**Cifrari asimmetrici**

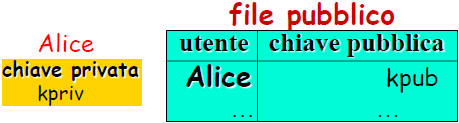
Nella crittografia asimmetrica vengono utilizzate due chiavi:

1. Una chiave **pubblica** per cifrare il messaggio;
2. Una chiave **privata** per decifrare il messaggio.

In questa tipologia di crittografia *non* vi sono problemi che riguardano la segretezza della chiave o tantomeno trovare un modo per scambiarsela in modo sicuro, semplicemente:

1. Il mittente genera una coppia composta da chiave pubblica e privata
2. Il mittente usa la chiave pubblica per cifrare e quella privata per decifrare.

Alla base di tutte le tecniche di cifratura asimmetrica vi è una funzione che è facile da calcolare e **difficile da invertire** (a meno che non si conoscono delle informazioni aggiuntive relative alla costruzione - trapdoor

**Schema della cifratura:**

1. Alice genera una coppia di chiavi: privata e pubblica
2. La chiave pubblica di Alice viene pubblicata su un file pubblico consultabile da tutti
3. Bob vuole cifrare un messaggio M e spedirlo ad Alice lungo un canale insicuro
4. Bob cifra il messaggio usando la chiave pubblica e poi invia C ad alice (C è una sequenza di bit), cioè:

cifratura di M per Alice è:

**Schema della decifratura:**

1. Alice riceve il testo cifrato C da Bob e lei lo vuole decifrare
2. Alice esegue il seguente algoritmo che prende in input C e la chiave privata di Alice:

decifratura di C è:

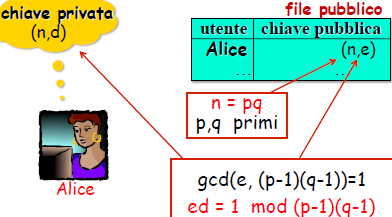
**Nota:** La cifratura la vedono anche gli altri utenti come anche l’algoritmo di decifratura, ma a differenza degli altri utenti, solo Alice conosce la sua chiave privata.

**Alcune considerazioni** sui cifrari asimmetrici:

* Chiunque può cifrare un messaggio per Alice, ma solo Alice può decifrare un messaggio che è stato cifrato appositamente per lei.
* **Non** ci sono chiavi condivise tra gli utenti: ogni utente genera una coppia di chiavi (*public key*, *private key*) e rende pubblica solo la chiave pubblica.
* Ogni utente memorizza una sola chiave (cioè la chiave privata)

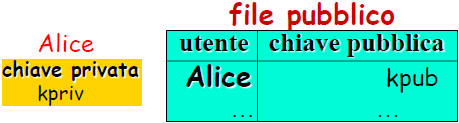
Differenza con quelli simmetrici e che le persone, prima di condividere il messaggio, **non si** devono incontrare per condividere le informazioni sulla chiave da usare per la cifratura e decifratura.

**RSA**

****Lo schema del RSA è il seguente:

1. A genera due numeri primi grandi,
2. A calcola e ;
3. A sceglie un numero tale che ;
4. A calcola usando l’algoritmo di Euclide Esteso;
5. A pubblica *n* ed *e* come sua chiave pubblica ;
6. A conserva *n* e *d* come sua chiave privata ;

**Nota**: la proprietà che sono primi garantisce sempre l’esistenza dell’inverso moltiplicativo *d*.

**Schema della cifratura RSA:**

1. Alice genera una coppia di chiavi: privata è la coppia e pubblica è la coppia
2. La chiave pubblica di Alice viene pubblicata su un file pubblico consultabile da tutti
3. Bob vuole cifrare un messaggio M e spedirlo ad Alice lungo un canale insicuro
4. Bob cifra il messaggio usando la chiave pubblica e poi invia C ad alice (C è una sequenza di bit), cioè:

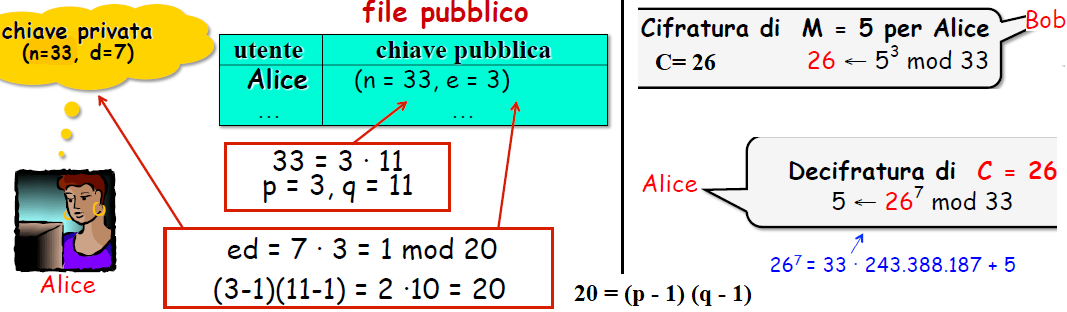
cifratura di M per Alice è:

**Schema della decifratura con RSA:**

1. Alice riceve il testo cifrato C da Bob e lei lo vuole decifrare
2. Alice esegue il seguente algoritmo che prende in input C e la chiave privata di Alice:

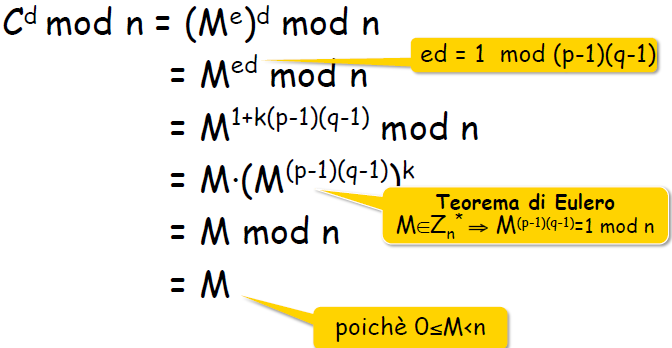
decifratura di C è:

Esempio di applicazione di RSA:



**NB:** Usare valori di *p*  e *q* abbastanza grandi altrimenti attraverso una ricerca esaustiva si può rompere RSA

**Correttezza decifratura con RSA**



Si nota che i valori *e* e *d* devono godere della proprietà che ogni volta che si cifra e decifra un messaggio M, questi due valori devono essere l’uno l’inverso l’altro. Questo vale perché *e, d* sono stati scelti in modo particolare.

