Quando si parla di bug bisogna sempre specificare l'obiettivo dell'attacco e la "gravità dell'attacco". Cosa ottengo?
- chi attacca? sarebbe chi prova a falsificare il dato, non posso essere io stesso, ma un altro. Tuttavia dovrebbe accedere al mio device. Ma quello lo faccio io!
- Verify C19: uso local time del telefono. Potrei usare orologio online. PEGGIO, perché ora devo proteggere il modo in cui prendo la data online, essendo elemento esterno.

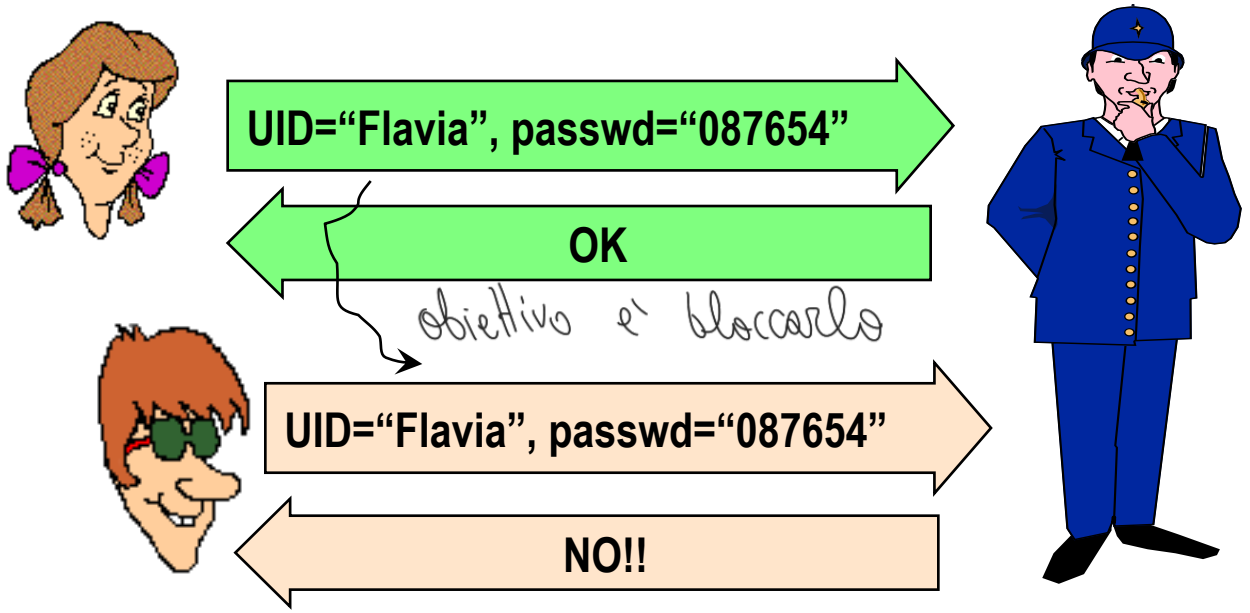
Extra details on PAP/CHAP authentication

OTP:

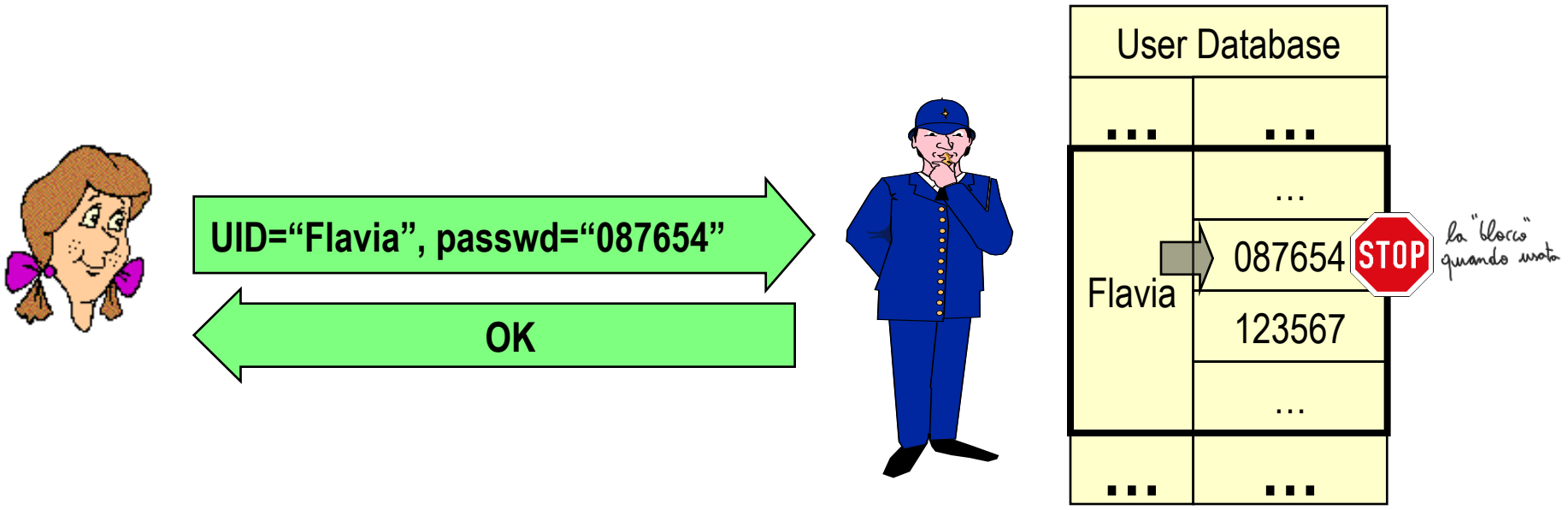
One-time passwd

➔**Is it possible to extend PAP so that user changes passwd at every (successful) attempt?**

⇨If it is, would prevent replay/playback attacks



One-time passwd: trivial... but...



➔N (large) passwd per user

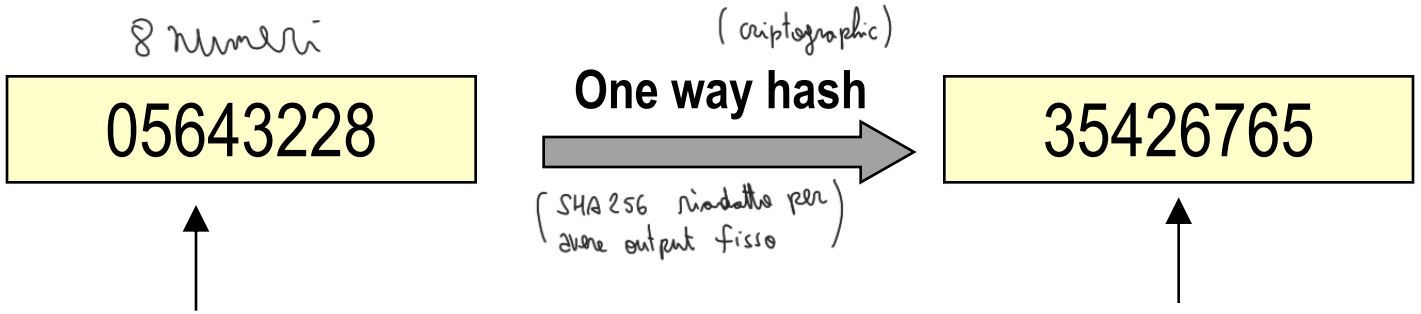
➔10.000.000++ users

➔HUGE DB!! Not viable ← scalabilità pessima

(costoso)

NON DEVO CAMBIARE NULLA NEL SISTEMA! (crea solo problemi)

Idea: hash chains



inizio da un valore e produco Hash

P[0] = starting point, "seed"
P[i] = H(P[i-1]) → hash del precedente
P[N] = last value

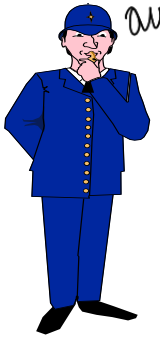
genero N OTP password, partendo solo dalla prima!

bisogna attaccare dove c'è la "configurazione!"

