Este documento técnico foi estruturado como um relatório detalhado e formal, adequado para ser transcrito para um documento Word. Ele contém um **Resumo Executivo Excepcional** (conforme solicitado), seguido por todas as seções e detalhes técnicos exigidos pelo material de origem.

**RELATÓRIO TÉCNICO: CRIPTOGRAFIA RSA**

**RESUMO**

O algoritmo RSA (Rivest-Shamir-Adleman), desenvolvido em 1977, permanece um pilar da segurança digital por ser um dos primeiros sistemas de **criptografia assimétrica** e de chave pública amplamente utilizados.

A segurança do RSA é intrinsecamente ligada à **dificuldade computacional de fatorar números grandes** (Problema da Fatoração). O algoritmo utiliza um par de chaves — pública e privada — sendo a pública usada para criptografar ou verificar assinaturas, e a privada, mantida em segredo, para descriptografar ou assinar.

**Fundamentos Essenciais:** O processo de geração de chaves envolve a escolha de dois primos grandes $p$ e $q$ para formar o módulo $n = p \times q$. A complexidade e a segurança dependem diretamente da função **Totiente de Euler** $\varphi(n) = (p-1) \times (q-1)$, que deve ser mantida secreta.

**Desempenho e Uso:** O RSA é significativamente mais lento (aproximadamente **1000x**) que algoritmos simétricos como o AES. Por essa razão, seu uso típico é **híbrido**: ele é empregado apenas para criptografar uma chave simétrica temporária, que então criptografa os dados grandes.

**Segurança Atual e Futura:**

* **Tamanho Mínimo de Chave:** Para garantir a segurança contra o melhor ataque clássico conhecido (GNFS), o padrão atual exige **2048 bits**, sendo **3072 bits** a recomendação futura.
* **Vulnerabilidade Quântica:** O RSA é vulnerável ao **Algoritmo de Shor** (1994), que pode quebrá-lo em tempo polinomial. Embora essa ameaça não seja imediata, a migração para a criptografia pós-quântica (PQC), como algoritmos baseados em *Lattice* (Kyber, Dilithium), já está sendo padronizada como contramedida futura.
* **Implementação Segura:** A robustez do RSA exige o uso de **Padding OAEP** para criptografia e **PSS** para assinatura, além de proteções contra ataques de canal lateral (*side-channel attacks*).

**1. INTRODUÇÃO E FUNDAMENTOS DO RSA**

**1.1 O que é RSA?**

RSA (Rivest-Shamir-Adleman) é um algoritmo de criptografia assimétrica desenvolvido em 1977. É reconhecido como um dos primeiros e mais amplamente utilizados sistemas de chave pública.

**1.2 Características e Princípio Básico**

O RSA possui as seguintes características principais:

1. **Assimétrico:** Utiliza duas chaves diferentes (pública e privada).
2. **Baseado em Problema Matemático:** Sua segurança depende da dificuldade de fatorar números grandes.
3. **Bidirecional:** Pode ser utilizado tanto para criptografar quanto para realizar **assinatura digital**.
4. **Ampla Adoção:** Serve como base para protocolos importantes como HTTPS e SSH.

O fluxo básico envolve Alice (o destinatário) e Bob (o remetente): Bob usa a chave pública de Alice para criptografar a mensagem, mas apenas Alice pode descriptografar utilizando sua chave privada.

**2. FUNDAMENTOS MATEMÁTICOS**

A fundação do RSA está na Teoria dos Números.

**2.1 Aritmética Modular**

A aritmética modular é central para o funcionamento do algoritmo. A notação $a \equiv b \pmod n$ indica que $a \mod n = b \mod n$. Uma propriedade crucial é que $(a^k) \mod n$ pode ser calculado eficientemente.

**2.2 Números Primos e Fatoração**

O **Teorema Fundamental da Aritmética** garante que todo número possui uma fatoração prima única. No RSA, o módulo $n$ é o produto de dois primos grandes $p$ e $q$ ($n = p \times q$). O **Problema da Fatoração** — encontrar $p$ e $q$ dado $n$ — é computacionalmente difícil para números grandes, garantindo a segurança do sistema.