**Efficienza delle computazioni**

RSA esegue le seguenti computazioni:

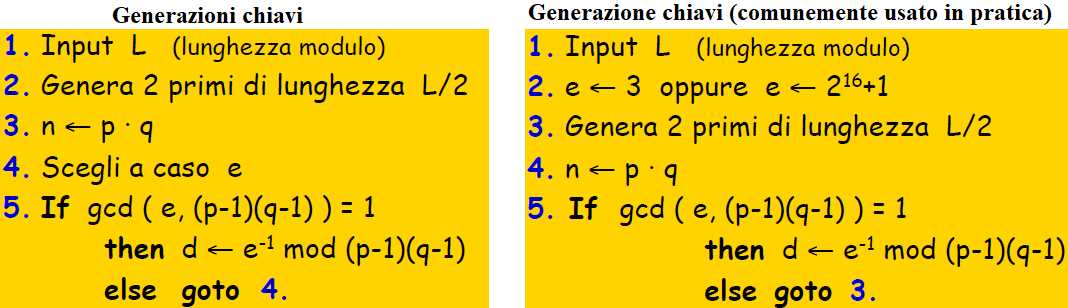
* Generazione dei numeri primi *p* e *q* (in genere di 1024 bit)
* Generazione di *e, d* = [Nota: è un inverso moltiplicativo]
* Elevazione a potenza modulare per cifratura e decifratura

**Generazione chiavi**

Prima si decide la lunghezza L del modulo: più modulo è lungo 🡪 maggiore sarò la sicurezza (non bisogna esagerare con la lunghezza altrimenti l’algoritmo si rallenta).

**Nota**: Se *n* = 2048 bit 🡪 *p* e *q* devono essere entrambi dei numeri primi di 1024 bit

Il passo 5 verifica se *e* è primo rispetto a (p-1) (q-1), perché se non è primo 🡪 non esiste



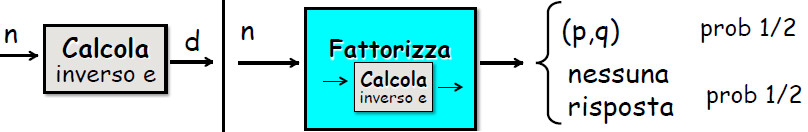
Da figura 2 si nota che l’esponente *e* viene scelto in modo fissato. Infatti, l’esponente *e* pubblico viene scelto in questo modo per **minimizzare** le operazioni per l’elevazione a potenza:

* (troppo piccolo, perciò si sceglie spesso l’altro)
* (Quello più usato: calcolo di richiede solo 16 quadrati ed una moltiplicazione

**Sicurezza generazione chiavi di RSA**

Per analizzare la sicurezza di RSA dobbiamo studiare quali sono i possibili attacchi su di esso. Il caso banale è che l’attaccante conoscendo la chiave pubblica vuole calcolare la chiave privata , *n* già lo conosce quindi **deve solo calcolare *d***. Per farlo basta calcolare:

1. Quindi, si dovrebbe fattorizzare *n* in modo da ottenere *p* e *q*
2. L’attaccante dovrebbe calcolare
3. Infine, l’attaccante dovrebbe calcolare *d* come indicato nella formula precedente

Tutto ciò però richiede **tempo esponenziale** a meno che non si ricorre all’algoritmo **Las vegas** che fattorizza *n* con probabilità .

Tale algoritmo prende in input , calcola con probabilità oppure non dà risposte con probabilità .

L’algoritmo **Las Vegas** usato dall’attaccante per fattorizzare *n* funziona in questo modo: (vedi figura)

Fattorizza *n* 🡪 Computa *d*; Computa *d* 🡪 Fattorizza *n*. Quindi computare *d* **è equivalente** a fattorizzare *n*

Ciò significa che l’attaccante non riesce a calcolare *d* a meno che non riesce a fare la fattorizzazione di *n*.

Dato *n*, calcolare due primi . Per valori grandi di *n* è un problema ritenuto computazionalmente difficile e quindi la complessità dell’algoritmo è **esponenziale.**

Attualmente, gli algoritmi capaci di fattorizzare in modo efficiente numeri molto grandi, sono tre:

* ***Quadratic Sieve:*** questo è l’algoritmo più veloce per numeri con meno di 150 cifre decimali; mentre per un numero maggiore di cifre è chiamata *Multiple Polynomial Quadratic Sieve*.
* ***Number Field Sieve (NFS)***: l'algoritmo *Multiple Polynomial Quadratic* *Sieve* è più veloce per numeri piccoli (da 110 a 135 cifre decimali a secondo delle implementazioni).
* ***Elliptic Curve Method (ECM)***: questo metodo è stato molto usato per trovare fattori di numeri a 38 cifre, ma non più grandi. Per numeri più grandi, gli altri algoritmi sono più veloci.

**Sicurezza cifratura RSA**

L’attaccante conosce la chiave pubblica e il messaggio cifrato , vuole calcolare il messaggio M. Quindi, in questa situazione l’attaccante non vuole fattorizzare *n* e trovare la chiave privata.

Se l’attaccante potesse fattorizzare *n* allora potrebbe anche computare il messaggio M:

1. Fattorizza *n*
2. Computa
3. Computa *d* usando questa equazione:
4. Ricava M decifrando C

Questo significa che se riesce a fattorizzare *n*, può sia decifrare il messaggio e sia calcolare *d*.

**Nota: Non si sa** se calcolare il messaggio M dal testo cifrato C sia equivalente a fattorizzare.

**Differenza tra cifrari simmetrici e asimmetrici**

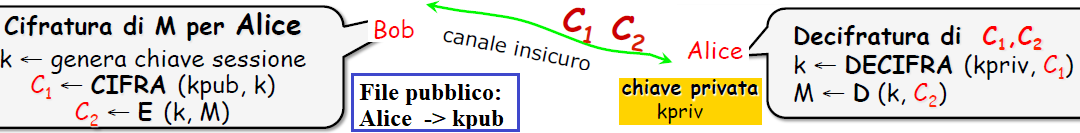
Grazie alla crittografia a chiave pubblica le chiavi private non vengono mai trasmesse ed è possibile implementare la firma digitale, viceversa la crittografia a chiave privata è molto più veloce ed è sufficiente in alcune situazioni, ad esempio quando si tratta di un singolo utente.

