Tópicos em Redes de Computadores

Autenticação com Chave Secreta, Diffie-Hellman & Sistemas Cliente/Servidor Seguros

Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/topredes

Sumário

- Vamos começar estudando um protocolos de autenticação mútua com chave secreta
- Em seguida: criação de chave secreta sobre canal de comunicação público: Diffie-Hellman
- Por fim: SSL e TLS e o Trabalho Prático

Protocolos de Autenticação

- Um protocolo de autenticação permite que duas partes se verifiquem mutuamente
- Ou seja: são quem dizem ser e não impostores
- Autenticou? Podem comunicar tranquilamente!

Autenticação com Chave Secreta

- No protocolo vamos considerar que ambas as partes já compartilham uma chave secreta K_{AB}
- A chave secreta foi compartilhada anteriormente, usando um meio seguro de comunicação – ou Diffie-Hellman, que vamos estudar a seguir
- No protocolo, as partes enviam desafios umas para as outras e devem receber as respostas adequadas

O Protocolo: Chave Secreta

- No primeiro protocolo, inicialmente Alice manda uma mensagem para Bob com sua identificação:

  Alice → A → Bob

- Bob não sabe se é Alice mesmo, ou Eva, a intrusora, gera um número aleatório DesafioB e envia para Alice:

  Bob → DesafioB → Alice

- Alice criptografa o DesafioB com a chave secreta e envia para Bob

- Alice → K_{AB}(DesafioB) → Bob

O Protocolo: Chave Secreta

- Alice também gera um número aleatório DesafioA e envia para Bob:

  Alice → DesafioA → Bob

- E Bob criptografa o DesafioA com a chave secreta compartilhada e envia de volta para Alice:

  Bob → K_{AB}(DesafioA) → Alice

- Pronto! Ambos já têm certeza que estão comunicando com as partes legítimas

O Protocolo: 3 Mensagens

- Que tal implementar este protocolo com apenas 3 mensagens?

  Alice → A, DesafioA → Bob

  Bob → DesafioB, K_{AB}(DesafioA) → Alice

  Alice → K_{AB}(DesafioB) → Bob

O Protocolo: 3 Mensagens

- Que tal implementar este protocolo com apenas 3 mensagens?

  Alice → A, DesafioA → Bob

  Bob → DesafioB, K_{AB}(DesafioA) → Alice

  Alice → K_{AB}(DesafioB) → Bob

- Infelizmente está sujeito a um ataque de reflexão

Ataque de Reflexão

- Eva, a invasora, começa fingindo que é Alice:

  Eva → A, DesafioE → Bob
- Bob inocentemente responde:

  Bob → DesafioB, K_{AB}(DesafioE) → Eva
- Pronto... Eva agora não tem como criptografar o DesafioB... como faz?...

Ataque de Reflexão

- Eva, a invasora, começa fingindo que é Alice:

  Eva → A, DesafioE → Bob
- Bob inocentemente responde:

  Bob → DesafioB, K_{AB}(DesafioE) → Eva
- Pronto... Eva agora não tem como criptografar o DesafioB... como faz?...
- Inicia uma nova seção com Bob usando DesafioB:

  Eva → A, DesafioB → Bob

Ataque de Reflexão

- Ou seja: Eva consegue que o próprio Bob criptografe DesafioB: 😱

Bob → UmNovoDesafioB, K_{AB}(DesafioB) → Eva

- Pronto: Eva pode então completar a sessão anterior e se autenticar com Bob:
- Eva → K_{AB}(DesafioB) → Bob
- A versão com mais mensagens pode sofrer um ataque muito semelhante...

Lições do Ataque de Reflexão

- Ataques podem ser imaginados nas situações mais inesperadas
- Neste caso é possível, por exemplo, usar duas chaves secretas, e cada parte sempre criptografa com a sua
- Os números aleatórios podem ser de conjuntos disjuntos: em um caminho números pares, em outro números ímpares
- Não permitir uma segunda autenticação, enquanto a primeira está em curso

Diffie-Hellman: Compartilhando a Chave Secreta

- Em diversas situações é necessário compartilhar uma chave secreta entre duas partes
- Como no protocolo que acabamos de ver – com as medidas extras de proteção definidas
- Se houver a possibilidade de usar criptografia de chave pública, ótimo!
  - desde que as chaves públicas sejam certificadas!
- Caso contrário: Diffie-Hellman!

