do tempo necessário ao processamento das operações de *hash* para verificar um conjunto de evidências. No PARM (Equação 5.14), no pior caso, o receptor deve fazer q operações de *hash* por p vezes (número de evidências por assinatura) pois deve-se seguir a cadeia de *hash* até encontrar uma evidência igual à recebida. No PARM2, o funcionamento é semelhante, mas não inclui ações relacionadas ao SAIDA nem a códigos já que os mesmos não são usados (Equação 5.15).

Por fim, o tempo gasto por receptores no ALPS (Equação 5.16), assim como os demais, é definido pelo número de operações de *hash* necessárias para verificar uma assinatura. Para verificar uma evidência, deve-se, no pior caso, seguir a cadeia de *hash* em até duas sub-matrizes ($2 \times b$) até encontrar uma evidência igual à recebida, sendo b o número de linhas de uma sub-matriz. No entanto, a autenticidade de um *chunk* é representada por uma assinatura contendo p evidências, ou seja, este processo deve ser feito p vezes.

$$CC_{PARM} = t_S + t_{EC} + q \times p \times t_H \tag{5.14}$$

$$CC_{PARM2} = q \times p \times t_H \tag{5.15}$$

$$CC_{ALPS} = 2 \times b \times p \times t_H \tag{5.16}$$

As equações 5.17, 5.18 e 5.19 indicam a **sobrecarga de comunicação por *chunk*** imposta pelos esquemas estudados (esta métrica corresponde à quantidade adicional de bytes a ser enviada para que os receptores possam verificar a autenticidade dos dados). O tamanho da assinatura do PARM a ser enviada juntamente ao *chunk* corresponde a p evidências de c bytes cada, como pode ser observado na Equação 5.17. No caso do PARM2 (Equação 5.18), além da assinatura, junto ao *chunk* são enviados k valores da tabela de uso, cada um com $\lg q$ bytes (utilizados para obter o número de *hashes* necessários para a verificar as evidências) e o número de sequência (i bytes). Já no ALPS (Equação 5.19) são enviados por *chunk* p evidências (assinatura) de c bytes e o contador de segurança (d bytes), utilizado nos cômputos de *hash* como uma medida extra de segurança.

$$CO_{PARM} = p \times c \tag{5.17}$$

$$CO_{PARM2} = p \times c + k \times \lg q + i \tag{5.18}$$

$$CO_{ALPS} = p \times c + d \tag{5.19}$$

Por fim, a **sobrecarga de armazenamento** nos receptores proporcionada por cada esquema é calculada a partir das Equações 5.20, 5.21 e 5.22 (esta métrica representa a quantidade total de bytes que deve ser dedicada ao armazenamento de estruturas de dados devido ao mecanismo de assinatura no receptor). No PARM (Equação 5.20), deve-se considerar as sobrecargas impostas pelo tamanho da chave pública e pelos valores da tabela de uso que precisam ser armazenados pelos receptores. A chave pública é representada por k evidências de c bytes cada e os valores da tabela de uso são representados por k valores de tamanho $\lg q$. Em contraste, no PARM2 (Equação 5.21), além da chave pública e da tabela de uso, deve-se considerar o tamanho dos *buffers* (w *buffers* são compostos por k evidências de c bytes) e o tamanho do número de sequência (i bytes). No ALPS (Equação 5.22) armazena-se a chave pública, os *buffers* e o contador de segurança (d bytes) utilizados no cômputo dos *hashes*. A chave pública também corresponde a k valores de c bytes cada e os w *buffers* também contém k evidências de c bytes.

$$SO_{PARM} = k \times c + k \times \lg q \tag{5.20}$$

$$SO_{PARM2} = k \times c \times w + k \times c + i \tag{5.21}$$

$$SO_{ALPS} = k \times c \times w + k \times c + d \tag{5.22}$$

6 RESULTADOS DA AVALIAÇÃO

Tendo como base as equações definidas no capítulo anterior, são apresentados nesse capítulo os resultados das simulações dos esquemas de autenticação estudados quando utilizados como parte de uma aplicação P2P de *live streaming*, seja para conteúdo convencional ou de alta definição. Inicialmente, são explicados os cenários utilizados nas simulações, na Seção 6.1. Na sequência, na Seção 6.2, avalia-se o comportamento do número de *chunks* por bloco por se tratar do parâmetro de maior impacto nos esquemas de amortização (HEFEEDA; MOKHTARIAN, 2010). Com base nesta análise e nos valores de entrada obtidos no trabalho realizado por Hefeeda e Mokhtarian, avalia-se o comportamento das métricas nos esquemas de amortização. Além disso, é feita uma análise de sensibilidade para tentar definir os melhores valores possíveis para os parâmetros de entrada dos esquemas de assinatura leve, permitindo assim uma avaliação precisa de desempenho dos mesmos (Seção 6.3). Por fim, é apresentado o comportamento da combinação dos esquemas mais eficientes de amortização e de asinatura leve (Seções 6.4 e 6.5).

6.1 Cenários Avaliados

A duração das sessões pode variar arbitrariamente e, como não existe uma duração típica para uma sessão de *streaming* P2P, são avaliados cenários com duração de 10 minutos (curta duração) e de 2 horas (longa duração). Em relação à qualidade de transmissão, são consideradas duas opções: qualidade convencional, com resolução de 280 linhas verticais, e alta definição (*high definition* - HD), com 1080 linhas verticais.

Em cenários de alta definição, a largura de banda necessária varia de 5 Mbps até 15 Mbps, de acordo com experimentos reportados por fabricantes de equipamento de alta definição (CISCO, 2009). A combinação das variações dos atributos duração e qualidade da sessão porporciona quatro cenários distintos, como pode ser observado na Tabela 6.1. A partir desses cenários, é possível verificar o desempenho dos esquemas de autenticação em cenários com diferentes requisitos. No presente trabalho, assume-se que *chunks* possuem 1.500 bytes, sendo assim possível transmitir um *chunk* em um único pacote em muitas das redes convencionais).

Tabela 6.1: Cenários Modelados

Cenário	Conv. Curta	HD Curta	Conv. Longa	HD Longa
Resolução	320x280	1920x1080	320x280	1920x1080
Duração	curta	curta	longa	longa
Sessão (min)	10	10	120	120
Largura de banda (Mbps)	0.384	16	0.384	16
Quantidade de *Chunks*	25,200	104,400	302,400	1,252,800
Taxa (*chunks*/s)	42	174	42	174
Volume de Dados (MB)	36.04	149.34	432.58	1792.14

6.2 Esquemas de Amortização de Assinatura

Para a análise do presente trabalho, foram selecionados os esquemas Butterfly Graph Chaining e cSAIDA, previamente estudados por Hefeeda e Mokhtarian , assim como os parâmetros de entrada apontados no trabalho deles. No presente trabalho, foi adicionada a análise do esquema de amortização Linear Digests.

Segundo Hefeeda e Mokhtarian, o **número de *chunks* por bloco** (n) é o parâmetro mais importante pois impacta diretamente no desempenho dos esquemas de amortização. Eles apontam 100 *chunks* como tamanho mínimo do bloco suficiente para todos os esquemas de amortização estudados. Fizemos uma análise semelhante a deles em relação a esse parâmetro, incorporando na análise o esquema Linear Digests. Para isto, fixamos os demais parâmetros (tendo como base os valores da Tabela 6.2) e variamos o número de *chunks* por bloco, observando as métricas de desempenho, como pode ser visto na Figura 6.1. Desta forma, é possível verificar qual o valor ideal para o tamanho do bloco. Na Figura 6.1, o eixo y (em escala logarítmica) representa as sobrecargas (atraso total; sobrecarga de comunicação por *chunk*; custo computacional por *chunk*; e tamanho do *buffer* no receptor), enquanto o eixo x (em escala linear) representa o número de *chunks* por cada bloco (n).

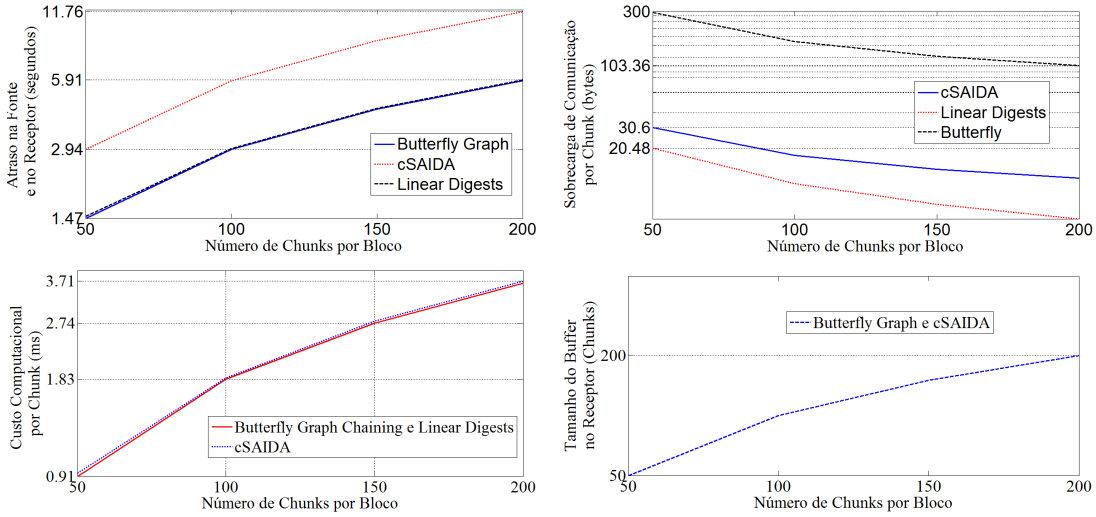


Figura 6.1: Avaliação das sobrecargas e custos a partir da variação do número de *chunks* por bloco

Como conclusão dessa análise em particular, obtivemos o mesmo valor ideal indicado

no referido trabalho, conforme explicado a seguir: 128 *chunks* por bloco. Isto se justifica devido às altas sobrecargas proporcionadas para valores acima de 128. Além disso, foi visto que a medida que aumenta o tamanho do bloco em *chunks*, o atraso na fonte e no receptor aumenta proporcionalmente por um fator de 0.03 e a sobrecarga de comunicação por *chunk* diminui. Já o custo computacional por *chunk* sofre um acréscimo conforme se aumenta o número de *chunks* por bloco (no caso do custo por bloco, aumenta-se o mesmo em uma proporção de 2.38 através do aumento de n). Por fim, observou-se que existe um acréscimo no tamanho do *buffer* dos esquemas cSAIDA e Butterfly Graph conforme o valor de n aumenta. Note que o esquema Linear Digests não utiliza *buffer* nos receptores.

A segunda análise realizada diz respeito às **métricas de desempenho** dos esquemas de amortização (custo computacional por bloco, sobrecarga de comunicação, atraso imposto pelo processo de autenticação e tamanho do *buffer*). A partir dos parâmetros de entrada e das equações dispostas na Seção 5.2, foram obtidos os resultados dessas métricas. Para isto, foram utilizados os valores contidos na Tabela 6.2 nos respectivos parâmetros de entrada (estes valores foram obtidos com base no trabalho realizado por Hefeeda e Mokhatarian). O resultado dessa análise pode ser observado na Tabela 6.3.

