# Accordo su chiavi

## (key agreement)

Corso di Sicurezza
a.a. 2019-20

**Alfredo De Santis**

Dipartimento di Informatica
Università di Salerno

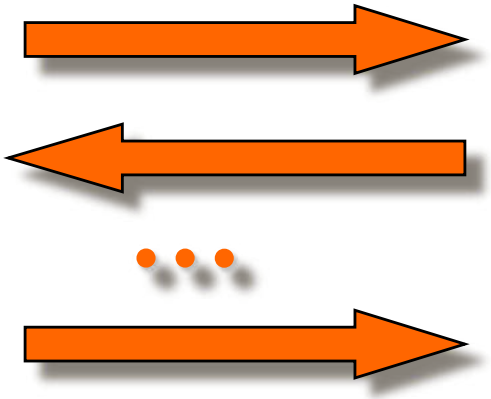
**ads@unisa.it**

**http://www.di-srv.unisa.it/~ads**

**Marzo 2020**

# Accordo su una chiave

Alice



Bob

K


K

# Accordo su chiavi: indice

- Protocollo Diffie-Hellman
- Problema del logaritmo discreto
- Generazione parametri
- Uso come cifrario a chiave pubblica
- Forward Secrecy

# Diffie-Hellman [1976]

Whitfield Diffie

Martin Hellman

Stanford University

W. Diffie and M.E. Hellman,
New directions in cryptography,
*IEEE Trans. on Inform. Theory* **22** (1976), 644-654.

# Diffie-Hellman [1976]

primo p, generatore g di $Z_p^*$

Alice

Bob

# Generatori di $Z_p^*$

g è generatore di $Z_p^*$ se $\{ g^i \mid 1 \leq i \leq p-1 \} = Z_p^*$

**Esempio:**

g = 2 è un generatore di $Z_{11}^*$

$2^{10} = 1024 = 1 \bmod 11$
$2^1 = 2 \bmod 11$
$2^8 = 256 = 3 \bmod 11$
$2^2 = 4 \bmod 11$
$2^4 = 16 = 5 \bmod 11$
$2^9 = 512 = 6 \bmod 11$
$2^7 = 128 = 7 \bmod 11$
$2^3 = 8 \bmod 11$
$2^6 = 64 = 9 \bmod 11$
$2^5 = 32 = 10 \bmod 11$

# Generatori di $Z_n^*$

- Ordine di $\alpha \in Z_n^*$ = il più piccolo intero positivo r tale che $\alpha^r = 1 \bmod n$
- $\alpha$ è generatore di $Z_n^*$ se ha ordine $\phi(n)$

**Teorema di Eulero**
$x \in Z_n^* \Rightarrow x^{\phi(n)} = 1 \bmod n$

# Generatori di $Z_n^*$

- Ordine di $\alpha \in Z_n^*$ = il più piccolo intero positivo r tale che $\alpha^r = 1 \bmod n$
- $\alpha$ è generatore di $Z_n^*$ se ha ordine $\phi(n)$

Teorema di Eulero
$x \in Z_n^* \Rightarrow x^{\phi(n)} = 1 \bmod n$

- $Z_n^*$ ha un generatore $\Leftrightarrow$ $n = 2, 4, p^k, 2p^k$, con p primo e $k \geq 1$
  - Se p è primo, allora $Z_p^*$ ha un generatore

# Potenze in $Z_{19}^*$

| a | $a^2$ | $a^3$ | $a^4$ | $a^5$ | $a^6$ | $a^7$ | $a^8$ | $a^9$ | $a^{10}$ | $a^{11}$ | $a^{12}$ | $a^{13}$ | $a^{14}$ | $a^{15}$ | $a^{16}$ | $a^{17}$ | $a^{18}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 16 | 13 | 7 | 14 | 9 | 18 | 17 | 15 | 11 | 3 | 6 | 12 | 5 | 10 | 1 |
| 3 | 9 | 8 | 5 | 15 | 7 | 2 | 6 | 18 | 16 | 10 | 11 | 14 | 4 | 12 | 17 | 13 | 1 |
| 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 | 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 |
| 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 | 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 |
| 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 | 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 |
| 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 |
| 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 |
| 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 | 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 |
| 10 | 5 | 12 | 6 | 3 | 11 | 15 | 17 | 18 | 9 | 14 | 7 | 13 | 16 | 8 | 4 | 2 | 1 |
| 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 |
| 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 |
| 13 | 17 | 12 | 4 | 14 | 11 | 10 | 16 | 18 | 6 | 2 | 7 | 15 | 5 | 8 | 9 | 3 | 1 |
| 14 | 6 | 8 | 17 | 10 | 7 | 3 | 4 | 18 | 5 | 13 | 11 | 2 | 9 | 12 | 16 | 15 | 1 |
| 15 | 16 | 12 | 9 | 2 | 11 | 13 | 5 | 18 | 4 | 3 | 7 | 10 | 17 | 8 | 6 | 14 | 1 |
| 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 | 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 |
| 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 | 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 |
| 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 |

# Generatori di $Z_n^*$

- Ordine di $\alpha \in Z_n^*$ = il più piccolo intero positivo r tale che $\alpha^r = 1 \bmod n$
- a è generatore di $Z_n^*$ se ha ordine $\phi(n)$

- Se $\alpha$ è un generatore di $Z_n^*$, allora
  - $Z_n^* = \{\alpha^i \bmod n \mid 0 \leq i \leq \phi(n)-1\}$
  - $b = \alpha^i \bmod n$ è un generatore di $Z_n^* \Leftrightarrow \gcd(i,\phi(n))=1$

Esempio:
2 è un generatore in $Z_{19}^*$
- $3=2^{13} \bmod 19$ è un generatore perchè gcd(13,18)=1
- $13=2^5 \bmod 19$ è un generatore perchè gcd(5,18)=1
- $5=2^{16} \bmod 19$ non è un generatore perchè gcd(16,18)=2

# Potenze in $Z_{19}^*$

| $a$ | $a^2$ | $a^3$ | $a^4$ | $a^5$ | $a^6$ | $a^7$ | $a^8$ | $a^9$ | $a^{10}$ | $a^{11}$ | $a^{12}$ | $a^{13}$ | $a^{14}$ | $a^{15}$ | $a^{16}$ | $a^{17}$ | $a^{18}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 16 | 13 | 7 | 14 | 9 | 18 | 17 | 15 | 11 | 3 | 6 | 12 | 5 | 10 | 1 |
| 3 | 9 | 8 | 5 | 15 | 7 | 2 | 6 | 18 | 16 | 10 | 11 | 14 | 4 | 12 | 17 | 13 | 1 |
| 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 | 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 |
| 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 | 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 |
| 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 | 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 |
| 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 |
| 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 |
| 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 | 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 |
| 10 | 5 | 12 | 6 | 3 | 11 | 15 | 17 | 18 | 9 | 14 | 7 | 13 | 16 | 8 | 4 | 2 | 1 |
| 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 |
| 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 |
| 13 | 17 | 12 | 4 | 14 | 11 | 10 | 16 | 18 | 6 | 2 | 7 | 15 | 5 | 8 | 9 | 3 | 1 |
| 14 | 6 | 8 | 17 | 10 | 7 | 3 | 4 | 18 | 5 | 13 | 11 | 2 | 9 | 12 | 16 | 15 | 1 |
| 15 | 16 | 12 | 9 | 2 | 11 | 13 | 5 | 18 | 4 | 3 | 7 | 10 | 17 | 8 | 6 | 14 | 1 |
| 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 | 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 |
| 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 | 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 |
| 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 |

# Generatori di $Z_n^*$

- Il numero di generatori di $Z_n^*$ è $\phi(\phi(n))$
  - Se p è primo, il numero di generatori di $Z_p^*$ è $\phi(p-1)$

# Diffie-Hellman [1976]

primo p, generatore g di $Z_p^*$

scelgo $x \in Z_p$

Alice

scelgo $y \in Z_p$

Bob

# Diffie-Hellman [1976]

primo p, generatore g

scelgo $x \in Z_p$

scelgo $y \in Z_p$

$g^x \bmod p$

Alice

Bob

# Diffie-Hellman [1976]

primo p, generatore g

scelgo $x \in Z_p$

scelgo $y \in Z_p$

Alice

$g^x \bmod p$

$g^y \bmod p$

Bob

# Diffie-Hellman [1976]

primo p, generatore g

scelgo $x \in Z_p$

scelgo $y \in Z_p$

Alice

$g^x \bmod p$

$g^y \bmod p$

Bob


$K = g^{xy} \bmod p$
$= (g^y)^x \bmod p$

$K = g^{xy} \bmod p$
$= (g^x)^y \bmod p$

# Diffie-Hellman: "piccolo" esempio

primo 11, generatore 2

scelgo x=3

scelgo y=4

$8 = 2^3 \bmod 11$

$5 = 2^4 \bmod 11$

Alice

Bob


$K=4=(2^4)^3 \bmod 11$

$K=4=(2^3)^4 \bmod 11$

# Diffie-Hellman: esempio

primo 25307, generatore 2

scelgo x=3578

scelgo y=19956

$6113 = 2^{3578} \bmod 25307$

$7984 = 2^{19956} \bmod 25307$

Alice

Bob

$K=3694=7984^{3578}$


$K=3694=6113^{19956}$

# Logaritmo discreto

La sicurezza di molte tecniche crittografiche si basa sulla intrattabilità del logaritmo discreto, ad es.:

- Crittosistema ElGamal
- Accordo su chiavi Diffie-Hellman
- Firme digitali DSS

Dati a,n,b calcolare x tale che $\mathbf{a^x = b \mod n}$

Esempio: $3^x = 7 \mod 13$ soluzione $x = 6$

# Logaritmo discreto:
## Complessità algoritmi

Dati a,n,b calcolare x tale che $a^x = b \mod n$

Vari algoritmi:

- Trial multiplication
- Baby-step giant-step
- Pollard's rho algorithm
- Pohlig-Hellman algorithm
- Index Calculus

# Logaritmo discreto:
## Complessità algoritmi

Dati a,n,b calcolare x tale che $a^x = b \mod n$

Se n è primo, i migliori algoritmi hanno complessità

$$L_n[a,c] = O(e^{(c+o(1))(\ln n)^a(\ln\ln n)^{1-a}})$$

con $c > 0$ ed $0 < a < 1$

**Miglior algoritmo:** Number field sieve

tempo medio euristico $L_n[1/3, 1.923]$

$\left(\frac{64}{9}\right)^{1/3} = 1,922999$

# Logaritmo discreto: record computazioni

- Primo, 130-digit, 431-bit: giugno 2005, A. Joux e R. Lecier
  - 3 settimane, 1.15 GHz 16-processor HP AlphaServer GS1280
- Primo, 160-digit, 530-bit: febbraio 2007, T. Kleinjung
  - Vari pc ed un cluster
- Primo, 160-digit, 530-bit: febbraio 2014, C. Bouvier, P. Gaudry, L. Imbert, H. Jeljeli, E. Thomé
- Primo, 232-digit, 768-bit: giugno 2016, T. Kleinjung, C. Diem, A. K. Lenstra, C. Priplata, C. Stahlke
  - Computazione iniziata a febbraio 2015, ha richiesto l'equivalente di circa 6600 core years di un Intel Xeon E5-2660 at 2.2 GHz
- Primo, 240 digit, 795 bit: dicembre 2019, F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann
  - Calcolato anche la fattorizzazione di RSA-240
  - Per entrambi, tempo totale equivalente al lavoro di 4.000 anni di un singolo computer con processore 2.1GHz Intel Xeon Gold 6130 CPUs as a reference
  - Number Field Sieve algorithm, usando open-source CADO-NFS software

# Scelta della sfida di 768-bit

- Scelta del primo
  - il primo numero *primo safe* > $2^{766}\pi$
  - numero primo safe se è nella forma 2q+1 con q primo
  - p = $\lfloor 2^{766}\pi \rfloor$ + 62762
- Generatore g=11
- Target t=$\lfloor 2^{766}e \rfloor$
- Calcolato $\log_g(t)$ =
  325923617918270562238615985978623709128341338833721058543950813521768156295091638348030637920237175638117352442299234041658748471079911977497864301995972638266781162575370644813703762423329783129621567127479417280687495231463348812

# Scelta della sfida di 795-bit

- Scelta del primo
  - il primo numero *primo safe* dopo RSA-240
  - numero primo safe se è nella forma 2q+1 con q primo
  - p = RSA-240 + 49204
- Scelta del generatore $g$=5
- Scelta del valore target
  - Frase "The magic words are still Squeamish Ossifrage»
  - Trasformata in decimale
  - Target=774356626343973985966622216006087686926705588649958206166317147722421706101723470351970238538755049093424997
- Calcolato $\log_g$(Target) = 926031359281441953630949553317328555029610991914376116167294204758987445623653667881005480990720934875482587528029233264473672441500961216292648092075981950622133668898591866811269289825060051277283214267512441114123717673755472250458517 16

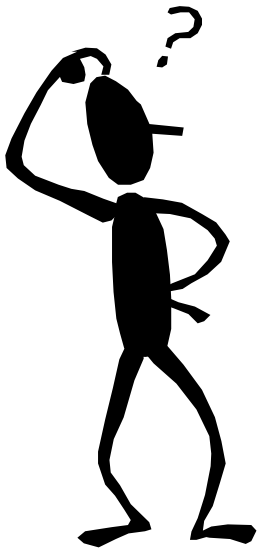
# CADO-NFS

- Crible Algébrique: Distribution, Optimisation - Number Field Sieve
- Implementazione in C/C++ dell'algoritmo Number Field Sieve (NFS) per fattorizzare interi e calcolare logaritmi discreti in campi finiti
- http://cado-nfs.gforge.inria.fr/

# Logaritmo discreto: record computazioni

- $GF(2^{613})$, settembre 2005, A. Joux e R. Lercier
  - 17 giorni, 4 nodi 1.3GHz 16-processori di un Itanium 2-based Bull computer Teranova
- $GF(2^{809})$, aprile 2013, R. Barbulescu, C. Bouvier, …
- $GF(2^{24 \cdot 257})$, 6168-bit, maggio 2013, A. Joux
- $GF(2^{18 \cdot 513})$, 9234-bit, gennaio 2014, R. Granger, T. Kleinjung, J. Zumbrägel
- $GF(3^{6 \cdot 71})$, 676-bit, 2010, T. Hayashi
- $GF(3^{6 \cdot 137})$, 1303-bit, gennaio 2014, F. Rodríguez-Henríquez
- $GF(3^{6 \cdot 163})$, 1551-bit, gennaio 2014, G. Adj, A. Menezes, T. Oliveira, F. Rodríguez-Henríquez

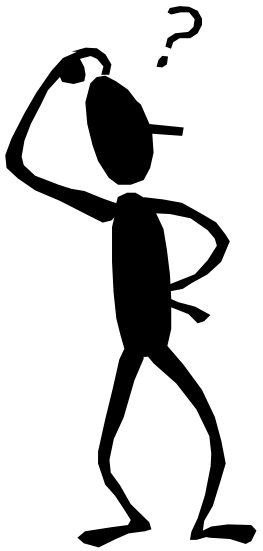
# Problema di Diffie-Hellman




**Input:** primo p, generatore g,
$g^x \bmod p$, $g^y \bmod p$
**Calcolare:** $g^{xy} \bmod p$

Il miglior algoritmo conosciuto calcola prima il logaritmo discreto $x \leftarrow \log_{g,p}(g^x \bmod p)$

# Problema di Diffie-Hellman




**Input:** primo p, generatore g,
$g^x \bmod p$, $g^y \bmod p$
**Calcolare:** $g^{xy} \bmod p$

Il miglior algoritmo conosciuto calcola prima il logaritmo discreto $x \leftarrow \log_{g,p}(g^x \bmod p)$

… ma non si sa se sono equivalenti!

