# SCELTE

## Username e password

- Lo username dell'utente deve essere in minuscolo, senza cifre né caratteri speciali.
- I nomi utente identificano univocamente gli utenti. Per questo motivo, ogni username deve essere unico. La lunghezza del nome utente deve essere al massimo di 20 caratteri.
- Le password devono includere almeno 1 lettera maiuscola, 1 lettera minuscola, 1 carattere speciale e 1 cifra. La lunghezza della password deve essere di almeno 8 caratteri.

## Public/Private Keys e Symmetric Key

- Il client nel momento in cui vuole registrarsi conosce già la chiave pubblica del Server, quindi si suppone che precedentemente ne abbia già fatto richiesta e l'abbia ricevuta insieme a un certificato (rilasciato da una CA) che ne dimostra la sua autenticità. Tale chiave è memorizzata in una cartella *PublicKey* in un file nominato *Server.pem.*
- Il client genera una chiave simmetrica effimera ogni qual volta che si connette al Server sia per registrarsi che per effettuare il login. Tale chiave non viene salvata da nessuna parte.

## Encryption and Decryption

- Il client utilizza la chiave pubblica RSA per trasportare in maniera sicura la chiave simmetrica generata.
- La chiave simmetrica viene utilizzata da client e server per criptare la comunicazione sia in fase di registrazione che di login; in particolare in quest'ultima viene utilizzata per trasportare in maniera sicura le chiave pubbliche DHE.
- Il protocollo utilizzato per criptare è AES-GCM. La dimensione dell'IV è di 12 bytes mentre il tag è grande 16 bytes. Essi sono inviati in chiaro, concatenati al testo cifrato e generati per ogni criptazione-decriptazione.
- La scelta di usare AES-GCM garantisce:
    1. **Confidenzialità:** Solo chi possiede la chiave segreta può decifrare i dati originali. Quindi, AES-GCM garantisce che solo le parti autorizzate possano accedere ai dati in chiaro.
    2. **Autenticità/Integrità dei dati e non malleabilità**: AES-GCM incorpora un meccanismo di autenticazione basato su Galois/Counter Mode (GCM). Questo meccanismo non solo cifra i dati ma include anche un tag di autenticazione, calcolato tramite una funzione di hash crittograficamente sicura. Questo tag è appeso ai dati cifrati e consente al destinatario di verificare se i dati sono stati alterati durante la trasmissione. Se i dati vengono modificati, il tag di

autenticazione non corrisponderà e la verifica fallirà, evidenziando la possibile manipolazione dei dati.

L'alternativa al tag di autenticazione sarebbe potuta essere l'utilizzo delle firme digitali, per fornire oltra a autenticità e integrità anche non ripudio, ma supponiamo che il server sia una fonte affidabile che non tenta di truffarci, pertanto abbiamo indirizzato il protocollo per garantire sicurezza da terze parti.