Tabela 6.2: Valores dos Parâmetros de Entrada dos Esquemas de Amortização

Símbolo	**Significado**	**Valor**
n	número de *chunks* por bloco	128 *chunks*
t_H	tempo para computar um *hash*	0.1 ms
t_V	tempo de verificação de uma assinatura digital	324 μs
h	número de bytes na saída de um *hash*	20 bytes
P_k	tamanho da chave pública	256 bytes
L	tamanho de um *chunk*	1500 bytes
Q	número de linhas do butterfly graph	32 linhas
V_{min}	número mínimo de *chunks* verificados	102 *chunks*
Rep	número de assinaturas duplicadas em um bloco	8 assinaturas
α	taxa de envio de *chunks*	42 *chunks*/s

Tabela 6.3: Sobrecargas dos Esquemas de Amortização de Assinatura

Esquema	**Custo Computacional por Bloco**	**Atraso**	**Comunicação Sobrecarga**	**Tamanho do Buffer**
Butterfly	300.32 ms	3.04 s	91 bytes	128 *chunks*
cSAIDA	304.32 ms	6.09 s	7.61 bytes	128 *chunks*
Linear Digests	300.32 ms	3.07 s	2 bytes	1 *chunk*

A partir da análise realizada no presente trabalho, apresentada na Tabela 6.3, observou-se o seguinte:

- Linear Digests e Butterfly Graph Chaining apresentam o menor custo computacional por *chunk*, embora sejam similares (cerca de 2.34ms por *chunk*);
- a menor sobrecarga de comunicação por *chunk* foi apresentada pelo Linear Digests: 2 bytes;

- Butterfly Graph Chaining obteve o menor atraso total (na fonte e no receptor): 3.04s;

- Linear Digests não demanda o uso de *buffers* de *chunks*, como os demais.

Hefeeda e Mokhtarian concluíram que o TFDP (HABIB et al., 2005) é o esquema mais eficiente entre os estudados por eles, tendo como base o custo computacional. Isto acontece pois não é necessário verificar uma assinatura digital para cada bloco. No entanto, para criar a árvore Merkle no TFDP, é necessário ter todo o fluxo de vídeo previamente à transmissão. Sendo assim, o TFDP foi desconsiderado em nossa análise pois só é aplicável em transmissões de vídeo sob demanda (VoD).

Dentre os esquemas estudados na presente dissertação, Hefeeda e Mokhtarian apontam o esquema Butterfly Graph Chaining como de melhor desempenho no que tange o custo computacional. Além disso, eles concluíram que o cSAIDA é o esquema que proporciona a menor sobrecarga de comunicação entre todos os esquemas estudados. A partir de 50 *chunks* por bloco, a sobrecarga de comunicação dos esquemas se estabiliza. No entanto, a partir dos resultados obtidos no presente trabalho, pôde-se notar que o esquema Linear Digests (não incluído nas análises de Hefeeda e Mokhtarian) apresenta um desempenho similar ao Butterfly Graph Chaining em algumas métricas (custo computacional) e até mesmo superior em outras (sobrecarga de comunicação).

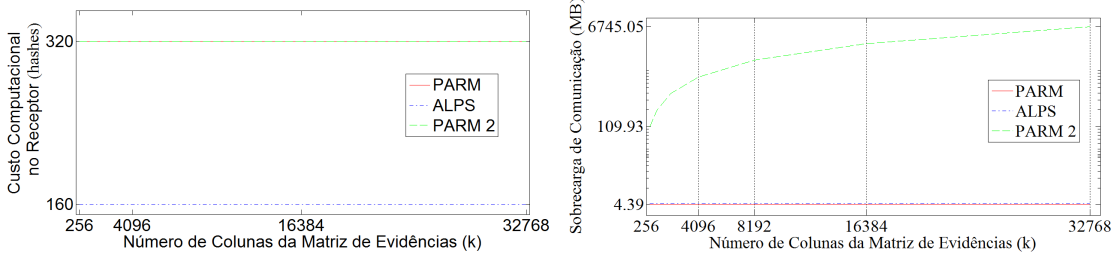
6.3 Esquemas de Assinatura Leve

O desafio inicial para uma comparação justa entre os esquemas leves de assinatura digital (PARM, PARM2 e ALPS) é ajustar o conjunto de parâmetros a serem usados. No início desta seção, são avaliados os esquemas para entender seus comportamentos e qual a influência de cada parâmetro através de uma análise de sensibilidade. Nesta análise, partiu-se de valores arbitrados com base nas referências bibliográficas, e avaliou-se o impacto da variação do valor de cada parâmetro, com base nas equações da Seção 5.3. Através de um processo com múltiplas iterações (que será explicado na sequência), se chegou ao melhor conjunto de parâmetros para cada esquema. Não é possível garantir, entretanto, que essas escolhas são ótimas, porque seria impossível avaliar todas as combinações. No final da seção, avalia-se as métricas de desempenho dos esquemas leves tendo como base os valores dos parâmetros de entrada encontrados na análise de sensibilidade realizada.

A seguir, é apresentada a análise de sensibilidade relativa aos seguintes parâmetros: (1) número de colunas da matriz, k; (2) número de linhas da matriz no PARM e PARM2, q; (3) número de linhas da sub-matriz no ALPS, b; e (4) número de evidências por assinatura, p. Todos os gráficos utilizam escala logarítmica no eixo y e escala linear no eixo x.

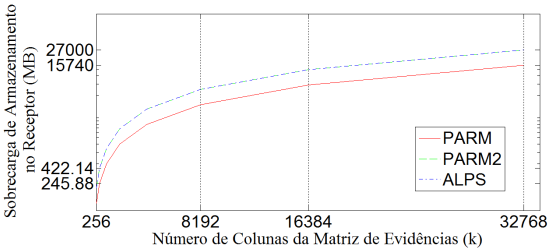
A Figura 6.2 apresenta três gráficos que ilustram os resultados da análise de sensibilidade relativa ao **número de colunas da matriz de evidências** (k). No eixo x desses gráficos, apresenta-se o valor de k. Na Figura 6.2 (a), o eixo y representa o custo computacional (operações de *hash* necessárias para verificar a autenticidade de um *chunk* no receptor). Já na Figura 6.2 (b), o eixo y indica a sobrecarga de comunicação (quantidade adicional de bytes a ser enviada para que os receptores possam verificar a autenticidade dos dados, em MB). Por fim, na Figura 6.2 (c), a sobrecarga de armazenamento no receptor (quantidade total de bytes que deve ser dedicada ao armazenamento de estruturas de dados devido ao mecanismo de assinatura no receptor, em MB) é representada no eixo y.

Como pode ser visto na Figura 6.2 (a), o custo computacional permancece constante, independentemente do valor de k. Ou seja, esta métrica não é influenciada pelo número de colunas pois a verificação da assinatura é influenciada pela quantidade de *hashes* necessários (normalmente relacionada com o número de linhas da matriz, q) e pelo número de evidências (p) que são enviadas em cada assinatura. No entanto, o valor de k influencia linearmente a sobrecarga de comunicação (Figura 6.2 (b)) do esquema de assinatura leve PARM2 devido aos valores da tabela de uso que são enviados juntamente a mensagem (vide Seções 4.3.4 e 4.3.5). Nos demais esquemas, a variação do número de colunas não impacta na sobrecarga de comunicação pois estes valores não precisam ser enviados para verificar a autenticidade dos *chunks*. Já em relação à sobrecarga de armazenamento (Figura 6.2 (c)), conforme aumenta-se o valor de k, maior (ordem linear) é a sobrecarga em todos os esquemas analisados devido ao aumento do tamanho da chave pública (k valores em cada chave, conforme pode ser observado nas Equações 5.20, 5.21 e 5.22 da Seção 5.3), aumento dos *buffers* (no caso do ALPS e PARM2, Seções 4.3.6 e 4.3.5) e aumento na quantidade de valores da tabela de uso que precisa ser armazenada nos receptores (no PARM, Seção 4.3.4).



(a) Custo computacional a partir da variação do número de colunas da matriz

(b) Sobrecarga de comunicação a partir da variação do número de colunas da matriz



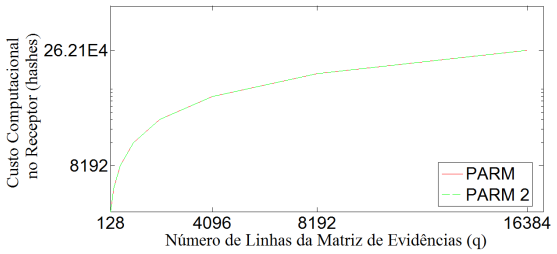
(c) Sobrecarga de armazenamento a partir da variação do número de colunas da matriz

Figura 6.2: Avaliação das métricas de desempenho a partir da variação do número de colunas da matriz e fixação dos demais parâmetros.

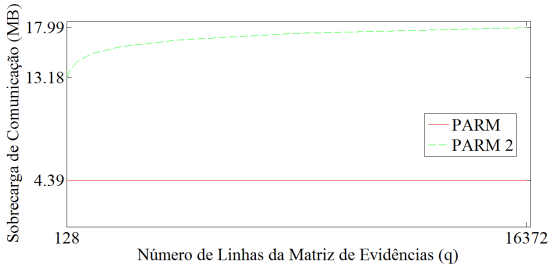
A Figura 6.3 contém a avaliação das sobrecargas em função do **número de linhas da matriz de evidências** nos esquemas PARM e PARM2 (parâmetro q). No eixo x, apresenta-se os valores de q. Na Figura 6.3 (a), o custo computacional no receptor (operações de *hash*) é representado no eixo y. Já na Figura 6.3 (b), o eixo y indica a sobrecarga de comunicação (MB). Finalmente, na Figura 6.3 (c), a sobrecarga de armazenamento no receptor (MB) é representada no eixo y.

O custo computacional é aumentado linearmente em ambos os esquemas PARM e PARM2 quando se aumenta o número de linhas da matriz, conforme pode ser visto na Figura 6.3 (a). Quanto mais se eleva o valor de q, maior é o custo computacional proporcionado por eles devido a um aumento no tamanho da matriz (q linhas por k colunas, Equações 5.14 e 5.15 da Seção 5.3), resultando assim em um maior número de *hashes* que

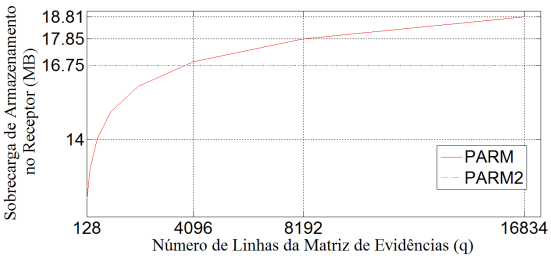
precisam ser computados durante a verificação da assinatura. Além disso, quanto maior for o valor de q, maior será (ordem linear) a sobrecarga de comunicação do PARM2 devido ao tamanho dos valores da tabela de uso (indicados por $\lg q$ na Equação 5.18 da Seção 5.3), enquanto o PARM não sofre alterações por não precisar enviar tais dados pela rede (Figura 6.3 (b)). Por fim, a sobrecarga de armazenamento no receptor do esquema PARM sofre um acréscimo linear devido a estes valores, enquanto no PARM2 permanece constante por não precisar armazenar os valores da tabela de uso (Figura 6.3 (c)).



(a) Custo computacional a partir da variação do número de linhas da Matriz



(b) Sobrecarga de comunicação a partir da variação do número de linhas da Matriz



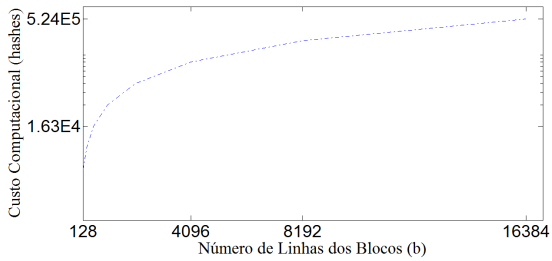
(c) Sobrecarga de armazenamento a partir da variação do número de linhas da Matriz

Figura 6.3: Avaliação das métricas de desempenho a partir da variação do número de linhas da matriz e fixação dos demais parâmetros.

