

# Introdução à Verificação Automática de Protocolos de Segurança com Scyther

**Diego Kreutz** (UNIPAMPA), Rodrigo Mansilha (UNIPAMPA)  
**Silvio E. Quincozes** (UFF), Tadeu Jenuário (UNIPAMPA)  
**João Otávio Chervinski** (Monash University)

# Quem somos?



**debian:~\$** whoami  
Diego Kreutz

- Professor@**UNIPAMPA**
- Coordenador@**UniHacker.Club**
- Interesses:
  - Segurança
  - Redes
  - Sistemas Distribuídos



**UNIHACKER.CLUB**  
PROGRAMA UNIVERSIDADE HACKER



# Quem somos?



**debian:**~\$ whoami  
Rodrigo Mansilha

- Professor@UNIPAMPA
- Interesses:
  - Segurança
  - Redes
  - Sistemas Distribuídos

<https://sites.unipampa.edu.br/rodrigomansilha/>



**PPGES**  
Programa de Pós-Graduação  
em Engenharia de Software



# Quem somos?



- Doutorando@UFF
- Interesses:
  - Segurança
  - Redes
  - Internet das Coisas (IoT)

**debian:**~\$ whoami  
Silvio Quincozes



# Quem somos?



**debian:~\$** whoami  
Tadeu Jenuário

- ❑ Graduando@**UNIPAMPA**
  - ❑ Engenharia de Software
- ❑ Interesses:
  - ❑ Segurança
  - ❑ Desenvolvimento de Software



Engenharia de Software



**UNIHACKER.CLUB**  
PROGRAMA UNIVERSIDADE HACKER



**L E A**

Laboratório de Estudos  
Avançados em Computação

# Quem somos?



**debian:**~\$ whoami  
João Chervinski

- Doutorando@**Monash**
- Membro@**UniHacker.Club**
- Interesses:
  - Segurança
  - Blockchains
  - Criptomoedas



**MONASH**  
University



**UNIHACKER.CLUB**  
PROGRAMA UNIVERSIDADE HACKER

# Nível do minicurso



# Intermediário a Avançado

# Conteúdo do minicurso

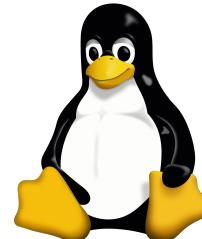


<https://s4a.in/github>

# Conhecimentos desejados

```
~ — errc2020@d: ~/scyther-linux-v1.1.3 — ssh scyther
errc2020@d:~/scyther-linux-v1.1.3$ ls
batcher.sh
Changelog.txt
combos-book.sh
combos-ike.sh
combos-ikev0.sh
combos-ikev1.sh
combos-ikev2.sh
combos-iso.sh
generate-attack-graphs.py
GNU-General-Public-License.txt
Gui
Images
INSTALL.md
INSTALL.txt
json-scyther.py
make-bsub.py
mpa.spdl
notes-brutus-mpa.txt
errc2020@d:~/scyther-linux-v1.1.3$
```

Linux e  
terminal /  
linha de  
comando



# Conhecimentos desejados

```
#!/usr/bin/env python
import pika
import sys

connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='logs.unihacker.club'))
channel = connection.channel()

channel.exchange_declare(exchange='logs', exchange_type='fanout')

message = ' '.join(sys.argv[1:]) or "Info: Coletando Dados!"

channel.basic_publish(exchange='logs', routing_key='', body=message)

print(" Dados enviados %r" % message)

connection.close()
```

Lógica de  
programação  
e codificação

# Conhecimentos desejados



Noções de  
Redes de  
Computadores

# Conhecimentos desejados

INFORMATION  
SECURITY



Princípios e  
Primitivas de  
Segurança da  
Informação

- ❑ Introdução
- ❑ A ferramenta Scyther
- ❑ O protocolo ACS
- ❑ O protocolo WMF
- ❑ O protocolo NS
- ❑ O protocolo GNSL
- ❑ Considerações finais

# Introdução



# LGPD

(Lei Geral de Proteção de Dados)

LEI N° 13.709, DE 14 DE AGOSTO DE 2018.

# LGPD: Proteção de Dados



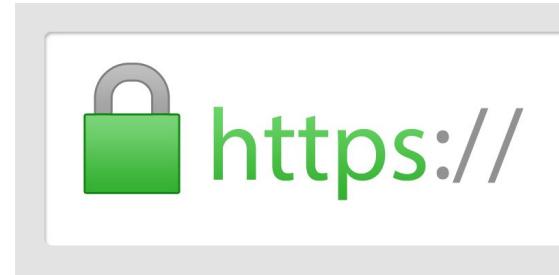
- ❑ em armazenamento
- ❑ em processamento
- ❑ em trânsito

# Protocolos de Segurança



- ❑ Transfer Layer Security (TLS)
- ❑ Secure Shell (SSH)
- ❑ Internet Security Protocol (IPSec)
- ❑ Diffie-Hellman (DH)
- ❑ Internet Key Exchange (IKE)

# HTTPS = Segurança?



HTTP + TLS = HTTPS

<https://s4a.in/httpsBR>

# HTTPS = Segurança?

<https://s4a.in/httpsBR>

Versão	5k+ sites
SSLv2	2%
SSLv3	5%
TLS 1.0	76%
TLS 1.1	80%
TLS 1.2	97%
TLS 1.3	35%

0% dos  
sites  
suportam  
**APENAS**  
**TLS 1.3**

# Projeto de Protocolos de Segurança



- ❑ Especificação
- ❑ Verificação
- ❑ Implementação
- ❑ Verificação de Código

# Projeto de Protocolos de Segurança



- Especificação
- Verificação
- Implementação
- Verificação de Código

Semântica  
agnóstica de  
verificação

# Projeto de Protocolos de Segurança



- Especificação
- Verificação
- Implementação
- Verificação de Código

Semântica  
específica da  
ferramenta

# Projeto de Protocolos de Segurança



- Especificação
- Verificação
- Implementação
- Verificação de Código

Tradução  
para  
linguagem de  
programação

# Projeto de Protocolos de Segurança



- Especificação
- Verificação
- Implementação
- Verificação de Código

Adição de  
semântica de  
linguagem de  
verificação

# Projeto de Protocolos de Segurança



- ✓ Especificação
- ✓ Verificação
- ❑ Implementação
- ❑ Verificação de Código

Foco deste  
minicurso

# Especificação: semântica

---

1. Alice → Bob [Alice,  $E_K(\text{dadosA})$ ]

---

# Especificação: semântica

---

1. Alice → Bob [Alice,  $E_K(\text{dadosA})$ ]

---

2. Bob → Alice [Bob,  $E_K(\text{dadosB})$ ]

---

# Especificação: semântica

- 
1. Alice → Bob [Alice,  $E_K(\text{dadosA})$ ]

---

  2. Bob → Alice [Bob,  $E_K(\text{dadosB})$ ]

---

  3. Bob, Alice  $K \leftarrow H(K \parallel \text{dadosA} \parallel \text{dadosB})$
-

# Especificação: semântica

1. Alice → Bob [Alice,  $E_{Ke}(\text{dadosA})$ ],  $\text{HMAC}_{Kh}$

Curiosidade: PFS, PCS, **PQS**

arXiv: <https://s4a.in/chavesKeKh>

ACM ToPS: <https://s4a.in/dVRmID>

Chaves  
distintas  
para cifra e  
HMAC

# Especificação: semântica

- 
1. Alice → Bob [Alice,  $E_{Ke}(\text{dadosA})$ ],  $\text{HMAC}_{Kh}$
  2. Bob → Alice [Bob,  $E_{Ke}(\text{dadosB})$ ],  $\text{HMAC}_{Kh}$
-

# Especificação: semântica

- 
1. Alice → Bob [Alice,  $E_{Ke}(dadosA)$ ],  $HMAC_{Kh}$
  2. Bob → Alice [Bob,  $E_{Ke}(dadosB)$ ],  $HMAC_{Kh}$
  3. Bob, Alice  $Ke \leftarrow H(Ke \parallel dadosA \parallel dadosB)$
-

# Especificação: semântica

- 
1. Alice → Bob [Alice,  $E_{Ke}(dadosA)$ ],  $HMAC_{Kh}$

---

  2. Bob → Alice [Bob,  $E_{Ke}(dadosB)$ ],  $HMAC_{Kh}$

---

  3. Bob, Alice  $Ke \leftarrow H(Ke \parallel dadosA \parallel dadosB)$

---

  4. Bob, Alice  $Kh \leftarrow H(Kh \parallel dadosA \parallel dadosB)$
-

- ✓ Introdução
- ❑ A **ferramenta Scyther**
- ❑ O protocolo ACS
- ❑ O protocolo WMF
- ❑ O protocolo NS
- ❑ O protocolo GNSL
- ❑ Considerações finais

# A ferramenta Scyther



- ❑ Semânticas Operacionais
- ❑ Práticas com Scyther

# Semânticas Operacionais

- ❑ Termos atômicos
- ❑ Tipos predefinidos
- ❑ Função hash
- ❑ Chaves simétricas
- ❑ Chaves assimétricas
- ❑ Tipos básicos de eventos
- ❑ Eventos de afirmação (claim)

# Termos atômicos

- ❑ **fresh** : valores pseudo-aleatórios
- ❑ **var** : armazena dados recebidos
- ❑ **const** : constantes locais

# Tipos predefinidos

- ❑ **Agent** : agenda das comunicações
- ❑ **Function** : define um termo como função
- ❑ **Nonce** : termos que armazenam valores

# Função hash

- ❑ finalidade: resumo criptográfico
- ❑ declaração: **hashfunction H**
- ❑ notação:  $H(\text{dado})$

# Chave simétricas

- ❑ finalidade: encriptação simétrica
- ❑ notação: { dados } termo
- ❑ exemplo: { dados } k (Alice, Bob)

# Chave assimétricas

- ❑ finalidade: encriptação assimétrica
- ❑ notação:  $\text{sk}(\text{Alice}), \text{pk}(\text{Alice})$
- ❑ exemplo:  $\{\text{dados}\} \text{pk}(\text{Alice})$

# Tipos básicos de eventos

- ❑ **send** : enviar dados
- ❑ **recv** : receber dados
- ❑ exemplo:

**send\_1(Alice, Bob, { dado } pk (Bob) )**

**recv\_1(Alice, Bob, { dado } **pk (Bob)** )**

# Eventos de afirmação (claim)

- ❑ **claim** : modelar propriedades de segurança
- ❑ exemplos:

**claim**(Bob , Secret , dado)

**claim**(Bob , Nisynch)

# Práticas com Scyther

- ❑ Sistema Operacional: Debian Linux
- ❑ Versão da Scyther: 1.1.3
- ❑ Download: <https://s4a.in/scyther>
- ❑ Descompactar:

```
tar xzfv scyther-linux-v1.1.3.tgz
```

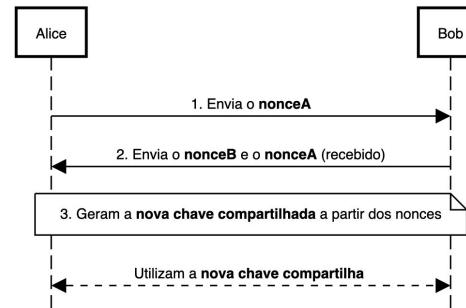
# Parâmetros da Scyther

-a, --auto-claims

-r, --max-runs=<int>

-A, --all-attacks

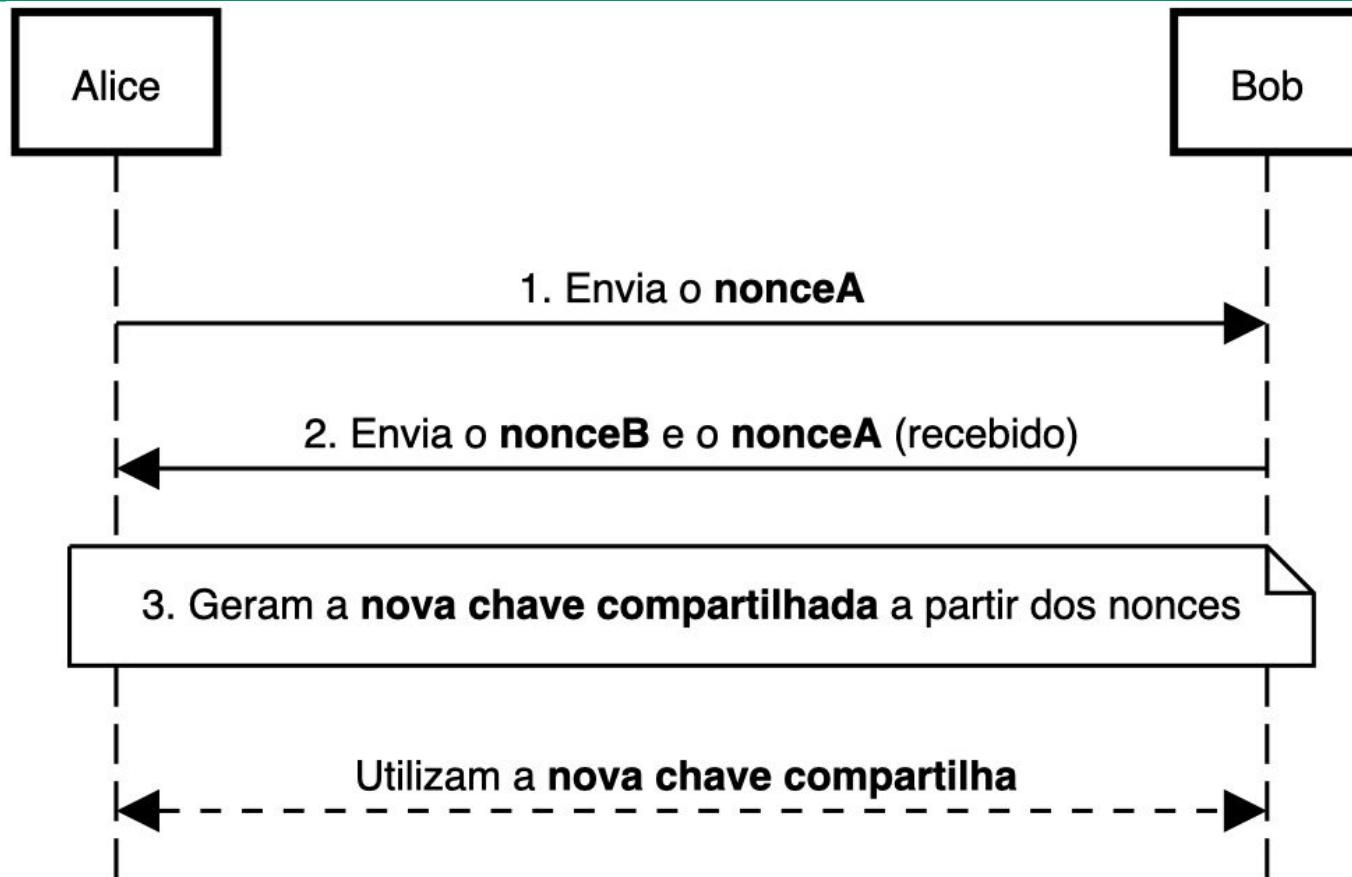
# Diagramas: ferramenta



Ferramenta: <https://sequencediagram.org>

Código dos Diagramas: <https://s4a.in/github>

# Protocolo exemplo (diagrama)



# Protocolo exemplo (especificação)

---

1. Alice → Bob [Alice,  $E_{pk_{Bob}}(\text{nonceA})$ ]

---

# Protocolo exemplo (especificação)

- 
1. Alice → Bob [Alice,  $E_{pk_{Bob}}(\text{nonceA})$ ]

---

  2. Bob → Alice [Bob,  $E_{pk_{Alice}}(\text{nonceB}, \text{nonceA})$ ]

---

# Protocolo exemplo (especificação)

- 
1. Alice → Bob [Alice,  $E_{pk_{Bob}}(\text{nonceA})$ ]

---

  2. Bob → Alice [Bob,  $E_{pk_{Alice}}(\text{nonceB}, \text{nonceA})$ ]

---

  3. Bob, Alice  $K \leftarrow H(\text{nonceA} \parallel \text{nonceB})$
-

# Protocolo na semântica Scyther

```
1 const pk: Function;  
2 secret sk: Function;  
3 inversekeys (pk,sk);  
4 const Eve: Agent;  
5 untrusted Eve;  
6 protocol exemplo(Alice,Bob,Eve){
```

# Protocolo na semântica Scyther

```
1 const pk: Function;  
2 secret sk: Function;  
3 inversekeys(pk,sk);  
4 const Eve: Agent;  
5 untrusted Eve;  
6 protocol exemplo(Alice,Bob,Eve){
```

# Protocolo na semântica Scyther

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice,Bob,{nonceA}pk(Bob));  
11    recv_2(Bob,Alice,{nonceA,nonceB}pk(Alice));  
12    claim(Alice,Secret,nonceA);  
13    claim(Alice,Secret,nonceB);  
14    claim(Alice,Nisynch);  
15 }
```

# Protocolo na semântica Scyther

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice,Bob,{nonceA}pk(Bob));  
11    recv_2(Bob,Alice,{nonceA,nonceB}pk(Alice));  
12    claim(Alice,Secret,nonceA);  
13    claim(Alice,Secret,nonceB);  
14    claim(Alice,Nisynch);  
15 }
```

# Protocolo na semântica Scyther

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice,Bob,{nonceA}pk(Bob));  
11    recv_2(Bob,Alice,{nonceA,nonceB}pk(Alice));  
12    claim(Alice,Secret,nonceA);  
13    claim(Alice,Secret,nonceB);  
14    claim(Alice,Nisynch);  
15 }
```

# Protocolo na semântica Scyther

```
16    role Bob{  
17        var nonceA: Nonce;  
18        fresh nonceB: Nonce;  
19        recv_1(Alice, Bob, {nonceA}pk(Bob));  
20        send_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
21        claim(Bob, Secret, nonceA);  
22        claim(Bob, Secret, nonceB);  
23        claim(Bob, Nisynch);  
24    }  
25 }
```

# Protocolo na semântica Scyther

```
16  role Bob{  
17      var nonceA: Nonce;  
18      fresh nonceB: Nonce;  
19      recv_1(Alice, Bob, {nonceA}pk(Bob));  
20      send_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
21      claim(Bob, Secret, nonceA);  
22      claim(Bob, Secret, nonceB);  
23      claim(Bob, Nisynch);  
24  }  
25 }
```

# Protocolo na semântica Scyther

```
16  role Bob{  
17      var nonceA: Nonce;  
18      fresh nonceB: Nonce;  
19      recv_1(Alice, Bob, {nonceA}pk(Bob));  
20      send_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
21      claim(Bob, Secret, nonceA);  
22      claim(Bob, Secret, nonceB);  
23      claim(Bob, Nisynch);  
24  }  
25 }
```

# Verificação com a Scyther

Protocolos na semântica Scyther:

<https://s4a.in/github>

d:~\$ cd ~/

d:~\$ git clone

<https://github.com/scyther-lea/errc2020.git>

# Verificação com a Scyther

<https://s4a.in/scytherSite>

d:~\$ cd ~/

d:~\$ wget

https://people.cispa.io/cas.cremers/downloads/scyther/scyther-linux-v1.1.3.tgz

d:~\$ tar xzvf scyther-linux-v1.1.3.tgz

d:~\$ cd scyther-linux-v1.1.3

d:~\$ cp ../errc2020/scyther/\*.spdl .

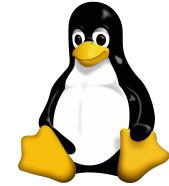
# Verificação com a Scyther