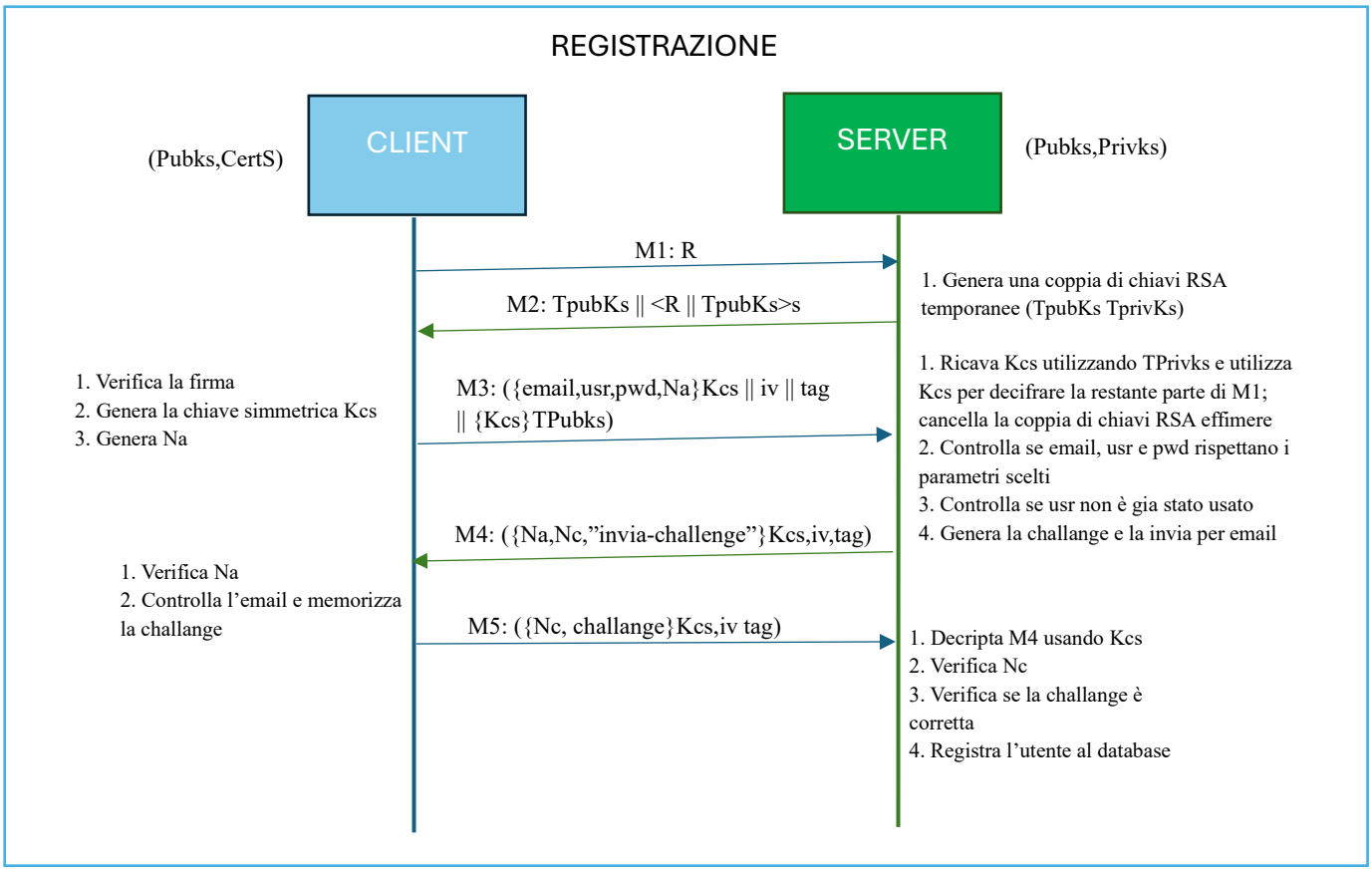
## No-replay attack

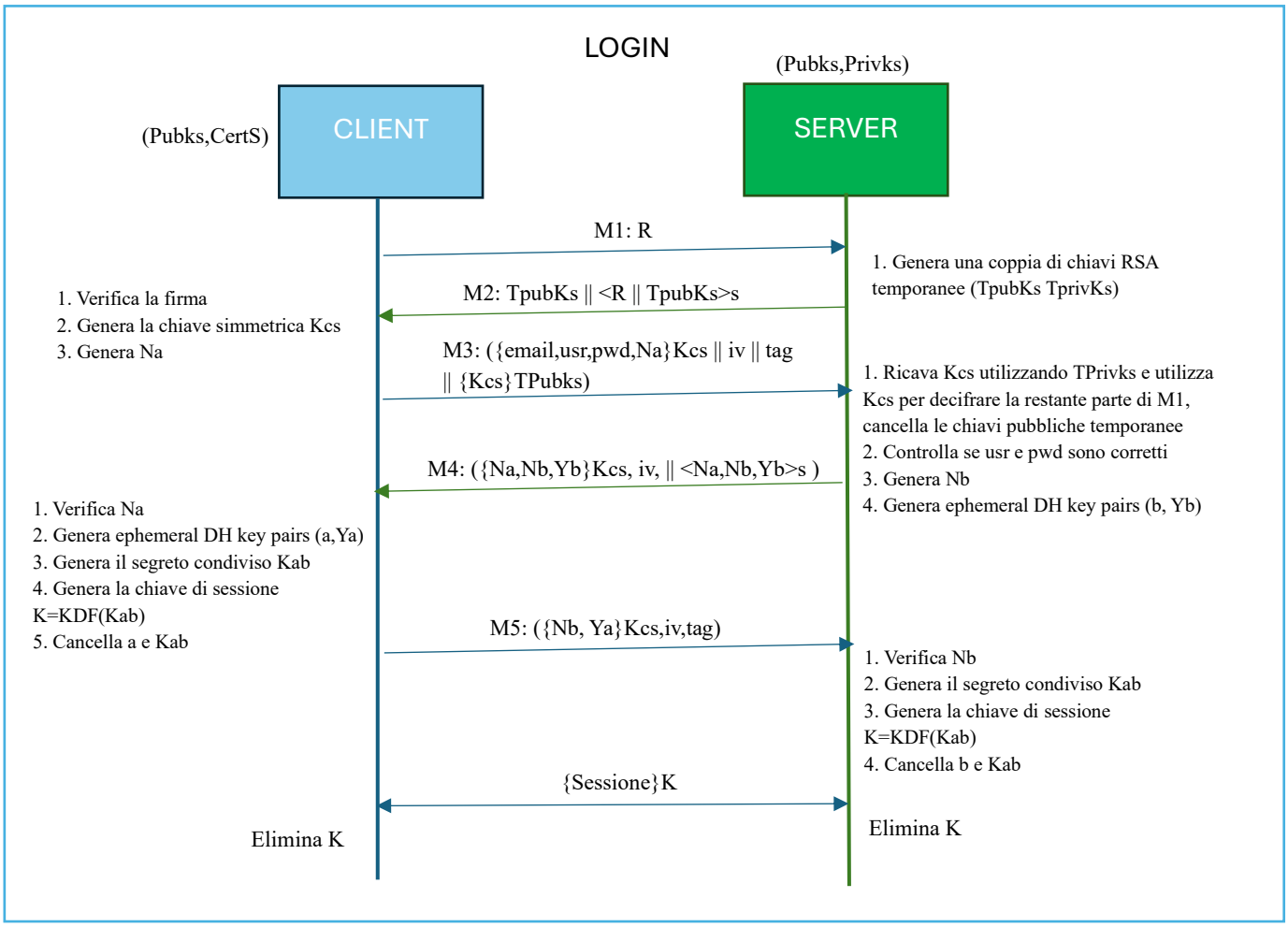
Per evitare il no-replay attack, come si può vedere dalla figura, vengono generati dei nonce per ogni messaggio, il ricevitore mantiene una lista di nonces utilizzati e verifica se il nonce nel messaggio è nuovo o è già stato ricevuto (controllando appunto se è gia presente nella lista), in questo caso viene generato un errore di replay attack. Inoltre, il nonce ricevuto viene rinviato indietro, questo è utile per 2 ragioni: la prima è garantire che entrambi i partecipanti della comunicazione siano effettivamente parte della sessione corrente e la seconda che ogni messaggio sia legato ai precedenti in modo sicuro. Questo rafforza la protezione contro gli attacchi di replay e assicura l'integrità e la sequenzialità della comunicazione.