One-time passwd: practical

offline

P[0]



auth P[0] ... P[N] P[N+1] — da mantenere

Hash è uni-direzionale, faccio in modo che ciò che invia Flavia, applicando Hash, sia confrontabile dal poliziotto

Flavia: P_0, P_1, P_2, P_3, P_{3+1} — Poliziotto: $H(P_2)$, $H(P_3)$, salvo $H(P_3)$

P_{3+1} ← stored at ① in DB with 'Flavia', il resto il server lo dimentica

come prenderlo?

N = 3

Compute P[0]…P[N]

$P_{[0]} = 3120$
$P[1] = 1214$
$P[2] = 5123$
$P[3] = 4134$ ← start
$P[3+1] = 7991$ (N+1)

Compute & store Flavia → P[N+1] = 7991

Il server ha SEMPRE P(i+1) rispetto al client che invia H(P(i)) = P(i+1) partono dalla fine.

4134

UID="Flavia", passwd= P[N]

5123

UID="Flavia", passwd= P[N-1]

… … …

UID="Flavia", passwd= P[i-1]

If H(P[N])==P[N+1] OK; store P[N]

↳ tolgo Flavia [7991] e metto [4134]

vedo se H[5123] = 4134

Stored P[i]
If H(P[i-1])==P[i]
OK; store P[i-1]

se perdo pw?

mando P[i-1] e poliziotto rifà H per vedere se corrisponde ad altro valore memorizzato

One-time passwd benefits

- **Passwd/code in clear = OK**
- **Relaxed assumption on server-side security: improved robustness against server-side attacker**
  - ⇨Authenticator only stores USED passwd
  - ⇨no way to predict next one (1-way hash)
- **Authenticator only stores 1 value**
  - ⇨Same complexity as in ordinary PAP
- **Issues:**
  - ⇨Large N to prevent frequent renegotiation
  - ⇨Client side = vulnerable (must store passwd seed or whole vector)

One-time password: mainly in 2-factor authentication

solo qui non va bene, ma è comodo per la gente.

- **Different requirements**
  - One-time authorization token
    - Generated on a different device (e.g. phone, keycard)
    - Received on a different channel (e.g. SMS, email, …)
  - Must be human-friendly
    - Main low level tech issue: how to truncate Hash (e.g. 160 bits) into 6 digits. See dedicated RFCs below
  - Different deployment model and relevant assumptions
    - Both server and clients are assumed to be secure
      - » no more strict need to use hash chains

(se attacco poliziotto fa multa, PIN lo ho l'altro che è sicuro. I voi PIN sono clear!

One-time password in 2-factor authentication

➔HOTP – HMAC-based OTP

- ⇨Uses HMAC, not plain hash
  - ➔(what's this? More later!)
- ⇨Does not use a chain, but a counter
- ⇨HOTP(K,C) = Truncate(HMAC-SHA-256(K,C))
- ⇨Details in RFC 4226

➔TOTP – Time-based OTP

- ⇨Turn time into counter
  - ➔TOTP = HOTP(K, T)
  - ➔T = (Current Unix time - T0) / X
- ⇨details in RFC 6238

CHAP: what about mutual authentication?

CHAP and mutual authentication / 1

(identificano tutti e 2)

CHALLENGE = 135623

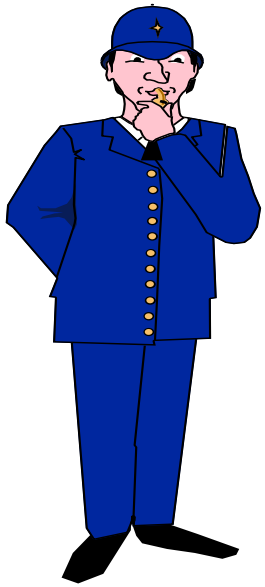
RESP: Flavia, H(sharedsecret, 135623)

ACK

CHALLENGE = 324567

RESP: servername, H(sharedsecret, 324567)

ACK



PAP non ha senso per tale scopo.

Usage of a shared Secret… good idea, Easy to manage!

Good idea??

MAI usare protocollo ottimo per scopo X per uno scopo Y diverso (anche se simili)

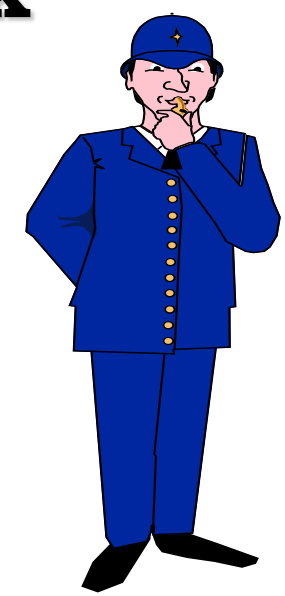
CHAP and mutual authentication /2
Reflection attack

la blocco!

CHALLENGE = 135623

non è al protocollo, non viene controllato

CHALLENGE = 135623



RESP: servername, H(sharedsecret, 135623)

ACK

RESP: client, H(sharedsecret, 135623)

Posso fixarlo??

ACK

VERY risky!!.

Attacker:

→replays server challenge

→accept computed resp

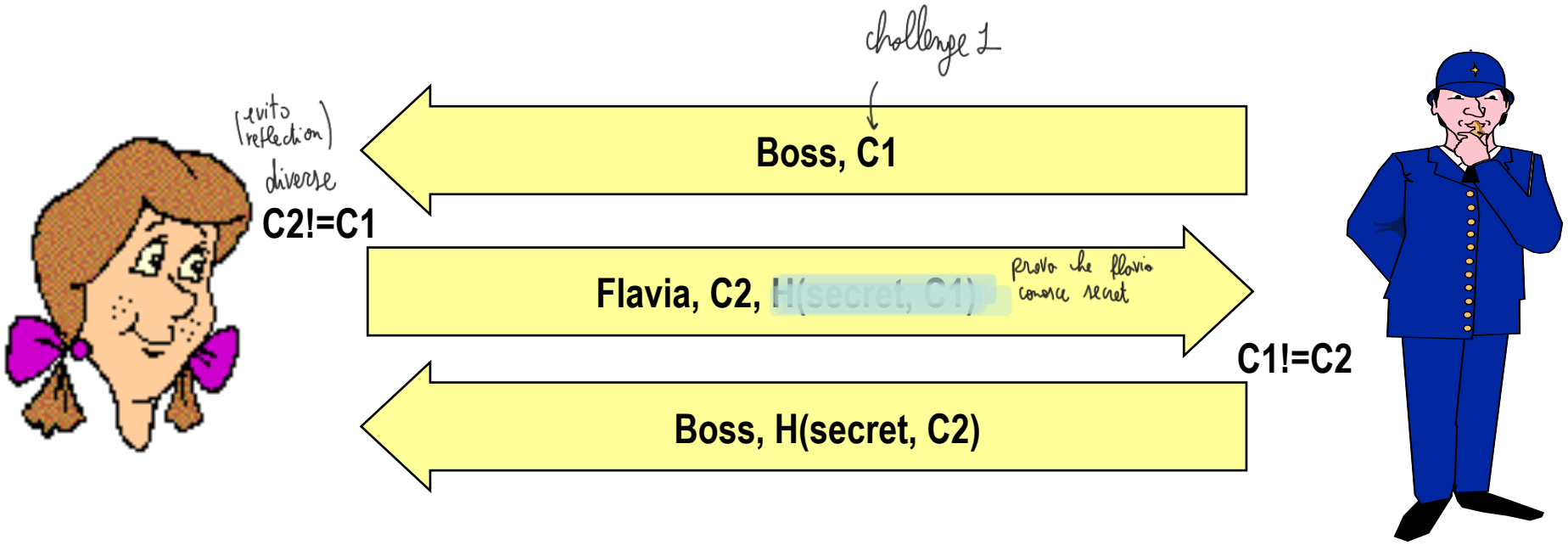
→uses resp to authenticate!!