**Cifrari ibridi**

I cifrari ibridi utilizzano una chiave di sessione, cioè si cifra una chiave di sessione e mai un messaggio.

La chiave pubblica serve solo per cifrare con la chiave di sessione che viene generata sul momento e il messaggio M viene quindi **cifrato in due parti**. Nella seconda cifratura del messaggio (cioè ci sta la cifratura del messaggio fatta con la chiave di sessione *k.* Bob invia ad Alice lungo il canale insicuro

Per la **decifrazione** viene prima calcolato *k* avendo la chiave privata e la prima parte del cifrato e poi si calcola il contenuto del messaggio avendo in input il risultato precedente e la seconda parte del cifrato.



**Altri attacchi ad RSA**

Vi sono comunque altri attacchi non basati sul problema della fattorizzazione come: (vedi prossima pagina)

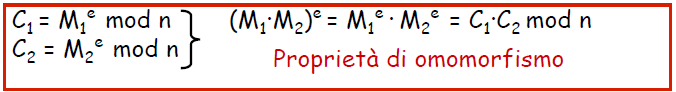
* **Chosen ciphertext attack**.
* **Common modulus attack**; (vedi come approfondimento)
* **Low exponent attack**. (vedi come approfondimento)

**Chosen ciphertext attack**

L’attaccante sceglie un testo cifrato di cui non conosce la decifrazione, dalla decifrazione di questo testo cifrato riesce a trovare la soluzione al vero obiettivo che è decifrare C.

Grazie alla **proprietà dell’omomorfismo** il sistema può essere rotto, si può costruire come prodotto di *C*

(moltiplicato) qualcosa che si conosce (*x* nel nostro caso).

**Obiettivo:** decifrare **C** (=

L’attaccante compie i seguenti passi:

1. Scegli *x* a caso
2. Calcola da (attaccante sta calcolando la decifrazione di un altro testo cifrato diverso da C, ovvero . Quindi:
3. Avvia la fase di **decifrazione** e otteniamo la coppia
4. Usando la coppia calcola
5. Usa per calcolare M, cioè:

**Nota:**

**Common Modulus Attack**

Stesso modulo *n* per diverse chiavi pubbliche:

* Chiave di Alice: Chiave di Bob:

Stesso messaggio M inviato ai vari utenti:

* Cifratura per Alice: Cifratura per Bob:

È semplice risalire al messaggio M usando l’algoritmo di **Euclide-Esteso** per calcolare *x, y* tali che:

**Low Exponent Attack**

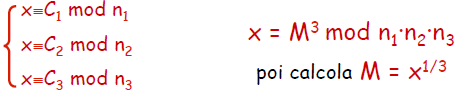
Stesso *e* per diverse chiavi pubbliche: supponiamo che

* Chiave di Alice: Chiave di Bob: ; Chiave di Eva:

Stesso messaggio M inviato ai vari utenti:

* Cifratura per Alice: Cifratura per Bob:
* Cifratura per Eva:

È semplice risalire ad M usando il **teorema cinese del resto** per calcolare la soluzione di:



**RSA: Attacchi ad implementazioni**

* **Timing Attack:** ricava i bit di *d* uno alla volta, analizzando il tempo richiesto per l’esponenziazione modulare (cioè per la decifratura)
* **Power Attack:** Ricava *d* analizzando la potenza consumata da una smartcard durante la decifratura. Cioè più moltiplicazioni si fanno e più energia si consuma

Le possibili contromisure sono:

* Ritardo costante (tutte le esponenziazioni richiedono lo stesso tempo. Dato un intervallo fisso e anche se l’implementazione ha finito di lavoro, bisogna aspettare la fine dell’intervallo per rilasciare l’output)
* Ritardo casuale nel calcolare un certo C (introduce “rumore” per confondere l’avversario)
* Blinding (moltiplica il cifrato per un numero casuale prima di decifrare).

# Analisi del consumo della banda utilizzata e scambio di dati

In questa sezione saranno illustrati e commentati i risultati dell’esperimento descritto a grandi linee nella metodologia in merito al consumo di banda e allo scambio di dati nell’esecuzione delle seguenti funzionalità quali:

Chiamata e videochiamata senza condivisione schermo;

Chiamata e videochiamata con condivisione schermo senza audio di sistema;

Chiamata e videochiamata con condivisione schermo e con audio di sistema.

In dettaglio, l’esperimento consisterà nell’eseguire una cattura della durata di 5 minuti mediante Wireshark. Al termine della cattura si eseguiranno i seguenti passi:

Si analizzerà, mediante la scheda “Conversation”, l’indirizzo IP a cui sarà associato il maggior numero di scambio di pacchetti, tra queste verrà considerata la conversazione che inizierà al tempo 0 e che avrà una durata maggiore di 300 secondi (vedi figura 5.1).

Trovato l’indirizzo IP, si utilizzerà il servizio “whois” per verificare se tale indirizzo fa parte del netblock dedicato alle varie piattaforme (vedi figura 5.2).

Una volta assicurato questo, dalla schermata “Conversation”, si andranno a considerare le informazioni relative alla dimensione totale relativa ai pacchetti scambiati tra le due parti (Bytes)

Per ottenere informazioni relative al consumo di banda, si utilizzerà lo strumento di Wireshark che permette di fare il plotting di grafici. A questo punto, si definirà un filtro per ottenere le informazioni relative ai bit/1s trasmessi utilizzando l’indirizzo IP sopra citato (vedi figura 5.3)

Si esporteranno i valori mostrati dal grafico in formato “.csv” al fine di poter calcolare una media con tali valori.

Questo procedimento verrà ripetuto per 5 volte per ciascuna funzionalità che si andrà ad esaminare in modo tale da ottenere dei risultati accurati e quindi non dettati da eventi aleatori.