# Che parametri scegliere?

Bisogna tener conto dell' expected security life

**Table 4: Recommended algorithms and minimum key sizes**

| Algorithm security lifetimes | Symmetric key algorithms (Encryption & MAC) | FFC (e.g., DSA, D-H) | IFC (e.g., RSA) | ECC (e.g., ECDSA) |
|---|---|---|---|---|
| Through 2010 (min. of 80 bits of strength) | 2TDEA[23]<br>3TDEA<br>AES-128<br>AES-192<br>AES-256 | Min.:<br>$L = 1024$;<br>$N = 160$ | Min.:<br>$k$=1024 | Min.:<br>$f$=160 |
| Through 2030 (min. of 112 bits of strength) | 3TDEA<br>AES-128<br>AES-192<br>AES-256 | Min.:<br>$L = 2048$<br>$N = 224$ | Min.:<br>$k$=2048 | Min.:<br>$f$=224 |
| Beyond 2030 (min. of 128 bits of strength) | AES-128<br>AES-192<br>AES-256 | Min.:<br>$L = 3072$<br>$N = 256$ | Min.:<br>$k$=3072 | Min.:<br>$f$=256 |

NIST SP 800-57,
"Recommendation for Key Management, Part 1: General (Revised)", Marzo 2007

# Che parametri scegliere?

| Security Strength | Symmetric key algorithms | FFC (e.g., DSA, D-H) | IFC (e.g., RSA) | ECC (e.g., ECDSA) |
|---|---|---|---|---|
| ≤ 80 | 2TDEA | $L = 1024$<br>$N = 160$ | $k = 1024$ | $f = 160\text{-}223$ |
| 112 | 3TDEA | $L = 2048$<br>$N = 224$ | $k = 2048$ | $f = 224\text{-}255$ |
| 128 | AES-128 | $L = 3072$<br>$N = 256$ | $k = 3072$ | $f = 256\text{-}383$ |
| 192 | AES-192 | $L = 7680$<br>$N = 384$ | $k = 7680$ | $f = 384\text{-}511$ |
| 256 | AES-256 | $L = 15360$<br>$N = 512$ | $k = 15360$ | $f = 512+$ |

Non approvati per la protezione di informazioni del Governo Federale USA

Inclusi negli standard NIST

Non ancora inclusi negli standard NIST per motivi di interoperabilità e di efficienza

NIST SP 800-57 Part 1 Revision 4 (pp. 52-53)
"Recommendation for Key Management Part 1: General", Gennaio 2016
http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4

# Scelta dei parametri

- Come scegliere p e g?

# Scelta di un generatore


Scegli_Generatore_Naive (p)

1. Scegli a caso $g$ in $Z_p^*$
2. **If** $\{g^i | 1 \leq i \leq p-1\} = Z_p^*$

   **then** trovato

   **else** goto 1.

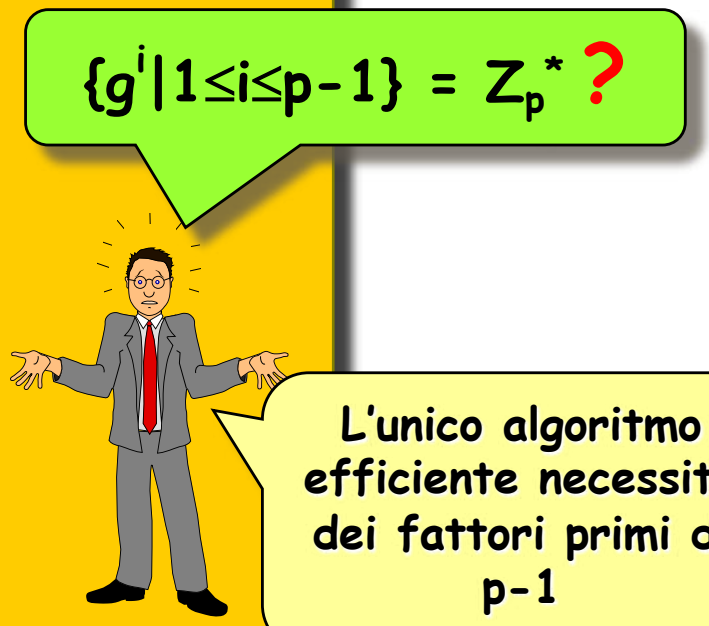
# Scelta di un generatore


## Scegli_Generatore_Naive (p)

1. Scegli a caso $g$ in $Z_p^*$
2. **If** $\{g^i | 1 \le i \le p-1\} = Z_p^*$

   **then** trovato

   **else** goto 1.



# Scelta di un generatore

p primo, $p-1 = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

$g$ è un generatore di $Z_p^* \Leftrightarrow$

$$\begin{cases} g^{(p-1)/p_1} \neq 1 \mod p \\ \dots \\ g^{(p-1)/p_k} \neq 1 \mod p \end{cases}$$

# Scelta di un generatore

p primo, $p-1 = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

g è un generatore di $Z_p^* \Leftrightarrow$

$$\begin{cases} g^{(p-1)/p_1} \neq 1 \mod p \\ \dots \\ g^{(p-1)/p_k} \neq 1 \mod p \end{cases}$$

**Esempio**

- 11 primo, $p-1 = 10 = 2 \cdot 5$
- 2 è un generatore di $Z_{11}^*$ perché

$$2^{(11-1)/2} = 2^5 = 10 \neq 1 \mod 11$$
$$2^{(11-1)/5} = 2^2 = 4 \neq 1 \mod 11$$

# Scelta di un generatore

p primo, $p-1 = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

g è un generatore di $Z_p^* \Leftrightarrow$

$$\begin{cases} g^{(p-1)/p_1} \neq 1 \mod p \\ \dots \\ g^{(p-1)/p_k} \neq 1 \mod p \end{cases}$$

**Esempio**

- 11 primo, $p-1 = 10 = 2 \cdot 5$
- 3 **non** è un generatore di $Z_{11}^*$ perché

$$3^{(11-1)/2} = 3^5 = 243 = 1 \mod 11$$
$$3^{(11-1)/5} = 3^2 = 9 \neq 1 \mod 11$$

# Scelta di un generatore

p primo, $p-1 = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

g è un generatore di $Z_p^* \Leftrightarrow$

$$\begin{cases} g^{(p-1)/p_1} \neq 1 \mod p \\ \dots \\ g^{(p-1)/p_k} \neq 1 \mod p \end{cases}$$

**Scegli_generatore** ( p, $(p_1,e_1,p_2,e_2,\dots,p_k,e_k)$ )

1. g ← elemento scelto a caso in $Z_p^*$
2. **if** ($g^{(p-1)/p_1} \neq 1 \mod p$ **and** … **and** $g^{(p-1)/p_k} \neq 1 \mod p$)
   **then** esci
   **else** go to 1.

# Probabilità successo singola iterazione

- Numero di generatori modulo un primo p è

$$\phi(\phi(p)) = \phi(p-1)$$

$$> (p-1) / (6 \cdot \ln\ln(p-1))$$

per ogni intero $n \geq 5$,
$\phi(n) > n/(6 \ln\ln n)$

# Probabilità successo singola iterazione

- Numero di generatori modulo un primo p è

$$\phi(\phi(p)) = \phi(p-1)$$

$$> (p-1) / (6 \cdot \ln\ln(p-1))$$

per ogni intero $n \geq 5$, $\phi(n) > n/(6 \ln\ln n)$

- Probabilità che un elemento a caso in $Z_p^*$ sia generatore

$$= \frac{\phi(\phi(p))}{\phi(p)} > \frac{p-1}{\phi(p) \cdot 6\ln\ln(p-1)} = \frac{1}{6 \cdot \ln\ln(p-1)}$$

# Analisi di Scegli_generatore

**Numero medio di iterazioni** $< 6 \cdot \ln\ln(p-1)$

| | |
|---|---|
| 512 bit | $6 \cdot \ln\ln(2^{512}) \approx 35{,}23$ |
| 1024 bit | $6 \cdot \ln\ln(2^{1024}) \approx 39{,}38$ |
| 2048 bit | $6 \cdot \ln\ln(2^{2048}) \approx 43{,}54$ |

# Generazione chiavi Diffie-Hellman

1. Scegli a caso 2 numeri primi $p_1$ $p_2$
2. $p \leftarrow 1 + 2p_1p_2$
3. Se p non è primo, go to 1.
4. $g \leftarrow$ **Scegli_generatore**$(p,(2,1,p_1,1,p_2,1))$

# Accordo su chiave
# Diffie-Hellman: sicurezza

- E' sicuro contro attaccanti passivi

- Non è sicuro contro attacchi *man-in-the-middle*

# Diffie-Hellman

## attacco man-in-the-middle

primo p, generatore g di $Z_p^*$

scelgo $x \in Z_p$

scelgo $z \in Z_p$

scelgo $y \in Z_p$

$g^x \bmod p$

Alice

$g^z \bmod p$

$g^z \bmod p$

$g^y \bmod p$

Bob

$K_A = g^{xz} \bmod p$
$= (g^z)^x \bmod p$

$K_A = g^{xz} \bmod p$
$= (g^x)^z \bmod p$
$K_B = g^{zy} \bmod p$
$= (g^y)^z \bmod p$

$K_B = g^{zy} \bmod p$
$= (g^z)^y \bmod p$

# Accordo su chiave
# Diffie-Hellman: sicurezza

- E' sicuro contro attaccanti passivi

- Non è sicuro contro attacchi *man-in-the-middle*
- Occorre autenticazione

# Crittografia a chiave pubbica con Diffie-Hellman

Possiamo usare Diffie-Hellman come cifrario asimmetrico?

# Cifrari asimmetrici


file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| ... | ... |

# Cifratura

chiave privata
kpriv

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| ... | ... |

canale insicuro

Bob

Devo cifrare il messaggio
M ed inviarlo ad Alice

# Cifratura

Alice

chiave privata
kpriv

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| ... | ... |

C

canale insicuro

Bob

Cifratura di M per Alice
C ← CIFRA(kpub, M)

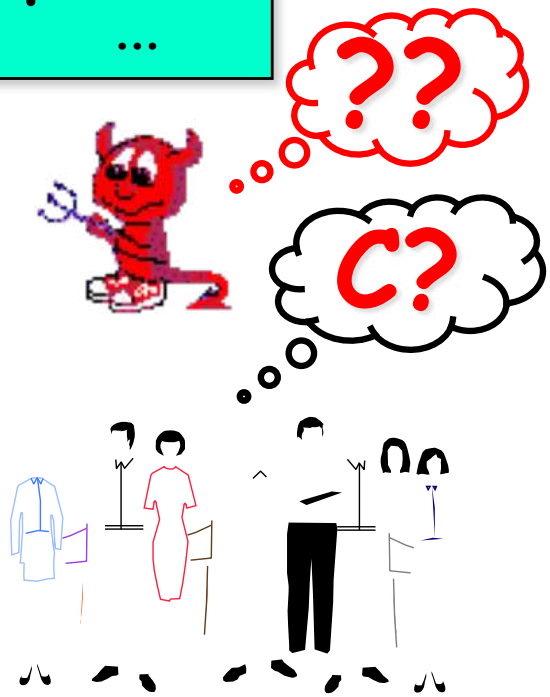
# Decifratura

Devo decifrare il messaggio cifrato *c*

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| ... | ... |



# Decifratura

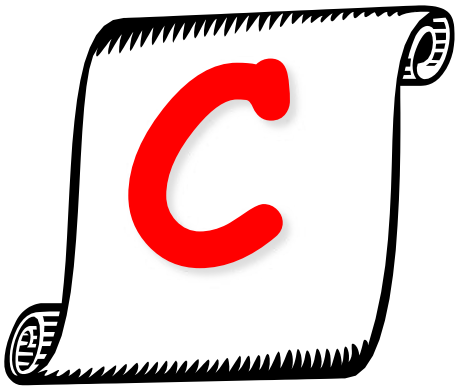
chiave privata
kpriv

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| ... | ... |

Decifratura di C
M ← DECIFRA (kpriv, C)