Without any info on real client !!

Yes but this happened because you used two unilateral authentication protocols!

Let's combine them into a single Mutual authentication with Challenge-Response

Pippo's mutual authentication protocol (seriously, don't do this!)

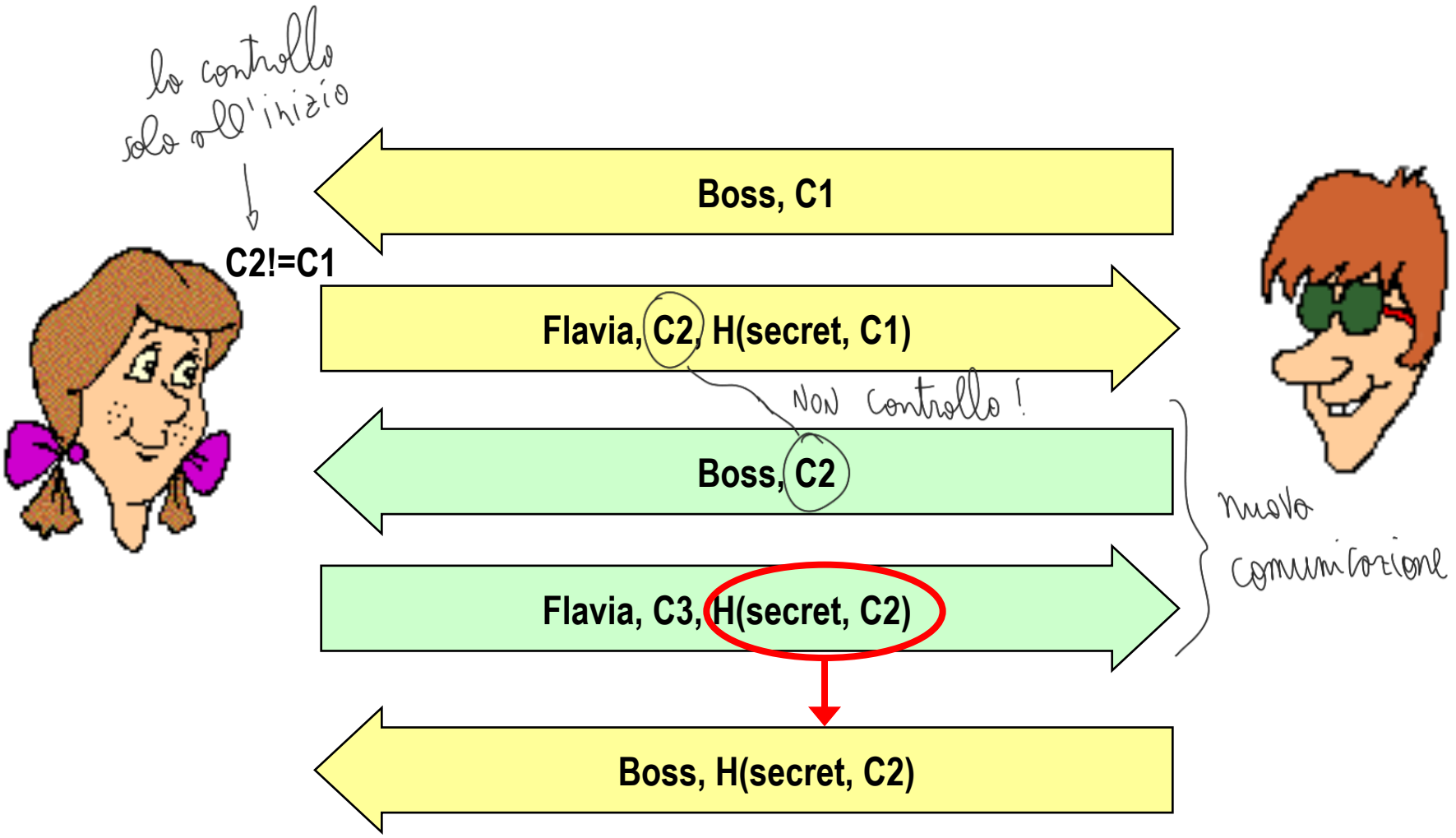


Flavia shows knowledge of secret over C1

Boss shows knowledge of secret over C2!=C1

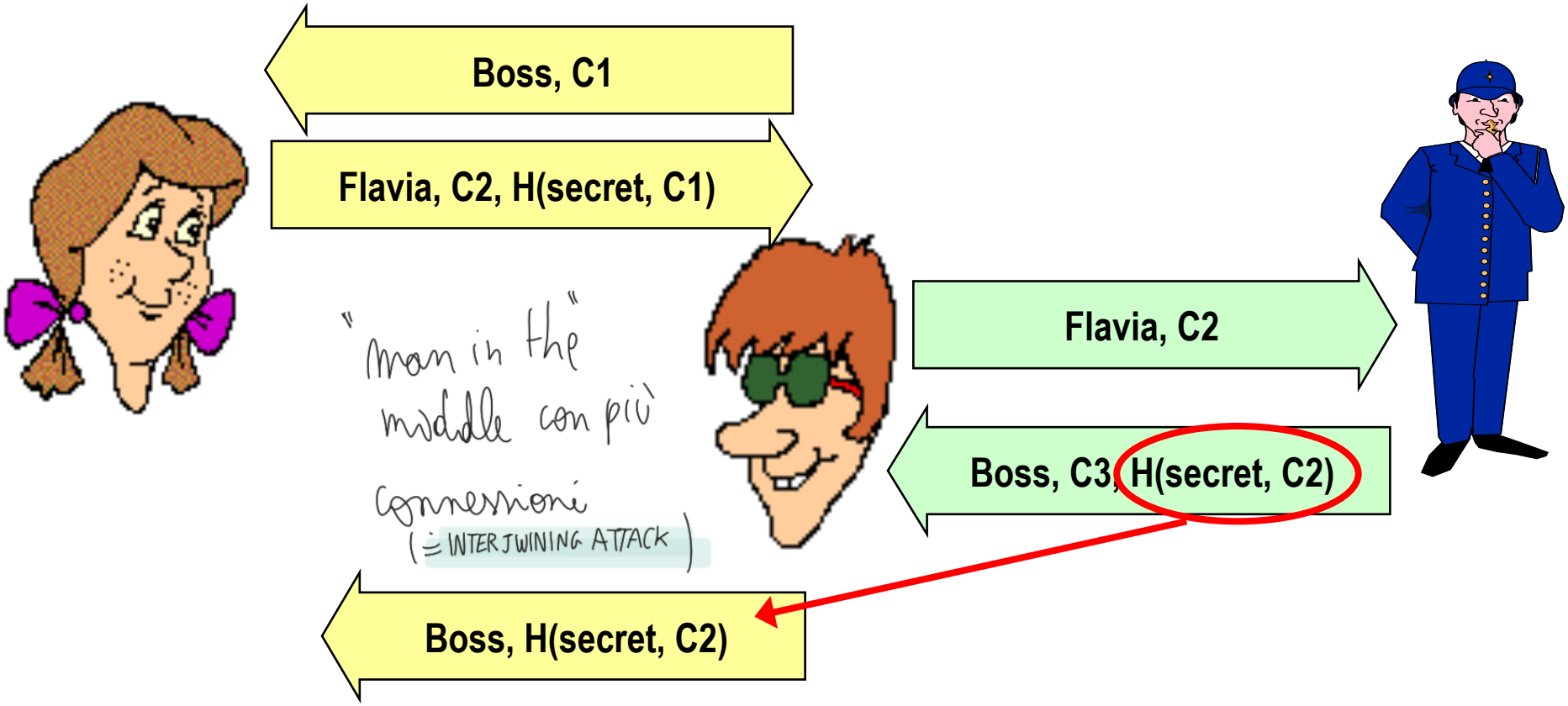
Where is the flaw?