O Protocolo de Diffie-Hellman

- O primeiro passo: Alice e Bob têm que escolher dois números *grandes* (p.ex. 512 bits) N e G, sendo N um número primo
- Estes números (N e G) não precisam ser secretos! Compartilhados publicamente
- Mas cada um deles (Alice e Bob) agora precisa de um número grande secreto, que só eles sabem
- Alice escolhe X e Bob escolhe Y → secretos!

Diffie-Hellman em Ação

- No primeiro passo Alice manda para Bob:

  Alice → (N, G, G^x mod N) → Bob
- Bob então responde para Alice:

  Bob → G^Y mod N → Alice
- Localmente Alice calcula:
- $(G^Y \bmod N)^x = G^{XY} \bmod N$
- Localmente Bob calcula:
- $(G^X \bmod N)^Y = G^{XY} \bmod N$

Eva em Ação

- Eva observa toda a comunicação
- Ela conhece G e N, mas não conhece nem X nem Y
- O ponto é que sabendo G^x mod N é computacionalmente difícil determinar X
- Mesma coisa para G^Y mod N e Y

Um Exemplo

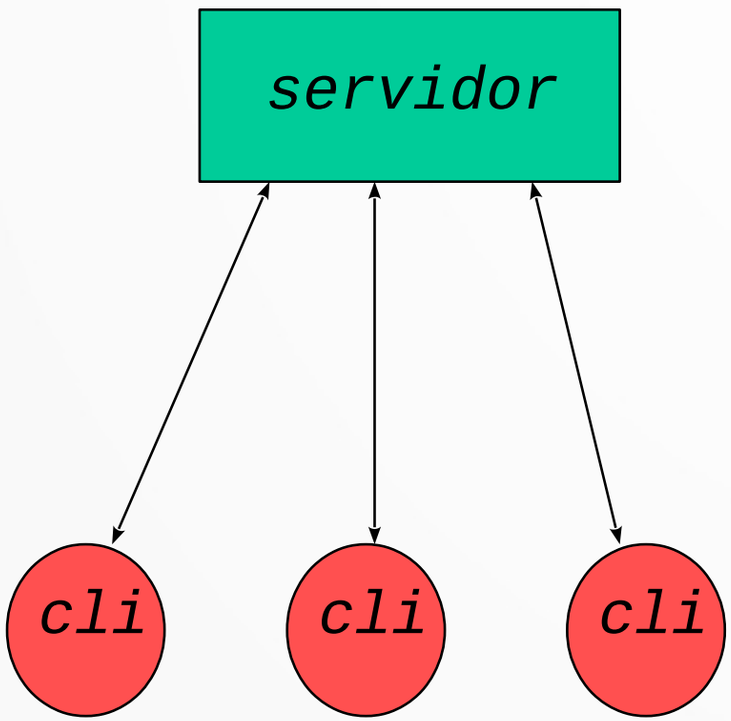
- Vamos ver um exemplos com números pequenos para entendermos melhor: N=47 e G=3
- Alice escolhe como número secreto X=8
- Bob escolhe como número secreto Y=10
- Alice envia para o Bob: $3^8 \bmod 47 = 28$
- Bob envia para Alice: $3^{10} \bmod 47 = 17$
- Alice calcula: $17^8 \bmod 47 = 4$
- Bob calcula: $28^{10} \bmod 47 = 4$