Alice



# Chiavi DH

chiave privata
(p,g,a)

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | $(p,g,g^a \bmod p)$ |
| ... | ... |

# Cifratura DH

chiave privata
(p,g,a)

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | $(p,g,g^a \bmod p)$ |
| ... | ... |

canale insicuro

Bob

Devo cifrare il messaggio
M ed inviarlo ad Alice

# Cifratura DH

chiave privata
(p,g,a)

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | $(p,g,g^a \bmod p)$ |
| ... | ... |

canale insicuro

$g^b \bmod p$, C

Bob

Cifratura di M per Alice

$g^b \bmod p$

$k \leftarrow (g^a \bmod p)^b \bmod p$

$C \leftarrow ENC(k,M)$

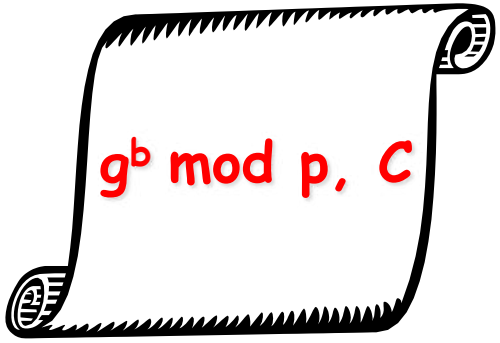
# Decifratura DH

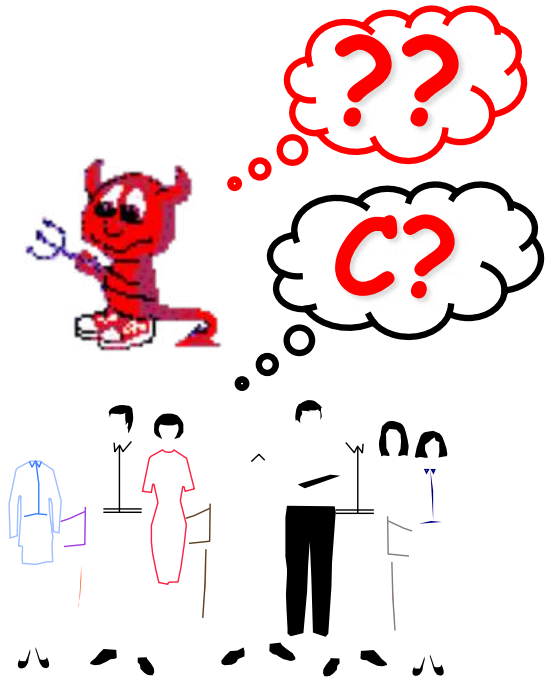
Devo decifrare il messaggio cifrato $(g^b \bmod p, C)$

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | $(p, g, g^a \bmod p)$ |
| ... | ... |



# Cifratura DH

chiave privata
(p,g,a)

Alice

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | $(p,g,g^a \bmod p)$ |
| ... | ... |

Decifratura di $g^b \bmod p$, $C$

$k \leftarrow (g^b \bmod p)^a \bmod p$

$M \leftarrow DEC(k,C)$

# Crittografia a chiave pubbica RSA e Diffie-Hellman

- RSA è il cifrario asimmetrico più diffuso
- Motivazioni:
  - Storiche
  - Commerciali

    Verisign fondata come spin-off di RSA Security nel 1995

    
    
  - Algoritmi di firma digitale

# Accordo su chiavi: indice

- Protocollo Diffie-Hellman
- Problema del logaritmo discreto
- Generazione parametri
- Uso come cifrario a chiave pubblica
- Forward Secrecy

# Forward Secrecy

- Proprietà di protocolli per accordo su chiavi
- Compromissione della chiave privata
  - ma chiavi di sessione non compromesse
  - cioè, non permette la decifrazione dei messaggi precedenti

# Forward Secrecy

- Proprietà di protocolli per accordo su chiavi
- Compromissione della chiave privata
  - ma chiavi di sessione non compromesse
  - cioè, non permette la decifrazione dei messaggi precedenti

- Realizzazione

# Forward Secrecy

- Proprietà di protocolli per accordo su chiavi
- Compromissione della chiave privata

  - ma chiavi di sessione non compromesse
  - cioè, non permette la decifrazione dei messaggi precedenti
- Realizzazione
  - Diffie-Hellman autenticato in cui la chiave privata cambia sempre

# Ephemeral Diffie-Hellman (DHE)

Alice

Hello

primo p, generatore g, $g^y$ mod p
$firma_{Bob}$(p, g, $g^y$ mod p)

scelgo $x \in Z_p$

$g^x$ mod p

scelgo $y \in Z_p$

Bob


$K = g^{xy} \bmod p = (g^y)^x \bmod p$

$K = g^{xy} \bmod p = (g^x)^y \bmod p$

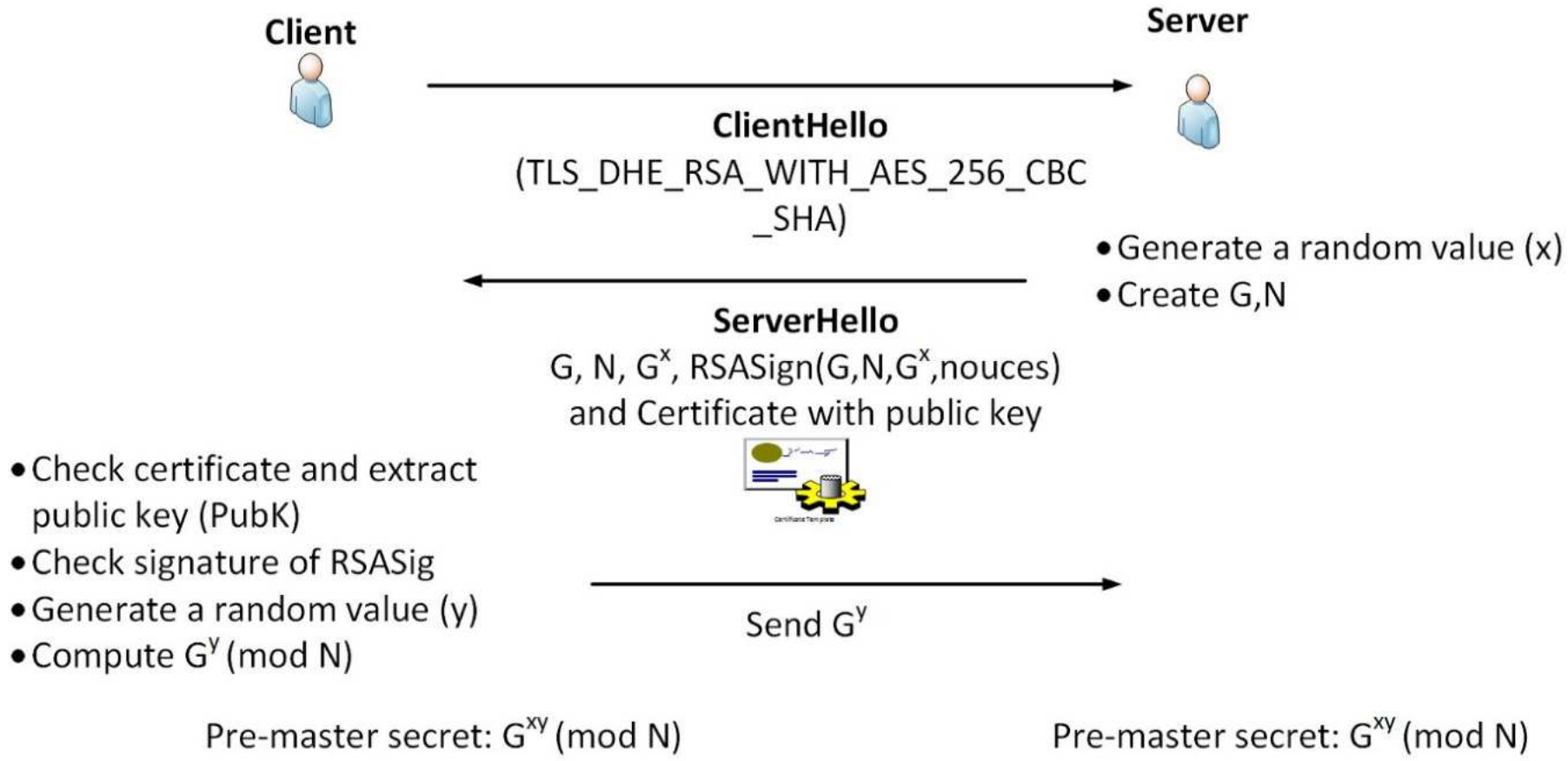
# Ephemeral Diffie-Hellman (DHE)

- Il valore y è scelto a caso ad ogni richiesta
- E viene cancellato dopo la generazione della chiave condivisa

# Ephemeral Diffie-Hellman (DHE)

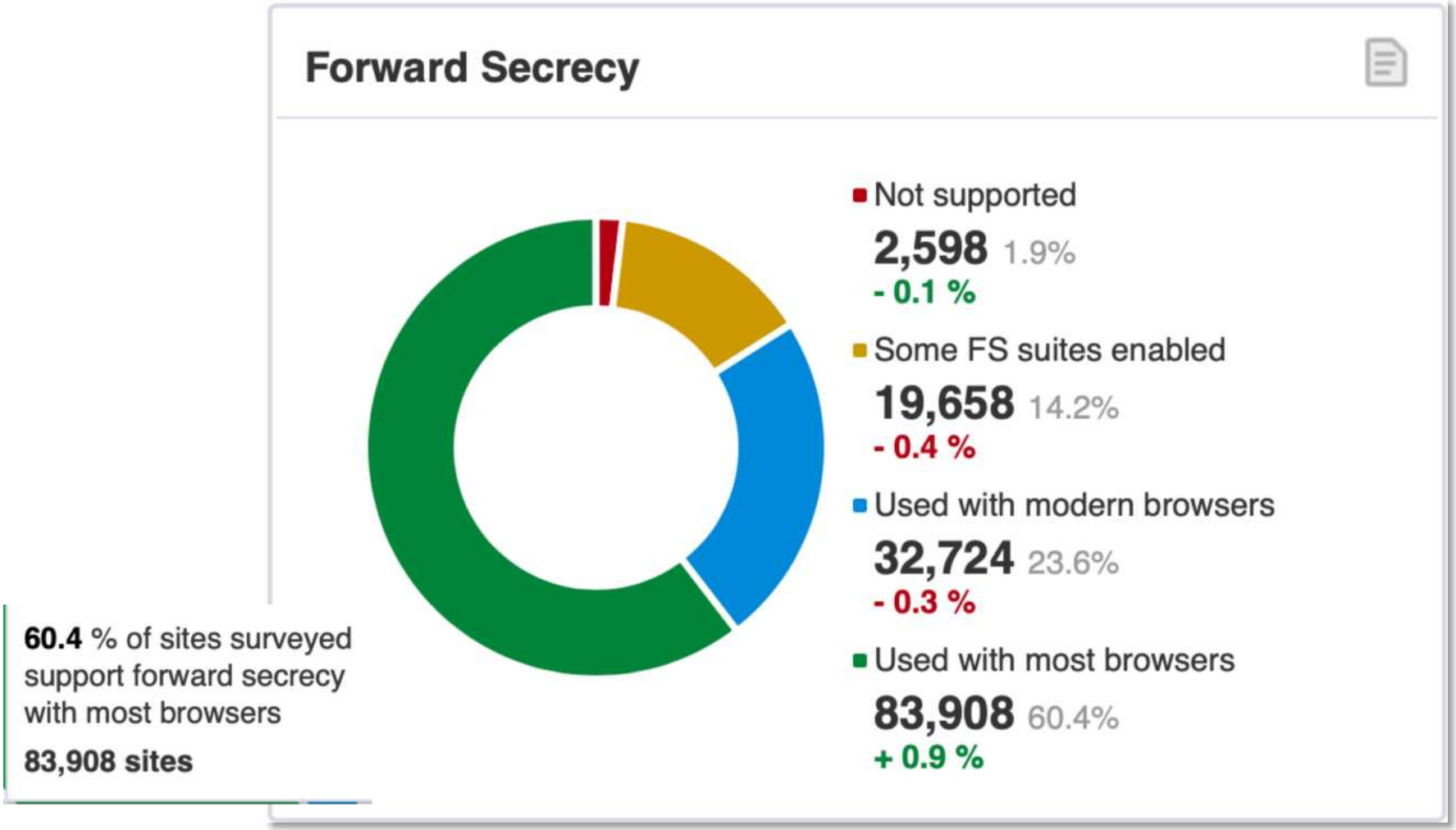
- Il valore y è scelto a caso ad ogni richiesta
- E viene cancellato dopo la generazione della chiave condivisa
- Vediamo una implementazione in TLS 1.3

# Ephemeral Diffie-Hellman with RSA
# DHE-RSA



# Survey uso TLS

Monthly Scan: April 03, 2020




https://www.ssllabs.com/ssl-pulse/

# Forward Secrecy in Signal

## Signal Protocol 

- cifratura end-to-end per *instant messaging*
- usa il Double Ratchet Algorithm
- Primitive: Elliptic-curve Diffie-Hellman con Curve25519, AES-256 e HMAC-SHA256

# Forward Secrecy in Signal

Signal Protocol 

- cifratura end-to-end per *instant messaging*
- usa il Double Ratchet Algorithm
- Primitive: Elliptic-curve Diffie-Hellman con Curve25519, AES-256 e HMAC-SHA256

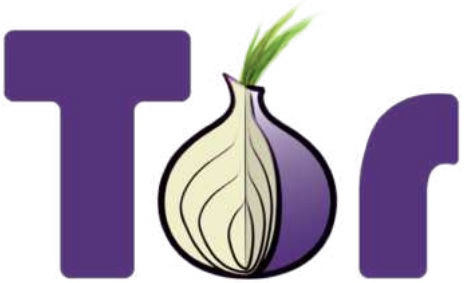
- Uso del Double Ratchet Algorithm
  - o implementazione personalizzata
    - Facebook Messanger
    - Skype
    - Whatsapp
    - …