Reflection attack!



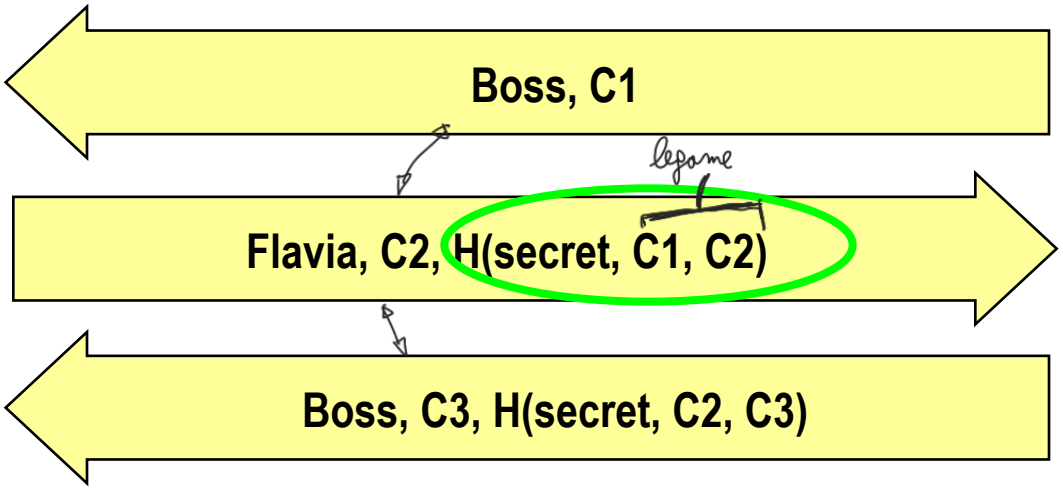
Let's then prevent reflection!

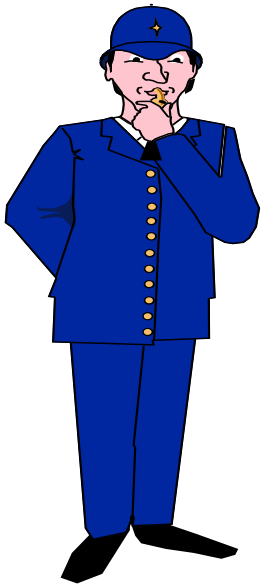
➔ **Intertwining attack:**
Attacker may use "other" party!



Let's try to fix this

voglio mutua autenticazione crittograficamente collegate anche se le due sessioni sono scollegate. Voglio Hash dipendenti! Dati C_1 e C_2, non le voglio slegate





Hash BOTH challenges together!

Cryptographically binds challenges in same handshake

i.e. binds the two authentication directions together!

Does not work

illusione ottica

Boss, C1

Flavia, C2, H(secret, C1, C2)

Boss, C2

Flavia, C3, H(secret, C2, C3)

Boss, C3, H(secret, C2, C3)

NUOVA SESSIONE

Too many nonces!!

Minimize nonces

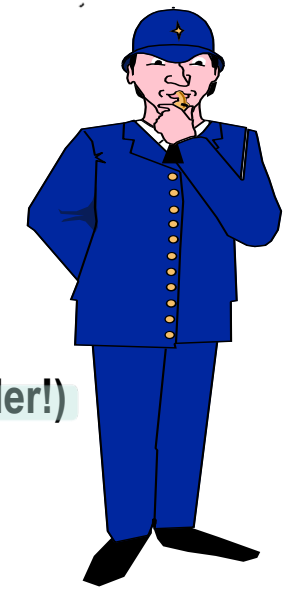
C_1 e C_2 "single" mutual section challenge
freshness generata da entrambi!

Boss, C1

Flavia, C2, H(secret, C1, C2)

Same challenges, but different values (order!)

Boss, H(secret, C2, C1)



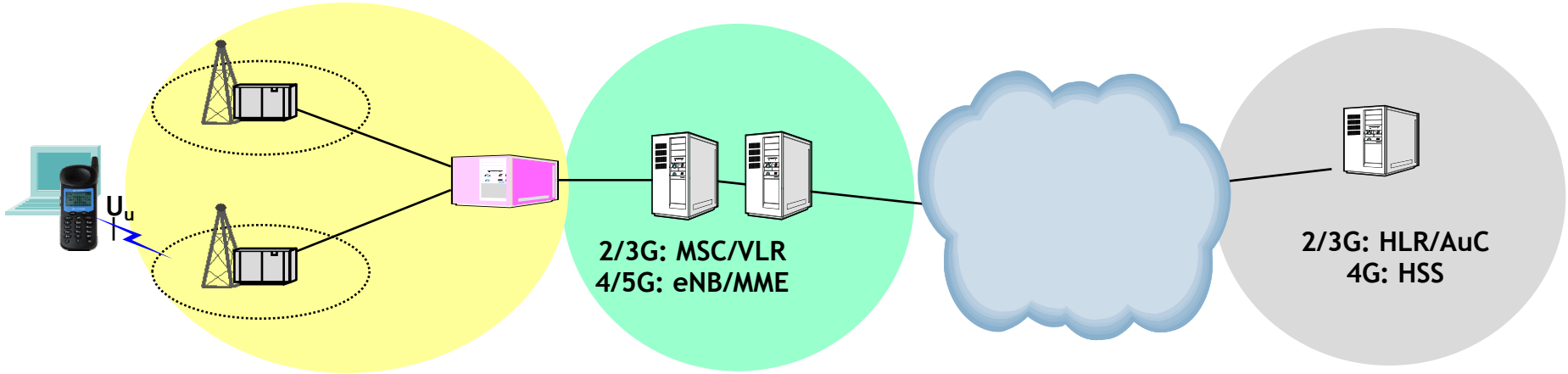
posso cambiare solo lui, posso metterlo in Hash