Le seguenti figure fanno riferimento ad una cattura di Wireshark eseguita per analizzare la funzionalità di una videochiamata con condivisione schermo e audio di sistema offerta da Microsoft Teams.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 5.1 - Schermata “Conversation” di Wireshark



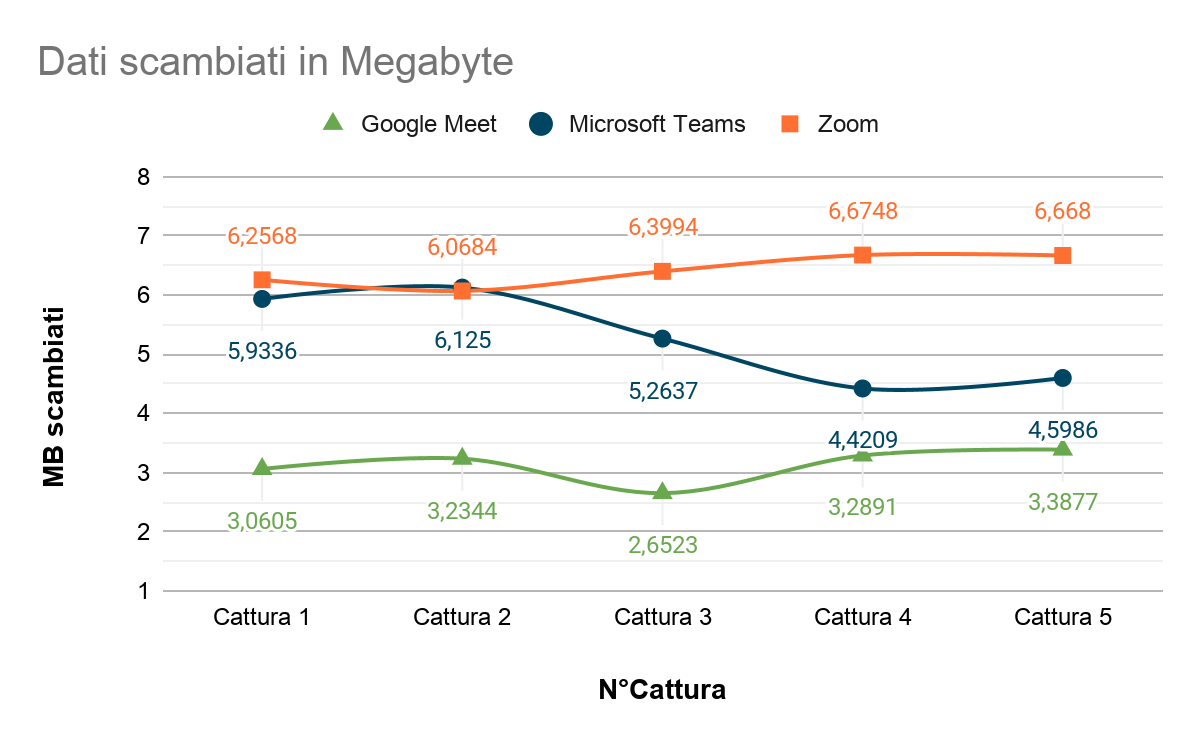
Figura 5.2 - Schermata “Grafici I/O” con filtro impostato

## Chiamata senza condivisione schermo

Dal grafico si può notare come Google Meet ha un consumo di banda minore rispetto ai due competitor, inoltre esso così come Zoom genera un consumo di banda costante rispetto a Microsoft Teams che molto probabilmente non appena possibile sfrutta tutta la banda a disposizione.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,1725 |
| Google Meet | 0,0840 |
| Microsoft Teams | 0,1425 |



Per quanto concerne la quantità di dati scambiati, anche in questo caso Google Meet risulta la piattaforma che ha scambiato il minor quantitativo di informazioni, mantenendo la stessa qualità percepibile[[1]](#footnote-1) degli altri competitor.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

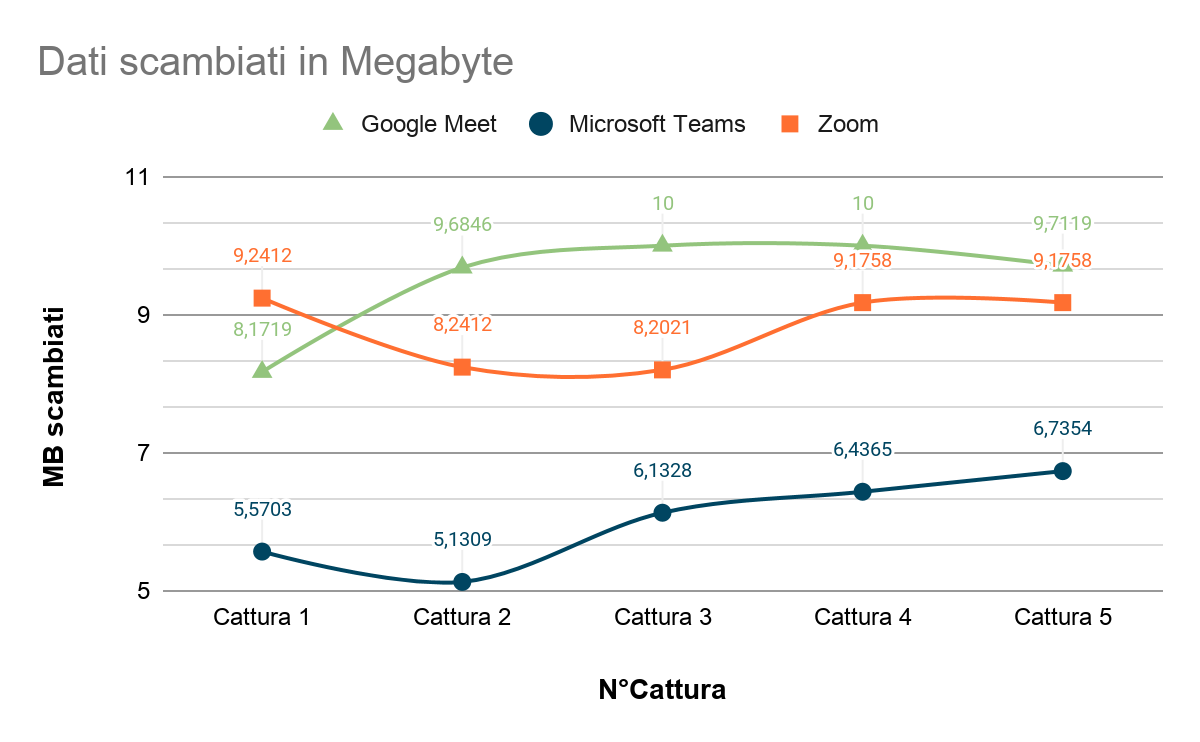
|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 6,4135 |
| Google Meet | 3,1248 |
| Microsoft Teams | 5,2684 |