# Forward Secrecy in TOR

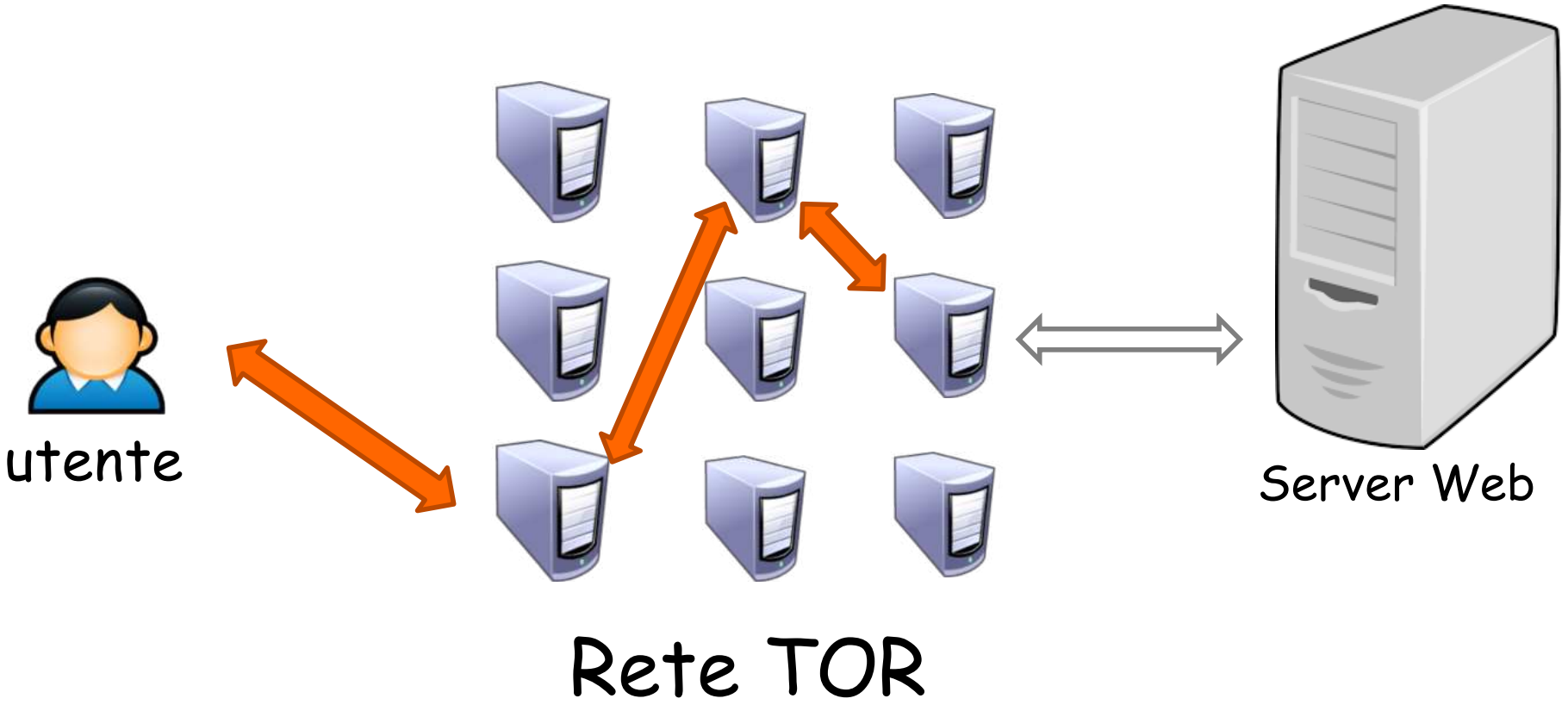
Onion Routing

- U.S. Naval Research Laboratory, 1998
- Layer di cifrature
  - Una cifratura per ogni nodo del cammino

The Onion Router (TOR)