```
d:~$ ./scyther.py --all-attacks --max-runs=5  
protocolo_exemplo.spdl
```

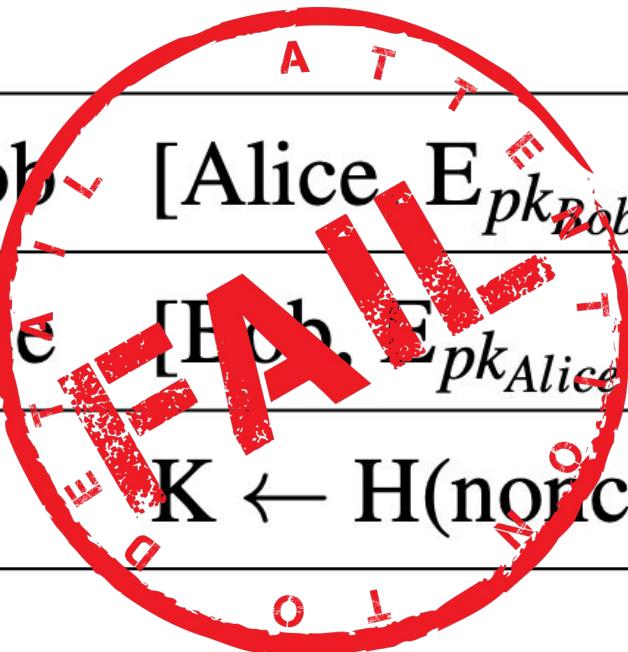
```
./scyther.py --all-attacks --max-runs=5 protocolo_exemplo.spdl
```

Verification results:

```
claim id [exemplo,Alice1], Secret(nA) : No attacks.  
claim id [exemplo,Alice2], Secret(nB) : No attacks.  
claim id [exemplo,Alice3], Nisynch : No attacks.  
claim id [exemplo,Bob1], Secret(nA) : Exactly 1 attack.  
claim id [exemplo,Bob2], Secret(nB) : No attacks.  
claim id [exemplo,Bob3], Nisynch : Exactly 1 attack.
```



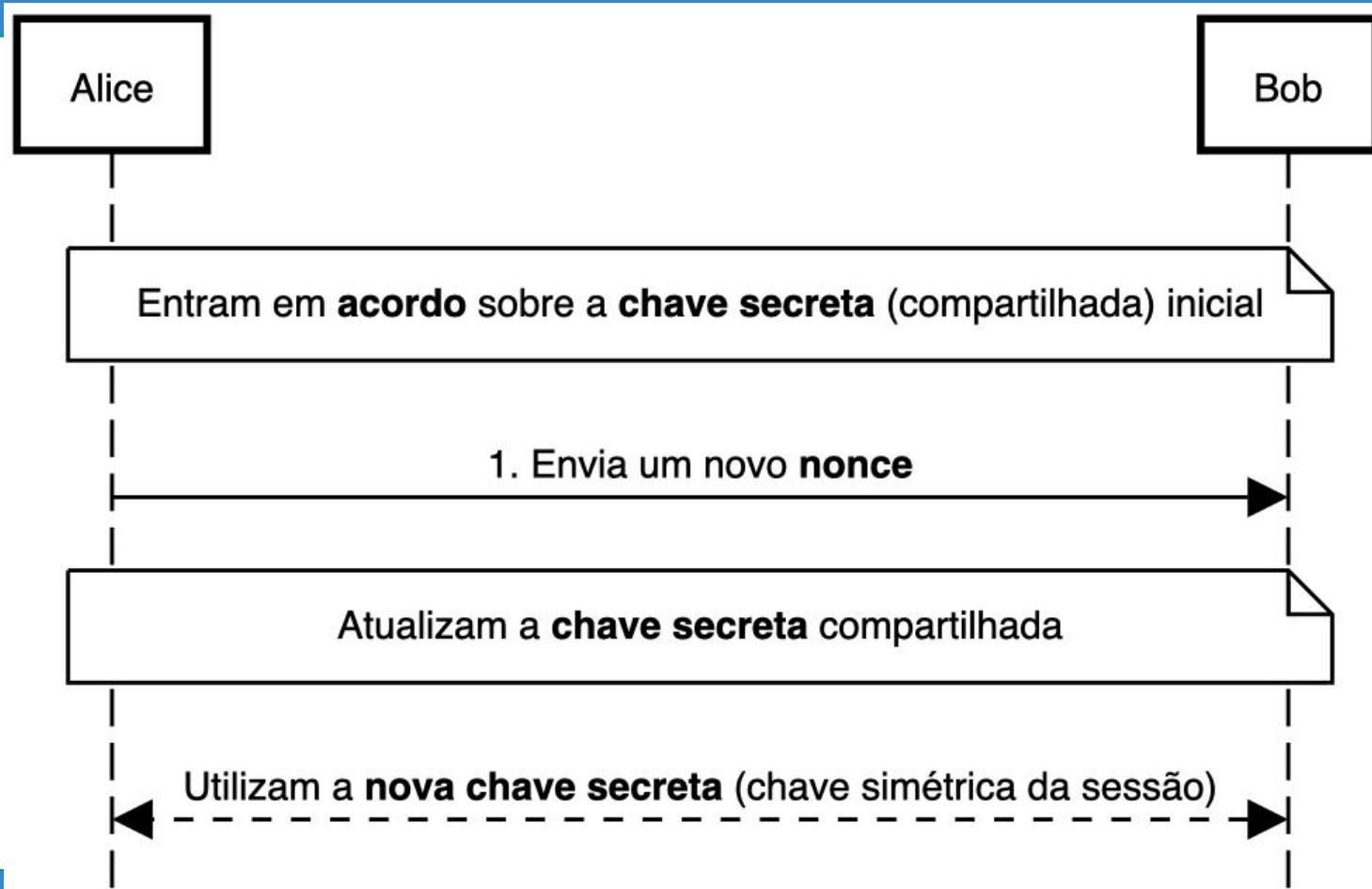
# Protocolo exemplo (vulnerable)

- 
1. Alice → Bob [Alice,  $E_{pk_{Bob}}(\text{nonceA})$ ]
  2. Bob → Alice [Bob,  $E_{pk_{Alice}}(\text{nonceB}, \text{nonceA})$ ]
  3. Bob, Alice  $K \leftarrow H(\text{nonceA} \parallel \text{nonceB})$
- 

# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ❑ O protocolo ACS
- ❑ O protocolo WMF
- ❑ O protocolo NS
- ❑ O protocolo GNSL
- ❑ Considerações finais

# O protocolo ACS (diagrama)



# O protocolo ACS (especificação)

---

1. Alice → Bob [E<sub>K</sub>(nonce)]

---

# O protocolo ACS (especificação)

---

1. Alice → Bob [ $E_K(\text{nonce})$ ]

---

2. Bob, Alice  $K \leftarrow H(K \parallel \text{nonce})$

---

# O protocolo ACS (adicionar HMAC)

---

1. Alice → Bob [ $E_K(\text{nonce})$ ], **HMAC**

---

2. Bob, Alice  $K \leftarrow H(K \parallel \text{nonce})$

---

**Exemplo:** <https://s4a.in/auth4app>

# O protocolo ACS (semântica Scyther)

```
1 secret K: SessionKey;  
2 const Eve: Agent;  
3 untrusted Eve;  
4 protocol ACS(Alice,Bob,Eve){
```

Há uma  
chave “k”  
(minúscula)  
padrão na  
Scyther

# O protocolo ACS (semântica Scyther)

```
1 secret K: SessionKey;
2 const Eve: Agent;
3 untrusted Eve;
4 protocol ACS(Alice,Bob,Eve){
```

# O protocolo ACS (semântica Scyther)

```
1 secret K: SessionKey;  
2 const Eve: Agent;  
3 untrusted Eve;  
4 protocol ACS(Alice,Bob,Eve){ }
```

# O protocolo ACS (semântica Scyther)

```
5   role Alice{  
6       fresh nonce: Nonce;  
7       send_1(Alice,Bob,{nonce}K(Alice,Bob));  
8       claim_Alice1(Alice,Secret,nonce);  
9       claim_Alice2(Alice,Secret,K);  
10      }  
11      role Bob{  
12          var nonce: Nonce;  
13          recv_1(Alice,Bob,{nonce}K(Alice,Bob));  
14          claim_Bob1(Bob,Secret,nonce);  
15          claim_Bob2(Bob,Secret,K);  
16      }
```

# O protocolo ACS (semântica Scyther)

```
5   role Alice{  
6       fresh nonce: Nonce;  
7       send_1(Alice,Bob,{nonce}K(Alice,Bob));  
8       claim_Alice1(Alice,Secret,nonce);  
9       claim_Alice2(Alice,Secret,K);  
10      }  
11      role Bob{  
12          var nonce: Nonce;  
13          recv_1(Alice,Bob,{nonce}K(Alice,Bob));  
14          claim_Bob1(Bob,Secret,nonce);  
15          claim_Bob2(Bob,Secret,K);  
16      }
```

# O protocolo ACS (semântica Scyther)

```
5   role Alice{  
6       fresh nonce: Nonce;  
7       send_1(Alice,Bob,{nonce}K(Alice,Bob));  
8       claim_Alice1(Alice,Secret,nonce);  
9       claim_Alice2(Alice,Secret,K);  
10      }  
11      role Bob{  
12          var nonce: Nonce;  
13          recv_1(Alice,Bob,{nonce}K(Alice,Bob));  
14          claim_Bob1(Bob,Secret,nonce);  
15          claim_Bob2(Bob,Secret,K);  
16      }
```

# O protocolo ACS (semântica Scyther)

```
5   role Alice{  
6       fresh nonce: Nonce;  
7       send_1(Alice,Bob,{nonce}K(Alice,Bob));  
8       claim_Alice1(Alice,Secret,nonce);  
9       claim_Alice2(Alice,Secret,K);  
10    }  
11    role Bob{  
12        var nonce: Nonce;  
13        recv_1(Alice,Bob,{nonce}K(Alice,Bob));  
14        claim_Bob1(Bob,Secret,nonce);  
15        claim_Bob2(Bob,Secret,K);  
16    }
```

# O protocolo ACS (semântica Scyther)

```
1 secret K: SessionKey;
2 const Eve: Agent;
3 untrusted Eve;
4 protocol ACS(Alice,Bob,Eve){
5   role Alice{
6     fresh nonce: Nonce;
7     send_1(Alice,Bob,{nonce}K(Alice,Bob));
8     claim_Alice1(Alice,Secret,nonce);
9     claim_Alice2(Alice,Secret,K);
10  }
11  role Bob{
12    var nonce: Nonce;
13    recv_1(Alice,Bob,{nonce}K(Alice,Bob));
14    claim_Bob1(Bob,Secret,nonce);
15    claim_Bob2(Bob,Secret,K);
16  }
17 }
```

# O protocolo ACS (verificação Scyther)

```
d:~$ ./scyther.py --all-attacks --max-runs=5  
protocolo_acs.spdl
```

```
./scyther.py --all-attacks --max-runs=5 protocolo_acs.spdl
```

Verification results:

```
claim id [ACS,Alice1], Secret(nonce) : No attacks.  
claim id [ACS,Alice2], Secret(K)      : No attacks.  
claim id [ACS,Bob1], Secret(nonce)   : No attacks.  
claim id [ACS,Bob2], Secret(K)       : No attacks.
```



# O protocolo ACS (aprovado pela Scyther)

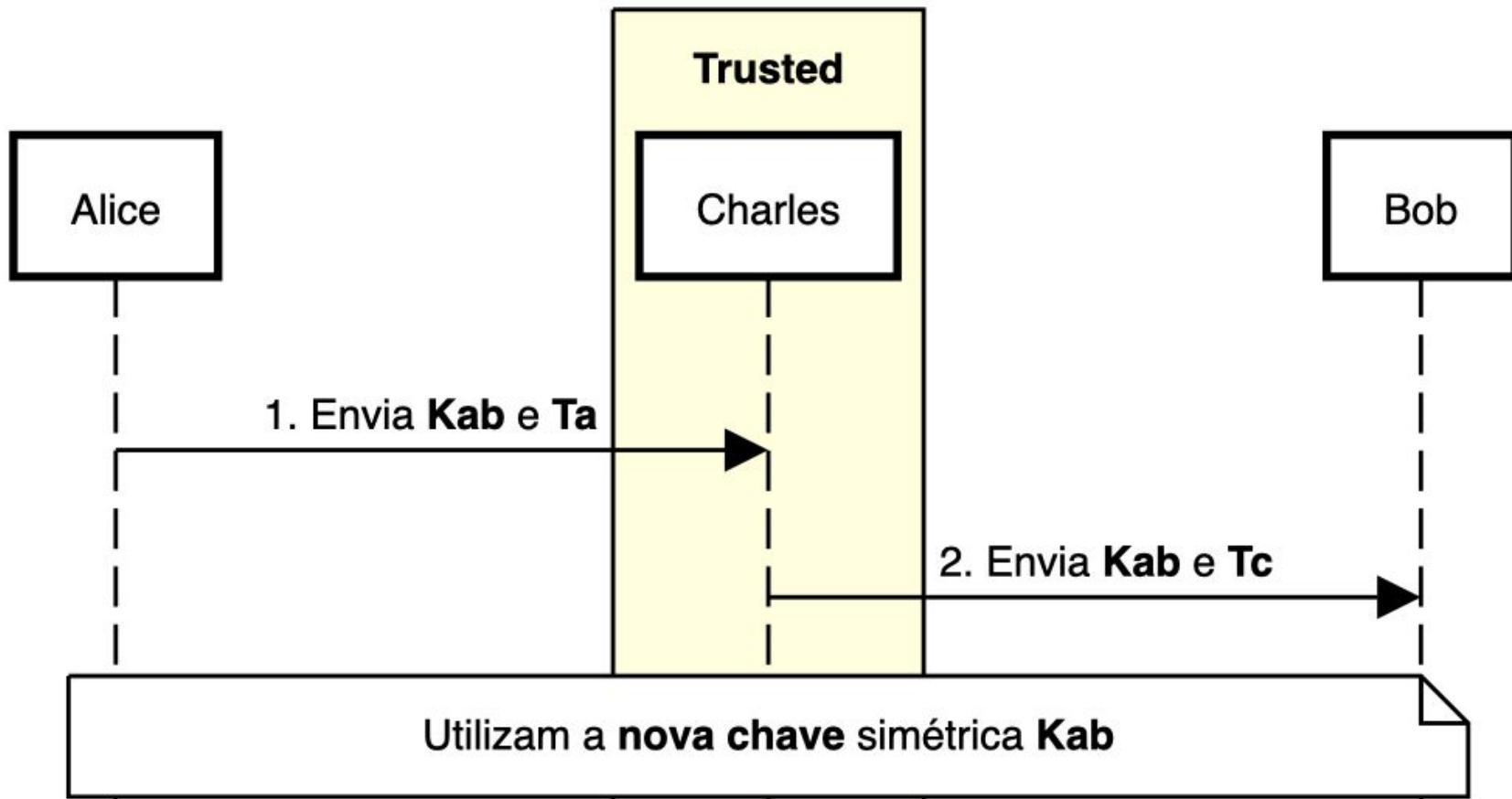
- 
1. Alice → Bob  $[E_K(\text{nonce})]$
  2. Bob, Alice  $K \leftarrow H(K \parallel \text{nonce})$
- 

APPROVED

# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ✓ O protocolo ACS
- O protocolo WMF**
- O protocolo NS
- O protocolo GNSL
- Considerações finais

# O protocolo WMF (diagrama)



# O protocolo WMF (especificação)

---

1. Alice → Charles [Alice,  $E_{K_{ac}}(T_a, Bob, K_{ab})$ ]

---

# O protocolo WMF (especificação)

---

1. Alice → Charles [Alice,  $E_{K_{ac}}(T_a, Bob, K_{ab})$ ]

---

2. Charles → Bob [ $E_{K_{bc}}(T_c, Alice, K_{ab})$ ]

# O protocolo WMF (semântica Scyther)

```
5   role Alice{  
6       fresh Kab: SessionKey;  
7       fresh Ta: TimeStamp;  
8       send_1(Alice,Charles,Alice,{Alice,Ta,Bob,Kab}k(Alice,Charles));  
9       claim_Alice1(Alice,Secret,Kab);  
10      claim_Alice2(Alice,Empty,(Fresh,Kab));  
11  }
```

# O protocolo WMF (semântica Scyther)

```
5   role Alice{  
6     fresh Kab: SessionKey;  
7     fresh Ta: TimeStamp;  
8     send_1(Alice,Charles,Alice,{ Alice,Ta,Bob,Kab }k(Alice,Charles));  
9     claim_ Alice1(Alice,Secret,Kab);  
10    claim_ Alice2(Alice,Empty,(Fresh,Kab));  
11 }
```

# O protocolo WMF (semântica Scyther)

```
5   role Alice{  
6     fresh Kab: SessionKey;  
7     fresh Ta: TimeStamp;  
8     send_1(Alice,Charles,Alice,{ Alice,Ta,Bob,Kab }k(Alice,Charles));  
9     claim_Alice1(Alice,Secret,Kab);  
10    claim_Alice2(Alice,Empty,(Fresh,Kab));  
11 }
```

# O protocolo WMF (semântica Scyther)

```
12   role Bob{  
13     var Tc: TimeStamp;  
14     var Kab: SessionKey;  
15     recv_2(Charles,Bob,{Charles,Tc,Alice,Kab}k(Bob,Charles));  
16     claim_Bob1(Bob,Secret,Kab);  
17     claim_Bob2(Bob,Nisynch);  
18     claim_Bob3(Bob,Empty,(Fresh,Kab));  
19 }
```

# O protocolo WMF (semântica Scyther)

```
12    role Bob{  
13        var Tc: TimeStamp;  
14        var Kab: SessionKey;  
15        recv_2(Charles,Bob,{Charles,Tc,Alice,Kab}k(Bob,Charles));  
16        claim_Bob1(Bob,Secret,Kab);  
17        claim_Bob2(Bob,Nisynch);  
18        claim_Bob3(Bob,Empty,(Fresh,Kab));  
19    }
```

# O protocolo WMF (semântica Scyther)

```
12  role Bob{  
13      var Tc: TimeStamp;  
14      var Kab: SessionKey;  
15      recv_2(Charles,Bob,{Charles,Tc,Alice,Kab}k(Bob,Charles));  
16      claim_Bob1(Bob,Secret,Kab);  
17      claim_Bob2(Bob,Nisynch);  
18      claim_Bob3(Bob,Empty,(Fresh,Kab));  
19 }
```

# O protocolo WMF (semântica Scyther)

```
20    role Charles{  
21        var Kab: SessionKey;  
22        var Ta: TimeStamp;  
23        fresh Tc:TimeStamp;  
24        recv_1( Alice,Charles,Alice,{Alice,Ta,Bob,Kab}k(Alice,Charles));  
25        send_2(Charles,Bob,{Charles,Tc,Alice,Kab}k(Bob,Charles));  
26    }
```

# O protocolo WMF (semântica Scyther)

```
20    role Charles{  
21        var Kab: SessionKey;  
22        var Ta: TimeStamp;  
23        fresh Tc:TimeStamp;  
24        recv_1( Alice,Charles,Alice,{Alice,Ta,Bob,Kab}k(Alice,Charles));  
25        send_2(Charles,Bob,{Charles,Tc,Alice,Kab}k(Bob,Charles));  
26    }
```

# O protocolo WMF (semântica Scyther)

```
20    role Charles{  
21        var Kab: SessionKey;  
22        var Ta: TimeStamp;  
23        fresh Tc:TimeStamp;  
24        recv_1( Alice,Charles,Alice,{Alice,Ta,Bob,Kab})k(Alice,Charles);  
25        send_2(Charles,Bob,{ Charles,Tc,Alice,Kab}k(Bob,Charles));  
26    }
```

# O protocolo WMF (semântica Scyther)

Exemplo WMF em: <https://s4a.in/github>

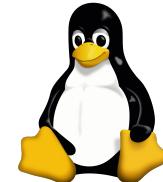
```
d:~$ git clone https://github.com/scyther-lea/errc2020.git  
d:~$ cd errc2020/scyther  
d:~$ vim protocolo_wmf.spdl
```

# O protocolo WMF (verificação Scyther)

```
./scyther.py --auto-claims --all-attacks protocolo_wmf.spdl
```

Verification results:

```
claim id [WMF, Alice3], Secret(Ta)      : No attacks.  
claim id [WMF, Alice4], Secret(Kab)      : No attacks within bounds.  
claim id [WMF, Alice5], Alive             : Exactly 1 attack.  
claim id [WMF, Alice6], Weakagree        : Exactly 1 attack.  
claim id [WMF, Alice7], Niagree           : No attacks.  
claim id [WMF, Alice8], Nisynch           : No attacks.  
claim id [WMF, Bob4], Secret(Kab)         : No attacks within bounds.  
claim id [WMF, Bob5], Secret(Tc)          : No attacks within bounds.  
claim id [WMF, Bob6], Alive               : Exactly 1 attack.  
claim id [WMF, Bob7], Weakagree          : Exactly 1 attack.  
claim id [WMF, Bob8], Niagree             : At least 3 attacks.  
claim id [WMF, Bob9], Nisynch             : At least 3 attacks.  
claim id [WMF, Charles1], Secret(Tc)       : No attacks within bounds.  
claim id [WMF, Charles2], Secret(Ta)       : No attacks within bounds.  
claim id [WMF, Charles3], Secret(Kab)       : No attacks within bounds.  
claim id [WMF, Charles4], Alive            : Exactly 1 attack.  
claim id [WMF, Charles5], Weakagree        : Exactly 1 attack.  
claim id [WMF, Charles6], Niagree           : At least 3 attacks.  
claim id [WMF, Charles7], Nisynch           : At least 3 attacks.
```



# O protocolo WMF (ataques)

- 
1. Alice → Charles [Alice,  $E_{K_{ac}}(T_a, Bob, K_{ab})$ ]
  2. Charles → Bob [ $E_{K_{bc}}(T_b, Alice, K_{ab})$ ]
- 



# O protocolo WMF (pontos fracos)

- ❑ Requer um **relógio global** (*timestamps*)
- ❑ Charles possui **acesso às chaves**
- ❑ Chave de sessão **Kab** definida por Alice
- ❑ **Ataques de *replay*** (intervalo do *timestamp*)
- ❑ Protocolo ***stateful*** (Charles)

# O protocolo WMF-Lowe (especificação)

- 
1. Alice → Charles [ $E_{K_{ac}}(T_a, Bob, K_{ab})$ ]
- 
2. Charles → Alice [ $E_{K_{ac}}(T_c, Alice, K_{ab})$ ]
- 
3. Bob → Alice [ $E_{K_{ab}}(\text{nonceB})$ ]
- 
4. Alice → Bob [ $E_{K_{ab}}(\text{nonceB}+1)$ ]

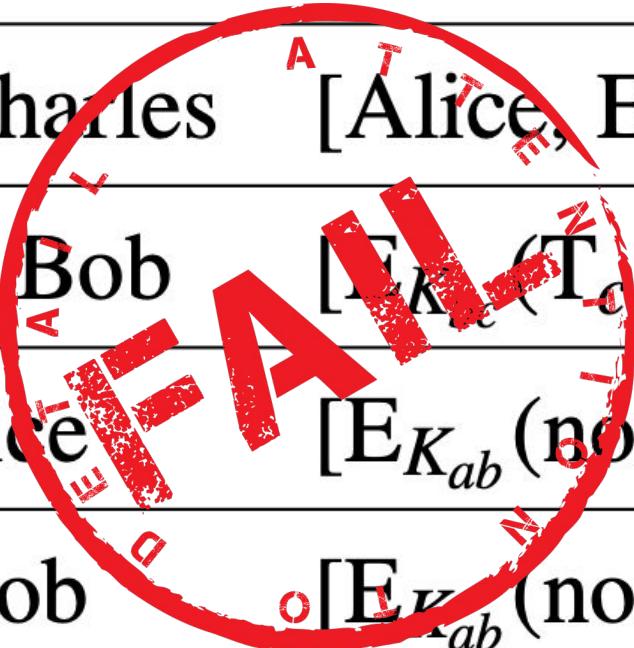
# O protocolo WMF-Lowe (verificação)

```
d:~$ ./scyther.py --all-attacks --max-runs=5  
protocolo_wmf_lowe.spdl
```



```
errc2020@d:~/scyther-linux-v1.1.3$ ./scyther.py --all-attacks --max-runs=5  
protocolo_wmf_lowe.spdl  
Verification results:  
claim id [WMF-Lowe,Alice1], Secret(Kab) : No attacks within bounds.  
[claim id [WMF-Lowe,Alice2], Nisynch : At least 3 attacks.]  
claim id [WMF-Lowe,Alice4], Secret(nonceB) : No attacks within bounds.  
claim id [WMF-Lowe,Bob1], Secret(Kab) : No attacks within bounds.  
[claim id [WMF-Lowe,Bob2], Nisynch : At least 3 attacks.]  
claim id [WMF-Lowe,Bob4], Secret(nonceB) : No attacks within bounds.
```

# O protocolo WMF-Lowe (ataques)

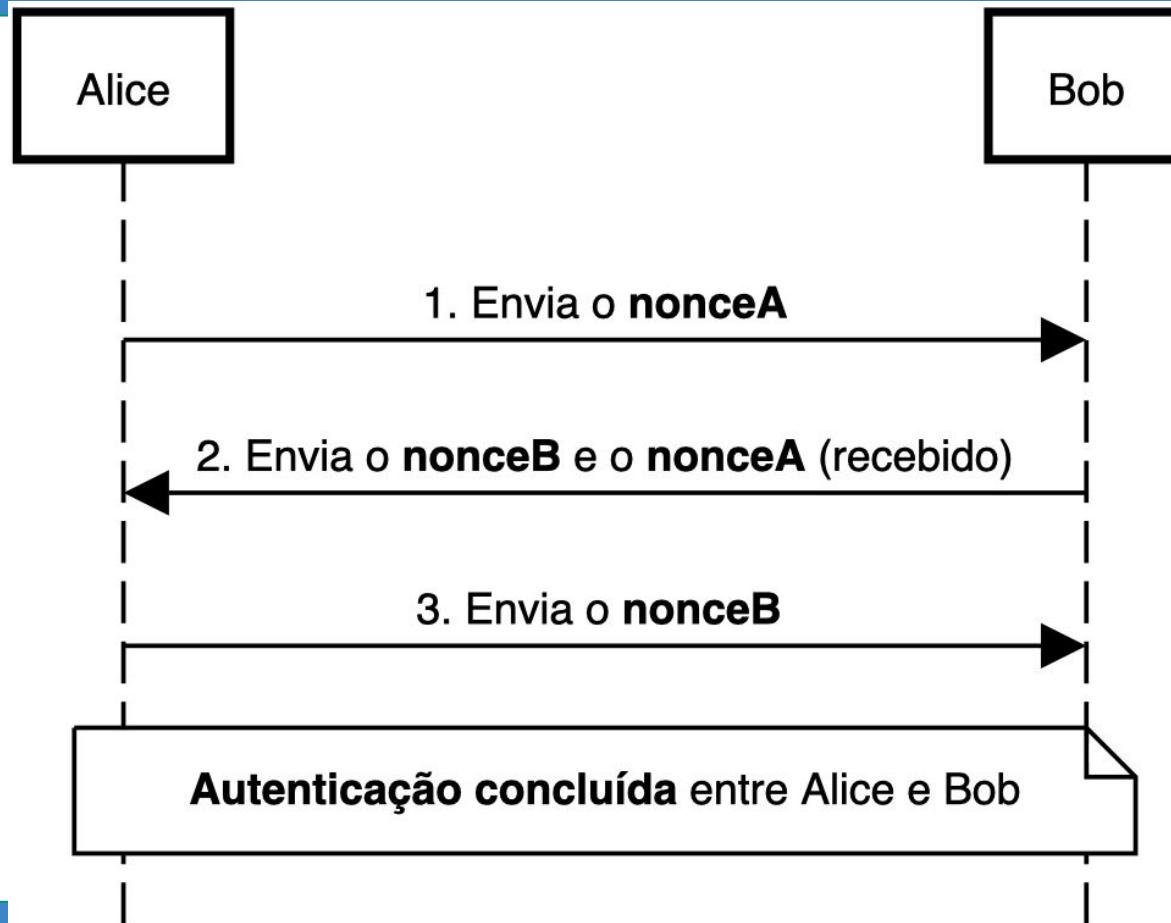
- 
1. Alice → Charles  $[A^T, E_{K_{ac}}(T_a, Bob, K_{ab})]$
  2. Charles → Bob  $[E_{K_{ac}}(T_c, Alice, K_{ab})]$
  3. Bob → Alice  $[E_{K_{ab}}(\text{nonceB})]$
  4. Alice → Bob  $[E_{K_{ab}}(\text{nonceB}+1)]$
- 

# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ✓ O protocolo ACS
- ✓ O protocolo WMF
- ❑ O protocolo NS
- ❑ O protocolo GNSL
- ❑ Considerações finais

# O protocolo NS (diagrama)

Autenticação para  
dois Participantes



# O protocolo NS (especificação)

---

1. Alice → Bob [E<sub>pk<sub>Bob</sub></sub>(Alice, nonceA)]

---

# O protocolo NS (especificação)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{nonceA})$ ]

---

  2. Bob → Alice [ $E_{pk_{Alice}}(\text{nonceA}, \text{nonceB})$ ]

---

# O protocolo NS (especificação)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{nonceA})$ ]

---

  2. Bob → Alice [ $E_{pk_{Alice}}(\text{nonceA}, \text{nonceB})$ ]

---

  3. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB})$ ]
-

# O protocolo NS (semântica Scyther)

```
1 const pk: Function;  
2 secret sk: Function;  
3 inversekeys (pk,sk);  
4 const Eve: Agent;  
5 untrusted Eve;  
6 protocol NS(Alice,Bob,Eve){
```

# O protocolo NS (semântica Scyther)

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice,Bob,{Alice,nonceA}pk(Bob));  
11    recv_2(Bob,Alice,{nonceA,nonceB}pk(Alice));  
12    send_3(Alice,Bob,{nonceB}pk(Bob));  
13    claim(Alice,Secret,nonceA);  
14    claim(Alice,Secret,nonceB);  
15    claim(Alice,Nisynch);  
16 }
```

# O protocolo NS (semântica Scyther)

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice, Bob, {Alice,nonceA}pk(Bob));  
11    recv_2(Bob, Alice, {nonceA,nonceB}pk(Alice));  
12    send_3(Alice, Bob, {nonceB}pk(Bob));  
13    claim(Alice, Secret, nonceA);  
14    claim(Alice, Secret, nonceB);  
15    claim(Alice, Nisynch);  
16 }
```

# O protocolo NS (semântica Scyther)

```
7   role Alice{  
8     fresh nonceA: Nonce;  
9     var nonceB: Nonce;  
10    send_1(Alice, Bob, {Alice, nonceA}pk(Bob));  
11    recv_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
12    send_3(Alice, Bob, {nonceB}pk(Bob));  
13    claim(Alice, Secret, nonceA);  
14    claim(Alice, Secret, nonceB);  
15    claim(Alice, Nisynch);  
16 }
```

# O protocolo NS (semântica Scyther)