## Chiamata con condivisione schermo senza audio di sistema

A simile risoluzione percepita, la piattaforma Microsoft Teams risulta utilizzare meno banda rispetto ai competitor. L’unica considerazione che può essere fatta è che guardando il grafico si può notare che Microsoft Teams a partire dalla seconda cattura comincia ad avere un andamento crescente, probabilmente il motivo è che non appena ha banda a disposizione ne sfrutta il necessario per garantire un migliore servizio.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,2347 |
| Google Meet | 0,2581 |
| Microsoft Teams | 0,0683 |



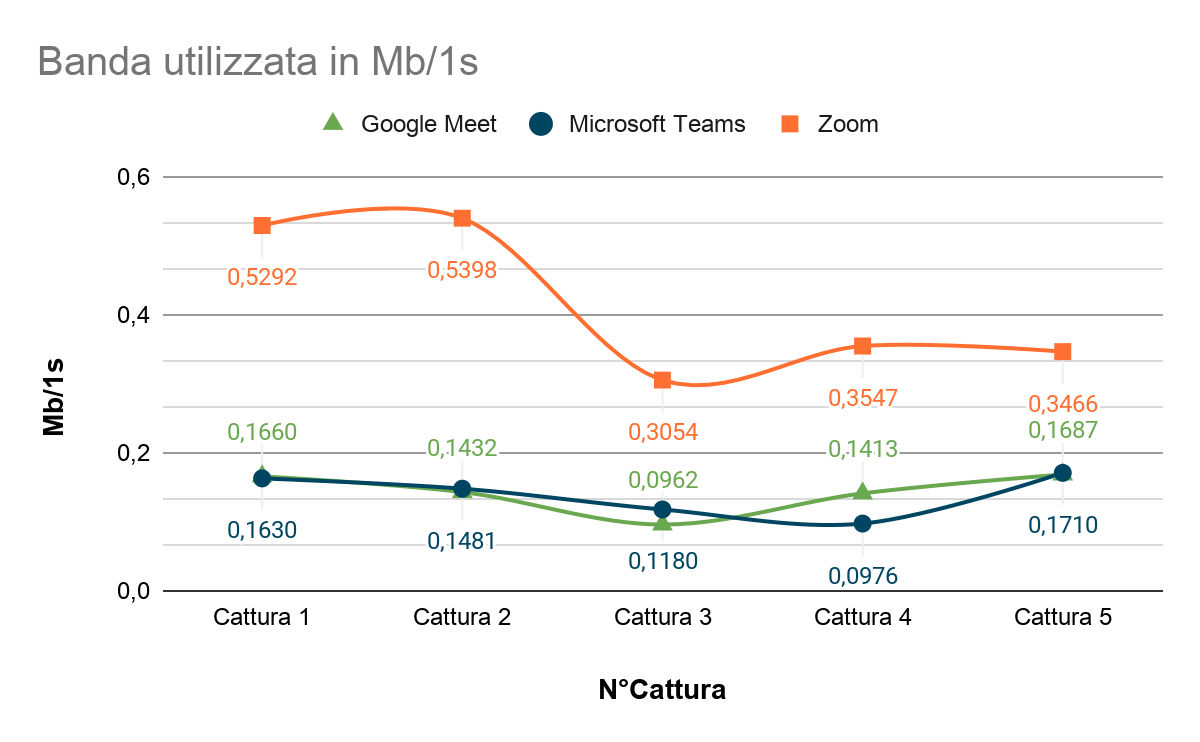
In questo esperimento si può notare come Microsoft Teams sia la piattaforma che scambi il minor quantitativo di dati.

Le osservazioni sono le stesse di quelle riportate per la banda, si può notare che Microsoft Teams a partire dalla seconda cattura tende ad avere un andamento crescente che lo potrebbe portare a uno scambio maggiore di dati rispetto allo stesso Zoom.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 8,9579 |
| Google Meet | 9,5137 |
| Microsoft Teams | 6,0011 |