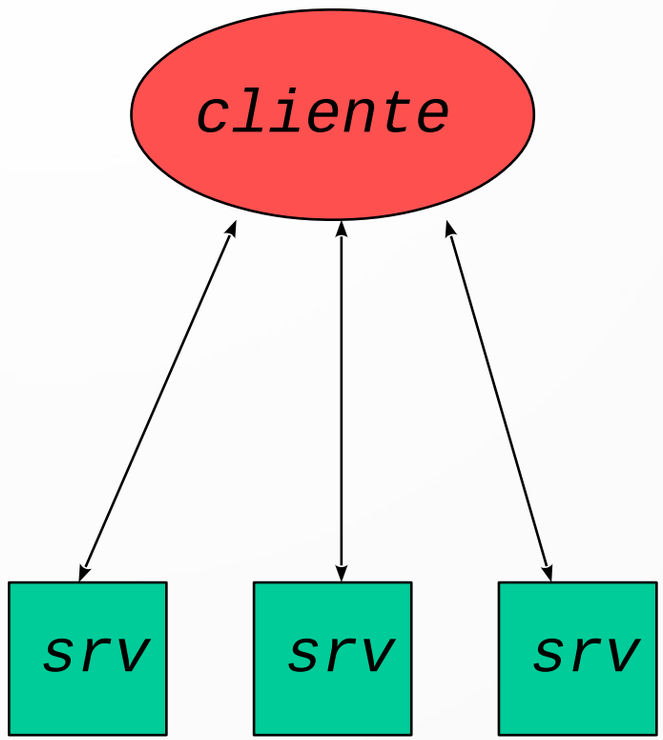
Sistemas Cliente/Servidor Seguros

Sistemas Cliente-Servidor (C/S)

- Dois processos de aplicação, executando em duas máquinas conectadas em rede, desejam se comunicar - como iniciar a comunicação?
- Solução: um dos processos permanece em execução, na escuta, aguardando uma [possível] comunicação vinda do outro processo
- O outro processo inicia a comunicação; oferece uma interface para um usuário que faz uso do serviço (humano ou processo)
- Cliente & Servidor

Clientes & Servidores





Ameaças aos Sistemas C/S

- Os servidores são os portais de entrada que o mundo usa para a organização
- É necessário autenticar & autorizar acessos
  - autorização: uma vez autenticado, o que pode fazer?
- Lapsos de segurança ou mesmo bugs podem levar a transtornos graves
- Como construir sistemas cliente-servidor seguros?

TLS: Transport Layer Security

- O TLS é a ferramenta mais usada hoje para programar sistemas Cliente/Servidor seguros
- Vastamente usado na Internet: HTTPS (*HyperText Transfer Protocol – Secure*), VoIP, transferência de arquivo segura, e-mail seguro, etc. etc. etc.
- O TLS foi desenvolvido a partir do SSL – *Secure Socket Layer*

SSL: Secure Socket Layer

- Socket: abstração para comunicação de processos
- Concretamente: uma API (interface de programação – *Application Programming Interface*) para programação de comunicação
- Não é exclusiva TCP/IP, mas certamente é seu uso mais importante/popular
- SSL: traz segurança para os sockets
- Segurança: sigilo + integridade + autenticidade

TLS: Transport Layer Security

- SSL: originalmente proposto pela Netscape (Web)
- Depois: padronizado pelo IETF e rebatizado TLS
  - Hoje na versão TLS 1.3 especificado no RFC 8446
- O cliente estabelece uma sessão segura com o servidor
- A sessão segura em geral é estabelecida em uma porta específica, HTTPS: 443 (enquanto HTTP: 80)