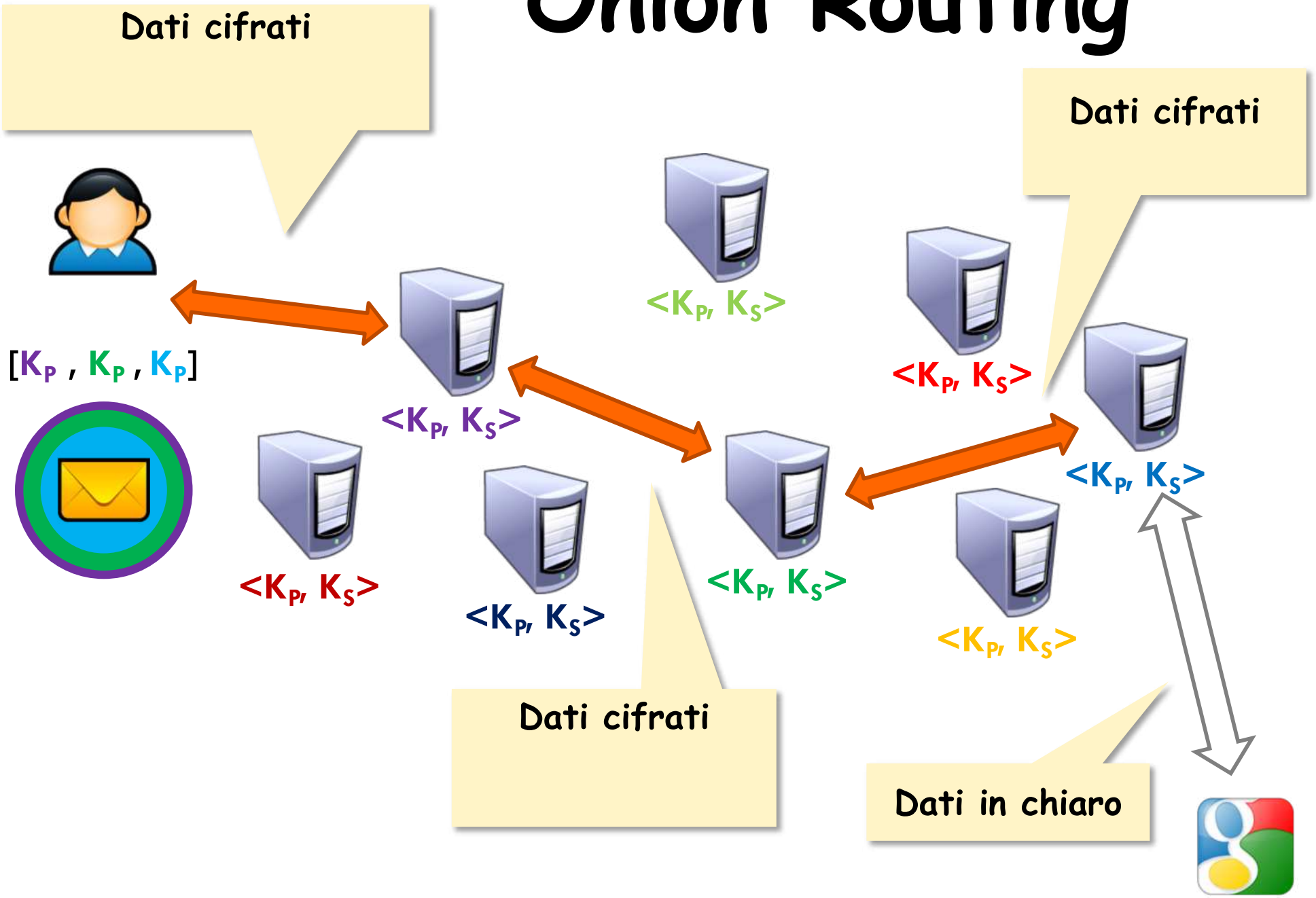


# The Onion Router (TOR)

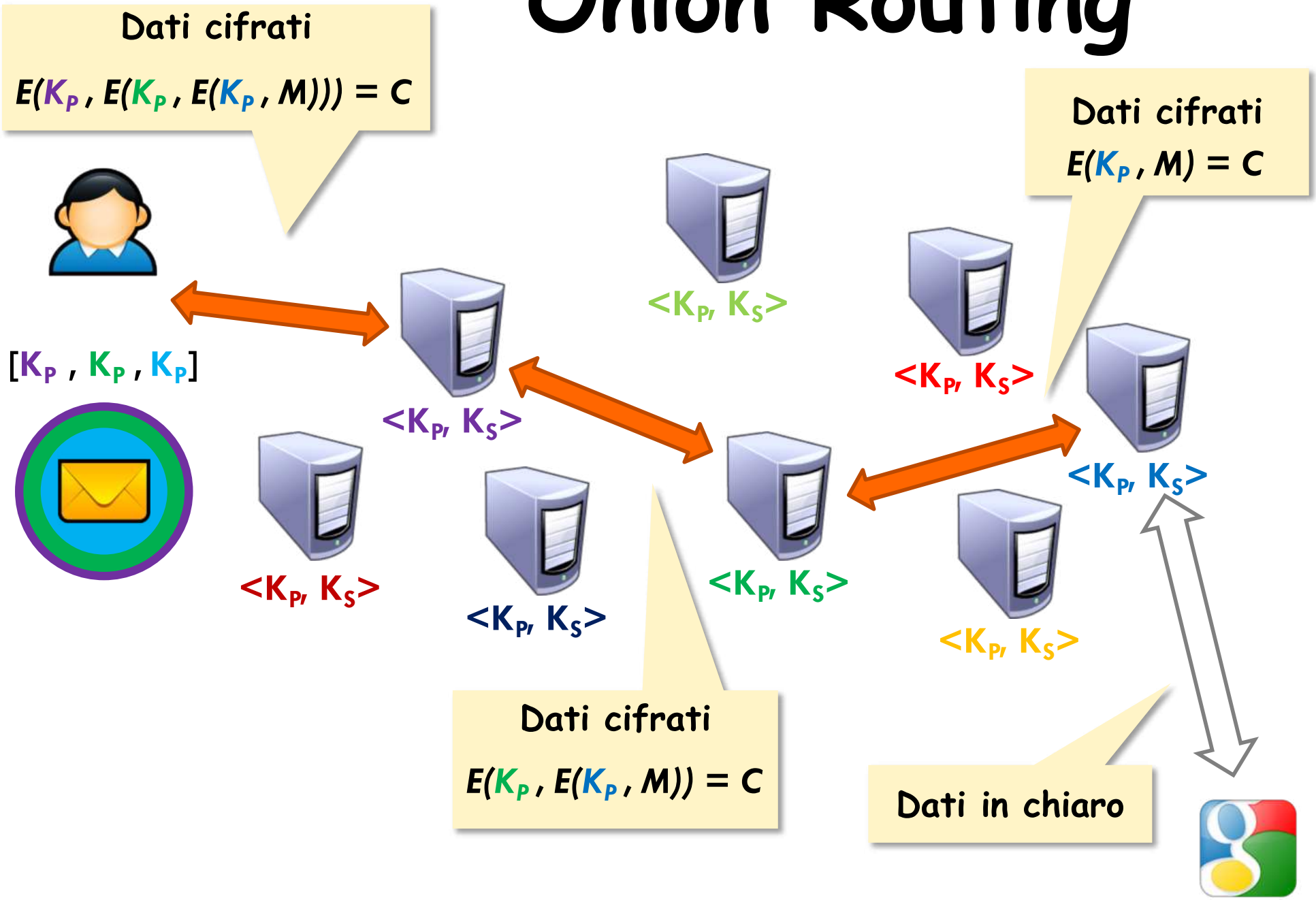


# The Onion Router (TOR)

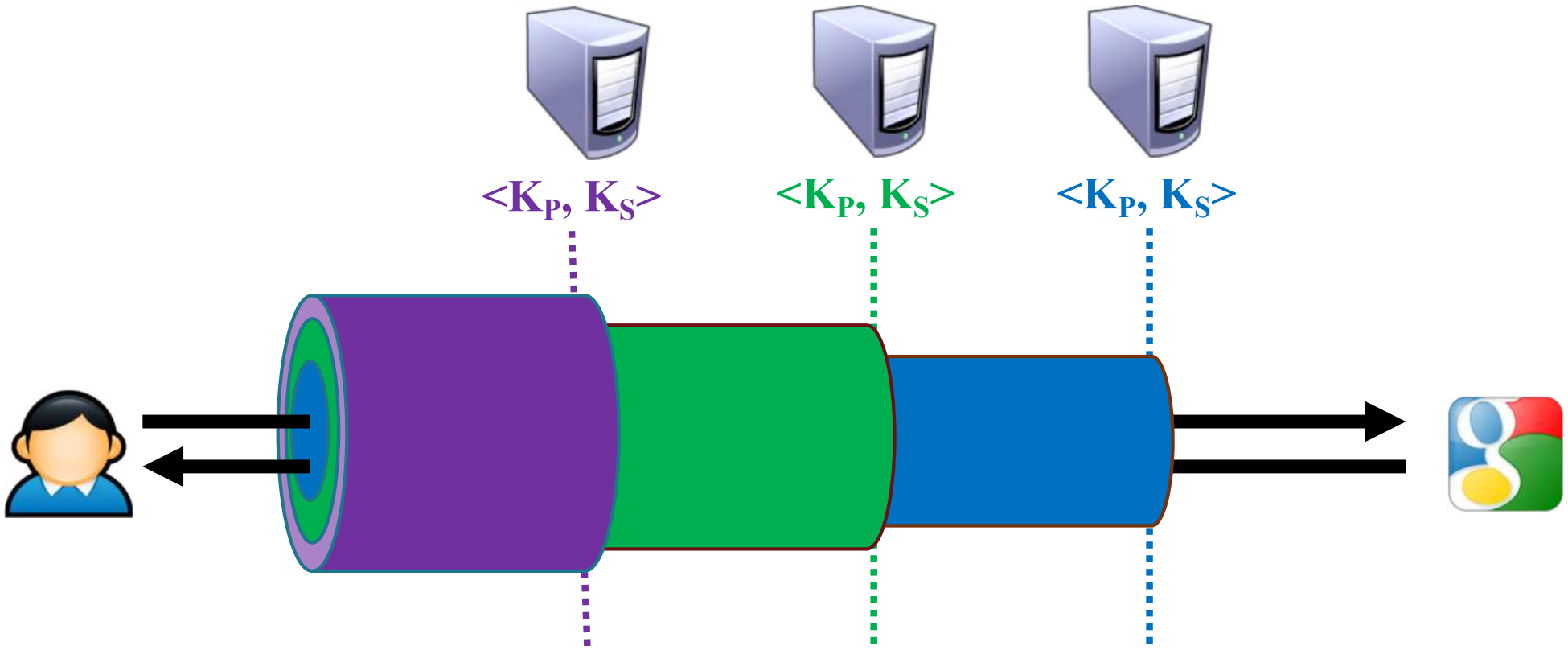
# Onion Routing



# Onion Routing



# Onion Routing

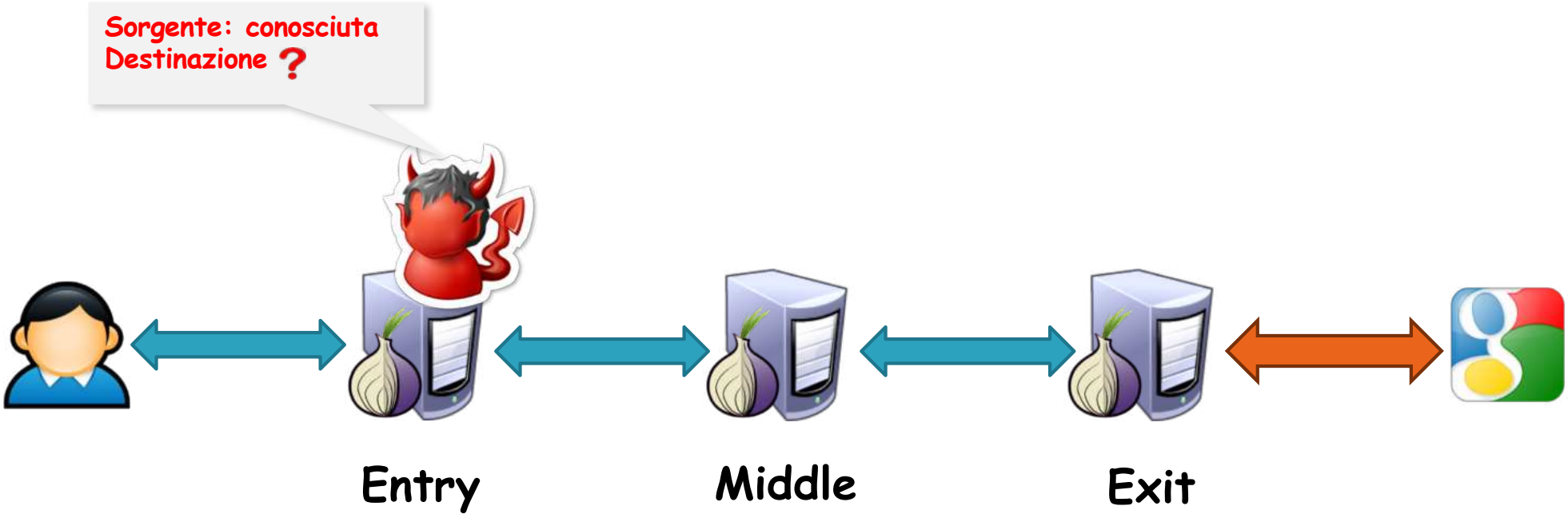



$E(K_P, E(K_P, E(K_P, M))) = C$

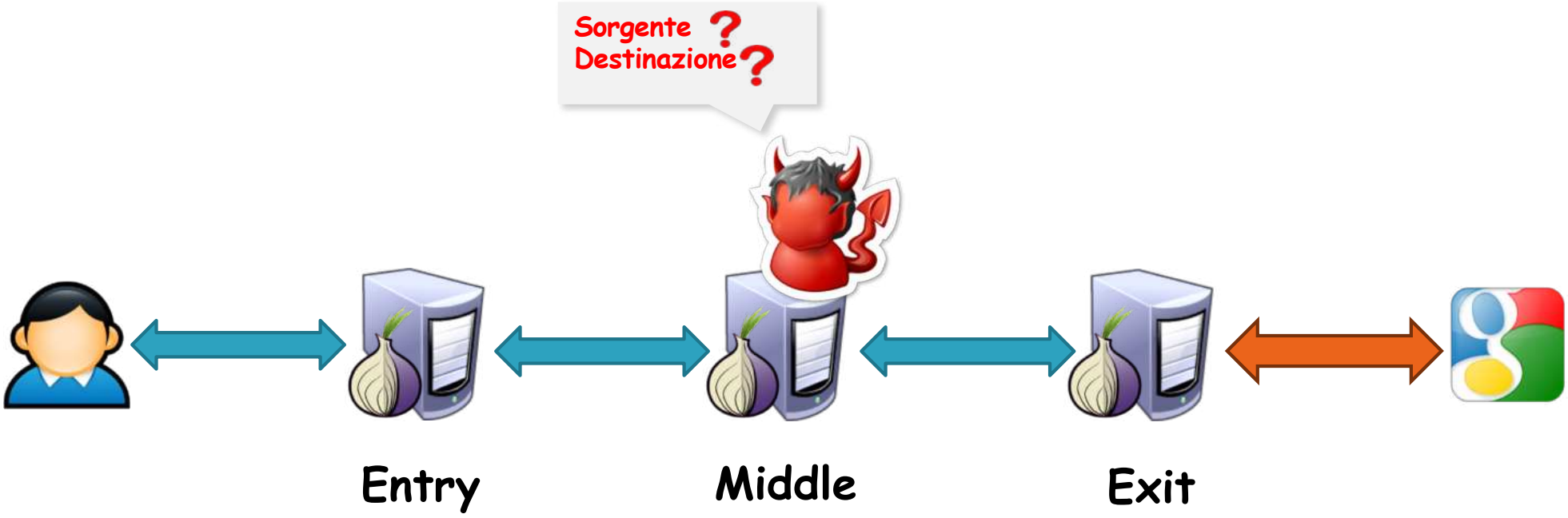
$E(K_P, E(K_P, M)) = C$

$E(K_P, M) = C$

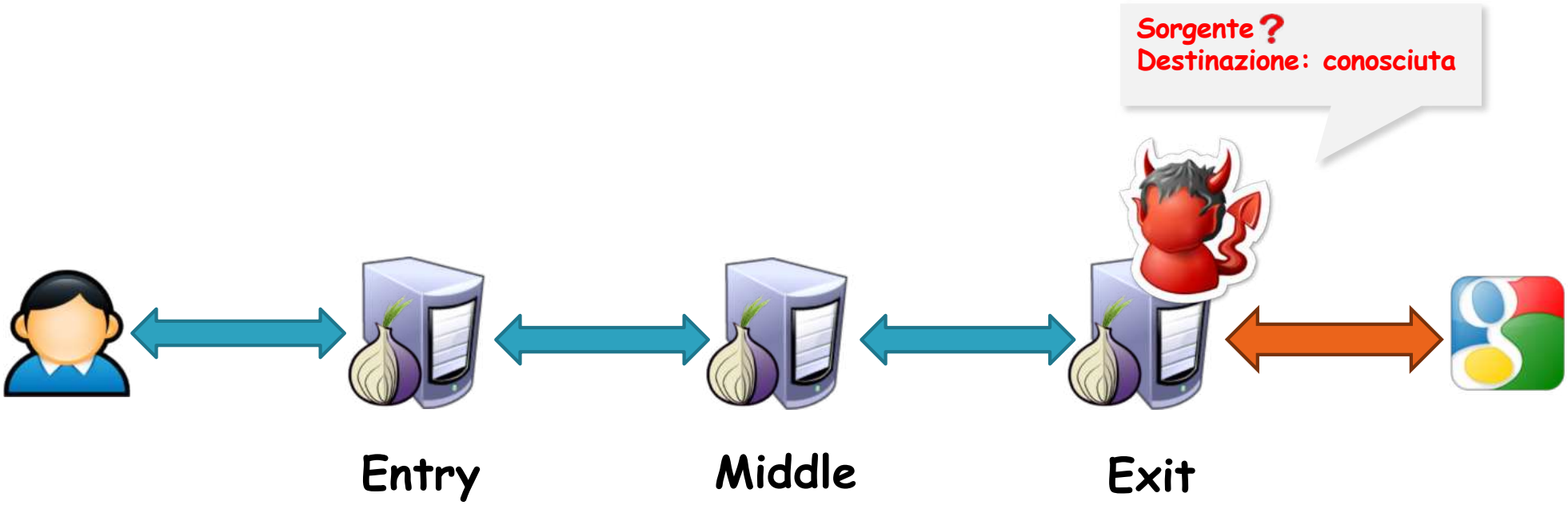
# Relay



# Relay



# Relay



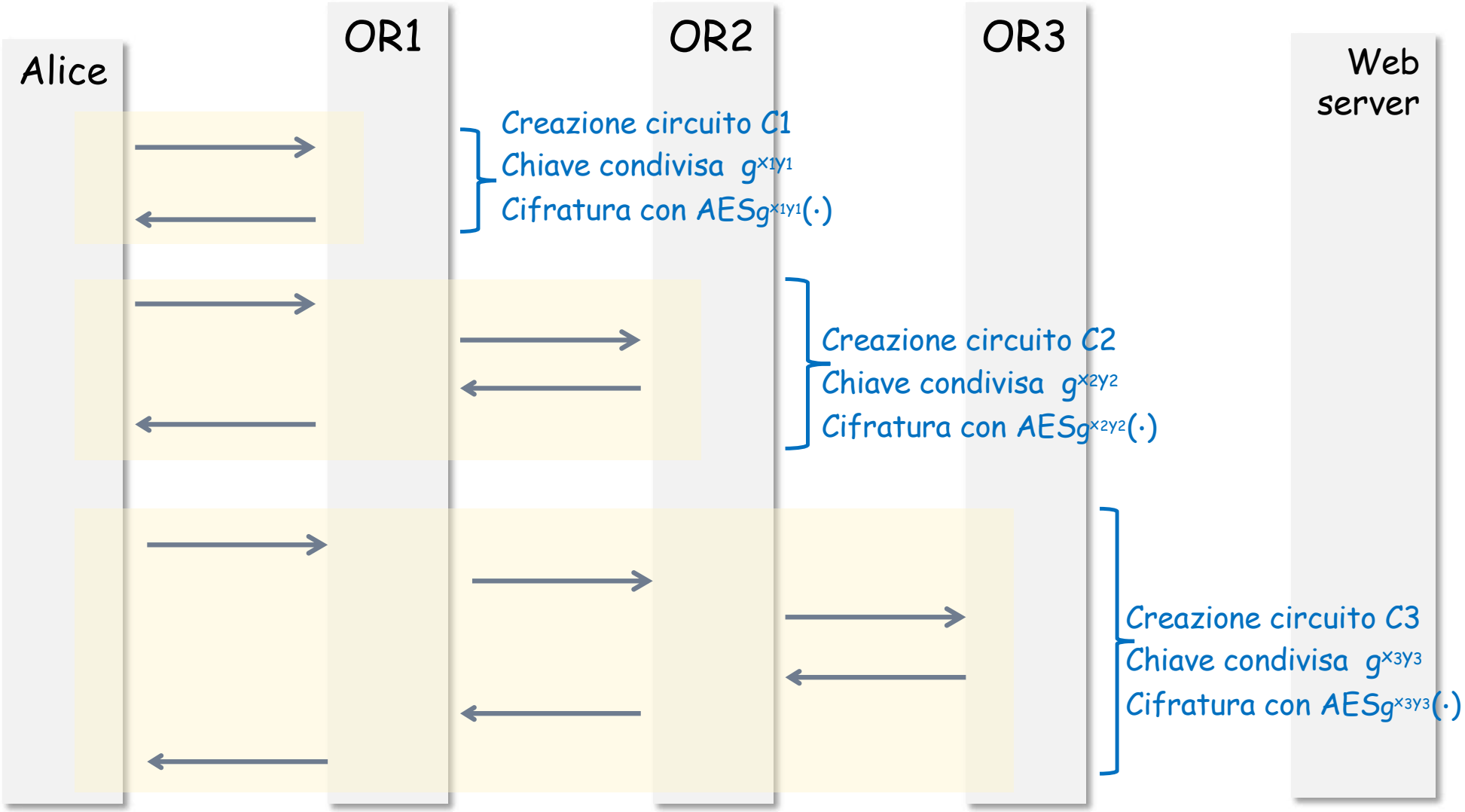
# Compromissione nodi TOR

"No adversary is truly global, but no adversary needs to be truly global. Eavesdropping on the entire Internet is a several-billion-dollar problem. Running a few computers to eavesdrop on a lot of traffic, a selective denial of service attack to drive traffic to your computers, that's like a tens-of-thousands-of-dollars problem." At the most basic level, an attacker who runs two poisoned Tor nodes—one entry, one exit—is able to analyse traffic and thereby identify the tiny, unlucky percentage of users whose circuit happened to cross both of those nodes. At present the Tor network offers, out of a total of around 7,000 relays, around 2,000 guard (entry) nodes and around 1,000 exit nodes. So the odds of such an event happening are one in two million (1/2000 x 1/1000), give or take."

