UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"Generazione di chiavi segrete con autoencoder"

Relatore: Prof. Stefano Tomasin
Correlatore: Dott. Francesco Ardizzon

Laureando: Pietro Biasetton

ANNO ACCADEMICO 2021 – 2022

Data di laurea 20 LUGLIO 2022

ABSTRACT

La tesi studia metodi di generazione di chiavi segrete tra due terminali ottenute dalle osservazioni del canale di comunicazione (ad esempio la risposta impulsiva del canale) fatte dai due terminali. Tipicamente le due osservazioni dello stesso canale non coincidono, a causa del rumore di stima e di possibili variazioni del canale se le osservazioni avvengono in tempi diversi. La tesi studia pertanto l'estrazione di sequenze di bit dai canali utilizzando tecniche di apprendimento automatico (machine learning) che realizzazione delle elaborazioni sulle osservazioni che includono anche la quantizzazione. Diverse soluzioni sono state studiate e confrontate mediante simulazione.

Indice

Introduzione **1**
Generazione di chiavi da canale radiomobile **3**
1.1 Algoritmi crittografici principali 3
1.2 Crittografia in sistemi di comunicazione radio a basso costo 4
1.3 Principi 4
1.4 Procedura per la generazione di chiavi private 5
Autoencoder **7**
2.1 Architettura 7
2.2 Parametri 8
2.3 Funzione di costo 9
2.4 Applicazioni degli autoencoder 9
Generazione di chiavi **11**
3.1 Quantizzatore non uniforme e autoencoder 11
3.2 Addestramento di un autoencoder 12
3.3 Quantizzatore differenziabile 12
3.4 Generazione di chiavi con autoencoder 14
3.5 Procedura di attacco 15
3.6 Confronto con lo stato dell'arte 16
Risultati delle simulazioni **19**
4.1 Dataset 19
4.2 Addestramento 20
4.3 Modelli utilizzati 20
4.4 Probabilità di errore 24
4.5 Informazione mutua tra le chiavi generate 27
4.6 Probabilità di errore di un attaccante 32
Conclusioni **37**
Bibliografia **39**

Introduzione

Il tema della sicurezza informatica è sempre più rilevante al giorno d'oggi. Il problema è di particolare interesse soprattutto nell'ambito della telecomunicazione radiomobile dove la natura dei canali di comunicazione permette a tutti gli utenti aventi accesso al canale entro un certo raggio di ricevere e possibilmente decodificare un segnale rendendo così la comunicazione vulnerabile ad attacchi esterni. Nell'ambito di reti contenenti dispositivi a basso costo il problema della sicurezza cresce di interesse in quanto non è applicabile la metodologia di crittografia dei messaggi più comunemente usata in quanto costosa dal punto di vista computazionale e delle infrastrutture dedicate. Un elemento fondamentale per il funzionamento di molti algoritmi di sicurezza è la disponibilità di chiavi segrete tra due dispositivi che vogliono comunicare, ovvero una sequenza di bit nota a entrambi ma ignota all'attaccante. Una soluzione è quella di generare lo stesso numero casuale su entrambi i terminali coinvolti in modo che questo costituisca la chiave crittografica utilizzata per codificare e decodificare i messaggi. La tesi vuole verificare se è possibile utilizzare delle reti neurali opportune per la generazione della chiave ai due terminali di un canale di telecomunicazione. Nello specifico nel Capitolo 1 si presenteranno le metodologie di codifica classiche e la metodologia più largamente usata nei sistemi di comunicazione radiomobile a basso costo, nel Capitolo 2 si darà una visione d'insieme dell'autoencoder ossia la rete neurale presa in considerazione per la risoluzione del problema che si concilierà poi come descritto nel Capitolo 3 con una operazione di quantizzazione al fine di ottenere delle chiavi private. Nel Capitolo 3 inoltre si descriverà la soluzione proposta per la generazione di chiavi con autoencoder, confrontandola con lo stato dell'arte. Verranno poi descritte le simulazioni effettuate e presentati i risultati nel Capitolo 4. Infine, nel Capitolo 5 si discuteranno le conclusioni della presente tesi.

Capitolo 1

Generazione di chiavi da canale radiomobile

Per fare fronte al requisito della confidenzialità del messaggio trasmesso in un canale si è sviluppata negli anni la crittografia ovvero la disciplina volta a rendere un messaggio incomprensibile a eventuali persone terze che non sono coinvolte nella comunicazione.

1.1 Algoritmi crittografici principali

Esistono due classi principali di algoritmi che si basano sull'utilizzo di chiavi: la crittografia simmetrica o a chiave privata e la crittografia asimmetrica o a chiave pubblica

Nella crittografia simmetrica la chiave è una sequenza di bit nota esclusivamente a mittente e destinatario. Gli algoritmi simmetrici permettono al mittente di criptare un messaggio con la stessa chiave che utilizzerà il destinatario per decriptarlo. Malgrado la maggior parte di tali algoritmi funzionino molto velocemente, questi presentano la criticità di dover far fronte alla distribuzione delle chiavi a tutte le parti che vogliamo possano accedere al messaggio, criticità che si amplifica al crescere del numero di utenti a cui vogliamo spedire il messaggio

Per far fronte ai problemi della crittografia simmetrica si vuole distinguere la chiave per criptare da quella per decriptare, nascono quindi gli algoritmi asimmetrici che consistono nell'introduzione di una seconda chiave. Negli algoritmi il destinatario comunicherà al mittente la propria chiave pubblica che verrà usata per criptare il messaggio, così il messaggio potrà essere decrittato solo dal destinatario in possesso della sua chiave privata. La chiave pubblica risulta quindi nota a tutti mentre la chiave privata è nota solo al destinatario del messaggio. Se però questi algoritmi risolvono tutte le criticità degli algoritmi simmetrici, la loro implementazione è poco efficiente soprattutto per lunghezze delle chiavi maggiori di 10 byte. Inoltre, questo metodo richiede una infrastruttura per la trasmissione sicura delle chiavi pubbliche.