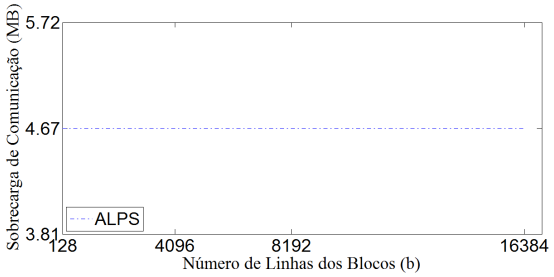
Na Figura 6.4, pode ser observado, a partir dos três gráficos contidos na mesma, o comportamento do esquema ALPS a partir da variação do **número de linhas das sub-matrizes de evidências** (b). No eixo x dos gráficos, apresenta-se os valores de b. Na Figura 6.4 (a), o custo computacional (*hashes*) é indicado no eixo y. Já a sobrecarga de comunicação (MB) é apresentada na Figura 6.4 (b), no eixo y. Por fim, na Figura 6.4 (c), o eixo y representa a sobrecarga de armazenamento no receptor (MB).

A partir da variação do número de linhas da sub-matriz (b), na Figura 6.4, tanto a sobrecarga de comunicação quanto a sobrecarga de armazenamento permanecem constantes. A sobrecarga de comunicação não é alterada pois apenas deve ser considerado o tamanho da assinatura (contém p evidências cada) e o contador de segurança, vide Equação 5.19 da Seção 5.3. Já a sobrecarga de armazenamento não sofre mudanças pois o número de linhas da sub-matriz não influencia no tamanho dos *buffers* e da chave pública (Equação 5.22). O aumento de b faz com que apenas o custo computacional seja aumentado linearmente já que o tamanho das sub-matrizes (corresponde a b linhas por k colunas na Equação 5.16, sendo que b deve ser bem menor que q para diminuir o custo proporcionado) cresce e, consequentemente, o número de operações de *hash* necessárias para a verificação das assinaturas aumenta.

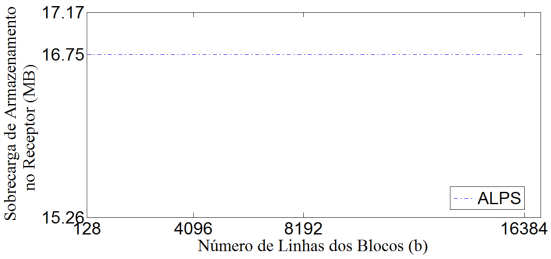
A Figura 6.5 apresenta a análise de sensibilidade realizada, tendo como base as métricas de desempenho (eixo y) e o **número de evidências por assinatura** (p), eixo x. A



(a) Custo computacional a partir da variação do tamanho das sub-matrizes do ALPS



(b) Sobrecarga de comunicação a partir da variação do tamanho das sub-matrizes do ALPS



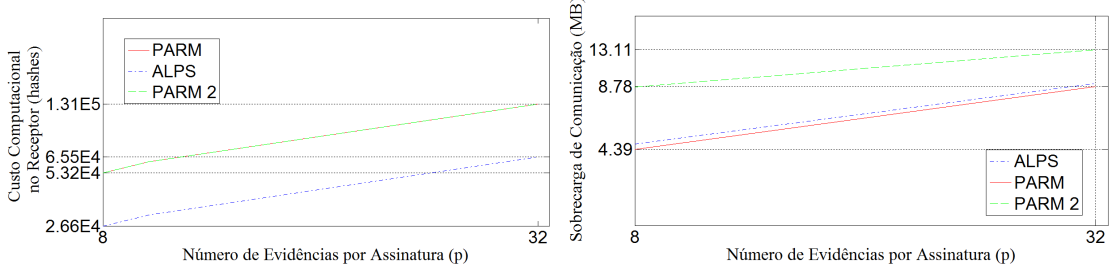
(c) Sobrecarga de armazenamento a partir da variação do tamanho das sub-matrizes do ALPS

Figura 6.4: Avaliação das métricas de desempenho a partir da variação do tamanho das sub-matrizes no ALPS e fixação dos demais parâmetros.

avaliação da variação de p apresenta o seguinte comportamento: o custo computacional (Figura 6.5 (a)) e a sobrecarga de comunicação (Figura 6.5 (b)) aumentam linearmente conforme o valor de p aumenta. Quanto maior for p, maior será o tamanho da assinatura. Sendo assim, o custo computacional aumenta devido à quantidade de evidências que precisam ser verificadas e a sobrecarga de comunicação aumenta devido a um maior número de evidências que precisam ser enviadas por assinatura pela rede. Enquanto isto, a sobrecarga de armazenamento no receptor (Figura 6.5 (c)) não sofre alterações devido à mudanças no número de evidências por assinatura pois o valor de p não influencia no tamanho dos *buffers* e no tamanho da chave pública, como pode ser observado nas Equações 5.20, 5.21 e 5.22 da Seção 5.3.

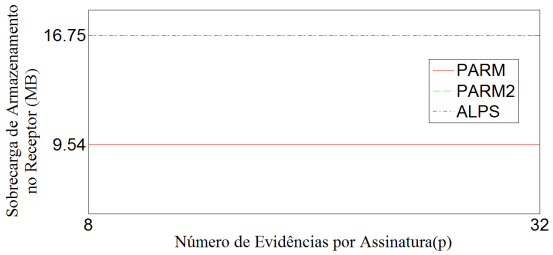
Os parâmetros de entrada dos esquemas foram descritos na Tabela 5.1 da Seção 4. Além desses, foi utilizado mais um parâmetro de entrada: nível mínimo de segurança. Trata-se do valor de nível de segurança (limite inferior) que não pode ser ultrapassado durante a transmissão. Caso a fonte tenha revelado evidências suficientes para que o nível de segurança seja inferior a esse limite, o esquema de assinatura não é mais considerado seguro. São utilizados três níveis mínimos de segurança durante as simulações: nível baixo (acima de 10^{15}); nível médio (acima de 10^{35}); e nível alto (acima de 10^{65}). Baseado nos valores indicados pelos autores dos esquemas, foram utilizados os seguintes conjuntos discretos de valores na avaliação dos parâmetros:

- colunas da matriz de evidências (k): [256, 512, 1024, 2048, 4096, 8192, 16384, 32768];
- linhas da matriz de evidências (q): [128, 256, 512, 1024, 2048, 4096, 8192, 16384];
- linhas da sub-matriz no ALPS (b): [128, 256, 512, 1024, 2048, 4096, 8192, 16384];
- evidências por assinatura (p): [8, 16, 32];



(a) Custo computacional a partir da variação do número de evidências por assinatura

(b) Sobrecarga de comunicação a partir da variação do número de evidências por assinatura



(c) Sobrecarga de armazenamento a partir da variação do número de evidências por assinatura

Figura 6.5: Avaliação das métricas de desempenho a partir da variação do número de evidências por assinatura e fixação dos demais parâmetros.

Baseado na análise das métricas de desempenho e no nível de segurança, o intervalo de valores foi reduzido já que algumas combinações apresentavam um nível de segurança muito baixo ou uma alta sobrecarga. Por exemplo, no PARM, o conjunto de valores possíveis de número de evidências por assinatura (p) foi alterado para [16, 32] quando o nível de segurança mínimo é alto (maior que 10^{65}). Outro exemplo é o intervalo de valores relativo ao número de colunas da matriz (k) que passou para [4096, 32768] quando considerado um alto nível de segurança. Sendo assim, após essas refatorações nos intervalos de valores, foram feitas as combinações entre os valores máximos e mínimos dos conjuntos de cada parâmetro, resultando assim nas quarenta **possíveis configurações** apresentadas nas Tabelas 6.4, 6.5 (PARM e PARM2); 6.6, 6.7, e 6.8 (ALPS).

Tabela 6.4: Configurações possíveis para o PARM e o PARM2 com um nível de segurança mínimo baixo e médio

Config	k	p	q
1	256	8	128
2	256	8	16384
3	256	32	128
4	256	32	16384
5	32768	8	128
6	32768	8	16384
7	32768	32	128
8	32768	32	16384

Caso o número de renovações parciais de chave de um esquema seja muito alto para uma determinada configuração, esta deve ser descartada por ser muito custosa tendo em vista a largura de banda necessária à autenticação. As Figuras 6.6 e 6.7 apresentam para

Tabela 6.5: Configurações possíveis para o PARM e o PARM2 com um nível de segurança mínimo alto

Config	k	p	q
9	1024	16	1024
10	1024	16	16384
11	1024	32	1024
12	1024	32	16384
13	32768	16	1024
14	32768	16	16384
15	32768	32	1024
16	32768	32	16384

Tabela 6.6: Configurações possíveis para o ALPS com um nível de segurança mínimo baixo

Config	k	p	b
17	256	8	128
18	256	8	16384
19	256	32	128
20	256	32	16384
21	32768	8	128
22	32768	8	16384
23	32768	32	128
24	32768	32	16384

Tabela 6.7: Configurações possíveis para o ALPS com um nível de segurança mínimo médio

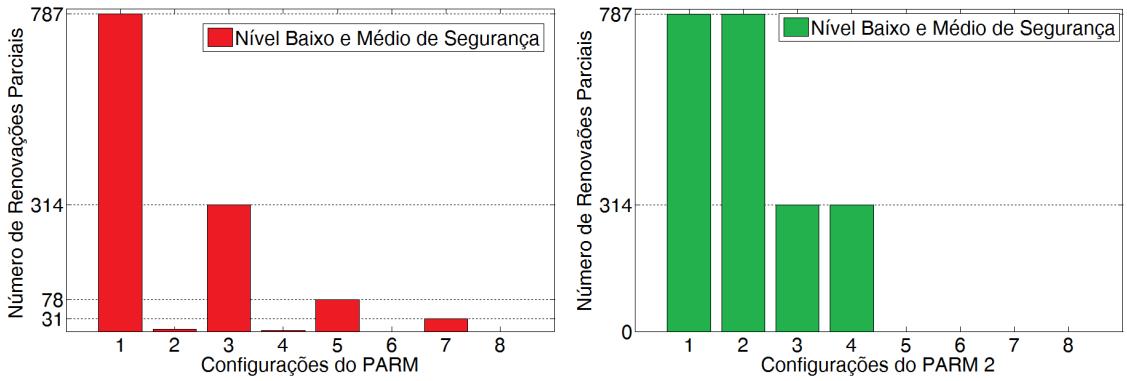
Config	k	p	b
25	256	16	128
26	256	16	16384
27	256	32	128
28	256	32	16384
29	32768	16	128
30	32768	16	16384
31	32768	32	128
32	32768	32	16384

Tabela 6.8: Configurações possíveis para o ALPS com um nível de segurança mínimo alto

Config	k	p	b
33	4096	16	1024
34	4096	16	16384
35	4096	32	1024
36	4096	32	16384
37	32768	16	1024
38	32768	16	16384
39	32768	32	1024
40	32768	32	16384

um cenário convencional o número de renovações parciais de chave pública necessárias aos esquemas PARM e PARM2, considerando, respectivamente, as primeiras 8 configurações e as 8 seguintes. Nestas figuras, no eixo x, são identificadas as 8 primeiras configurações (níveis baixo e médio de segurança) na Figura 6.6 e as 8 seguintes (nível alto) na Figura 6.7 (vide Tabelas 6.4 e 6.5). O eixo y representa o número de renovações parciais necessárias para cada configuração durante a transmissão.