**2.3 Função Totiente de Euler ($\varphi(n)$)**

A função $\varphi(n)$ representa a quantidade de números menores que $n$ que são **coprimos** com $n$. Para o RSA, onde $n = p \times q$, o cálculo é simples e deve ser mantido em segredo: $$\varphi(n) = (p-1) \times (q-1)$$

**2.4 Teorema de Euler e Inverso Modular**

O **Teorema de Euler** estabelece que se $\text{gcd}(a, n) = 1$, então $a^{\varphi(n)} \equiv 1 \pmod n$. O corolário $a^{(k\times\varphi(n) + 1)} \equiv a \pmod n$ é o que prova que o processo de descriptografia inverte o processo de criptografia.

O **Expoente Privado** ($d$) é determinado como o inverso modular do Expoente Público ($e$) módulo $\varphi(n)$, tal que $e \times d \equiv 1 \pmod{\varphi(n)}$. Este valor é encontrado usando o **Algoritmo Euclidiano Estendido**.

**3. O ALGORITMO RSA PASSO-A-PASSO**

O algoritmo é dividido em três fases:

**3.1 Fase 1: Geração de Chaves**

1. **Gerar Primos ($p$ e $q$):** Gere dois primos distintos ($p \ne q$) de tamanhos similares. Para uma chave de $n$ bits, $p$ e $q$ devem ter aproximadamente $n/2$ bits cada.
2. **Calcular o Módulo ($n$):** $n = p \times q$. Este valor é público.
3. **Calcular $\varphi(n)$:** $\varphi(n) = (p-1) \times (q-1)$. Este valor deve ser secreto.
4. **Escolher Expoente Público ($e$):** Escolha $e$ tal que $1 < e < \varphi(n)$ e $\text{gcd}(e, \varphi(n)) = 1$. O valor **$e = 65537$** é comumente usado por ser primo, eficiente (tem poucos bits '1' em binário), e amplamente testado.
5. **Calcular Expoente Privado ($d$):** Calcule $d$, o inverso modular de $e$ módulo $\varphi(n)$, usando o Algoritmo Euclidiano Estendido.

**Resultado:** A **Chave Pública** é $(n, e)$ e a **Chave Privada** é $(n, d)$.

**3.2 Fase 2: Criptografia**

Para criptografar uma mensagem $m$ (onde $m < n$) usando a chave pública $(n, e)$: $$c = m^e \mod n$$ $c$ é o texto cifrado.

**3.3 Fase 3: Descriptografia**

Para descriptografar o texto cifrado $c$ usando a chave privada $(n, d)$: $$m = c^d \mod n$$ $m$ é a mensagem original.

**4. DETALHES DE IMPLEMENTAÇÃO**

A implementação prática do RSA exige algoritmos eficientes para lidar com números de grande escala.

**4.1 Teste de Primalidade Miller-Rabin**

Este é um algoritmo probabilístico usado para verificar se $p$ e $q$ são primos. A probabilidade de erro (aceitar um número composto como primo) é menor que $(1/4)^k$, onde $k$ é o número de *rounds* de teste.

**4.2 Exponenciação Modular Rápida**

Também conhecido como algoritmo **"Square-and-Multiply"**.

* **Função:** Calcula $a^b \mod n$ eficientemente, evitando que números intermediários se tornem gigantescos.
* **Complexidade:** $O(\log \exp)$.
* É fundamental, pois esta é a operação mais custosa e frequente do RSA.

**4.3 Algoritmo Euclidiano Estendido (AEE)**

Este algoritmo encontra os coeficientes $x$ e $y$ para a equação $ax + by = \text{gcd}(a, b)$. No RSA, ele é usado para encontrar o inverso modular $d$ tal que $e \times d \equiv 1 \pmod{\varphi(n)}$.

**5. ANÁLISE DE SEGURANÇA E TAMANHO DAS CHAVES**

**5.1 Segurança Baseada no Problema da Fatoração**

A segurança depende da dificuldade de, dado $n = p \times q$, encontrar $p$ e $q$.

O melhor algoritmo clássico conhecido é o **General Number Field Sieve (GNFS)**. Para quebrar uma chave de 2048 bits, o GNFS exigiria aproximadamente $2^{112}$ operações, o que é inviável na computação atual.