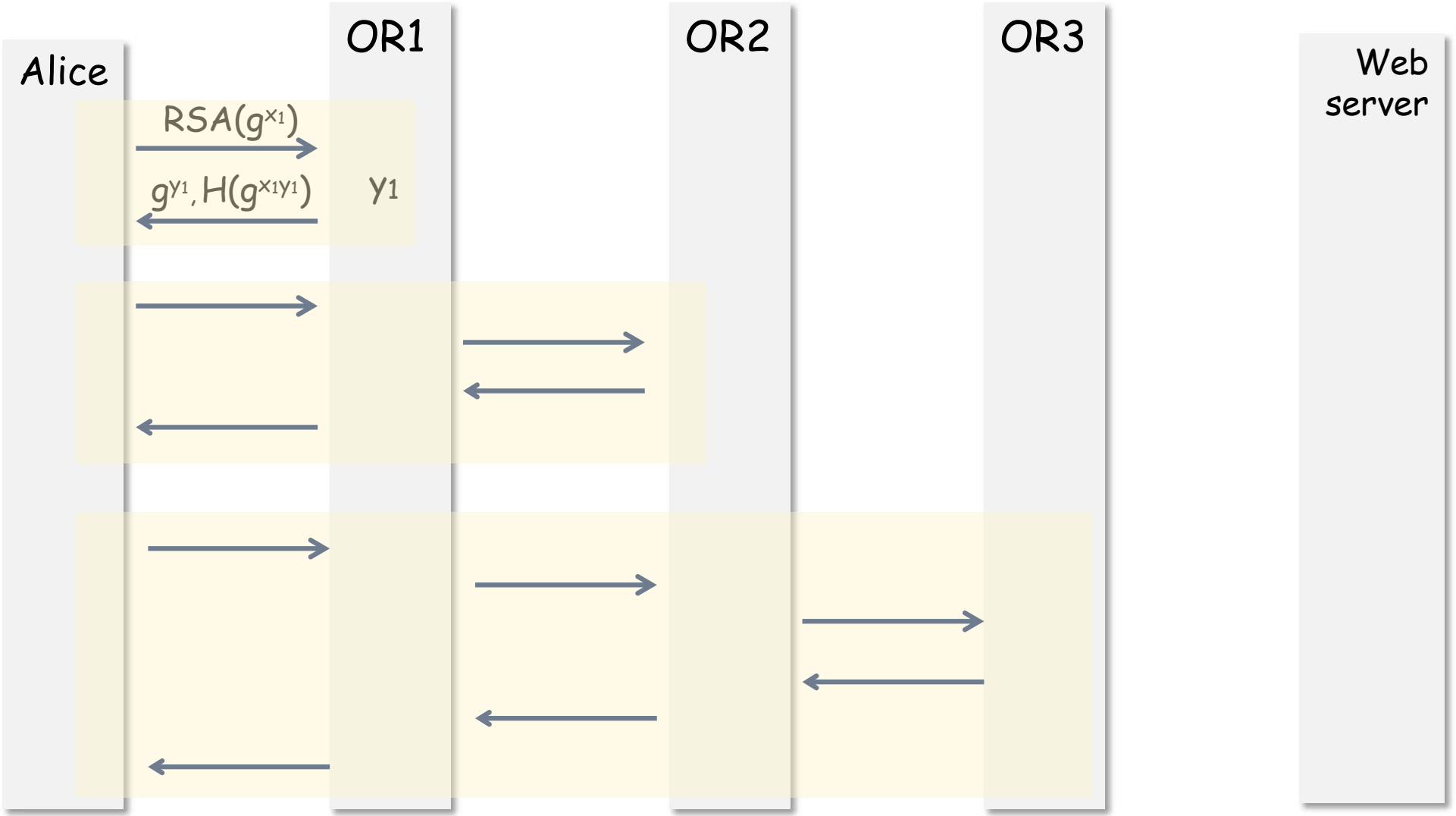
Nick Mathewson, cofondatore progetto TOR, 31 agosto 2016

https://arstechnica.com/information-technology/2016/08/building-a-new-tor-that-withstands-next-generation-state-surveillance/