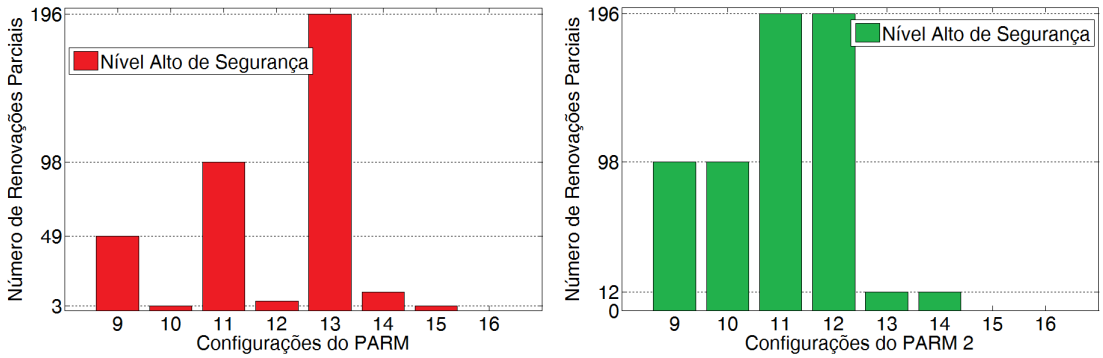
Além de avaliar o comportamento relativo à renovação parcial de chave, foram avaliadas todas as configurações possíveis para verificar quais conseguiam manter o nível de segurança mínimo em um cenário convencional. Nas figuras, contidas no Apêndice A para melhor organização, são apresentados todos os gráficos contendo essa avaliação, respeitando os níveis de segurança mínimo. Nos gráficos do Apêndice A, são apresentados os níveis de segurança dos esquemas de autenticação (eixo y, em escala logarítmica) em função do número de *chunks* enviados (eixo x, em escala linear).



(a) Renovações Parciais de Chave - PARM - Nível de Segurança Baixo e Médio

(b) Renovações Parciais de Chave - PARM 2 - Nível de Segurança Baixo e Médio

Figura 6.6: Número de renovações parciais de chave das configurações possíveis dos esquemas PARM e PARM2 em um cenário convencional com baixo e médio níveis de segurança



(a) Renovações Parciais de Chave - PARM - Nível de Segurança Alto

(b) Renovações Parciais de Chave - PARM 2 - Nível de Segurança Alto

Figura 6.7: Número de renovações parciais de chave das configurações possíveis dos esquemas PARM e PARM2 em um cenário convencional com alto nível de segurança

Algumas configurações foram descartadas da análise por não conseguirem manter o nível mínimo de segurança esperado ou por necessitarem de um alto número de renovações de chave. Como não existem avaliações reais prévias em outros trabalhos, foram descartadas em cada esquema as configurações com o maior número de renovações, ou seja, o pior caso para o mesmo nível de segurança. No caso do PARM, a configuração 1 foi descartada por necessitar de 787 renovações parciais de chave no cenário convencional de curta duração com nível de segurança mínimo baixo/médio e a configuração 13 no caso de uma transmissão com nível de seguraça mínimo alto (196 renovações parciais). Já no caso do PARM2, foram descartadas as configurações 1 e 2 (baixo e médio níveis de segurança); e 11 e 12 (alto nível de segurança). A seguir, todas as configurações descartadas são apresentadas:

- PARM: baixo nível de segurança (config 1); médio nível de segurança (configs 1, 2, 5 e 6); alto nível de segurança (configs 9, 10, 13 e 14);
- PARM2: baixo nível de segurança (configs 1, 2 e 5); médio nível de segurança (configs 1, 2, 5, 6 e 7); alto nível de segurança (configs 9, 10, 11, 12, 13 e 15);
- ALPS: baixo nível de segurança (configs 17, 18, 19 e 21); médio nível de segurança (configs 25, 26, 27, 28, 29, 31); alto nível de segurança (configs 33, 34, 35, 36, 37, 38 e 39).

Por fim, a partir desta análise prévia, é possível avaliar o desempenho dos esquemas leves de assinatura. Para isto, foram utilizadas as configurações apresentadas que não foram descartadas conforme explicado anteriormente (Tabelas 6.4, 6.5, 6.6, 6.7 e 6.8). Em relação às métricas de desempenho, foram analisados o custo computacional por *chunk* (operações de *hash*), sobrecarga de comunicação (em bytes) e sobrecarga de armazenamento (em kilobytes) para diferentes níveis mínimos de segurança (baixo, médio e alto). Seus resultados estão contidos nas Tabelas 6.9, 6.10 e 6.11.

Como conclusão dessa avaliação (contida nas Tabelas 6.9, 6.10 e 6.11), observou-se que o esquema de assinatura ALPS apresentou maior custo computacional no receptor que o PARM e o PARM2, o que pode influenciar na viabilidade da transmissão. Por exemplo, o menor custo apresentado pelo ALPS para um baixo nível de segurança foi 8192 *hashes* (configuração 23 da Tabela 6.6) enquanto que o PARM e PARM2 obtiveram 4096 *hashes*

Tabela 6.9: Sobrecargas dos Esquemas Leves de Assinatura com um nível de segurança mínimo baixo

Sobrecarga		**PARM**		**PARM2**		**ALPS**
Custo computacional por *chunk* (*hashes*)	Config 2:	131072	Config 3:	4096	Config 20:	1048576
	Config 3:	4096	Config 4:	524288	Config 22:	262144
	Config 4:	524288	Config 6:	131072	Config 23:	8192
	Config 5:	200121	Config 7:	4096	Config 24:	1048576
	Config 6:	131072	Config 8:	524288		
	Config 7:	4096				
	Config 8:	524288				
Sobrecarga de Comunicação por *chunk* (bytes)	Config 2:	32	Config 3:	671.44	Config 20:	132
	Config 3:	128	Config 4:	1210.89	Config 22:	36
	Config 4:	128	Config 6:	138134	Config 23:	132
	Config 5:	32	Config 7:	69181.05	Config 24:	132
	Config 6:	32	Config 8:	138230.11		
	Config 7:	128				
	Config 8:	128				
Sobrecarga de Armazenamento (*kilobytes*)	Config 2:	2.05	Config 3:	3	Config 20:	3
	Config 3:	1.53	Config 4:	3	Config 22:	3
	Config 4:	2.05	Config 6:	384	Config 23:	384
	Config 5:	192.43	Config 7:	384	Config 24:	384
	Config 6:	195.43	Config 8:	384		
	Config 7:	200.12				
	Config 8:	262.86				

Tabela 6.10: Sobrecargas dos Esquemas Leves de Assinatura com um nível de segurança mínimo médio

Sobrecarga		**PARM**		**PARM2**		**ALPS**
Custo computacional por *chunk* (*hashes*)	Config 3:	4096	Config 3:	4096	Config 30:	524288
	Config 4:	524288	Config 4:	524288	Config 32:	1048576
	Config 7:	4096	Config 8:	524288		
	Config 8:	524288				
Sobrecarga de comunicação por *chunk* (bytes)	Config 3:	128	Config 3:	671.44	Config 30:	68
	Config 4:	128	Config 4:	1210.89	Config 32:	132
	Config 7:	128	Config 8:	138230.11		
	Config 8:	128				
Sobrecarga de armazenamento (*kilobytes*)	Config 3:	1.53	Config 3:	3	Config 30:	384
	Config 4:	2.05	Config 4:	3	Config 32:	384
	Config 7:	200.12	Config 8:	384		
	Config 8:	262.86				

Tabela 6.11: Sobrecargas dos Esquemas Leves de Assinatura com um nível de segurança mínimo alto

Overhead		**PARM**		**PARM2**		**ALPS**
Custo computacional por *chunk* (*hashes*)	Config 11:	32768	Config 14:	262144	Config 40:	1048576
	Config 12:	524288	Config 16:	524288		
	Config 15:	32768				
	Config 16:	524288				
Sobrecarga de comunicação por *chunk* (bytes)	Config 11:	128	Config 14:	138166.11	Config 40:	132
	Config 12:	128	Config 16:	138230.11		
	Config 15:	128				
	Config 16:	128				
Sobrecarga de armazenamento (*kilobytes*)	Config 11:	7.01	Config 14:	384	Config 40:	384
	Config 12:	8.21	Config 16:	384		
	Config 15:	224.33				
	Config 16:	262.86				

como valor mínimo (configurações 3 e 7 da Tabela 6.4), como pode ser visto na Tabela 6.9. Por outro lado, PARM2 apresentou a maior sobrecarga de comunicação entre os esquemas leves devido aos valores da tabela de uso que devem ser enviados na mensagem, o que torna necessário um maior uso de largura de banda. Além disso, independente do nível mínimo de segurança, o maior impacto de sobrecarga de armazenamento é proporcionado pelo esquema de assinatura PARM2, fazendo com que os receptores precisem ter uma capacidade de armazenamento maior de dados.

6.4 Uso Combinado dos Esquemas de Amortização e de Assinatura Leve

Para verificar a viabilidade da autenticação em sistemas P2P de *live streaming*, tanto em cenários convencionais quanto em cenários de alta definição, torna-se necessário testar a forma mais rápida e menos custosa de autenticação de dados. Tendo isto em mente, foram analisados os resultados das Seções 6.2 e 6.3 para selecionar o esquema de amortização de assinatura digital e o esquema de assinatura leve mais eficientes. Desta forma, é possível testar tanto a eficiência de cada um individualmente quanto à de uma solução conjunta dos mesmos. No caso da solução em conjunto, o esquema selecionado da Seção 6.2 irá amortizar a assinatura do esquema da Seção 6.3.

O primeiro desafio foi escolher quais seriam os esquemas que deveriam ter seu desempenho avaliado. O esquema de amortização escolhido foi o Linear Digests. Embora o resultado da avaliação de algumas métricas do esquema Butterfly Graph Chaining tenha sido semelhante, o Linear Digests se mostrou mais vantajoso tendo em vista o menor tamanho de *buffer* nos receptores e a menor sobrecarga de comunicação (Seção 6.2). Já em relação aos esquemas leves, a questão decisiva na escolha do esquema mais eficiente foi a definição do que era mais importante: custo computacional ou sobrecarga de comunicação. Obviamente, este tipo de decisão está relacionada diretamente ao tipo de aplicação que se deseja utilizar. Para um receptor de baixa capacidade computacional, seria mais danoso ter um alto custo computacional na verificação de autenticidade. Sendo assim, selecionamos o esquema PARM2 pois ele apresentou um menor custo computacional que o esquema ALPS, conforme pode ser visto nos resultados apresentados na Seção 6.3.

O segundo desafio encontrado foi modelar as métricas mais relevantes da solução conjunta através de equações. A avaliação da solução combinada do PARM2 e Linear Digests (chamada de PARM2LD no presente trabalho) pode ser realizada com base nas equações da Tabela 6.12, que representam as métricas sobrecarga de comunicação por *chunk* (em bytes) e custo computacional por *chunk* (operações de *hash*).

A equação relativa à sobrecarga de comunicação por *chunk* de PARM2LD foi obtida a partir das Equações 5.3 (sobrecarga de comunicação por bloco de *chunks* do esquema de amortização Linear Digests, na Seção 5.2) e 5.18 (sobrecarga de comunicação por *chunk* do esquema de assinatura leve PARM2, na Seção 5.3) do Capítulo 5. Para um *chunk*, é necessário considerar a chave pública que deve ser distribuída de modo seguro (uma chave pública contém k valores de c bytes cada), a assinatura do *chunk* (p evidências por assinatura de c bytes cada), os valores da tabela de uso que devem ser enviados junto a mensagem (k valores de $\lg q$ bytes) e um número de sequência (i bytes), o que deve ser normalizado pelo número de *chunks* de um bloco (n).