## Chiamata con condivisione schermo e audio di sistema

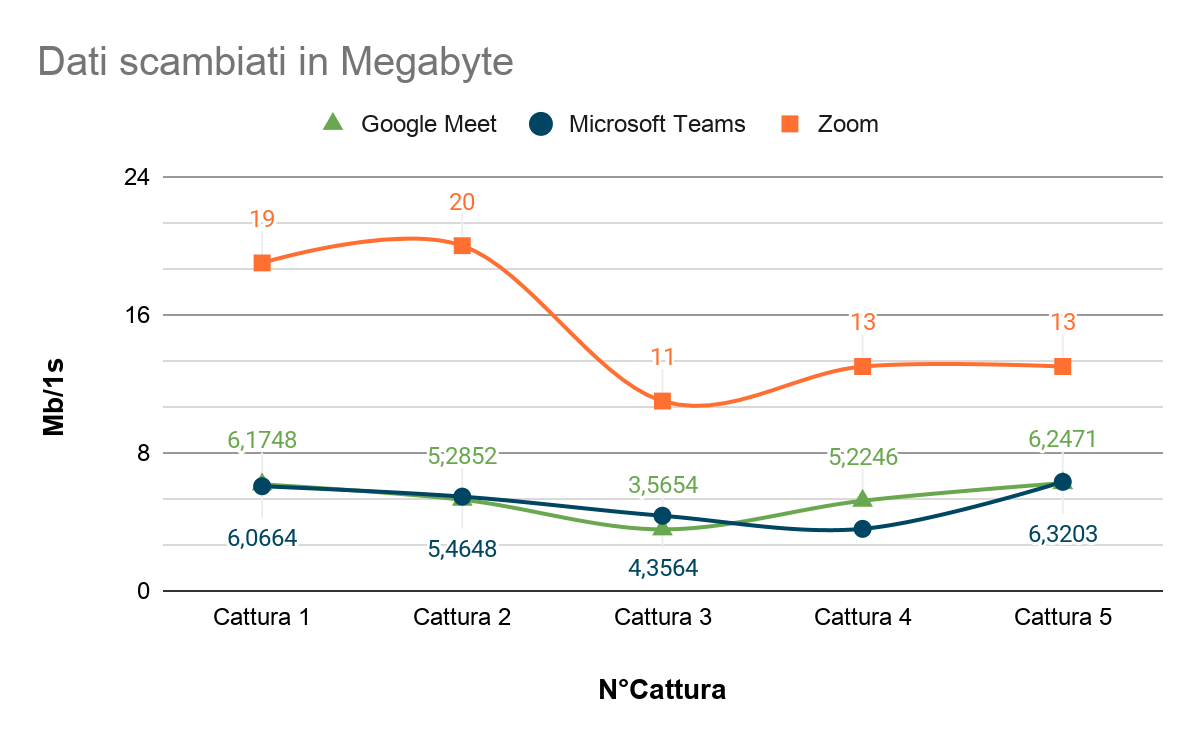


In questo esperimento Google Meet e Microsoft Teams risultano consumare in media una banda simile, rispettivamente 0.1430 e 0.1395, quindi, essendo che nessuna delle due prevale si considereranno ai fini dell’analisi gli altri fattori come RAM e CPU utilizzata.

Come si può notare dal grafico, Zoom dopo due catture comincia ad utilizzare meno banda attestandosi in media sullo 0.32, si rilascia quindi, a sviluppi futuri di effettuare più catture in modo da valutare con maggiore dettaglio il risultato ottenuto.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,4151 |
| Google Meet | 0,1430 |
| Microsoft Teams | 0,1395 |



Come nel caso della banda utilizzata, Google Meet e Microsoft Teams scambiano in media la stessa quantità di dati, in particolar modo, il primo in media scambia 5,2994 MB di dati e il secondo 5,1591 MB.

Anche in questo caso, essendo che nessuna delle due prevale si considereranno gli altri fattori selezionati per delineare la piattaforma migliore per questo tipo di esperimento.

L’osservazione in merito all’andamento per lo scambio di dati di Zoom è analogo a quello effettuato per la banda utilizzata.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 15,20 |
| Google Meet | 5,2994 |
| Microsoft Teams | 5,1611 |

## Videochiamata senza condivisione schermo

In questo caso Google Meet ha la meglio sulle altre due piattaforme, oltre ad avere un consumo di banda inferiore, nei 5 esperimenti ha avuto un consumo di banda costante, rispetto a Microsoft Teams che ha avuto picchi abbastanza considerevoli.

Occorre specificare che più volte è stato ripetuto questo esperimento, in particolare 2, per un totale di 10 catture, ma in tutti i casi Zoom e Microsoft Teams hanno presentato outlier dovuti molto probabilmente all’adattamento della banda in base a quella disponibile, altresì, bisogna specificare che i blocchi di 5 catture sono state effettuate in giorni differenti, ma i risultati sono stati gli stessi e cioè consumo di banda elevato per il tipo di esperimento che si sta analizzando e non costante nelle cinque catture.

Siccome questo comportamento anomalo si verifica soltanto con questa tipologia di esperimento si lascia agli sviluppi futuri il test con almeno 20 catture per studiarne maggiormente l’andamento.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

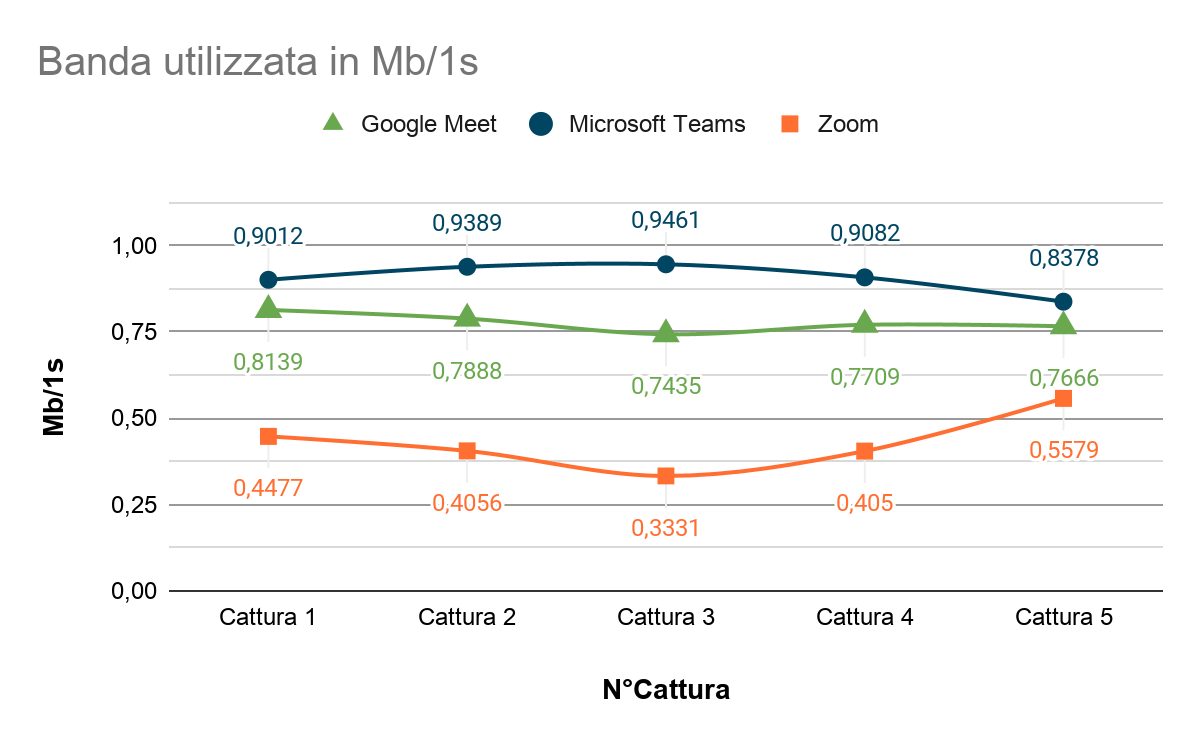
|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,7620 |
| Google Meet | 0,5906 |
| Microsoft Teams | 1,63996 |