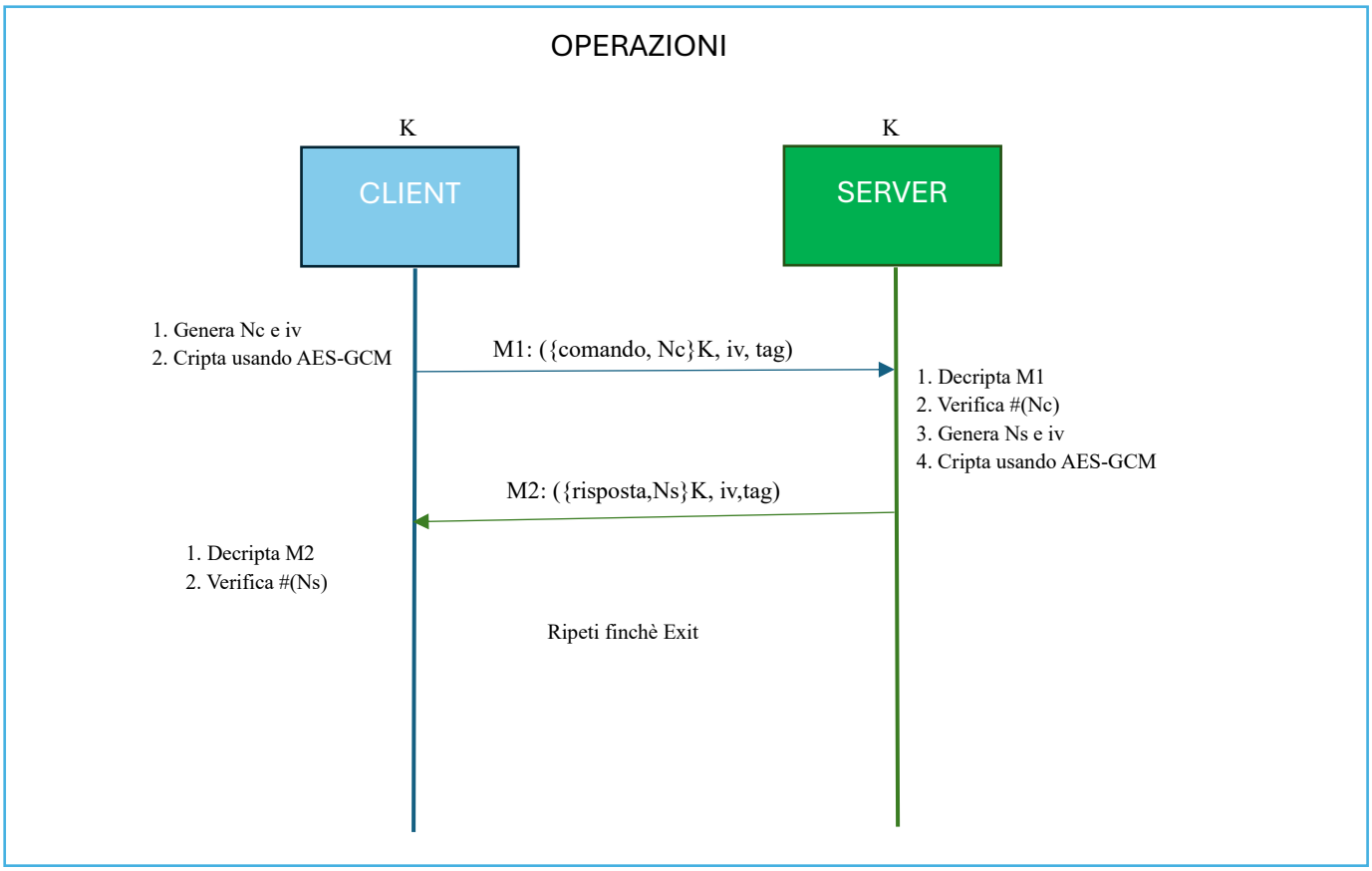
## Registrazione con protocollo challenge-response