No reflection possible anymore

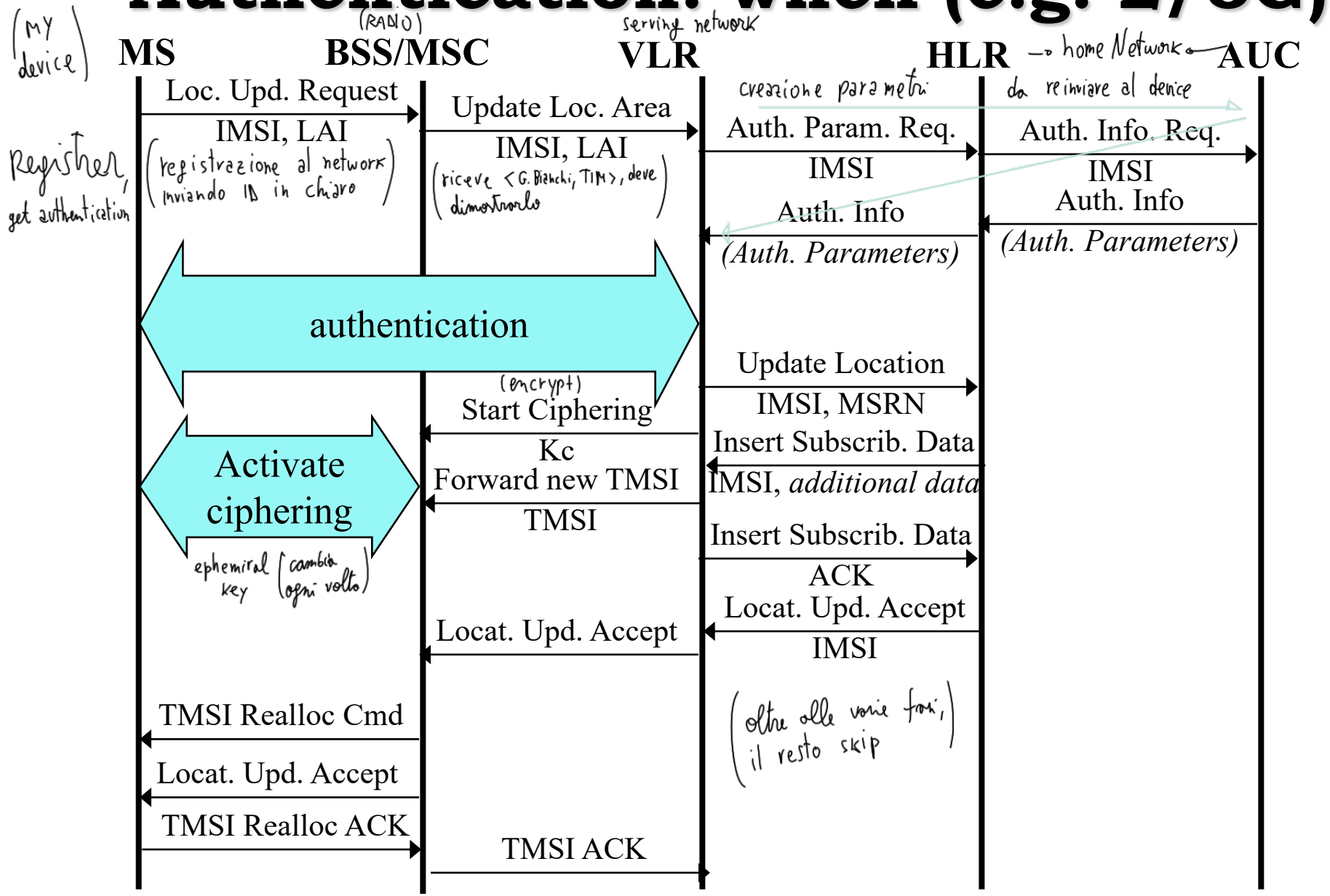
cioè entrambi contribuiscono ad 'aggiornare' le challenge

Challenge-Response authentication in Wireless Cellular systems

Architecture of wireless cellular systems (abstracted)



Authentication: when (e.g. 2/3G)



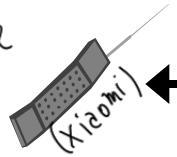
Authentication in 2G

"CHAP" approach, NON è CHAP

(unilateral)

SOLO TIM ha Ki

Real Giuseppe MS



classic challenge

Authentication Request

Challenge: 128 bit RAND

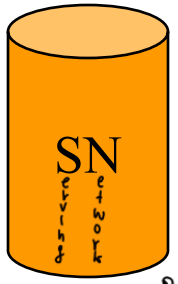
Giuseppe, Challenge

IMSI, RAND

Cambiano

SRES, Kc

H(Ki, RAND) next encryption key



NON mi fido, mando response senza mandare il segreto, non fornisco "long term secret Ki", sta in "SIM" e "HN server"

Ki in SIM TIM — RAND

Hash ≅ A3 — alg. A3 può essere proprietario, solo il provider deve conoscerlo

chip nella SIM che fa la computazione

SRES

MiddleMan può prendere RAND e SRES ma non Kc

SRES

classic Response

Authentication Response

Signed RESult: 32 bit SRES

Equal?