```
17    role Bob{  
18        var nonceA: Nonce;  
19        fresh nonceB: Nonce;  
20        recv_1(Alice, Bob, {Alice, nonceA}pk(Bob));  
21        send_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
22        recv_3(Alice, Bob, {nonceB}pk(Bob));  
23        claim(Bob, Secret, nonceA);  
24        claim(Bob, Secret, nonceB);  
25        claim(Bob, Nisynch);  
26    }
```

# O protocolo NS (semântica Scyther)

```
17   role Bob{  
18     var nonceA: Nonce;  
19     fresh nonceB: Nonce;  
20     recv_1(Alice, Bob, {Alice,nonceA}pk(Bob));  
21     send_2(Bob, Alice, {nonceA,nonceB}pk(Alice));  
22     recv_3(Alice, Bob, {nonceB}pk(Bob));  
23     claim(Bob, Secret, nonceA);  
24     claim(Bob, Secret, nonceB);  
25     claim(Bob, Nisynch);  
26 }
```

# O protocolo NS (semântica Scyther)

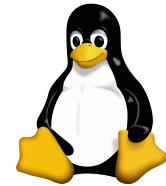
```
17   role Bob{  
18     var nonceA: Nonce;  
19     fresh nonceB: Nonce;  
20     recv_1(Alice, Bob, {Alice, nonceA}pk(Bob));  
21     send_2(Bob, Alice, {nonceA, nonceB}pk(Alice));  
22     recv_3(Alice, Bob, {nonceB}pk(Bob));  
23     claim(Bob, Secret, nonceA);  
24     claim(Bob, Secret, nonceB);  
25     claim(Bob, Nisynch);  
26 }
```

# O protocolo NS (verificação Scyther)

```
d:~$ ./scyther.py --all-attacks --max-runs=5  
protocolo_ns.spdl  
  
../scyther.py --all-attacks --max-runs=5 protocolo_ns.spdl
```

Verification results:

```
claim id [NS,Alice1], Secret(nonceA) : No attacks.  
claim id [NS,Alice2], Secret(nonceB) : No attacks.  
claim id [NS,Alice3], Nisynch       : No attacks.  
claim id [NS,Bob1],   Secret(nonceA) : Exactly 1 attack.  
claim id [NS,Bob2],   Secret(nonceB) : Exactly 1 attack.  
claim id [NS,Bob3],   Nisynch       : Exactly 1 attack.
```



# O protocolo NS (ataque)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{nonceA})$ ]

---

  2. Bob → Alice [ $E_{pk_{Alice}}(\text{nonceA}, \text{nonceB})$ ]

---

  3. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB})$ ]
- 

# O protocolo NS (ataque)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{nonceA})$ ]
  2. Bob → Alice [ $E_{pk_{Alice}}(\text{nonceA}, \text{nonceB})$ ]
  3. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB})$ ]
-

# O protocolo NS (correção - NSL)

1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{nonceA})$ ]
2. Bob → Alice [ $E_{pk_{Alice}}(\text{nonceA}, \text{nonceB}, \text{Bob})$ ]
3. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB})$ ]



# O protocolo NSL (verificação Scyther)

```
debian:~$ ./scyther.py --all-attacks --max-runs=5  
protocolo_nsl.spdl
```

```
./scyther.py --all-attacks --max-runs=5 protocolo_ns_corrigido.spdl
```

Verification results:

```
claim id [NS,Alice1], Secret(nonceA) : No attacks.  
claim id [NS,Alice2], Secret(nonceB) : No attacks.  
claim id [NS,Alice3], Nisynch : No attacks.  
claim id [NS,Bob1], Secret(nonceA) : No attacks.  
claim id [NS,Bob2], Secret(nonceB) : No attacks.  
claim id [NS,Bob3], Nisynch : No attacks.
```



# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ✓ O protocolo ACS
- ✓ O protocolo WMF
- ✓ O protocolo NS
- O protocolo GNSL
- Considerações finais

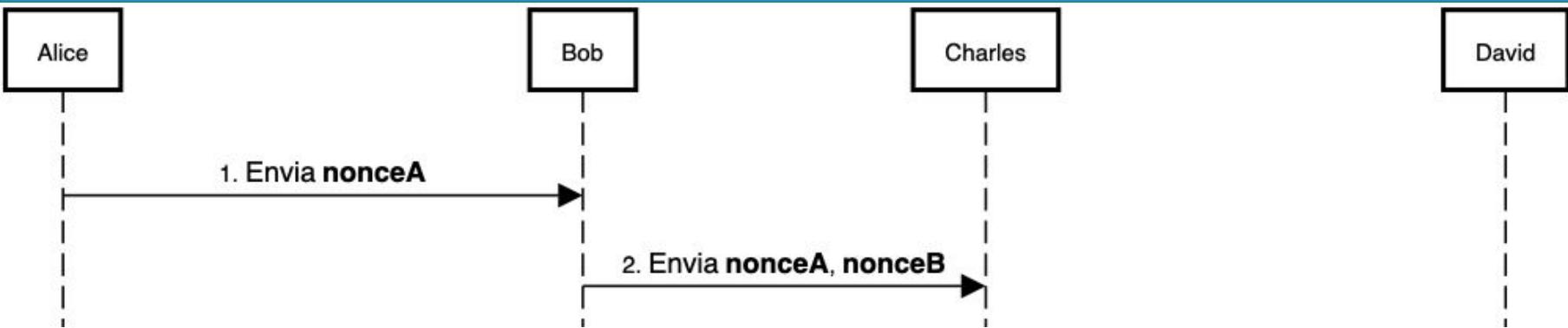
# O protocolo GNSL (conceito)

- ❑ NSL generalizado para múltiplas entidades
  - ❑ Substitui autenticação mútua aos pares
- ❑ Cada mensagem carrega informações sobre "todos"
  - ❑ Reduz o número de mensagens
  - ❑ Pressupõe confiança nas entidades autenticadas

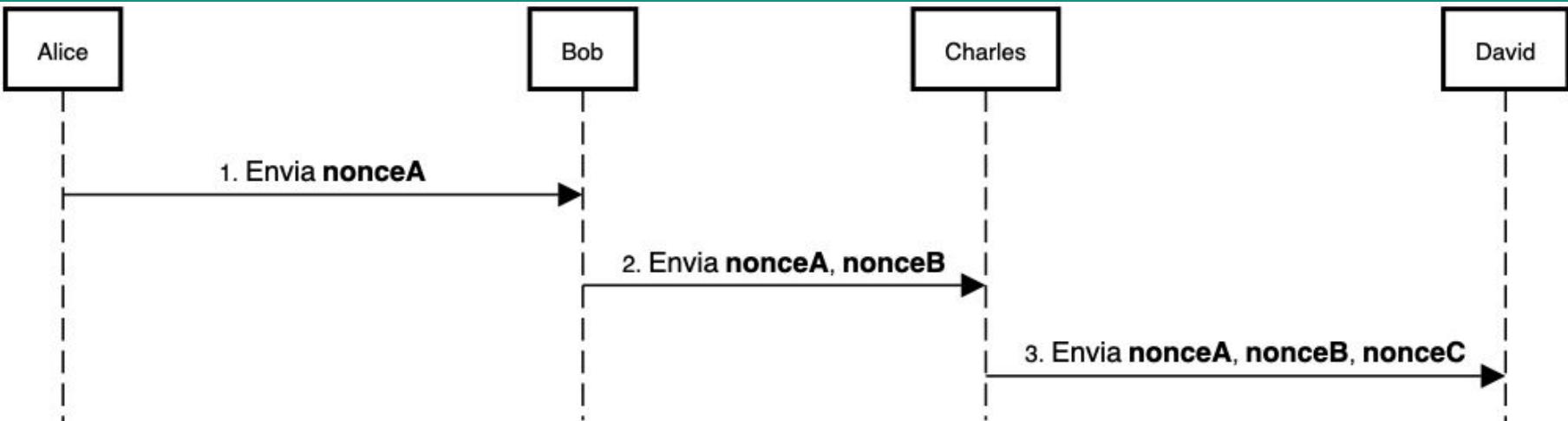
# O protocolo 4GNSL (diagrama)



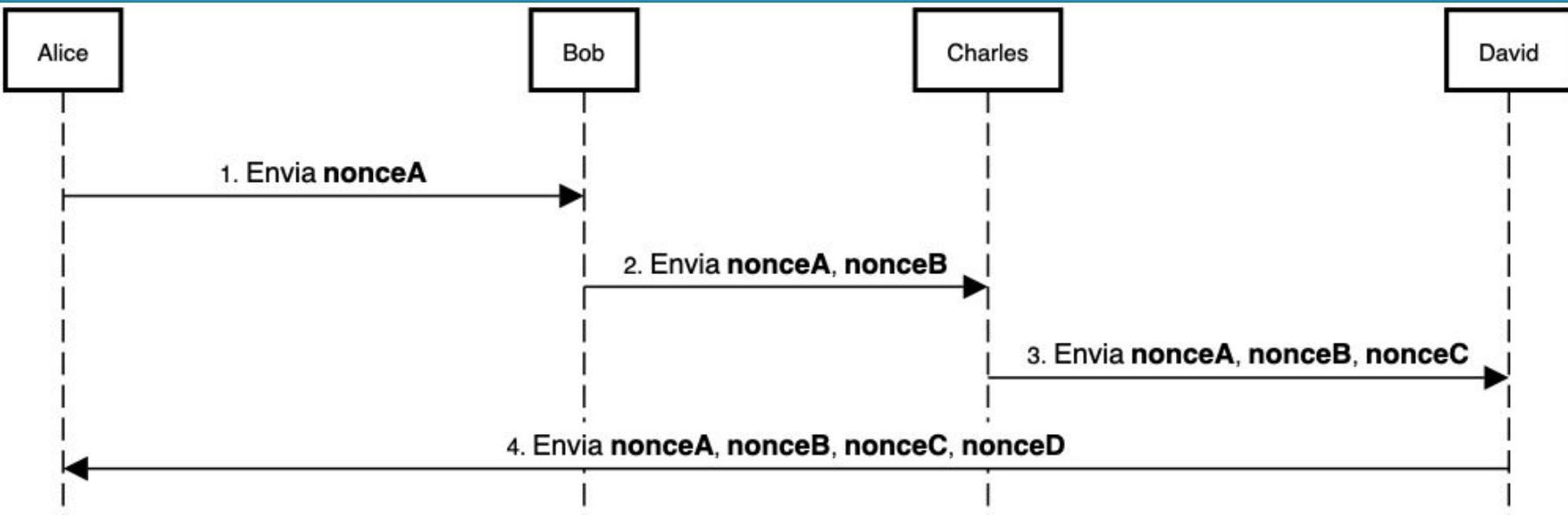
# O protocolo 4GNSL (diagrama)



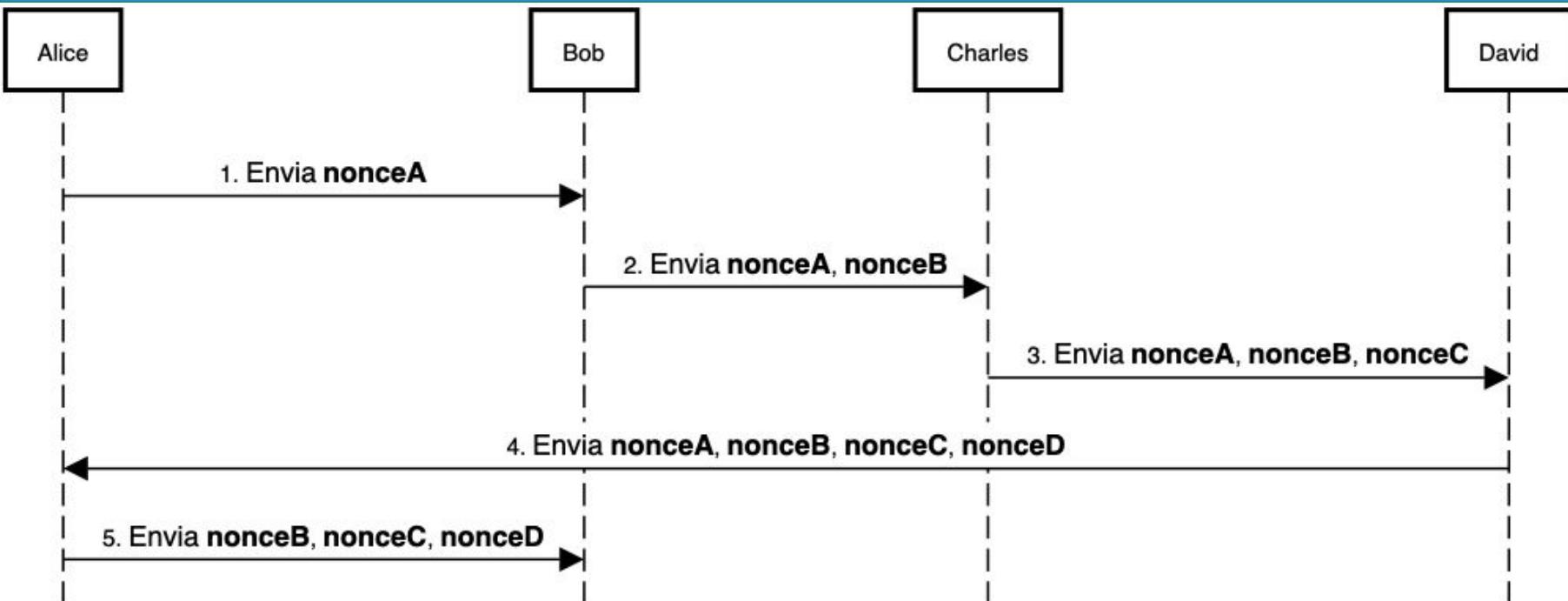
# O protocolo 4GNSL (diagrama)



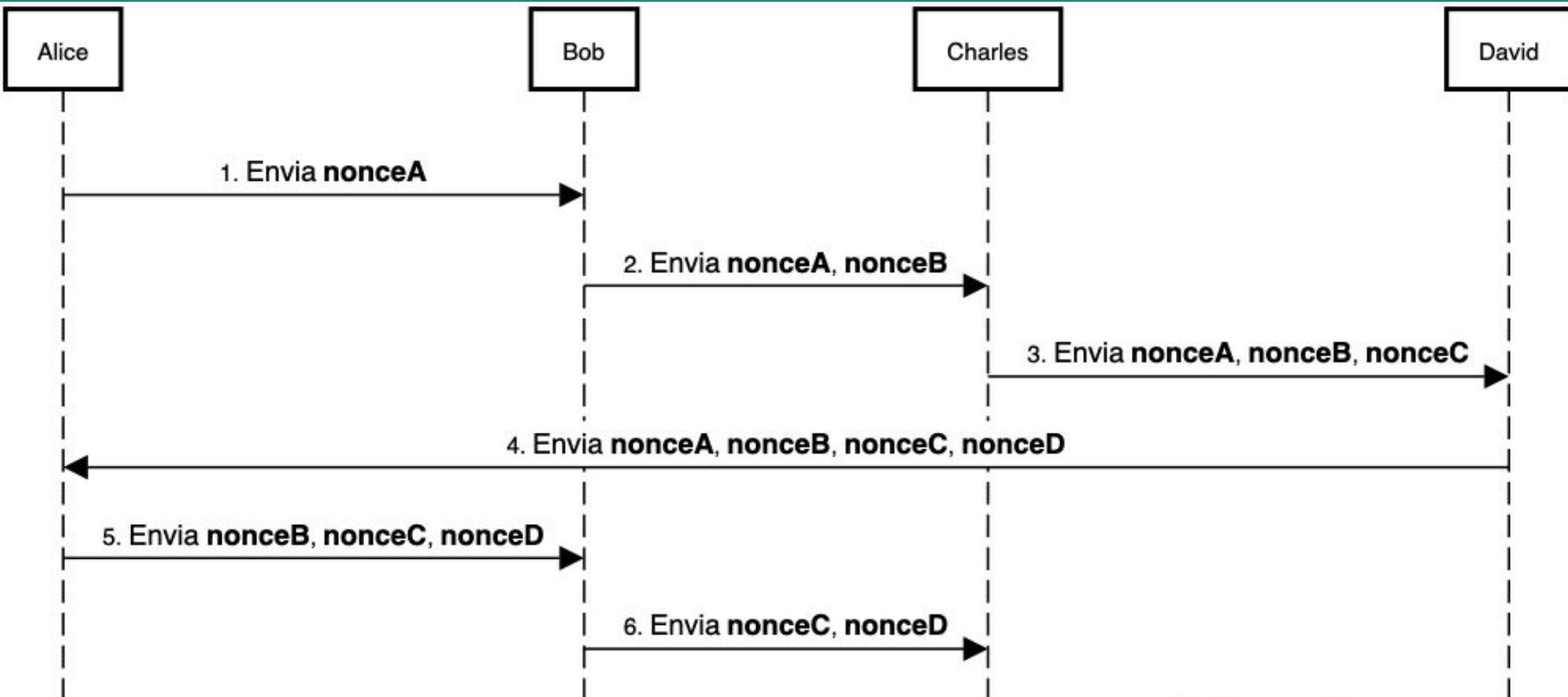
# O protocolo 4GNSL (diagrama)



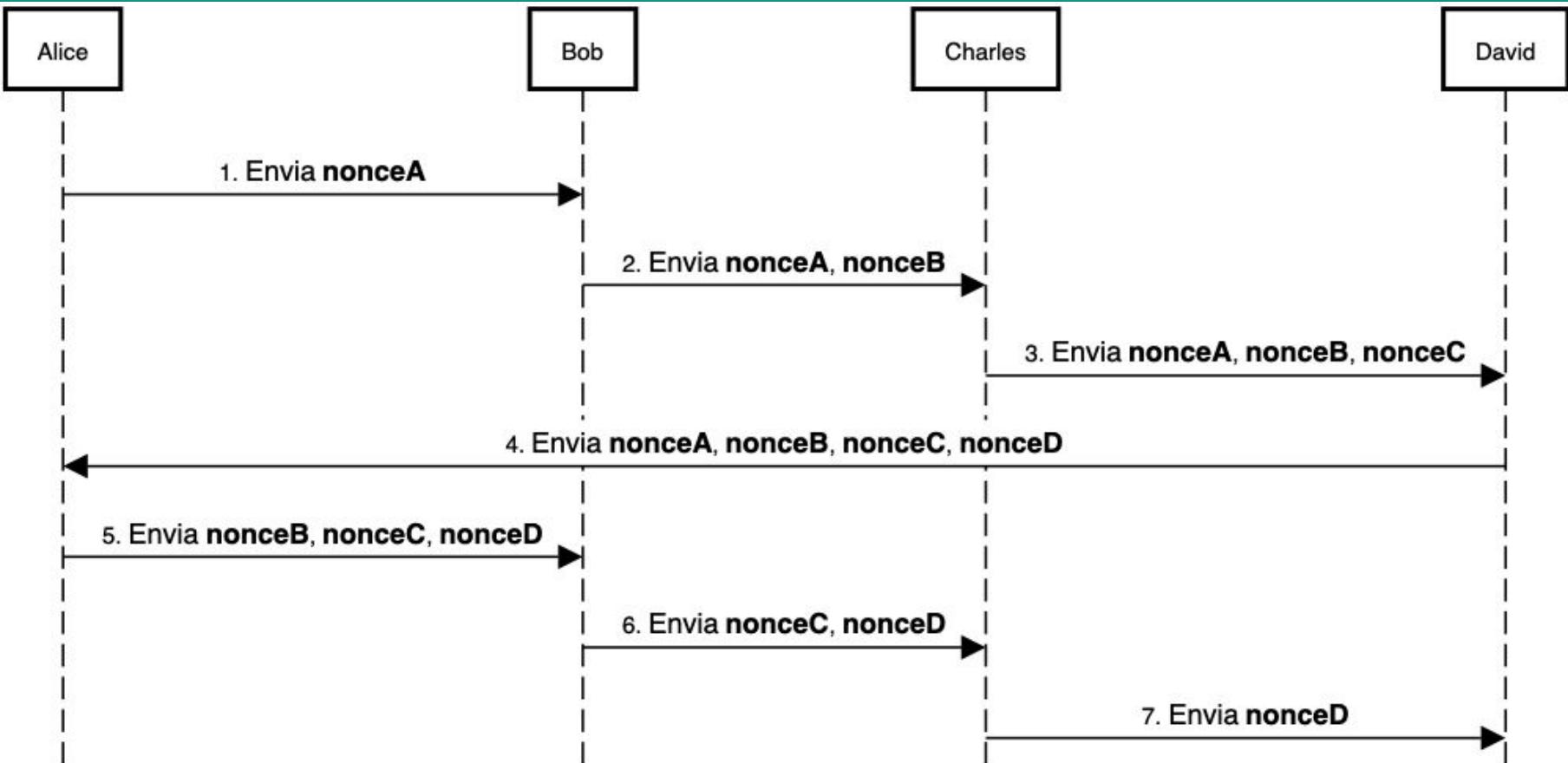
# O protocolo 4GNSL (diagrama)



# O protocolo 4GNSL (diagrama)



# O protocolo 4GNSL (diagrama)



# O protocolo 4GNSL (especificação)

---

1. Alice → Bob       $[E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})]$

---

# O protocolo 4GNSL (especificação)

---

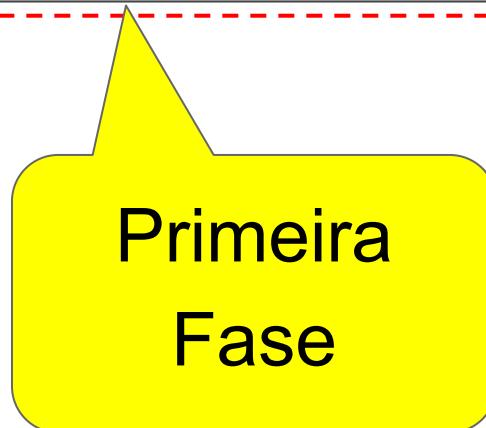
1. Alice → Bob       $[E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})]$

---

2. Bob → Charles       $[E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})]$

# O protocolo 4GNSL (especificação)

1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]



# O protocolo 4GNSL (especificação)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
  2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
  3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]
  4. David → Alice [ $E_{pk_{Alice}}(\text{Bob}, \text{Charles}, \text{David}, \text{nonceA}, \text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
-

# O protocolo 4GNSL (especificação)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
  2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
  3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]
  4. David → Alice [ $E_{pk_{Alice}}(\text{Bob}, \text{Charles}, \text{David}, \text{nonceA}, \text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  5. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
-

# O protocolo 4GNSL (especificação)

- 
- 1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
  - 2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
  - 3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]
  - 4. David → Alice [ $E_{pk_{Alice}}(\text{Bob}, \text{Charles}, \text{David}, \text{nonceA}, \text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  - 5. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  - 6. Bob → Charles [ $E_{pk_{Charles}}(\text{nonceC}, \text{nonceD})$ ]
-

# O protocolo 4GNSL (especificação)

- 
- 1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
  - 2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
  - 3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]
  - 4. David → Alice [ $E_{pk_{Alice}}(\text{Bob}, \text{Charles}, \text{David}, \text{nonceA}, \text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  - 5. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  - 6. Bob → Charles [ $E_{pk_{Charles}}(\text{nonceC}, \text{nonceD})$ ]
  - 7. Charles → David [ $E_{pk_{David}}(\text{nonceD})$ ]
-

# O protocolo 4GNSL (especificação)

Segunda  
Fase

1. Alice → B [E<sub>pk<sub>B</sub></sub>(Alice,Charles,David,nonceA)]
2. Bob → Ch [E<sub>pk<sub>Ch</sub></sub>(Bob,Charles,David,nonceA,nonceB)]
3. Charles → A [E<sub>pk<sub>A</sub></sub>(Charles,Bob,David,nonceA,nonceB,nonceC)]
4. David → Alice [E<sub>pk<sub>Alice</sub></sub>(Bob,Charles,David,nonceA,nonceB,nonceC,nonceD)]
5. Alice → Bob [E<sub>pk<sub>Bob</sub></sub>(nonceB,nonceC,nonceD)]
6. Bob → Charles [E<sub>pk<sub>Charles</sub></sub>(nonceC,nonceD)]
7. Charles → David [E<sub>pk<sub>David</sub></sub>(nonceD)]

# O protocolo 4GNSL (semântica Scyther)

Protocolo na semântica Scyther:  
<https://s4a.in/github>