Per i dati scambiati, Google Meet prevale rispetto alle altre due piattaforme a motivo del numero basso di informazioni scambiate. Come già detto precedentemente, nell’analisi della banda utilizzata, gli esperimenti di Microsoft Teams e quelli di Zoom sono stati ripetuti due volte ottenendo comunque risultati poco costanti e quindi con molti outlier.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 156,2 |
| Google Meet | 21,6 |
| Microsoft Teams | 67,4 |

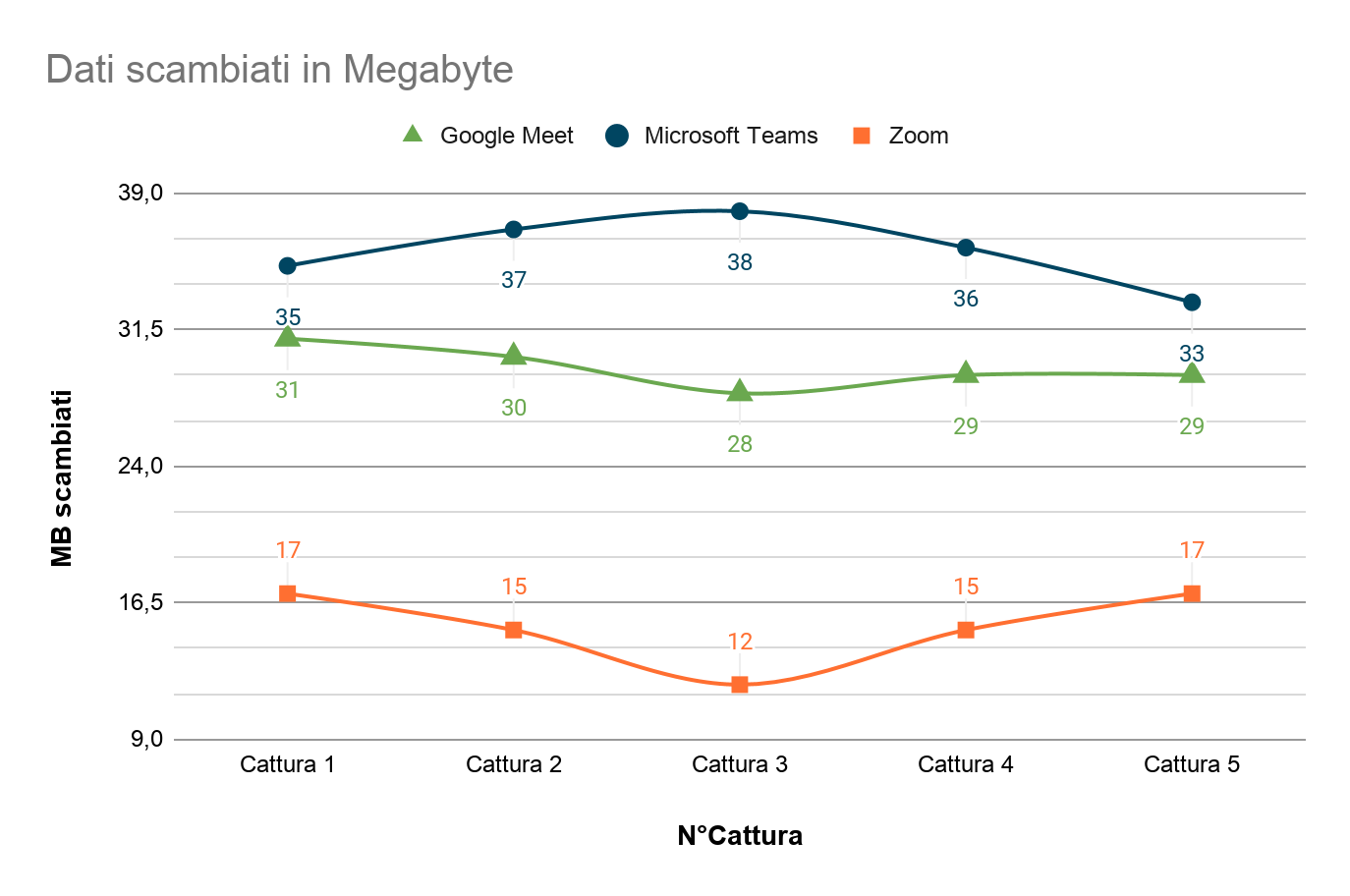
## Videochiamata con condivisione schermo senza audio di sistema



In questo caso la piattaforma zoom riesce, a simili livelli di qualità percepiti ad utilizzare una quantità di banda inferiore rispetto alle altre due piattaforme.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,4299 |
| Google Meet | 0,7767 |
| Microsoft Teams | 0,9064 |