Já a equação que diz respeito ao custo computacional por *chunk* de PARM2LD foi desenvolvida com base nas Equações 5.1 (custo computacional por bloco de *chunks* do esquema de amortização Linear Digests, na Seção 5.2) e 5.15 (custo computacional por

chunk do esquema de assinatura leve PARM2, na Seção 5.3) do Capítulo 5. Para verificar a autenticidade de um *chunk*, deve-se fazer o *hash* do *chunk* (L bytes de 64 em 64 bytes) e q *hashes* para cada evidência contida na assinatura (cada assinatura contém p evidências), o que deve ser normalizado por n.

A partir dos parâmetros de entrada dessas fórmulas (contidas na Tabela 6.12), é possível verificar o desempenho dessa solução conjunta. Caso haja avanços tecnológicos no hardware de mercado, basta ajustar os valores dos parâmetros de entrada, obtendo assim o desempenho atual da solução. Na próxima seção, a solução conjunta PARM2LD é avaliada com base nos valores para os parâmetros de entrada do esquema de amortização Linear Digests, contidos na Tabela 6.2 (Seção 6.2), e nas configurações apresentadas nas Tabelas 6.4 e 6.5 (Seção 6.3) para o esquema de assinatura leve PARM2.

Tabela 6.12: Métricas de Desempenho da Solução Conjunta entre o PARM2 e o Linear Digests

Métrica	**Equação**
Sobrecarga de Comunicação por *Chunk*	$CO_{PARM2LD} = (k \times c + p \times c + k \times \log q + i)/n$
Custo Computacional por *Chunk*	$CC_{PARM2LD} = (q \times p)/n + L/64$

6.5 Instanciação do Modelo considerando Hardware Convencional

Os resultados da avaliação do custo computacional no receptor tanto do PARM2 e Linear Digests quanto da solução conjunta (PARM2LD), nos quatro cenários apresentados na Seção 6.1 e com três diferentes níveis de segurança mínima (Seção 6.3), estão contidos na Tabela 6.13. Na Tabela 6.14, apresenta-se a análise das métricas de desempenho da solução conjunta.

Primeiramente, foi avaliada a viabilidade de uso dessas soluções nos cenários estudados. Para tal avaliação, foi definido no presente trabalho um coeficiente de viabilidade β que consiste na divisão do custo computacional total pelo tempo de transmissão total. Sendo assim, se β é maior ou igual a 1, a solução de autenticação é garantidamente inviável, e valores menores do que 1 podem ser viáveis. Pode-se estimar que valores próximos de 1 também sejam inviáveis porque há outros custos computacionais a considerar, como decodificação do vídeo. Os resultados dessa avaliação também estão apresentados na Tabela 6.13. Como conclusão, foi possível observar os seguintes pontos:

- entre as três soluções estudadas (PARM2, Linear Digests e PARM2LD), considerando um nível baixo ou médio de segurança, a solução conjunta (PARM2LD) apresentou o menor custo computacional, o que foi verificado através das configurações 3 (baixo e médio níveis de segurança) e 7 (baixo nível de segurança) da Tabela 6.4. Já para um nível alto de segurança, o melhor desempenho foi alcançado pelo esquema de amortização Linear Digests através da utilização do esquema clássico de assinatura digital RSA;

- o esquema de amortização Linear Digests, se utilizado individualmente, apenas pode ser usado na autenticação de transmissões convencionais. Em transmissões de alta definição, os valores de β calculados são 0.9 e 1. Conforme fora explicado anteriormente, valores próximos de 1 e superiores a 1 indicam a inviabilidade do esquema no cenário em questão;

- o esquema de assinatura leve PARM2, se utilizado individualmente, não apresenta valores de coeficiente (β) aceitáveis para os cenários estudados nem convencionais nem de alta definição. Na avaliação do mesmo, foram utilizadas as configurações 3, 4, 6, 7, 8, 14 e 16 contidas nas Tabelas 6.4 e 6.5 da Seção 6.3;

- considerando os esquemas estudados, em cenários de alta definição, a autenticação não é viável nem mesmo com baixos níveis de segurança. Sendo assim, é possível concluir que os esquemas atuais são incapazes de lidar com as demandas de um sistema P2P de *live streaming* para alta definição no que concerne autenticação de conteúdo. Quando esses sistemas estiverem em uso, serão vulneráveis a ataques de poluição de conteúdo.

Tabela 6.13: Avaliação do Custo Computacional Total

Esquema	**Seg.**		**Conv. curta**	β	**HD curta**	β	**Conv. longa**	β	**HD longa**	β
LDigests	Alto		3 min	0.3	9 min	0.9	25 min	0.21	2 horas	1
PARM2	Baixo	C-3,7:	3 horas	17.2	12 horas	72	2 dias	17.2	6 dias	72
		C-6:	4 dias	576	16 dias	2304	46 dias	552	7 meses	2277
		C-4,8:	16 dias	2300	64 dias	9200	183 dias	2200	2 anos	9100
	Médio	C-3:	3 horas	17.2	12 horas	72	2 dias	17.2	6 dias	72
		C-4,8:	16 dias	2300	64 dias	9200	183 dias	2200	2 anos	9100
	Alto	C-14:	8 dias	1100	32 dias	4500	91 dias	1100	1 ano	4500
		C-16:	16 dias	2300	64 dias	9200	183 dias	2200	2 anos	9100
PARM2LD	Baixo	C-3,7:	2.35 min	0.24	9.78 min	0.98	28.22 min	0.23	2 horas	1
		C-6:	44 min	4.4	3 horas	18	9 horas	4.5	2 dias	24
		C-4,8:	2.53 horas	17.34	11.57 horas	71.7	2 dias	24	6 dias	72
	Médio	C-3:	2.35 min	0.24	9.78 min	0.98	28.22 min	0.23	2 horas	1
		C-4,8:	2.53 horas	17.34	11.57 horas	71.7	2 dias	24	6 dias	72
	Alto	C-14:	1.45 horas	8.35	6.01 horas	35.35	17.40 horas	8.7	3 dias	36
		C-16:	2.53 horas	17.34	11.57 horas	71.7	2 dias	24	6 dias	72

Finalmente, foram analisadas as métricas de desempenho (custo computacional por *chunk* e sobrecarga de comunicação por *chunk*) da solução conjunta PARM2LD para diferentes níveis de segurança, o que pode ser observado na Tabela 6.14. Para isto, foram utilizadas as equações contidas na Tabela 6.12 (Seção 6.4) e os valores dos parâmetros de entrada das Tabelas 6.2 (Seção 6.2) e 6.4 (Seção 6.3). A partir desses resultados, indica-se o uso das configurações 3 e 7 do PARM2 caso o objetivo seja manter um nível de segurança baixo ou médio durante uma transmissão pois elas obtiveram os menores custos e sobrecargas entre as configurações estudadas neste contexto.

Tabela 6.14: Avaliação de Desempenho da Solução PARM2LD

Nível de segurança	**Baixo**		**Médio**		**Alto**	
Custo Computacional por *Chunk* (*hashes*)	Config 3:	56	Config 3:	56	Config 14:	2071
	Config 4:	4120	Config 4:	4120	Config 16:	4110
	Config 6:	1047	Config 8:	4110		
	Config 7:	56				
	Config 8:	4110				
Sobrecarga de Comunicação por *Chunk* (*bytes*)	Config 3:	13.24	Config 3:	13.24	Config 14:	2103.42
	Config 4:	17.46	Config 4:	17.46	Config 16:	2104
	Config 6:	2103.42	Config 8:	1060		
	Config 7:	1564.47				
	Config 8:	1060				

7 CONCLUSÕES E OPORTUNIDADES DE PESQUISA

As aplicações de *live streaming* em redes P2P vem sendo cada vez mais utilizadas, chegando a atrair milhões de usuários. Este tipo de aplicação pode transmitir vídeos convencionais ou de alta definição. Transmissões em alta definição são uma tendência do mercado, porém ainda não foram exploradas no contexto de *live streaming* em redes P2P. Neste trabalho, foram avaliados, em igualdade de condições, o desempenho dos principais esquemas de assinatura digital sob essas perspectivas para combater o ataque de poluição de conteúdo em aplicações P2P de *live streaming*. Para isto, foi feita uma comparação quantitativa de custos, sobrecargas e nível de segurança durante transmissões convencionais e em alta definição.

A partir da análise quantitativa realizada, o esquema leve de assinatura PARM2 obteve o melhor desempenho. Já em relação aos esquemas de amortização, o Linear Digests apresentou os menores custos e sobrecargas. Inicialmente, os esquemas mais eficientes foram selecionados e avaliados individualmente. O esquema de amortização Linear Digests, quando utilizado individualmente, apenas pode ser utilizado na autenticação em transmissões convencionais. E o esquema PARM2, quando utilizado individualmente, não apresentou um desempenho aceitável para os cenários estudados a partir da análise de sensibilidade realizada no presente trabalho. Por fim, foi analisada uma solução conjunta desses esquemas. A partir dessa análise em particular, foi possível observar que a autenticação é viável apenas com baixos e médios níveis de segurança, em cenários convencionais.

Sendo assim, foi possível concluir que os esquemas atuais são incapazes de lidar com as demandas de um sistema P2P de *live streaming* para alta definição no que concerne autenticação de conteúdo. Quando esses sistemas estiverem em uso, serão vulneráveis a ataques de poluição de conteúdo. Para atingir um nível de segurança aceitável nos cenários de alta definição, seria necessário ajustar os parâmetros levando a um aumento substancial das sobrecargas de processamento, comunicação e armazenamento. Em particular, no caso do PARM e PARM2, o maior impacto seria na sobrecarga de comunicação, devido ao maior número de renovações. No caso do ALPS, seria necessário aumentar substancialmente a matriz de evidências, o que resultaria em um alto custo computacional nos receptores devido às operações de *hash* para verificar assinaturas.

Como trabalhos futuros, duas ações imediatas são logo identificadas. A primeira, a implementação e avaliação experimental de cada esquema, permitindo uma avaliação mais precisa (embora os desafios e dificuldades associados às implementações sejam óbvios). A segunda e mais importante ação, um desafio proposto à comunidade de pesquisa, é a investigação de novos algoritmos de assinatura digital eficientes para *live streaming* de alta definição que proporcionem um nível de segurança aceitável.

REFERÊNCIAS

BARCELLOS, M. P.; GASPARY, L. P. Segurança em Redes P2P: princípios, tecnologias e desafios. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC 2006, Curitiba, PR. **Anais...** [S.l.: s.n.], 2006.

BELLARE, M.; ROGAWAY, P. Random oracles are practical: a paradigm for designing efficient protocols. In: CCS '93: PROCEEDINGS OF THE 1ST ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY, New York, NY, USA. **Anais...** ACM, 1993. p.62–73.

BORGES, A.; ALMEIDA, J.; CAMPOS, S. Combate a Poluição em Sistemas P2P de Mídia Contínua ao Vivo. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC). **Anais...** SBC, 2008.

CANETTI, R. et al. Multicast security: a taxonomy and some efficient constructions. In: INFOCOM '99. EIGHTEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES. PROCEEDINGS. IEEE. **Anais...** [S.l.: s.n.], 1999. v.2, p.708–716.

CHALLAL, Y.; BETTAHAR, H.; BOUABDALLAH, A. A taxonomy of multicast data origin authentication: issues and solutions. **IEEE Communications Surveys and Tutorials**, [S.l.], v.6, n.1-4, p.34–57, 2004.

CIRULLI, A.; DI PIETRO, R. PEAC: a probabilistic, efficient, and resilient authentication protocol for broadcast communications. In: SECURECOMM '08: PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON SECURITY AND PRIVACY IN COMMUNICATION NETOWRKS, New York, NY, USA. **Anais...** ACM, 2008. p.1–10.

CISCO. **Cisco Digital Media Systems Solution Overview**. Disponível em: `http://www.cisco.com`. Acesso em: 24 maio 2011.