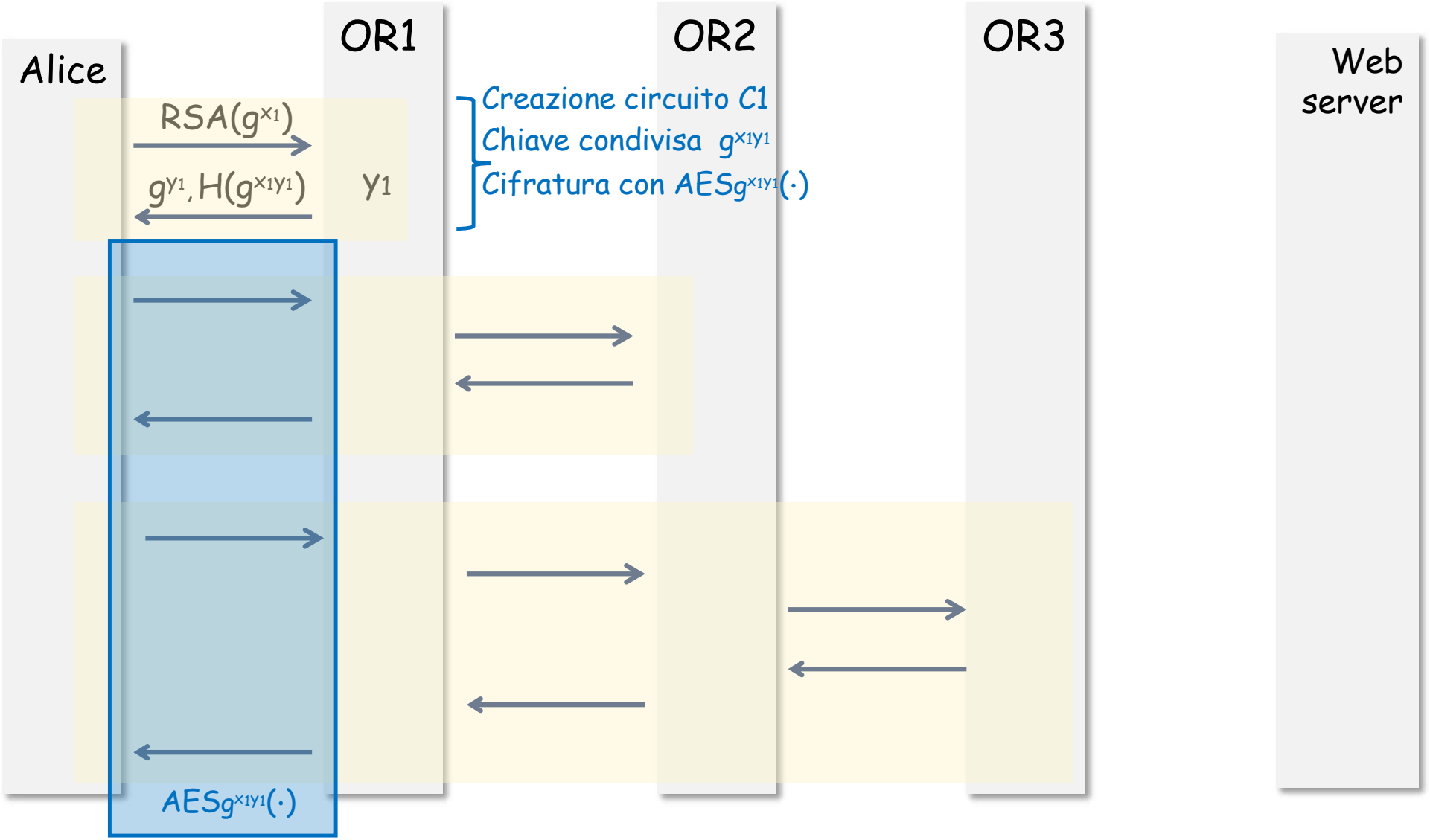
# Forward Secrecy in TOR



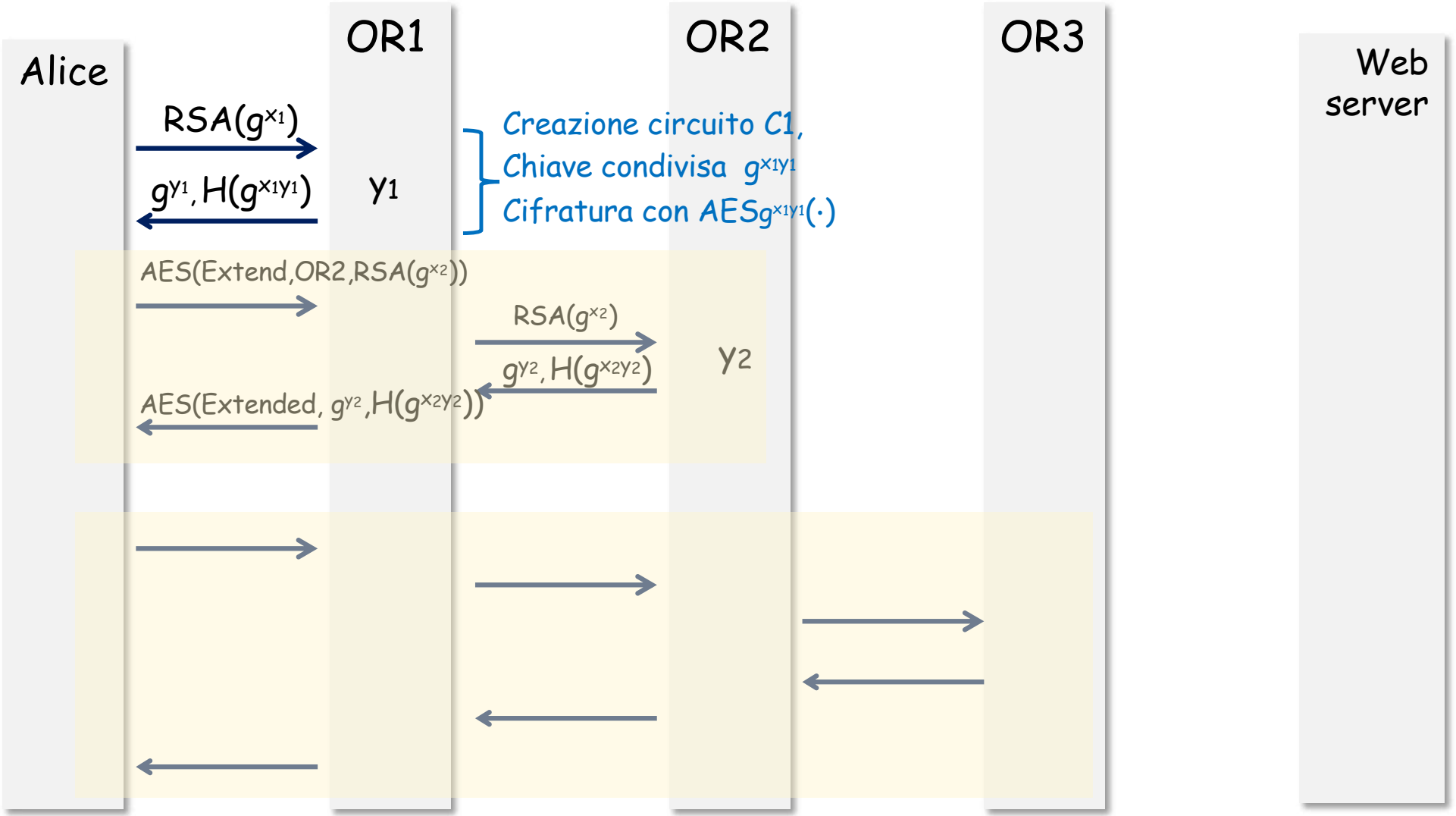
# Forward Secrecy in TOR



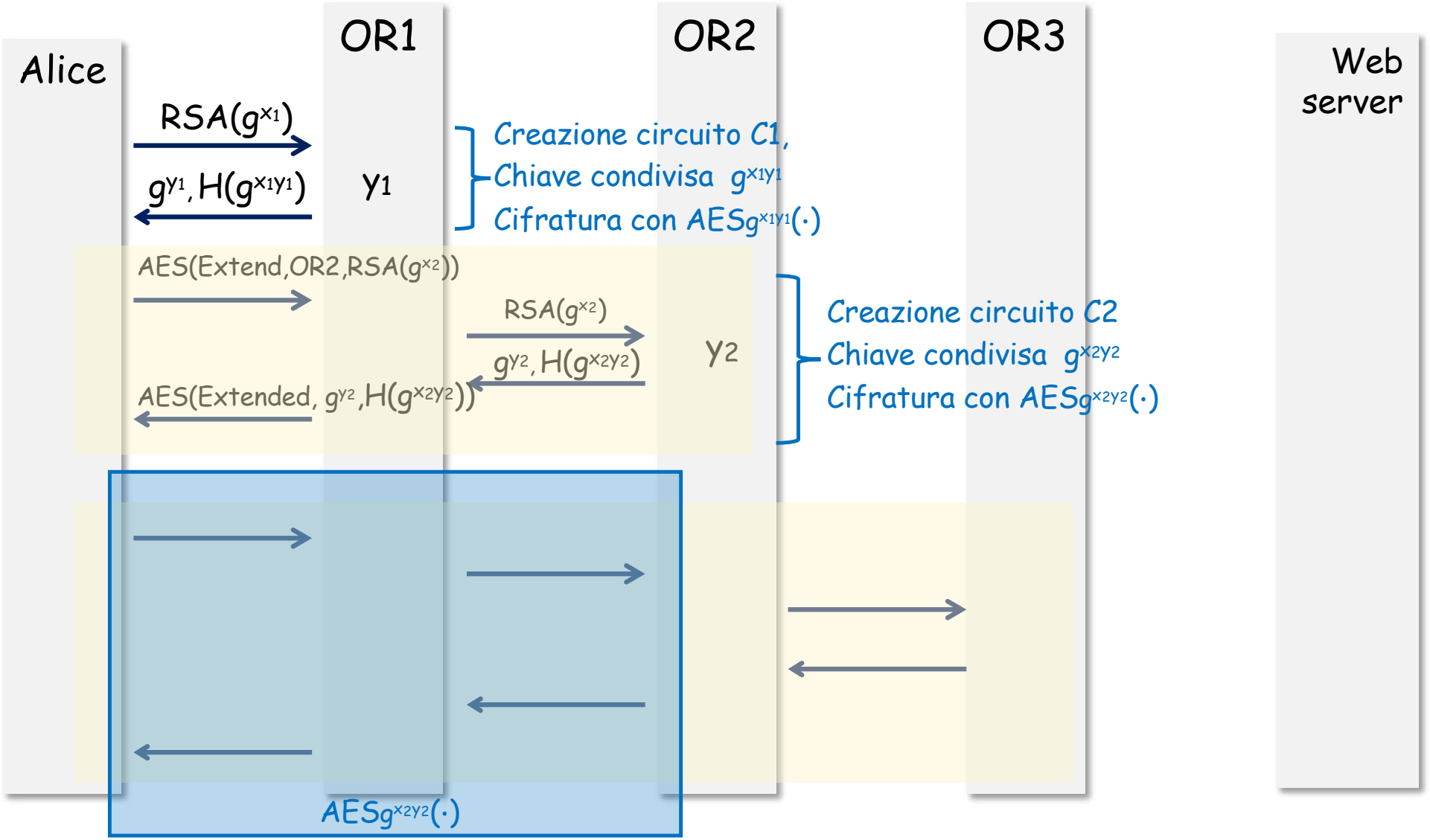
# Forward Secrecy in TOR



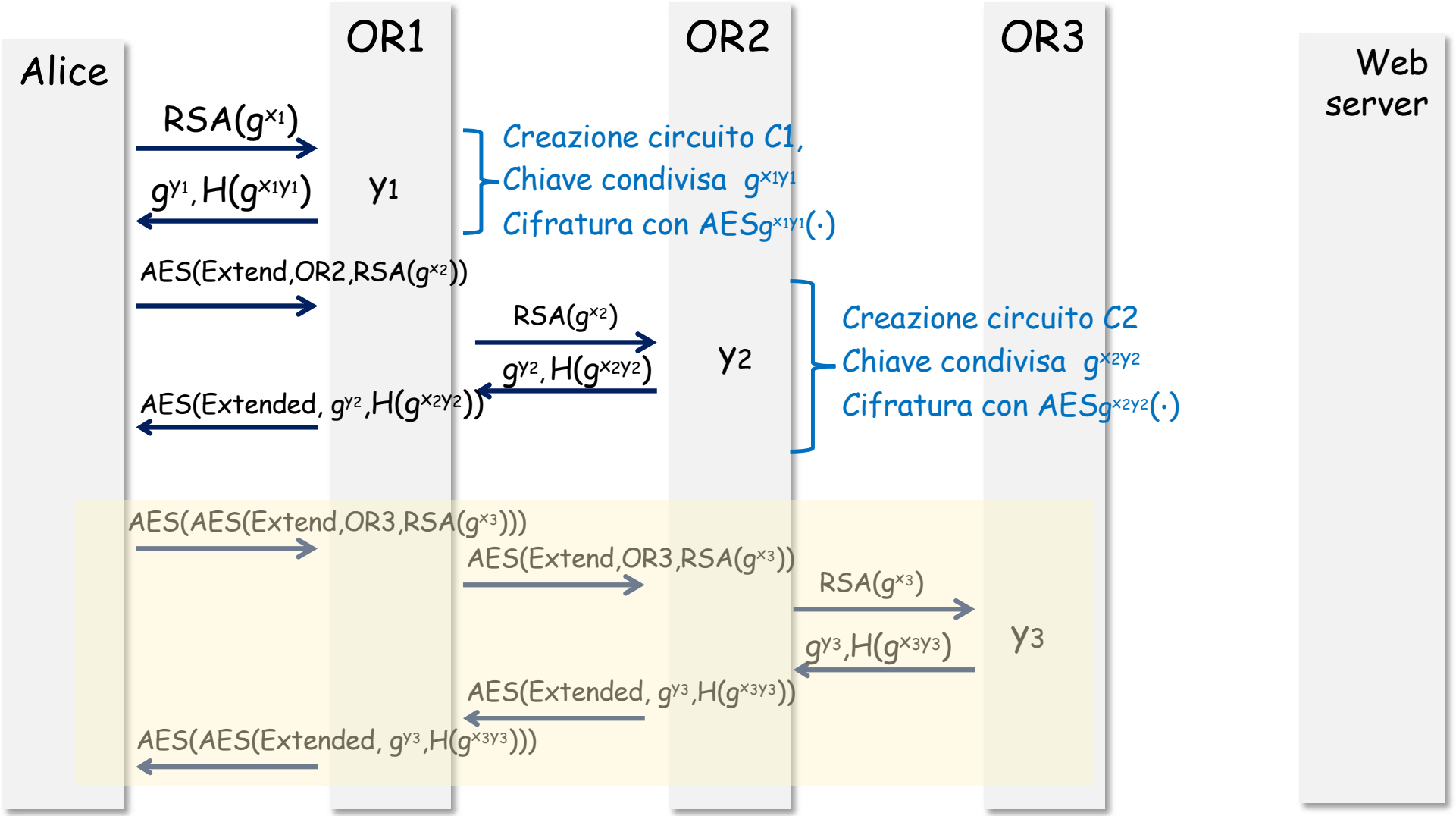
# Forward Secrecy in TOR



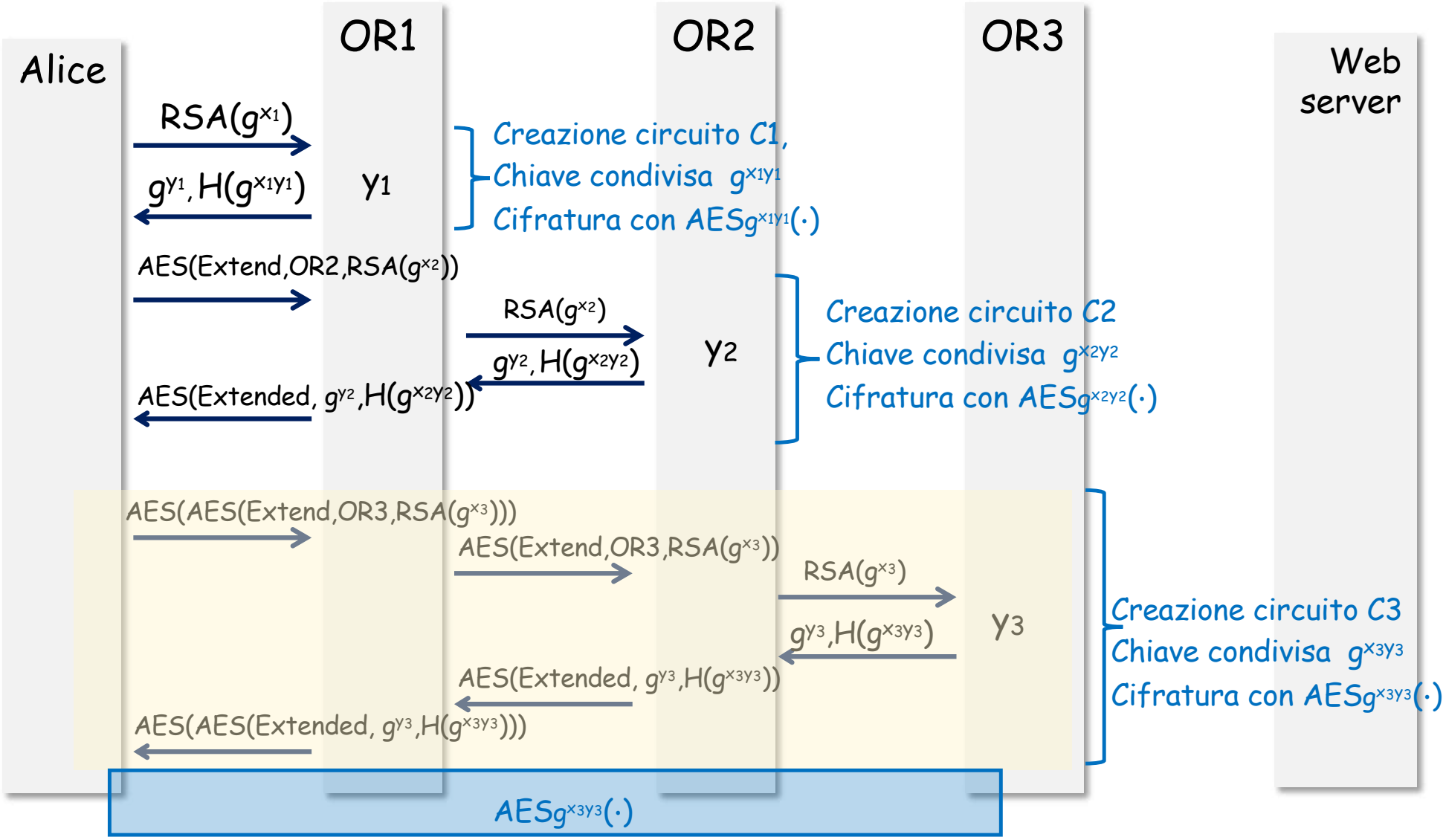
# Forward Secrecy in TOR



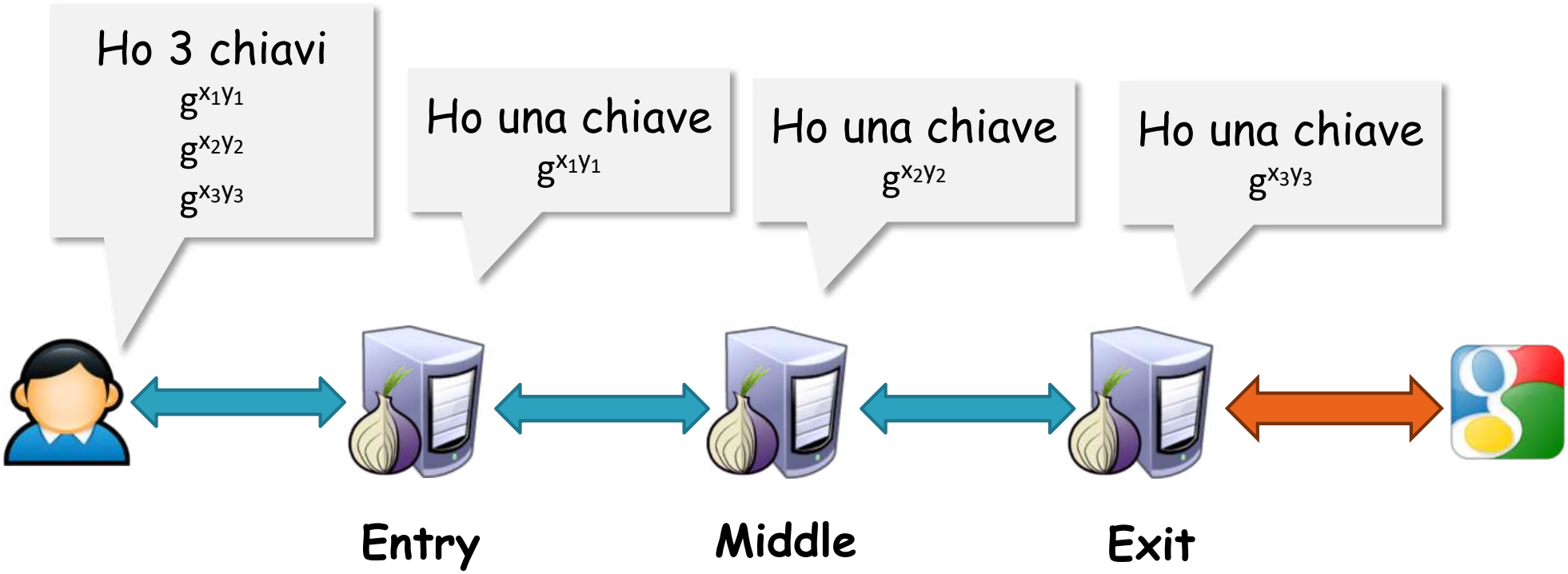
# Forward Secrecy in TOR



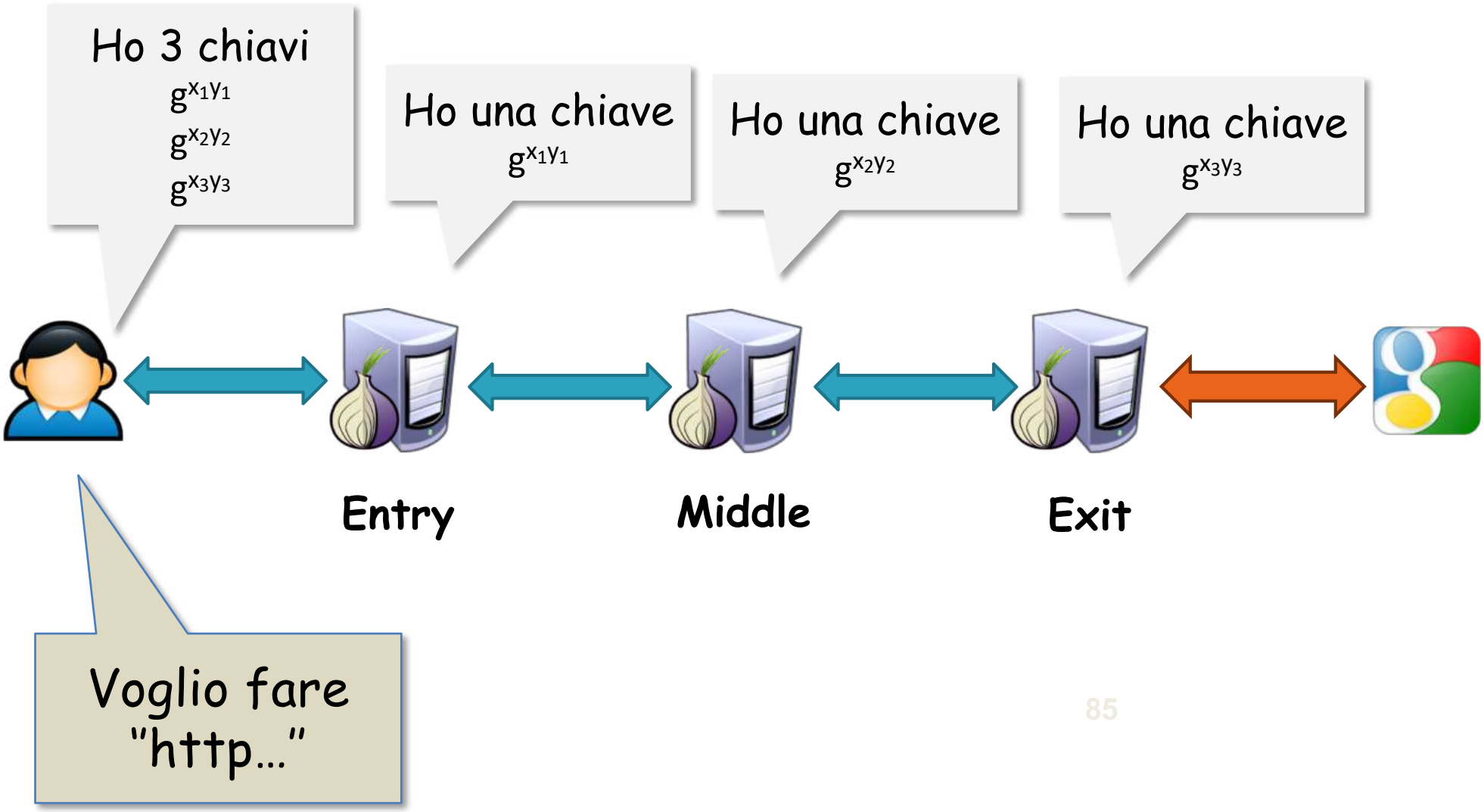
# Forward Secrecy in TOR



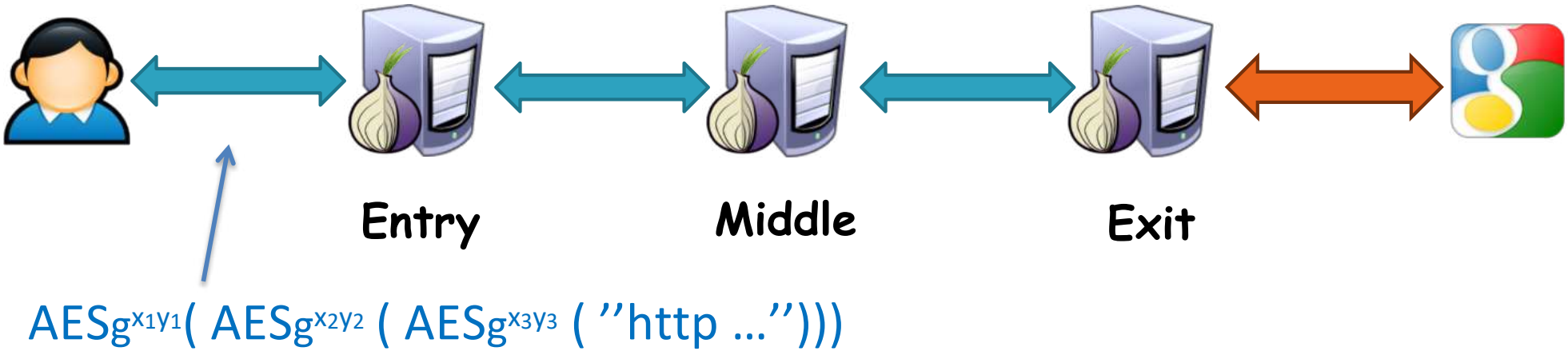
# Chiavi dopo creazione circuiti



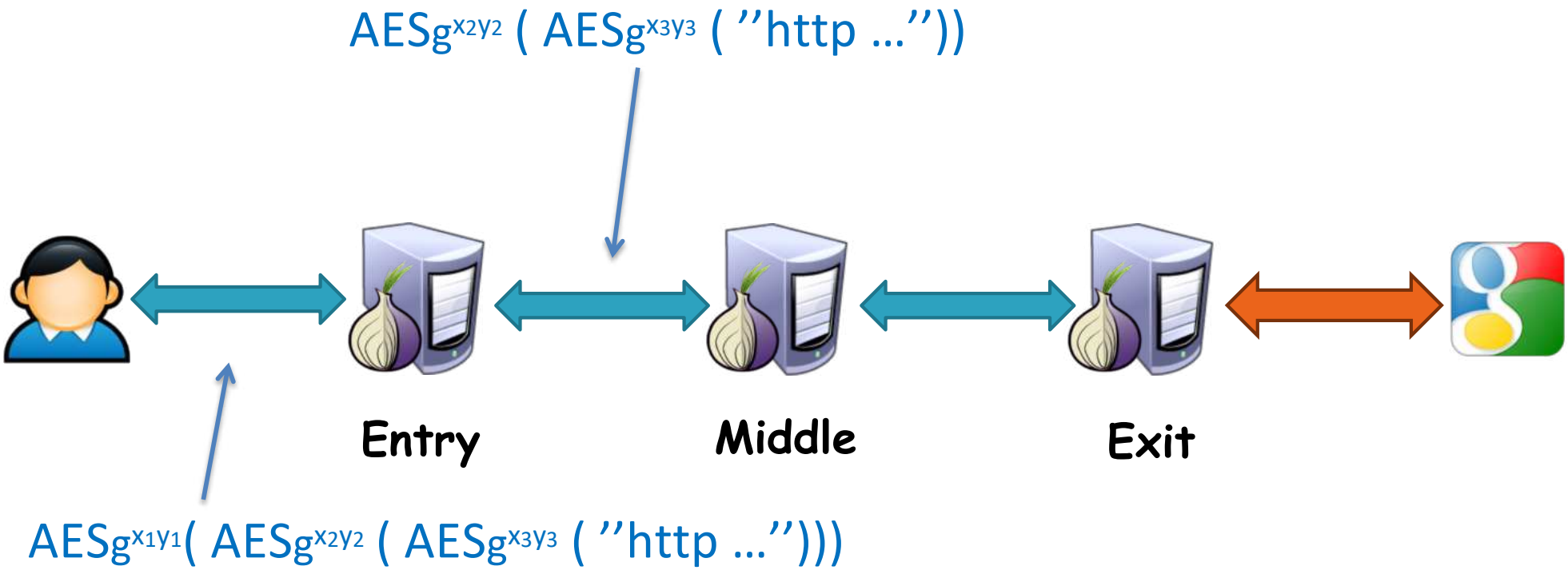