Yes: user autentico

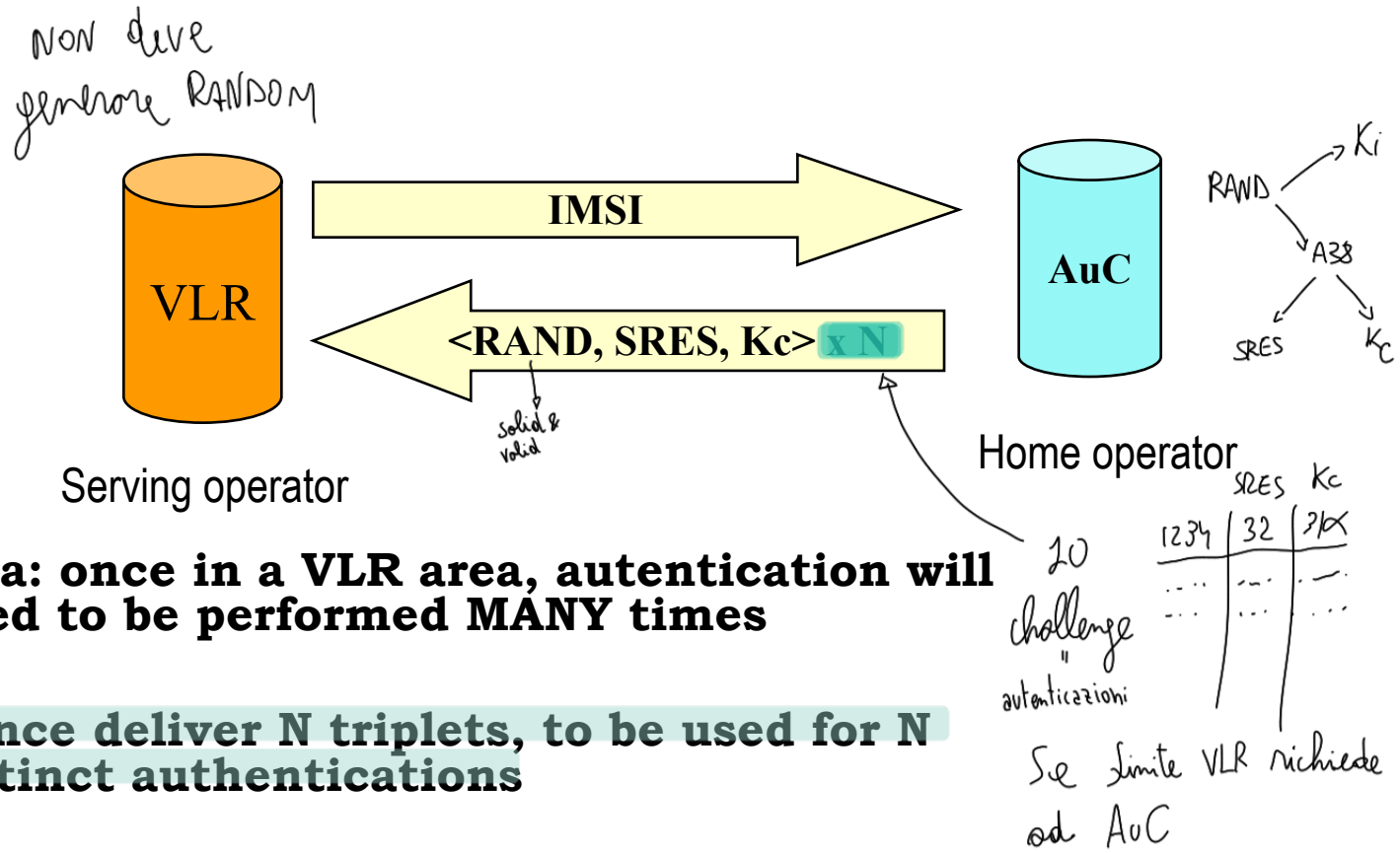
Ki — RAND

A8 — vale lo stesso di A3

produco Kc → userò per la prossima sessione

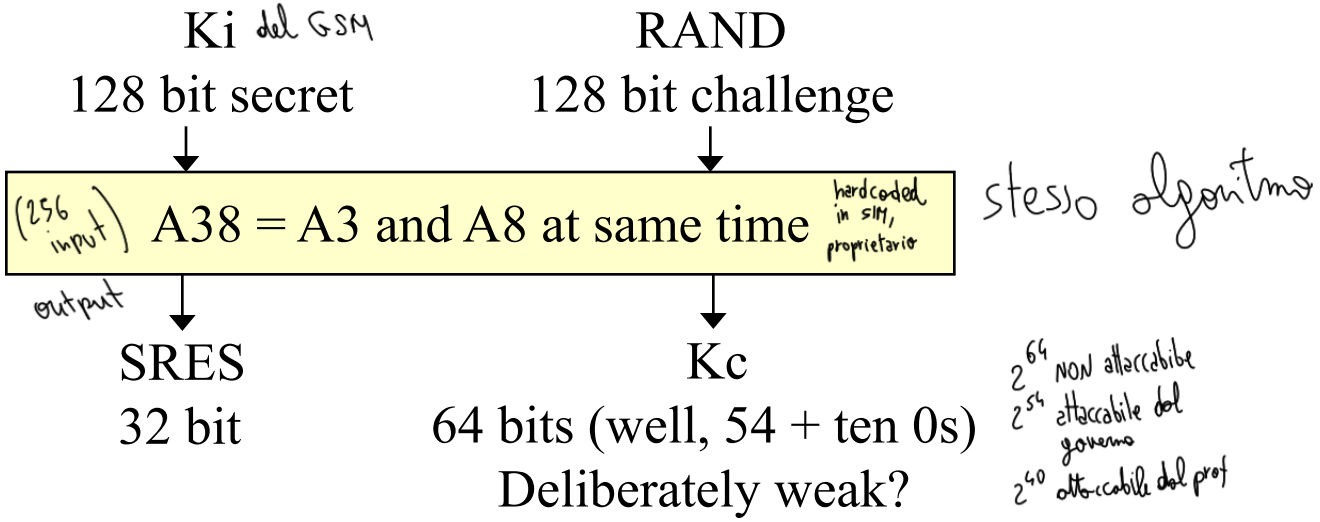
RAND fatto da SN, lo vorrei da TIM, bisogna "prendersi tutte le responsabilità"!

Triplets (Authentication Vector)



- **Idea: once in a VLR area, autentication will need to be performed MANY times**
- **Hence deliver N triplets, to be used for N distinct authentications**
- **IMPORTANT: VLR does NOT need to know authentication algo used (A3, A8)**
  - Triplet contains computed result by AuC
  - A3, A8 run inside the SIM (given by operator)

Authentication: details



➔Challenge response with: (analogie)

- ⇨Challenge → RAND
- ⇨Secret → Ki
- ⇨Hash → A3 algorithm

On the A3/A8 algorithms

- **Security by obscurity**
  - A3 algorithm CAN BE operator-specific!
  - But most vendors originally used algo today called COMP128
  - Non disclosed but…
    - Reverse engineered? Leaked out?
- **COMP128 broken in 1998**
  - http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html
  - Chosen challenge attack [1998, Briceno, Goldberg, Wagner]
    - Ki disclosed through about 150.000 queries with suitably selected challenges → approx 8h in 1998
    - Having Ki means cloning the card!
  - Better attack [2002, Rao, Rohatgi, Scherzer, Tinguely]
    - Less than 1 minute!
- **Lesson learned:**
  - Security by obscurity does NOT work!! Leave hash design to crypto experts

Over the air attach

➔No mutual authentication!

➔Rogue BTS may easily perform the attach

⇨run it for sufficient time

5G/3G/4G authentication: AKA Authentication and Key Agreement

(STRONGLY Simplified for our purposes)

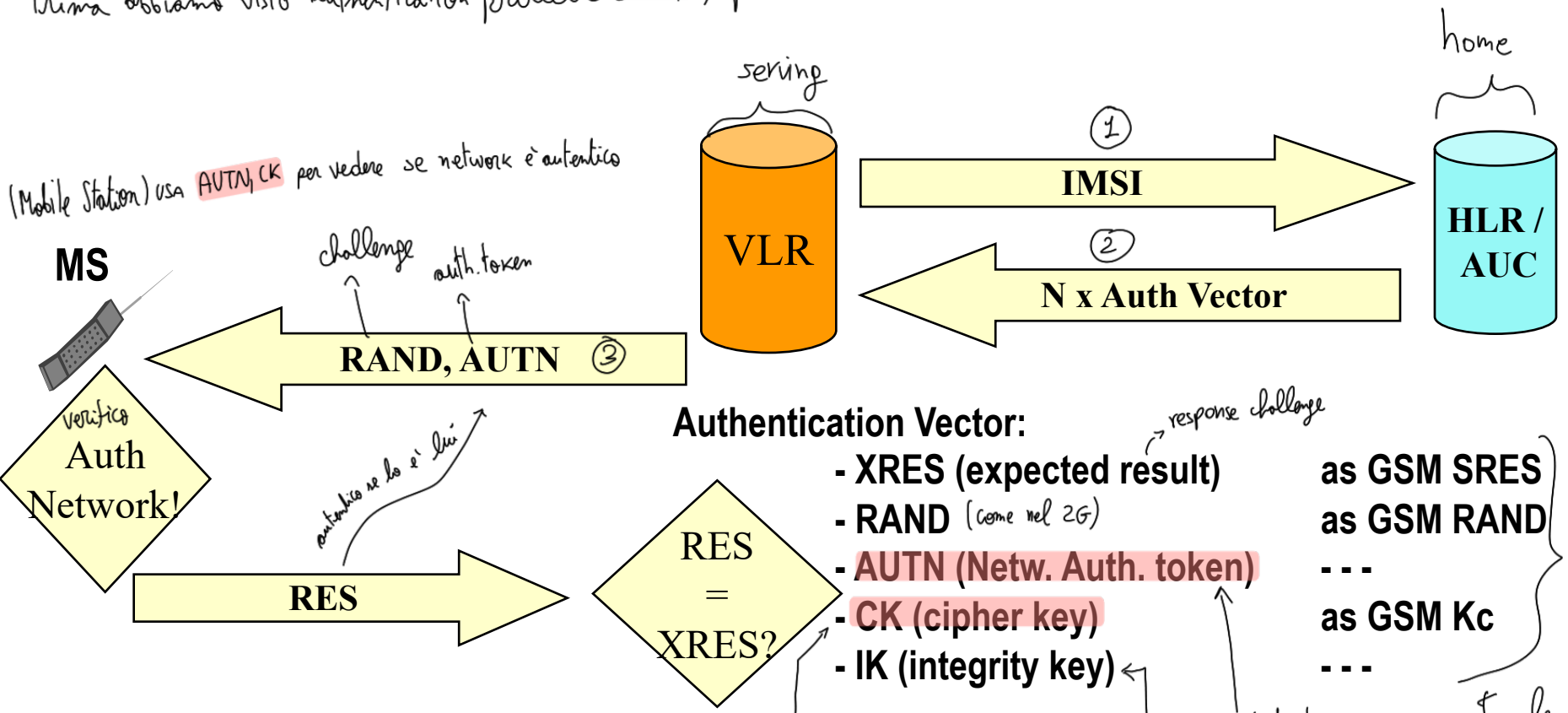
3G terminology

Major differences with 2G

- **Mutual authentication!**
  - Optimized…
  - Guaranteed freshness for auth parameters
- **More comprehensive security**
  - More keys and several extra details
    - (we will not focus on this)
- **Algorithms FIRST scrutinized by research community, THEN selected**
  - The opposite of security by obscurity ☺