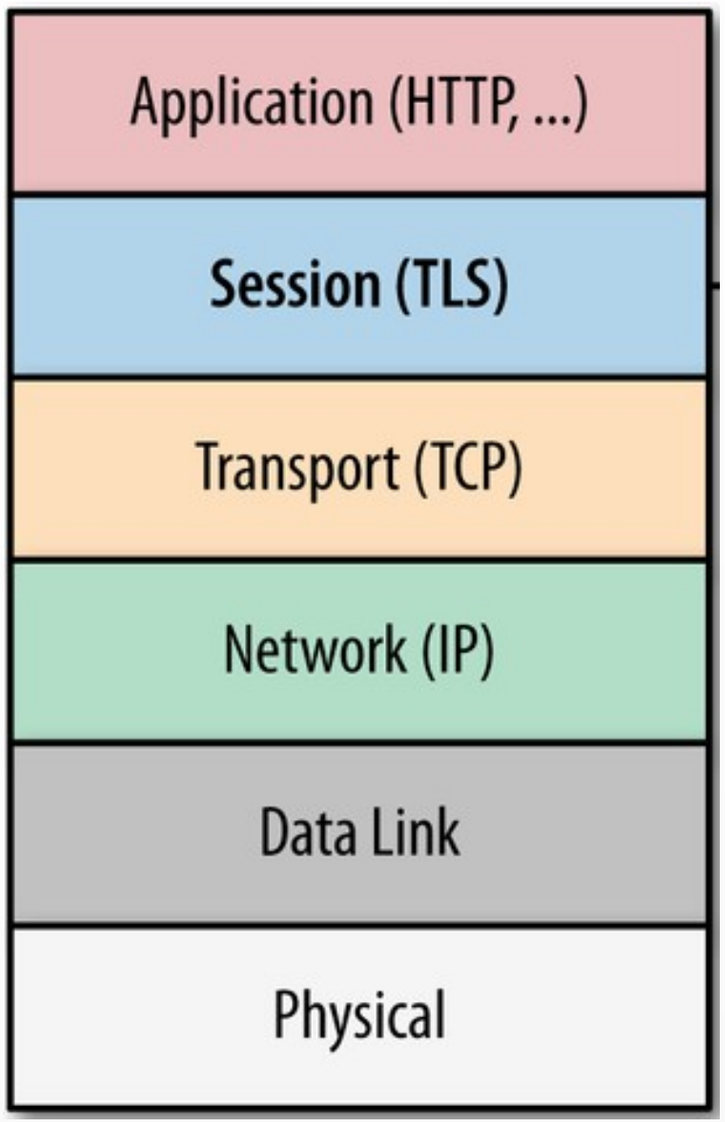
TLS: A Comunicação Segura

- Apenas no estabelecimento da sessão (*handshake*) cliente e servidor usam criptografia de chave pública
- Se autenticam, definem os parâmetros da comunicação e uma “chave mestra secreta” para a comunicação propriamente dita
- Diversos algoritmos criptográficos disponíveis/possíveis
- No *handshake* trocam informações sobre algoritmos possíveis (DES, AES, SHA, MD5, RSA,...)

TLS: Uma Camada Extra?

- TLS executa na aplicação...
- Mas efetivamente suporta aplicações seguras
- Sobre protocolo de trasnporte confiável: TCP
- Ou seja: efetivamente uma camada a mais ;-)

TLS: Uma Camada a Mais?



2ª Avaliação TopRedes

- A segunda avaliação da disciplina vai ser a construção de um sistema cliente/servidor seguro com TLS
- O trabalho vai ser feito BCC/IBM em dupla; Pós individual
  - defina sua dupla hoje mesmo! se tiver dificuldade entre em contato com o professor na próxima aula

O Sistema C/S Seguro

- O sistema propriamente dito vai ser muito simples:
  - Um KVS (*Key-Value Store*) seguro: servidor mantém uma base de dados, cliente pode consultar, alterar...
- O importante é que o sistema garanta: sigilo, integridade e autenticidade
- Cada uma destas propriedades deve ser demonstrada

O Sistema C/S Seguro

- Cada uma das propriedades deve ser demonstrada:
  - Sigilo: mostre os dados criptografados transmitidos (imprimindo junto a versão original)
  - Autenticidade: um invasor tenta comunicar com o servidor ou com o cliente, mostre que não consegue
  - Integridade: troque o valor de alguns bits da mensagem e mostre que não descriptografa corretamente

O que deve ser apresentado:

- É possível escolher a linguagem que quiser para implementar o trabalho
  - Sugestão: Python, podem usar C, C++, Java, outras linguagens
- Desta vez o relatório vai ser especial: explique como programar um sistema cliente-servidor seguro usando TLS na linguagem da sua escolha
- Formato sugerido: vídeo!
  - se não quiser: apresentação
  - se não quiser: página Web com texto
- Divulgue na Internet, nas redes sociais, no YouTube, faça deste uma contribuição do seu currículo!

O que deve ser apresentado:

- Cada dupla deve fazer uma página Web contendo:
  - O relatório: por exemplo link para o vídeo no YouTube
  - Ou a apresentação, ou o relatório em texto mesmo
- Logs de execução: se no vídeo você estiver confiante que demonstrou seu sistema bem – não precisa de logs na página!
- Código comentado, pense no usuário que viu seu vídeo e vai acessar o código para aprender

Conclusão

- Nesta aula começamos revendo o conceito de *hash*
- Em seguida examinamos um protocolo para autenticação usando chave secreta
- E estudamos o protocolo de Diffie-Hellman para estabelecimento de chave secreta
- Por fim: programação de sistemas cliente-servidor seguros com TLS e a 2ª avaliação da disciplina TopRedes

Obrigado!

Lembrando: a página da disciplina é:
https://www.inf.ufpr.br/elias/topredes