# Dopo creazione circuiti



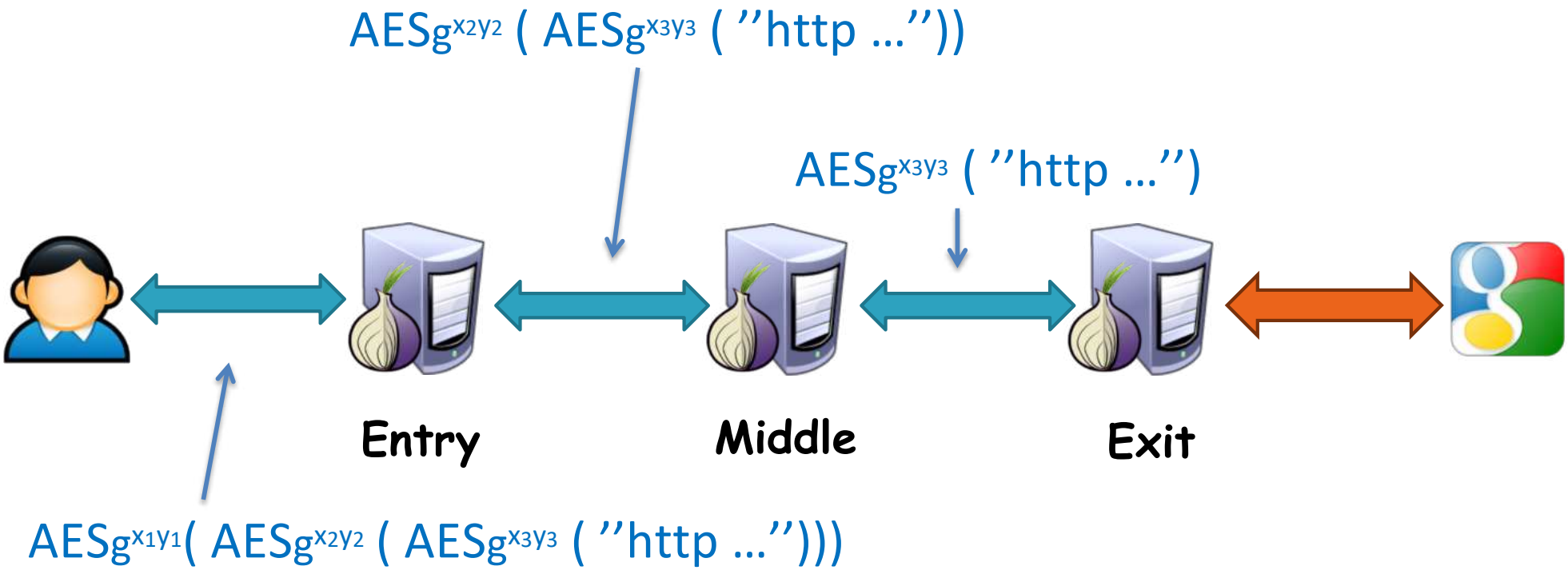
# Dopo creazione circuiti



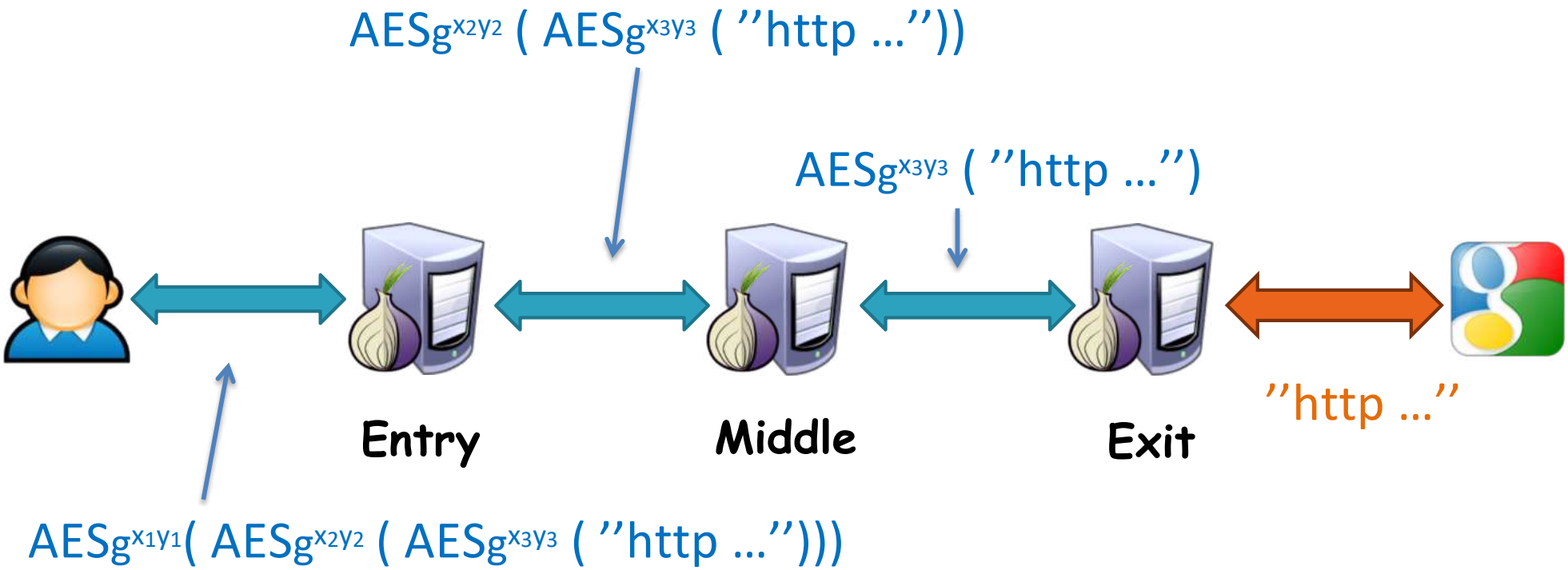
# Dopo creazione circuiti



# Dopo creazione circuiti



# Dopo creazione circuiti



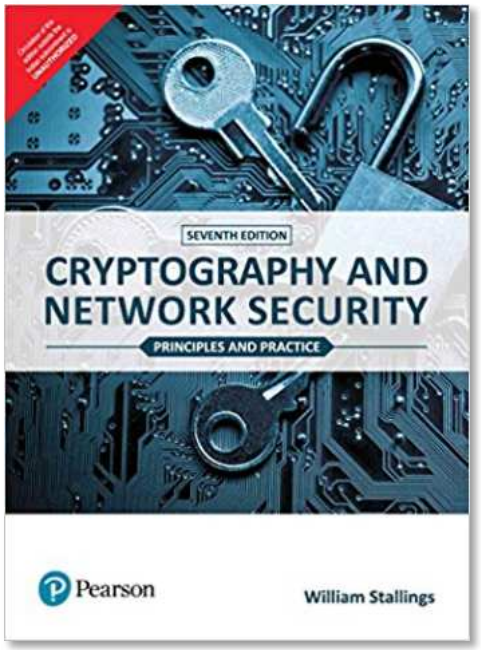
# Forward Secrecy

Il protocollo Diffie-Hellman viene frequentemente usato per ottenere Forward Secrecy per la velocità della generazione della chiave

# Bibliografia

Cryptography and Network Security: Principles and Practices
Prentice-Hall (7/Ed)
by W. Stallings, 2016

- Logaritmo Discreto, par. 2.8
- Diffie-Hellman, par. 10.1



# Domande?