COSTA, C. et al. Fighting pollution dissemination in peer-to-peer networks. In: SAC '07: PROCEEDINGS OF THE 2007 ACM SYMPOSIUM ON APPLIED COMPUTING, New York, NY, USA. **Anais...** ACM, 2007. p.1586–1590.

DHUNGEL, P. et al. The Pollution Attack in P2P Live Video Streaming: measurement results and defenses. In: WORKSHOP ON PEER-TO-PEER STREAMING AND IP-TV, P2P-TV '07. **Anais...** ACM, 2007. p.323–328.

FU, W.; JAIN, S.; VICENTE, M. R. **Global Broadband Quality Study Shows Progress, Highlights Broadband Quality Gap**. Disponível em: `http://www.cisco.com`. Acesso em: 24 maio 2011.

GENNARO, R.; ROHATGI, P. How to Sign Digital Streams. In: **Advances in Cryptology**: 17th annual international cryptology conference. [S.l.]: Springer Berlin, 1997. p.180–197. (Lecture Notes in Computer Science, v.1294).

GOLDREICH, O.; GOLDWASSER, S.; MICALI, S. How to construct random functions. **J. ACM**, New York, NY, USA, v.33, p.792–807, August 1986.

HABIB, A. et al. A Tree-Based Forward Digest Protocol to Verify Data Integrity in Distributed Media Streaming. **IEEE Transactions on Knowledge and Data Engeneering**, [S.l.], v.17, n.7, p.1010–1014, 2005.

HARIDASAN, M.; RENESSE, R. van. SecureStream: an intrusion-tolerant protocol for live-streaming dissemination. **Comput. Commun.**, Newton, MA, USA, v.31, p.563–575, February 2008.

HEFEEDA, M.; MOKHTARIAN, K. Authentication schemes for multimedia streams: quantitative analysis and comparison. **ACM Transactions on Multimedia Computing, Communication, and Applications**, New York, NY, USA, v.6, p.6:1–6:24, February 2010.

HEI, X. et al. Insights into PPLive: a measurement study of a large-scale p2p iptv system. In: IN PROCEEDINGS OF IPTV WORKSHOP, INTERNATIONAL WORLD WIDE WEB CONFERENCE. **Anais...** [S.l.: s.n.], 2006.

HEI, X. et al. A measurement study of a large-scale P2P IPTV system. **Multimedia, IEEE Transactions on**, [S.l.], v.9, n.8, December 2007.

JIN, X. et al. Detecting Malicious Hosts in the Presence of Lying Hosts in Peer-to-Peer Streaming. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA EXPO (ICME). **Proceedings...** IEEE, 2006. p.1537–40.

LIAO, X. et al. AnySee: peer-to-peer live streaming. In: INFOCOM. **Proceedings...** [S.l.: s.n.], 2006. p.1–10.

LIN, W. W.; SHIEH, S.; LIN, J.-C. A Pollution Attack Resistant Multicast Authentication Scheme Tolerant to Packet Loss. In: SSIRI '08: INTERNATIONAL CONFERENCE ON SECURE SYSTEM INTEGRATION AND RELIABILITY IMPROVEMENT. **Anais...** IEEE Computer Society, 2008. p.8–15.

LIN, Y.-J.; SHIEH, S.; LIN, W. W. Lightweight, Pollution-attack Resistant Multicast Authentication Scheme. In: ACM SYMPOSIUM ON INFORMATION, COMPUTER AND COMMUNICATIONS SECURITY, ASIACCS '06. **Anais...** ACM, 2006. p.148–156.

LYSYANSKAYA, A.; TAMASSIA, R.; TRIANDOPOULOS, N. Multicast Authentication in Fully Adversarial Networks. **Security and Privacy, IEEE Symposium on**, Los Alamitos, CA, USA, v.0, p.241, 2004.

LYSYANSKAYA, A.; TAMASSIA, R.; TRIANDOPOULOS, N. Authenticated error-correcting codes with applications to multicast authentication. **ACM Trans. Inf. Syst. Secur.**, New York, NY, USA, v.13, p.17:1–17:34, March 2010.

MAGHAREI, N.; REJAIE, R.; GUO, Y. Mesh or multiple-tree: a comparative study of live p2p streaming approaches. In: INFOCOM 2007: 26TH IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS. **Anais...** IEEE, 2007. p.1424–1432.

MARQUARDT, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. **SIAM Journal on Applied Mathematics**, [S.l.], v.11, n.2, p.431–441, 1963.

MEIER, R.; WATTENHOFER, R. ALPS: authenticating live peer-to-peer streams. In: SRDS '08: IEEE SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS, Los Alamitos, CA, USA. **Anais...** IEEE Computer Society, 2008. p.45–52.

MINER, S.; STADDON, J. Graph-Based Authentication of Digital Streams. In: SP '01: PROCEEDINGS OF THE 2001 IEEE SYMPOSIUM ON SECURITY AND PRIVACY, Washington, DC, USA. **Anais...** IEEE Computer Society, 2001. p.232.

MORAES, I. M. et al. Distribuição de Vídeo sobre Redes Par-a-Par: arquiteturas, mecanismos e desafios. **Simpósio Brasileiro de Redes de Computadores (SBRC)**, [S.l.], 2008.

PANNETRAT, A.; MOLVA, R. Efficient Multicast Packet Authentication. In: NDSS '03: NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM. **Anais...** Internet Society, 2003. p.1–12.

PARK, J. M.; CHONG, E. K. P.; SIEGEL, H. J. Efficient Multicast Packet Authentication Using Signature Amortization. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 2002. **Anais...** IEEE Computer Society, 2002. p.227–240.

PARK, J. M.; CHONG, E. K. P.; SIEGEL, H. J. Efficient multicast stream authentication using erasure codes. **ACM Transactions on Information and System Security**, [S.l.], v.6, n.2, p.258–285, 2003.

PARK, Y.; CHO, Y. The eSAIDA Stream Authentication Scheme. In: COMPUTATIONAL SCIENCE AND ITS APPLICATIONS - ICCSA. **Anais...** Springer, 2004. p.799–807. (Lecture Notes in Computer Science, v.3046).

PERRIG, A. The BiBa One-time Signature and Broadcast Authentication Protocol. In: CCS '01: 8TH ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY. **Anais...** ACM, 2001. p.28–37.

PERRIG, A.; CANETTI, R.; WATSON, I. T. J. Efficient authentication and signing of multicast streams over lossy channels. In: IN IEEE SYMPOSIUM ON SECURITY AND PRIVACY, Washington, DC, USA. **Anais...** IEEE Computer Society, 2000. p.56–73.

PERRIG, A. et al. The TESLA Broadcast Authentication Protocol. **RSA CryptoBytes**, [S.l.], v.5, n.Summer, p.2–13, 2002.

PIATEK, M. et al. Contracts: practical contribution incentives for p2p live streaming. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION - NSDI, 7. **Anais...** USENIX Association, 2010. p.81–94.

POYNTON, C. **Digital Video and HDTV Algorithms and Interfaces**. 1.ed. [S.l.]: Morgan Kaufmann Publishers, 2003.

RABIN, M. O. Efficient dispersal of information for security, load balancing, and fault tolerance. **Journal of the ACM**, New York, NY, USA, v.36, n.2, p.335–348, April 1989.

REDIESS, F. K. et al. Projeto de Hardware para a Compensação de Movimento do Padrão H.264/AVC de Compressão de Vídeo. **XIII Simpósio de Informática - Revista Hífen**, [S.l.], v.30, n.58, 2006.

REED, I. S.; SOLOMON, G. Polynomial Codes Over Certain Finite Fields. **Journal of the Society for Industrial and Applied Mathematics**, [S.l.], v.8, n.2, p.300–304, 1960.

REYZIN, L.; REYZIN, N. Better than BiBa: short one-time signatures with fast signing and verifying. In: ACISP '02: PROCEEDINGS OF THE 7TH AUSTRALIAN CONFERENCE ON INFORMATION SECURITY AND PRIVACY. **Anais...** Springer-Verlag, 2002. p.144–153.

RICHARDSON, I. E. **H.264 and MPEG-4 Video Compression**: video coding for next generation multimedia. 1.ed. New York, NY, USA: Wiley, 2003.

SILVA, A. P. C. da et al. Chunk Distribution in Mesh-Based Large-Scale P2P Streaming Systems: a fluid approach. **IEEE Transactions on Parallel and Distributed Systems**, Los Alamitos, CA, USA, v.22, p.451–463, 2011.

STANDARDISATION, I. O. for. **Short MPEG-2 description**. Disponível em: `http://mpeg.chiariglione.org/standards/mpeg-2/mpeg-2.htm`. Acesso em: 24 maio 2011.

STANDARDISATION, I. O. for. **Overview of the MPEG-4 Standard**. Disponível em: `http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm`. Acesso em: 24 maio 2011.

TOPIWALA, P.; RAJU, A.; SINGH, D. A Real-time H.264 High 4:4:4 codec. In: APPLICATIONS OF DIGITAL IMAGE PROCESSING XXXII. **Anais...** SPIE, 2009. v.7443, p.74430Z.

WONG, C. K.; LAM, S. S. Digital signatures for flows and multicasts. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.7, n.4, p.502–513, August 1999.

YANG, S. et al. The Content Pollution in Peer-to-Peer Live Streaming Systems: analysis and implications. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING. **Anais...** IEEE Computer Society, 2008. p.652–659.

ZHANG, X. et al. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In: IEEE INFOCOM. **Anais...** [S.l.: s.n.], 2005. v.3, p.2102–2111.

ZHANG, Z. et al. Stream Authentication Based on Generlized Butterfly Graph. In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING - ICIP. **Anais...** IEEE, 2007. p.121–124.

ZHANG, Z. et al. Generalized butterfly graph and its application to video stream authentication. **IEEE Trans. Cir. and Sys. for Video Technol.**, Piscataway, NJ, USA, v.19, n.7, p.965–977, 2009.

ZHANG, Z.; SUN, Q.; WONG, W.-C. A Proposal of Butterfly-graph Based Stream Authentication over Lossy Networks. In: ICME '05: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO. **Anais...** IEEE Computer Society, 2005. p.1–4.