**5.2 Tamanhos de Chave Recomendados**

O poder computacional necessário para quebrar o RSA aumenta exponencialmente com o tamanho da chave.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ano** | **Tamanho Mínimo (RSA)** | **Equivalência Simétrica** | **Status** |
| 2010 | 1024 bits | 80 bits | ❌ Quebrado |
| 2015 | 2048 bits | 112 bits | ✅ Seguro atual |
| **2025** | **3072 bits** | **128 bits** | ✅ **Recomendado** |

**5.3 Ataques e Contramedidas**

Ataques geralmente exploram falhas na implementação e não no fundamento matemático.

|  |  |  |
| --- | --- | --- |
| **Tipo de Ataque** | **Descrição** | **Contramedida** |
| **Ataque de Padding** (ex: Bleichenbacher) | Explora falhas no esquema de preenchimento de mensagens. | Usar **Padding OAEP** (Optimal Asymmetric Encryption Padding). |
| **Ataque de Canal Lateral** (*Side-Channel*) | Analisa tempo de execução ou consumo de energia para deduzir o expoente privado $d$. | Implementar operações em **tempo constante** (constant\_time\_exp). |
| **Expoente Privado Pequeno** | Se $d$ for muito pequeno, é vulnerável ao ataque de Wiener. | Garantir que $d$ tenha um número de bits adequado (ex: maior que $n/4$). |

**6. OTIMIZAÇÕES E LIMITAÇÕES**

**6.1 Otimização CRT (Teorema Chinês do Resto)**

A descriptografia é a operação mais lenta do RSA. O CRT pode ser usado para acelerar a descriptografia em cerca de **4x**. A técnica divide o cálculo de $m = c^d \mod n$ em dois módulos menores (módulo $p$ e módulo $q$) e depois combina os resultados. Para isso, a chave privada otimizada precisa armazenar $p$, $q$ e valores pré-computados como $d\_p$ e $d\_q$.

**6.2 Limitações Fundamentais**

1. **Tamanho da Mensagem:** A mensagem ($m$) deve ser menor que o módulo ($n$). Para 2048 bits, o máximo é de cerca de 255 bytes por bloco. A solução é usar o RSA em conjunto com criptografia simétrica (uso híbrido).
2. **Performance:** O RSA é aproximadamente **1000x mais lento que o AES**. É usado para criptografar chaves simétricas, e não os dados em si.

**6.3 Ataques Quânticos**

O **Algoritmo de Shor** (1994) é capaz de quebrar o RSA em tempo polinomial. Estima-se que, em 10 a 20 anos, essa ameaça possa se tornar prática. A contramedida é a migração para a **criptografia pós-quântica (PQC)**.

**7. COMPARAÇÃO E USO RECOMENDADO**

**7.1 RSA vs. Outros Algoritmos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspecto** | **RSA (2048 bits)** | **ECC (256 bits)** | **Kyber (Pós-Quântico)** |
| Segurança | ~112 bits | ~128 bits | ~128 bits |
| Tamanho Chave | 2048 bits | 256 bits | ~1KB |
| Velocidade | Lento | Mais rápido | Variável |
| Risco Quântico | ❌ Vulnerável | ❌ Vulnerável | ✅ Resistente |

**7.2 Uso em Produção vs. Uso Educacional**

O RSA é um algoritmo maduro (45+ anos de análise), versátil (criptografia e assinatura), e com fundamento matemático sólido.

|  |  |
| --- | --- |
| **✅ USAR RSA PARA:** | **❌ NÃO USAR RSA PARA:** |
| Troca de chaves simétricas (híbrido) | Criptografia de dados grandes |
| Assinatura digital (com PSS) | Novos sistemas críticos (preferir ECC) |

A implementação para fins educacionais, como a que usa chaves pequenas (ex: 512 bits) ou omite o *padding* seguro, **não é considerada segura**. Para produção, é obrigatório usar:

* Bibliotecas criptográficas testadas (OpenSSL, BoringSSL).
* Chaves de **2048 bits ou maiores**.
* **Padding OAEP** para criptografia e **PSS** para assinatura digital.
* Proteções contra *timing attacks*.