Authentication Vector

Prima abbiamo visto authentication protocol 3-WAY, qui MS non si autentica! Perché?



Authentication Vector:

- XRES (expected result) — response challenge — as GSM SRES
- RAND (come nel 2G) — as GSM RAND
- AUTN (Netw. Auth. token) — - - -
- CK (cipher key) — as GSM Kc
- IK (integrity key) — - - -

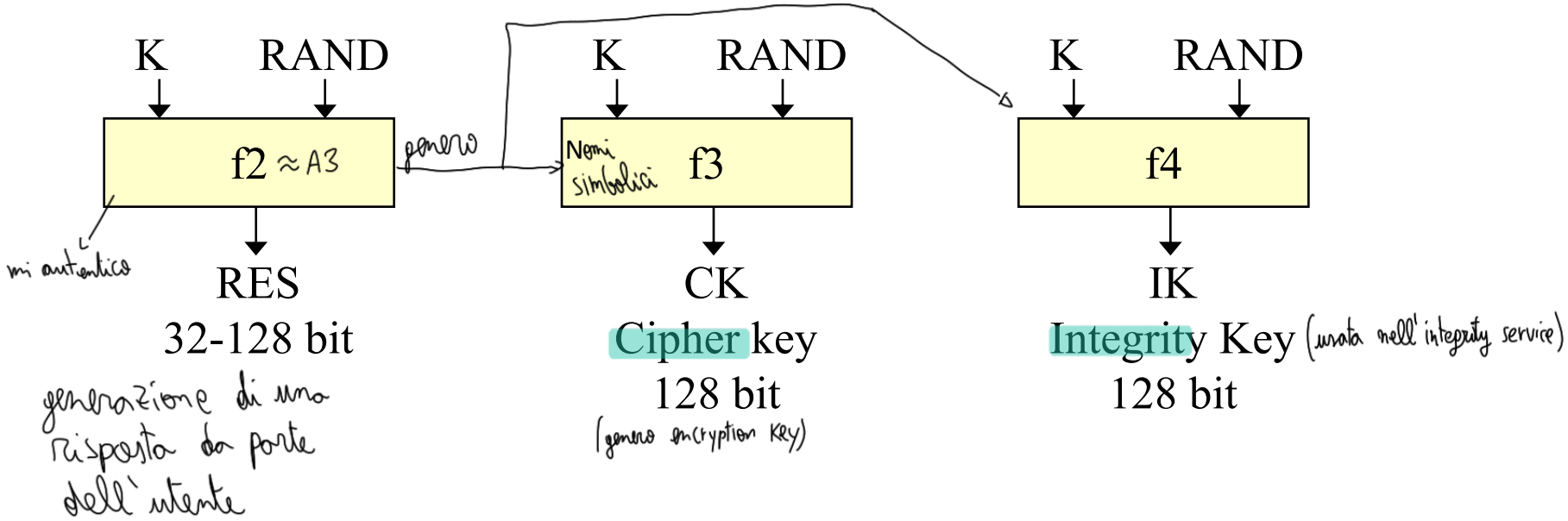
quintuple

integrity (≈ HMAC) + autenticazione usato da stazione per vedere se network è autentico

era output di A8, ora è alla encryption key di GSM

MS authentication

➔Usual challenge response, but different (public) algorithm

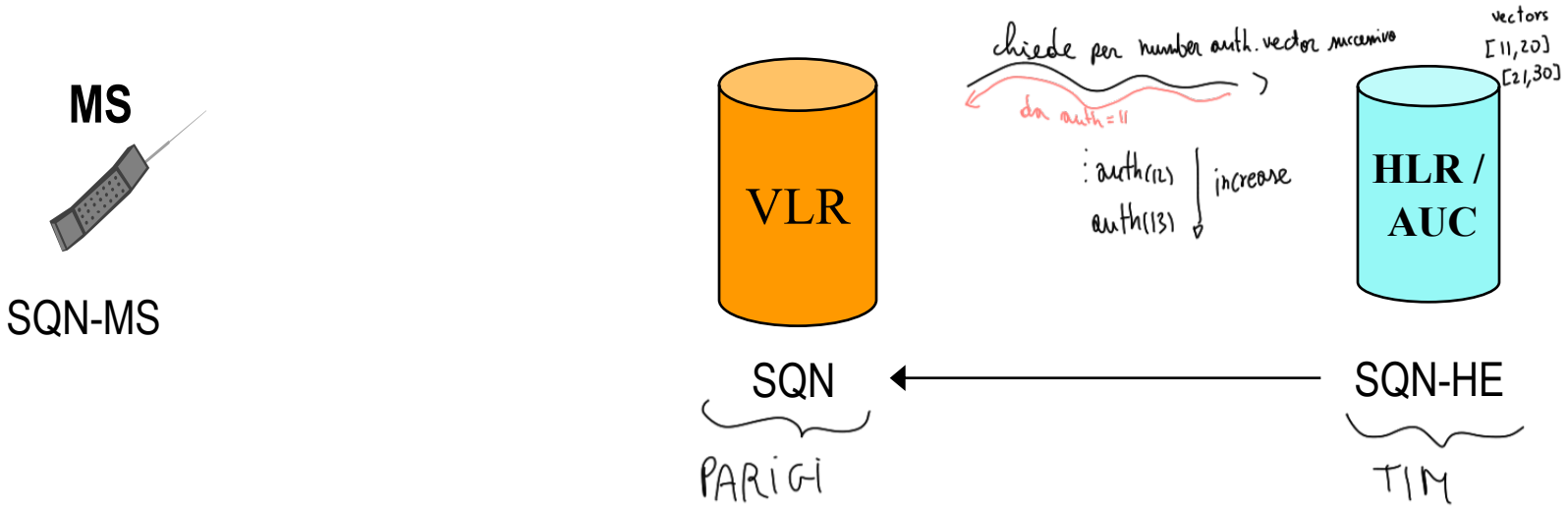


da utilizzare per encryption cipher con C_K, HMAC, IK

(L'ORDINE CONTA)

Network Authentication

- **MS should send a nonce…**
  - 1 extra message
- **Bright idea: use Sequence number as "implicit" nonce!**
  - Issue: MS and AuC must be (approx) synchronized
  - And robust procedures for resync must be specified



[assumo non ci siano problemi tecnici]

SEQ as nonce: idea

memorizzo in SIM l'auth

Current SQN-MS stored

tolleranza, non necessario

Check SQN = SQN-MS + 1 (or in appropriate Tolerance range to cope with lost msg)

verifico che non sia reply attack

new **SQN** (included in AUTN) (given by AuC)