APÊNDICE A NÍVEL DE SEGURANÇA MÍNIMO DOS ESQUEMAS LEVES DE ASSINATURA

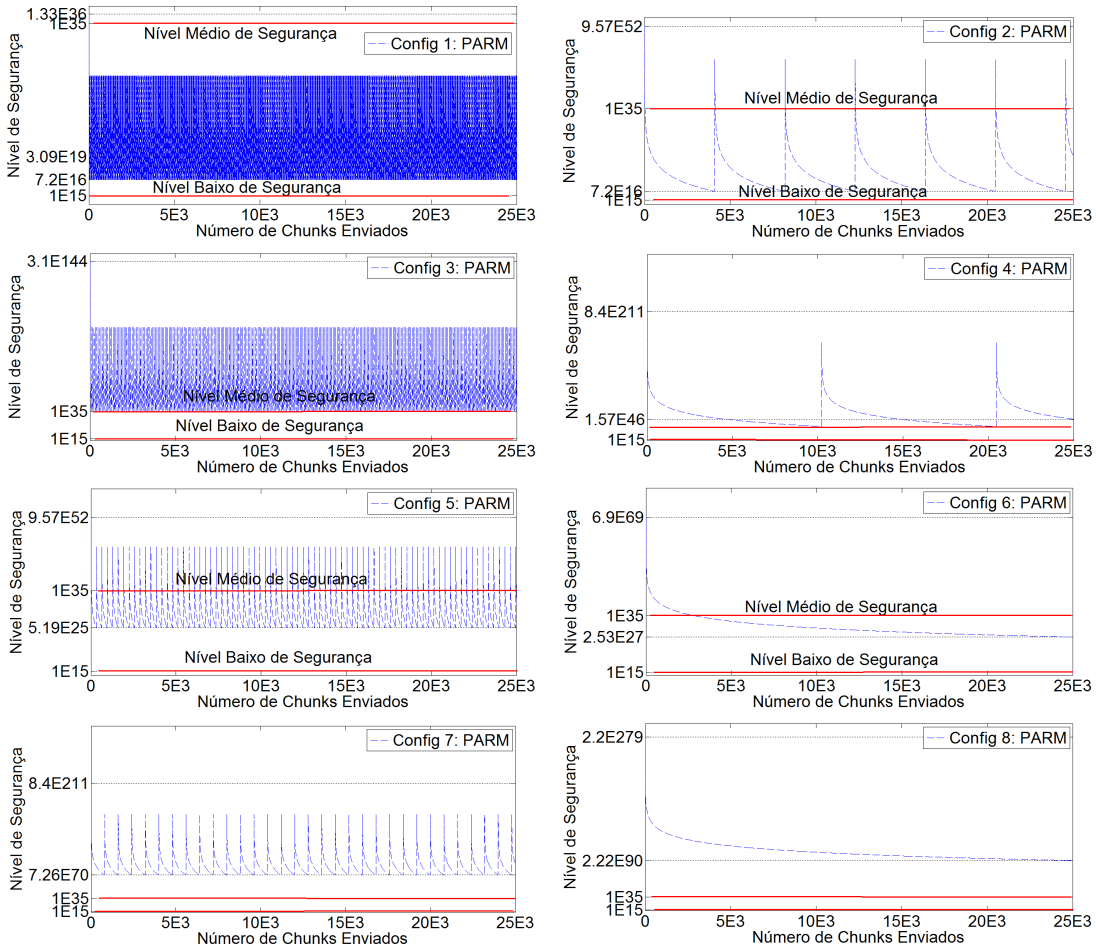


Figura A.1: Análise de segurança de nível médio e baixo das configurações possíveis do esquema PARM em um cenário convencional

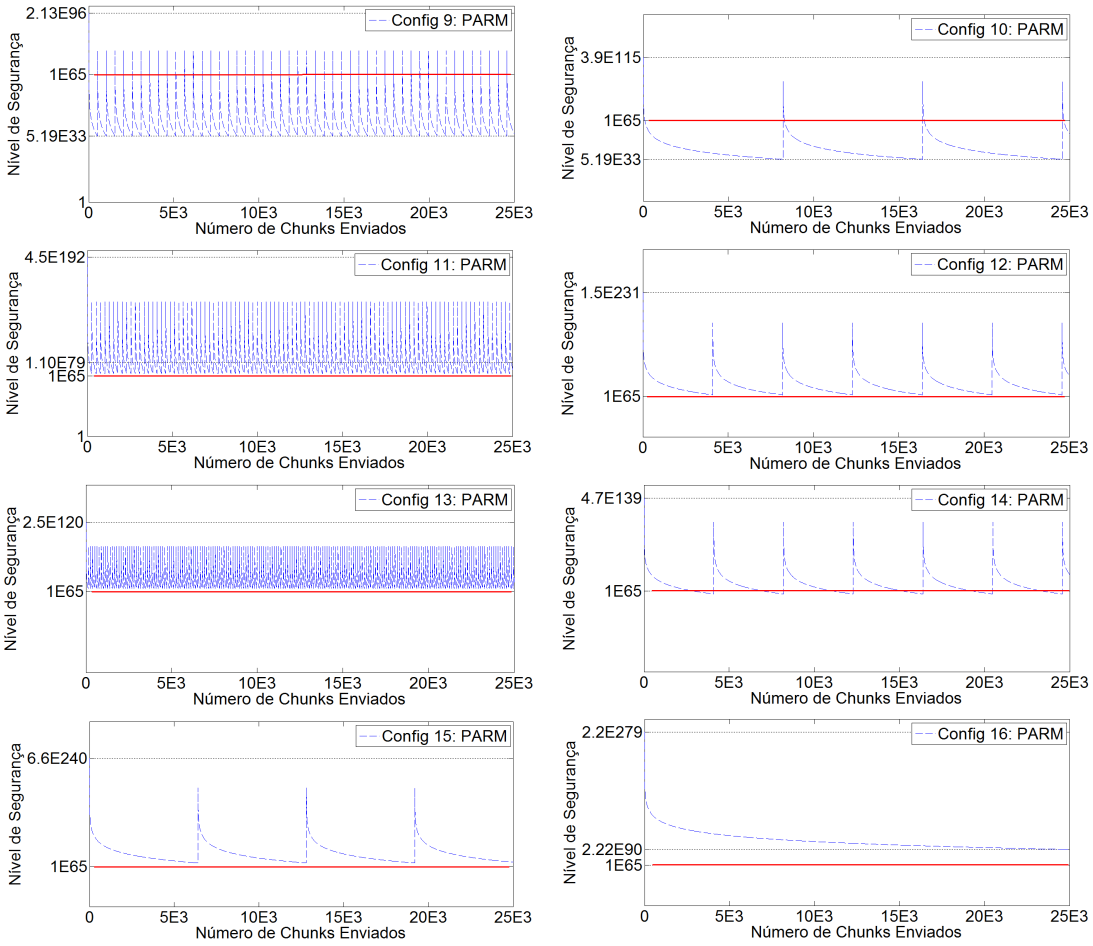


Figura A.2: Análise de segurança de nível alto das configurações possíveis do esquema PARM em um cenário convencional

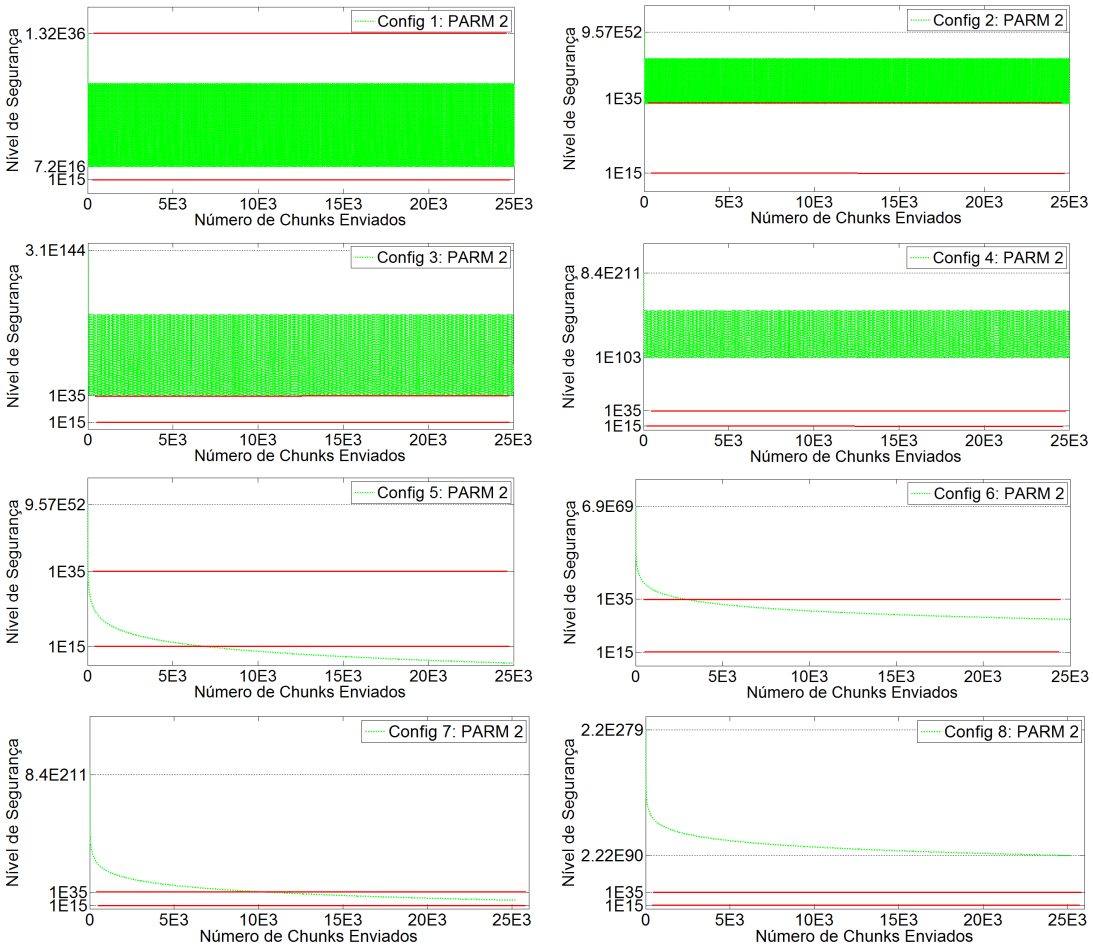


Figura A.3: Análise de segurança de nível médio e baixo das configurações possíveis do esquema PARM 2 em um cenário convencional

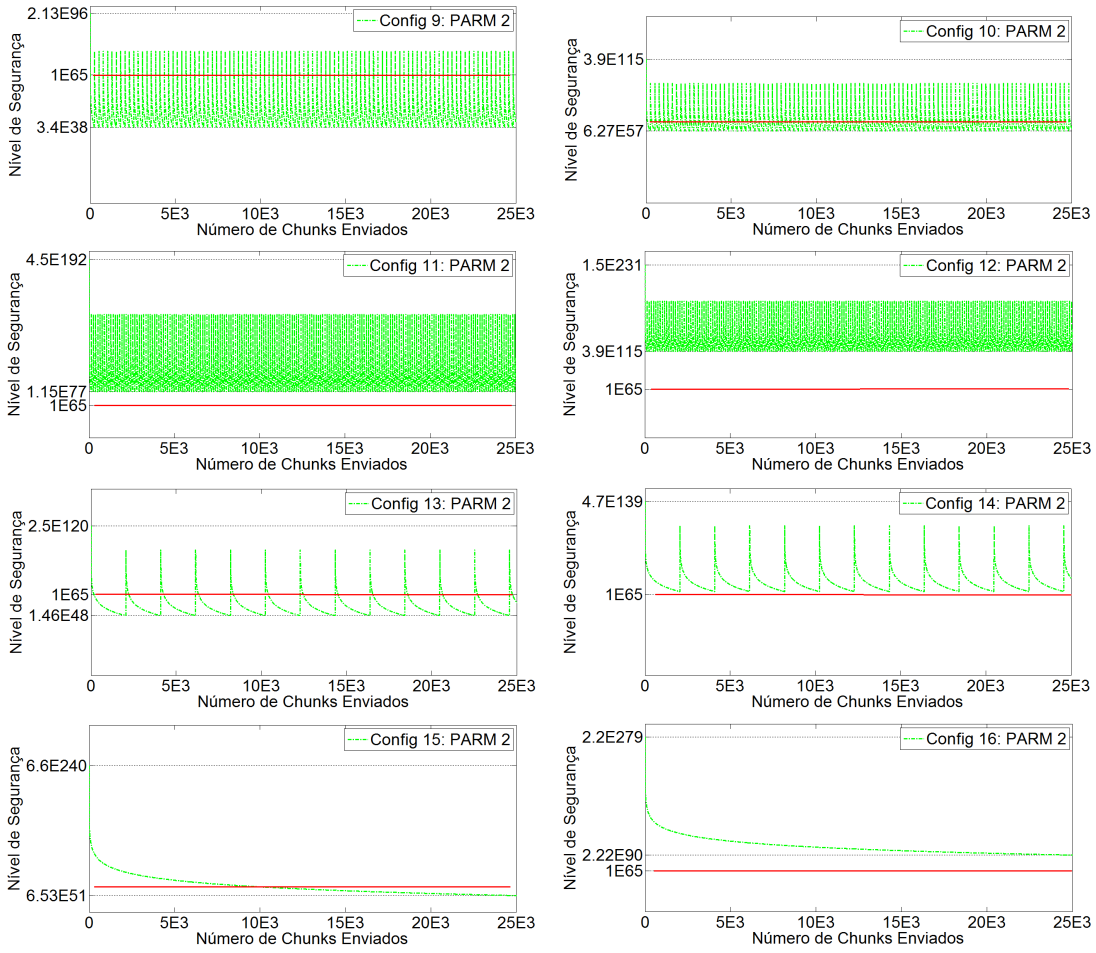


Figura A.4: Análise de segurança de nível alto das configurações possíveis do esquema PARM 2 em um cenário convencional