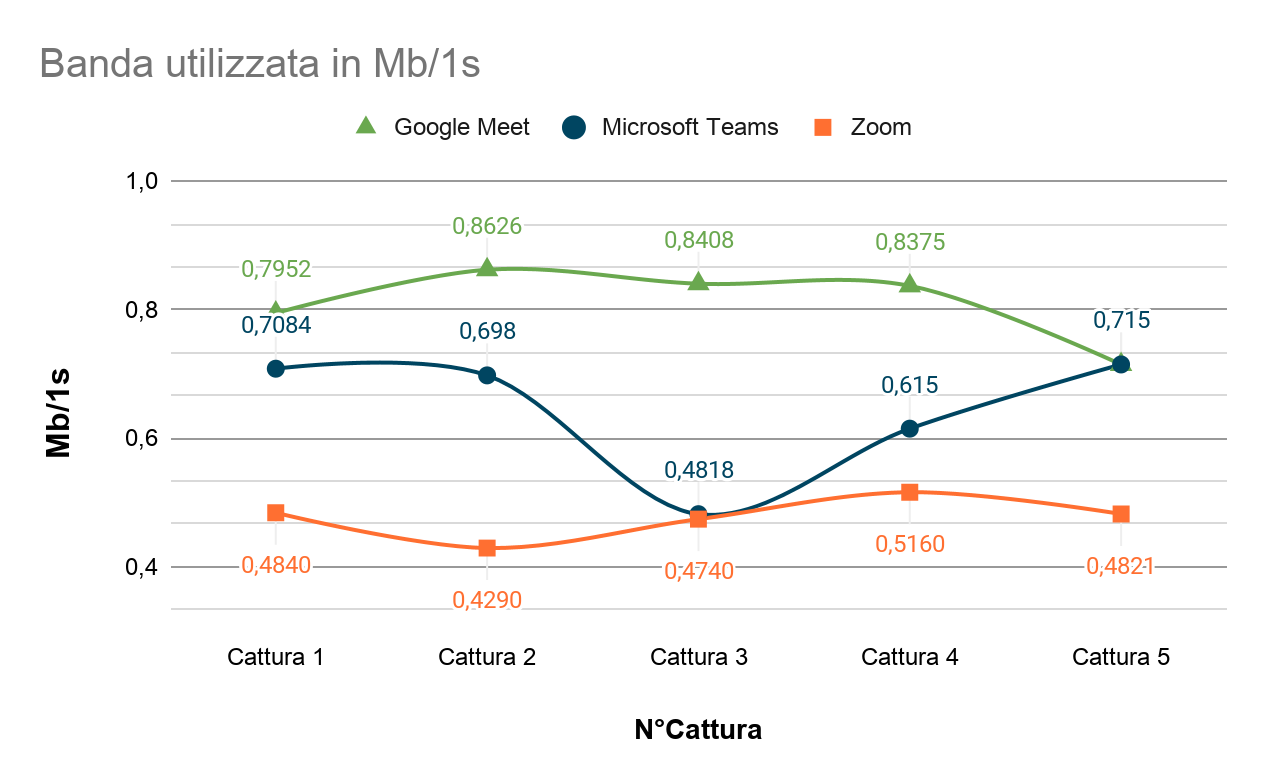


Per quanto riguarda i dati scambiati Zoom ha la meglio su Google Meet e Microsoft Teams con una media di dati scambiati pari a 15,2.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 15,2 |
| Google Meet | 29,4 |
| Microsoft Teams | 35,8 |

## Videochiamata con condivisione schermo e con audio di sistema

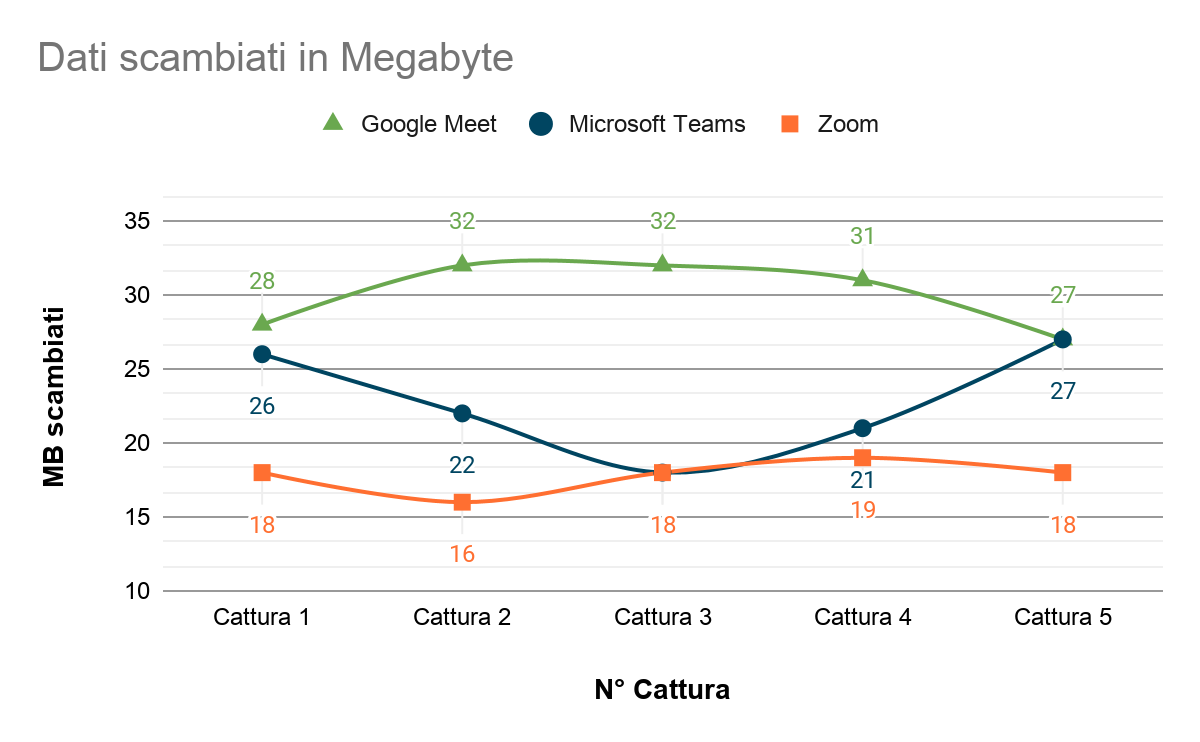


In tal caso si può notare come Zoom sia la piattaforma che consuma meno banda rispetto alle concorrenti. L’appunto in tal caso è che Microsoft Teams nella cattura 3 genera un outlier che andrà a inficiare poi nel calcolo della media finale.

Di seguito viene riportata la media dei Mbit/1s per ciascuna piattaforma:

|  |  |
| --- | --- |
| **Piattaforma** | **Mbit per secondo** |
| Zoom | 0,4770 |
| Google Meet | 0,8102 |
| Microsoft Teams | 0,6436 |

Si può notare come Zoom è la piattaforma che consuma il minor quantitativo di banda, seguito subito da Microsoft Teams.



In tal caso, Zoom risulta essere la piattaforma che scambia il minor numero di MB, subito seguito da Microsoft Teams che si distacca per solo 5 MB.

Di seguito viene mostrata la media dei dati scambiati dalle tre piattaforme nelle cinque catture effettuate:

|  |  |
| --- | --- |
| **Piattaforma** | **MB dati scambiati** |
| Zoom | 17,8 |
| Google Meet | 30,0 |
| Microsoft Teams | 22,8 |

1. Si è scelto di utilizzare l’adattamento della banda in base a quella disponibile senza ad andare a settare i parametri di qualità. [↑](#footnote-ref-1)