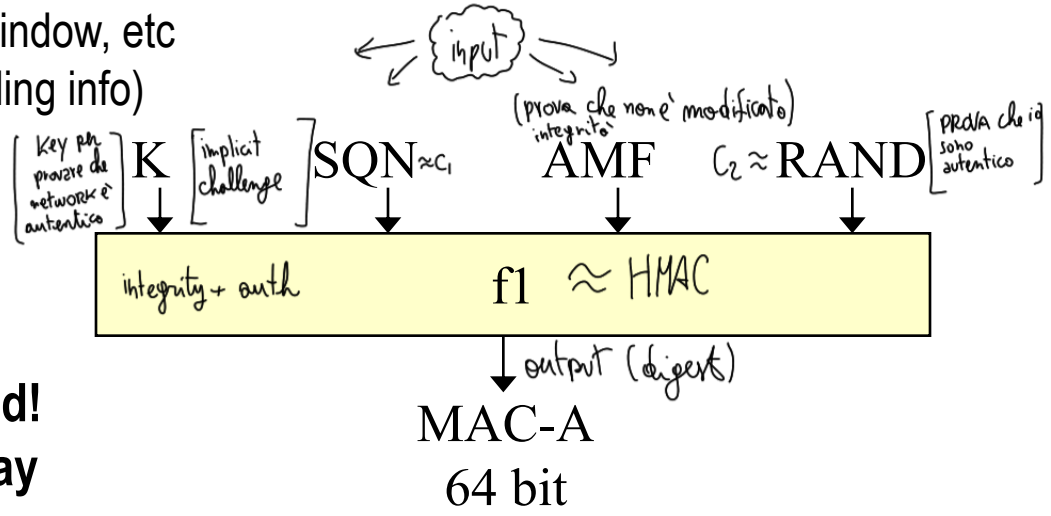
Use SQN as "implicit" challenge!!

→ seq è NONCE

Once auth OK, update local SQN

Network authentication: AUTN format

SQN Sequence number	AMF Auth & key mgmt field	MAC-A Message Auth code
48 bit	16 bit: carries info on which algo or key to use if choice available, sync window, etc (signalling info)	64 bit: allows MS to verify authenticity of Network!



MS: has all info needed to check that MAC-A transmitted by network is the same of MAC-A locally computed! SQN guarantees defense against replay

Minor detail: protecting SQN!

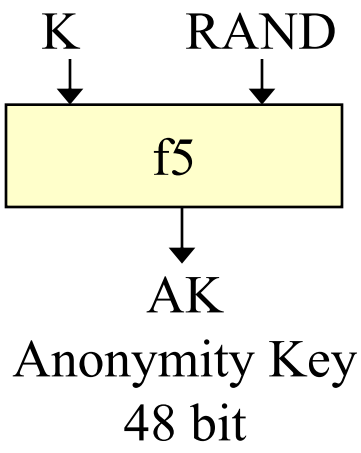
➔ **A privacy problem:** (location: uso TMSI, per disaccoppiare una autenticazione con una successiva)

⇨ By looking at SQN (stepwise increasing), eavesdropper may discriminate and track user!

➔ **Solution: mask SQN with Anonymity Key** [48 bit Pseudo Random]

voglio encrypt, ma non ho negoziato nessuna chiave

SQN xor AK Sequence number	AMF Auth & key mgmt field	MAC-A Message Auth code



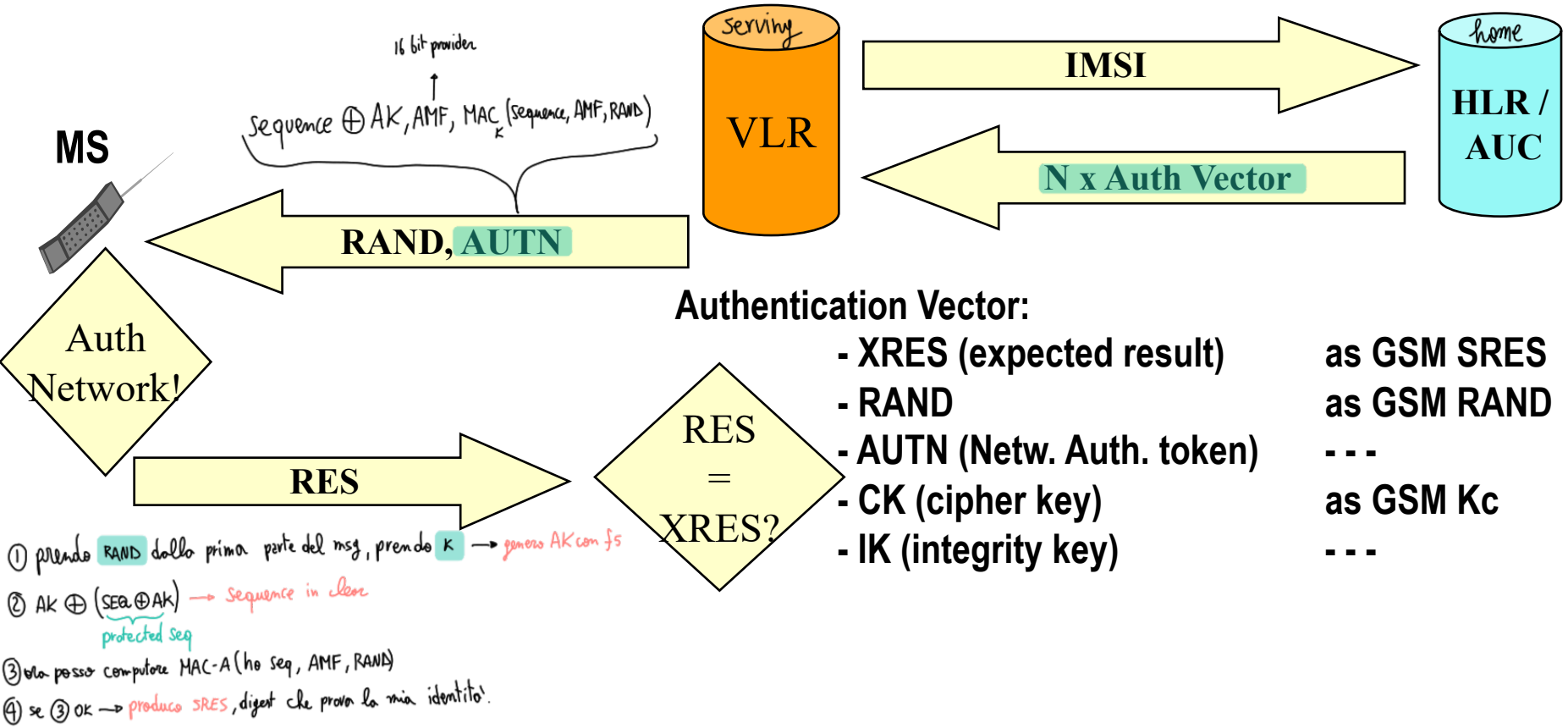
così generata: Key derivation Function è tecnica usata

seq. non mi protegge:
<41235...2, Tor Vergata>
<41235...3, Parioli>
seq. number NO OK

effimero, 1 msg
SOLUZIONE

RIVEDO CON LE NUOVE CONOSCENZE

Authentication Vector



Authentication Vector:

- XRES (expected result)	as GSM SRES
- RAND	as GSM RAND
- AUTN (Netw. Auth. token)	- - -
- CK (cipher key)	as GSM Kc
- IK (integrity key)	- - -