- Il client invia email username e password al server.
- Il server invia un email all'utente contenente una OTP di 6 cifre. Simuliamo l'invio dell'email tramite la creazione di una cartella nominata con lo username scelto dall'utente all'interno del quale viene creato un file contenente il numero a 6 cifre.
- Se l'utente inserisce la challenge correttamente, le sue credenziali vengono inserite in un file nominato "*Database.txt"* altrimenti la cartella con il suo nome viene eliminata e viene inviato un messaggio di errore. Le credenziali vengono salvate nel file seguendo l'ordine sotto riportato:

  ***email:username:salt$hash(salt || password)***


REGISTRAZIONE
CLIENT
SERVER
(Pubks,CertS)
(Pubks,Privks)
M1: R
M2: TpubKs || <R || TpubKs>s
M3: ({email,usr,pwd,Na}Kcs || iv || tag || {Kcs}TPubks)
M4: ({Na,Nc,"invia-challenge"}Kcs,iv,tag)
M5: ({Nc, challange}Kcs,iv tag)
1. Verifica la firma
2. Genera la chiave simmetrica Kcs
3. Genera Na
1. Genera una coppia di chiavi RSA temporanee (TpubKs TprivKs)
1. Ricava Kcs utilizzando TPrivks e utilizza Kcs per decifrare la restante parte di M1; cancella la coppia di chiavi RSA effimere
2. Controlla se email, usr e pwd rispettano i parametri scelti
3. Controlla se usr non è gia stato usato
4. Genera la challange e la invia per email
1. Verifica Na
2. Controlla l'email e memorizza la challange
1. Decripta M4 usando Kcs
2. Verifica Nc
3. Verifica se la challange è corretta
4. Registra l'utente al database


LOGIN
(Pubks,Privks)
CLIENT
SERVER
(Pubks,CertS)
M1: R
M2: TpubKs || <R || TpubKs>s
M3: ({email,usr,pwd,Na}Kcs || iv || tag || {Kcs}TPubks)
M4: ({Na,Nb,Yb}Kcs, iv, || <Na,Nb,Yb>s )
M5: ({Nb, Ya}Kcs,iv,tag)
{Sessione}K
1. Verifica la firma
2. Genera la chiave simmetrica Kcs
3. Genera Na
1. Genera una coppia di chiavi RSA temporanee (TpubKs TprivKs)
1. Ricava Kcs utilizzando TPrivks e utilizza Kcs per decifrare la restante parte di M1, cancella le chiavi pubbliche temporanee
2. Controlla se usr e pwd sono corretti
3. Genera Nb
4. Genera ephemeral DH key pairs (b, Yb)
1. Verifica Na
2. Genera ephemeral DH key pairs (a,Ya)
3. Genera il segreto condiviso Kab
4. Genera la chiave di sessione K=KDF(Kab)
5. Cancella a e Kab
1. Verifica Nb
2. Genera il segreto condiviso Kab
3. Genera la chiave di sessione K=KDF(Kab)
4. Cancella b e Kab
Elimina K
Elimina K


OPERAZIONI
K
K
CLIENT
SERVER
1. Genera Nc e iv
2. Cripta usando AES-GCM
M1: ({comando, Nc}K, iv, tag)
1. Decripta M1
2. Verifica #(Nc)
3. Genera Ns e iv
4. Cripta usando AES-GCM
M2: ({risposta,Ns}K, iv,tag)
1. Decripta M2
2. Verifica #(Ns)
Ripeti finchè Exit