```
d:~$ cd ~/  
d:~$ git clone https://github.com/scyther-lea/errc2020.git  
d:~$ cd ~/errc2020/scyther  
d:~$ vim protocolo_4gnsl.spdl
```

# O protocolo 4GNSL (verificação)

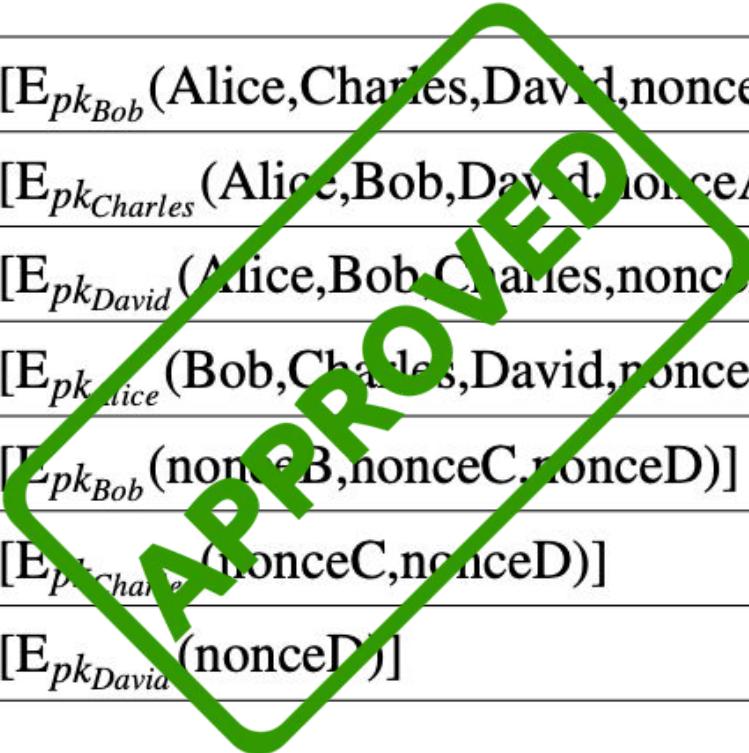
```
./scyther.py --auto-claims --all-attacks --max-runs=7 protocolo_4gnsl.spdl
```

Verification results:

claim id [4GNSL,Alice1], Secret(nonceA)	: Proof of correctness
claim id [4GNSL,Alice2], Secret(nonceD)	: No attacks within bounds.
claim id [4GNSL,Alice3], Secret(nonceC)	: No attacks within bounds.
claim id [4GNSL,Alice4], Secret(nonceB)	: Proof of correctness.
claim id [4GNSL,Alice5], Alive	: Proof of correctness.
claim id [4GNSL,Alice6], Weakagree	: Proof of correctness.
claim id [4GNSL,Alice7], Niagree	: Proof of correctness.
claim id [4GNSL,Alice8], Nisynch	: Proof of correctness.
claim id [4GNSL,Bob1], Secret(nonceB)	: No attacks within bounds.
claim id [4GNSL,Bob2], Secret(nonceD)	: No attacks within bounds.
claim id [4GNSL,Bob3], Secret(nonceC)	: No attacks within bounds.
claim id [4GNSL,Bob4], Secret(nonceA)	: No attacks within bounds.
claim id [4GNSL,Bob5], Alive	: No attacks within bounds.
claim id [4GNSL,Bob6], Weakagree	: No attacks within bounds.
claim id [4GNSL,Bob7], Niagree	: No attacks within bounds.
claim id [4GNSL,Bob8], Nisynch	: No attacks within bounds.



# O protocolo 4GNSL (aprovado Scyther)

- 
1. Alice → Bob [ $E_{pk_{Bob}}(\text{Alice}, \text{Charles}, \text{David}, \text{nonceA})$ ]
  2. Bob → Charles [ $E_{pk_{Charles}}(\text{Alice}, \text{Bob}, \text{David}, \text{nonceA}, \text{nonceB})$ ]
  3. Charles → David [ $E_{pk_{David}}(\text{Alice}, \text{Bob}, \text{Charles}, \text{nonceA}, \text{nonceB}, \text{nonceC})$ ]
  4. David → Alice [ $E_{pk_{Alice}}(\text{Bob}, \text{Charles}, \text{David}, \text{nonceA}, \text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  5. Alice → Bob [ $E_{pk_{Bob}}(\text{nonceB}, \text{nonceC}, \text{nonceD})$ ]
  6. Bob → Charles [ $E_{pk_{Charles}}(\text{nonceC}, \text{nonceD})$ ]
  7. Charles → David [ $E_{pk_{David}}(\text{nonceD})$ ]

# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ✓ O protocolo ACS
- ✓ O protocolo WMF
- ✓ O protocolo NS
- ✓ O protocolo GNSL
- Considerações finais

# Considerações finais

- ❑ Atualização de Chave Simétrica (ACS)
- ❑ Wide Mouth Frog (WMF)
- ❑ Needham-Schroeder (NS)
- ❑ Needham-Schroeder-Lowe (NSL)
- ❑ Generalized NS-Lowe (GNSL)

# Considerações finais

- ❑ Protocolos de segurança são cruciais
- ❑ Verificação automática é importante
- ❑ Falha do NS descoberta 17 anos depois
- ❑ Ferramentas: **Scyther**, Pro-Verif, CryptoVerif, AVISPA, Tamarin Prover, Coq, ...

# Roteiro

- ✓ Introdução
- ✓ A ferramenta Scyther
- ✓ O protocolo ACS
- ✓ O protocolo WMF
- ✓ O protocolo NS
- ✓ O protocolo GNSL
- ✓ Considerações finais

# Obrigado!



Diego Kreutz [diegokreutz@unipampa.edu.br](mailto:diegokreutz@unipampa.edu.br)

Rodrigo Mansilha [rodrigomansilha@unipampa.edu.br](mailto:rodrigomansilha@unipampa.edu.br)

Silvio E. Quincozes [sequincozes@gmail.com](mailto:sequincozes@gmail.com)

Tadeu Jenuário [tadeujenuarioo@gmail.com](mailto:tadeujenuarioo@gmail.com)

João Otávio Chervinski [joaoottavio@gmail.com](mailto:joaoottavio@gmail.com)

Conteúdo do minicurso: <https://s4a.in/github>