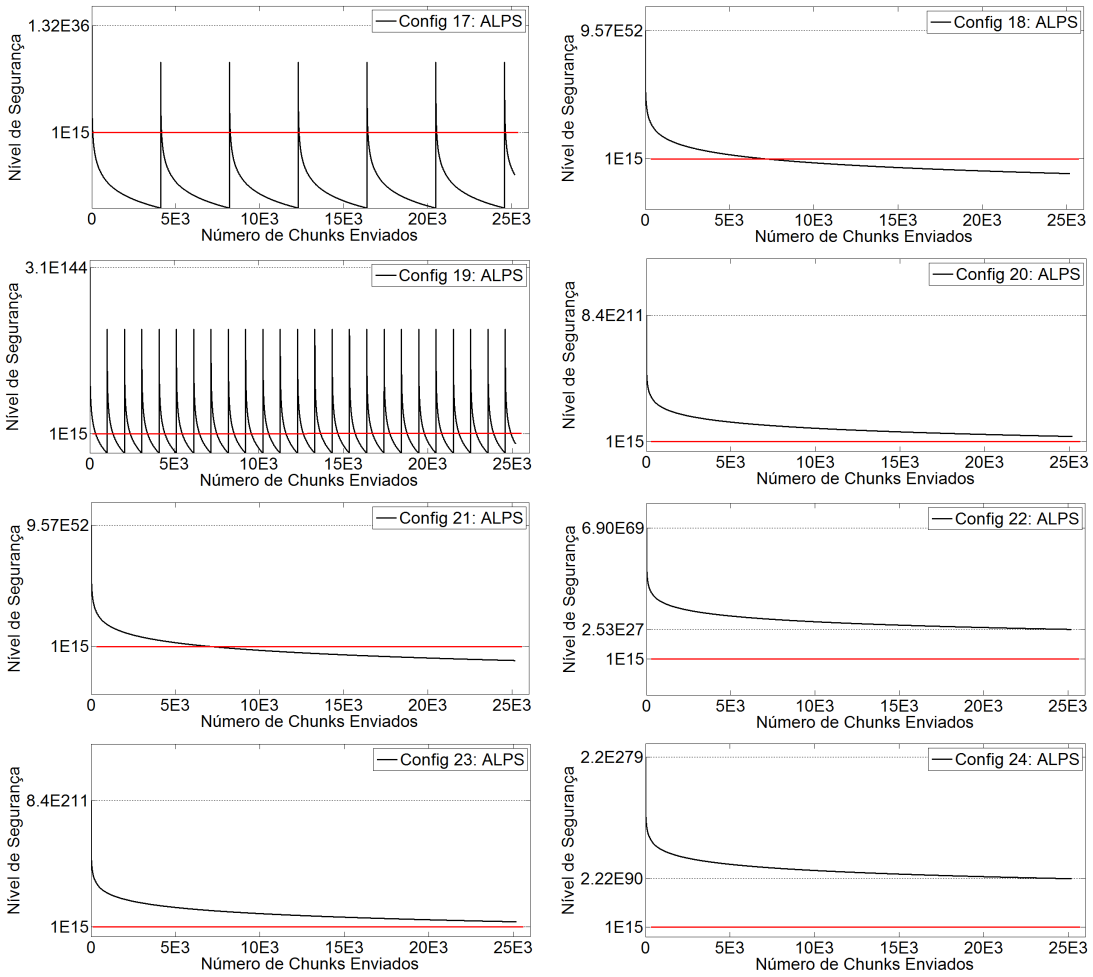


Figura A.5: Análise de segurança de nível baixo das configurações possíveis do esquema ALPS em um cenário convencional

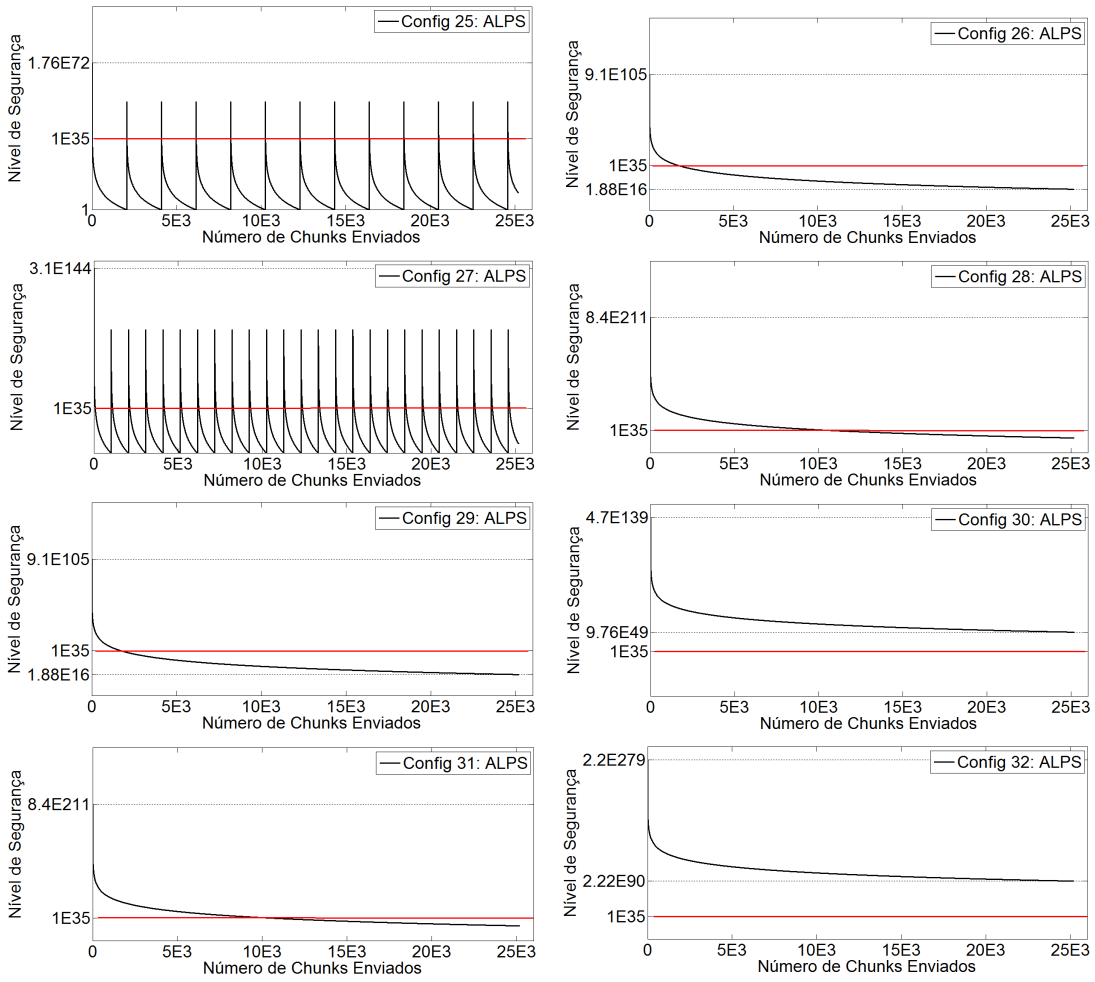


Figura A.6: Análise de segurança de nível médio das configurações possíveis do esquema ALPS em um cenário convencional

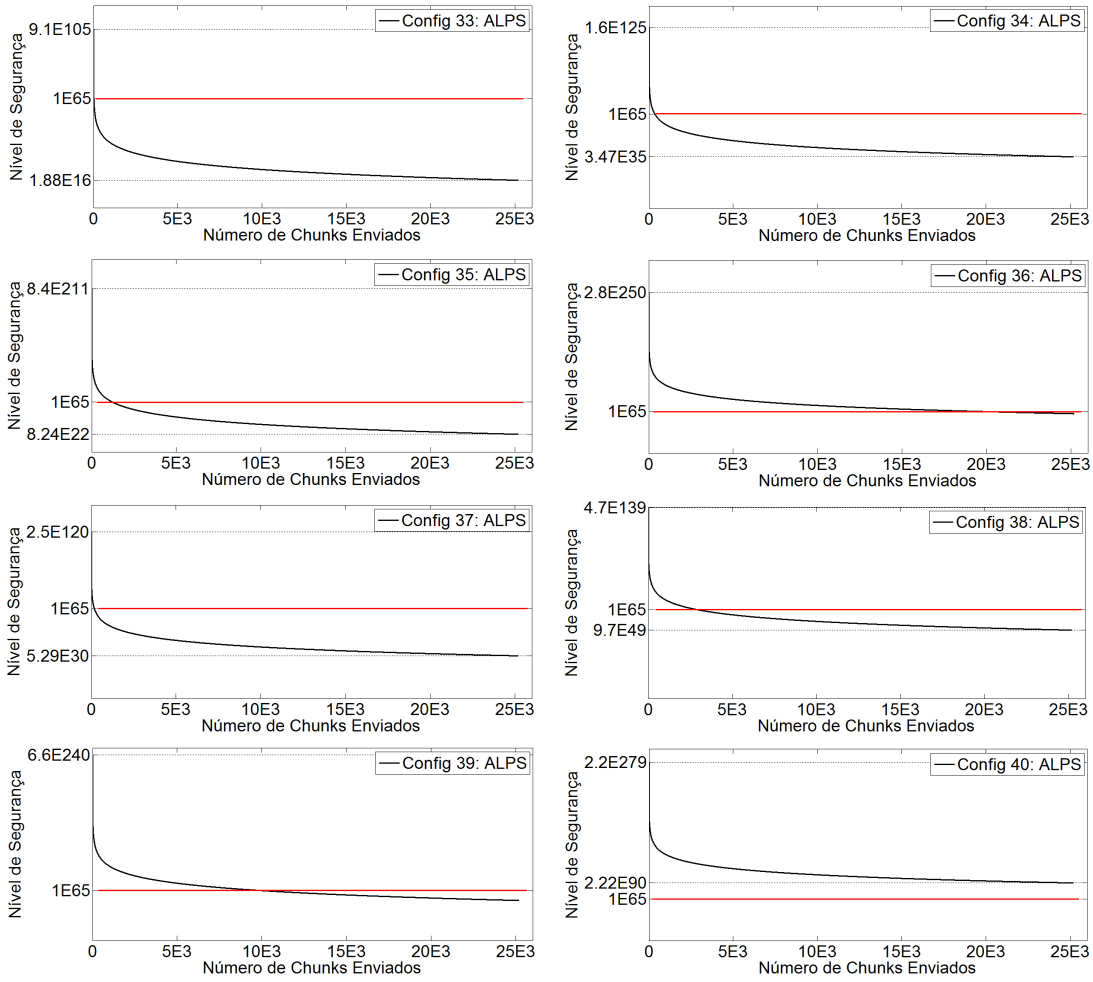


Figura A.7: Análise de segurança de nível alto das configurações possíveis do esquema ALPS em um cenário convencional