La soluzione più largamente adottata è quella di integrare le due tipologie di algoritmi in quella che viene definita crittografia mista. Si usa la crittografia a chiave pubblica per scambiare in modo sicuro chiavi simmetriche che verranno usate per crittografare il flusso di dati

1.2 Crittografia in sistemi di comunicazione radio a basso costo

La natura di un canale radiomobile permette a tutti gli utenti entro un certo raggio di ricevere e possibilmente decodificare un segnale rendendo questo tipo di comunicazione particolarmente vulnerabile ad eventuali attacchi esterni. È di fondamentale importanza, dunque, l'utilizzo di un algoritmo di crittografia dei messaggi. Se però oltre alla natura radiomobile del canale consideriamo anche che i dispositivi coinvolti nella comunicazione siano a basso costo gli algoritmi descritti nella sezione precedente non sono più applicabili a causa della loro elevata richiesta di potenza computazionale. Nasce quindi la necessità di una tecnica che basandosi sulla natura del canale e sugli algoritmi precedentemente presentati permetta di effettuare una codifica dei messaggi. Una soluzione consiste nella generazione casuale della chiave simmetrica per i due utenti legittimi coinvolti nella comunicazione.

1.3 Principi

La generazione delle chiavi private si fonda su tre principi:

- La variazione temporale del canale, da cui la chiave viene generata, che garantisce la casualità di scelta delle chiavi. La componente randomica in un canale può infatti essere introdotta dalle riflessioni nell'ambiente e dalle sue variazioni. Più velocemente il canale cambia nel tempo e nello spazio (a ragione della posizione del ricevitore) più la sua componente casuale cresce. Questa componente del canale è una fonte ideale per la generazione casuale delle chiavi. La generazione, infatti è importante che sia casuale in quanto una chiave non generata casualmente diminuirebbe significativamente lo spazio di ricerca di un eventuale attacco di forza bruta rendendo la crittografia più vulnerabile.
- La reciprocità del canale, che indica il fatto che il canale visto dall'utente A quando trasmette l'utente B sia lo stesso che vede l'utente B quando trasmette l'utente A. Le ragioni di una possibile non reciprocità sono legate al fatto che i due dispositivi A e B utilizzano componenti diverse per la ricezione e trasmissione che influenzano diversamente il canale.
- La decorrelazione spaziale, che implica che un attaccante posto a una distanza maggiore di mezza lunghezza d'onda del segnale da uno dei due utenti coinvolti nella comunicazione vedrà un canale non correlato con quello legittimo, pertanto, una chiave generata a partire dal canale legittimo sarà differente da una chiave generata dall'attaccante. Questo principio è stato verificato empiricamente.

1.4 Procedura per la generazione di chiavi private

La procedura per la generazione casuale di chiavi private consiste di quattro fasi: il sondaggio del canale, la quantizzazione, la conciliazione delle informazioni e l'amplificazione della privacy.

- La fase di stima del canale è volta a misurare la componente randomica del canale da parte di entrambi gli utenti. Questi scambiano tra di loro uno per volta dei segnali pilota che serviranno a misurare i parametri utili alla generazione delle chiavi. La fase si conclude quando entrambi gli utenti hanno accumulato sufficienti misure. Le stime di canale da parte degli utenti devono essere fatte alternatamente in quanto si suppone che il sistema in questione sia half-duplex e quindi i dispositivi possono trovarsi o in uno stato di trasmissione o in uno stato di ricezione ma non in entrambi contemporaneamente.
- La fase di quantizzazione mappa stime analogiche del canale in valori binari. Un quantizzatore è definito da due fattori: il numero di livelli di quantizzazione, ossia il numero di bit in cui ogni misurazione è convertita, e le soglie ossia i valori di riferimento per gli intervalli di quantizzazione.
- La fase di conciliazione delle informazioni ha lo scopo di far concordare gli utenti sulla stessa chiave in quanto dopo la fase di quantizzazione i bit solitamente non corrispondono. La conciliazione è portata a termine grazie all'utilizzo di protocolli, o codici di correzione degli errori, che implicano uno scambio di informazioni attraverso il canale pubblico.
- L'amplificazione della privacy cerca di mitigare il rilascio di informazioni sulla chiave allo attaccante avvenuto nella fase precedente. Questa fase può essere implementata da funzioni universali di hashing.

In questa tesi ci si concentra sulla fase della quantizzazione per estrarre il maggior numero di bit condivisi, riducendo lo scambio di informazioni nella fase di riconciliazione.

Capitolo 2

Autoencoder

Un *autoencoder* è una rete neurale introdotta nel 1986 da Rumelhart, Hinton e Williams in [1] con lo scopo di imparare a ricostruire i valori di ingresso con il minor errore possibile. Lo scopo è ottenere una rappresentazione informativa del dato di ingresso più compatta che potrà essere usata per varie applicazioni.

2.1 Architettura

Per meglio comprendere gli autoencoder bisogna fare riferimento alla loro architettura. Le componenti principali di un autoencoder sono tre: un encoder, un livello interno che descrive delle informazioni latenti e un decoder. Definiamo per gli autoencoder che seguono la trattazione le seguenti variabili:

- L: la dimensione dell'ingresso che corrisponde al numero di neuroni che compongono il livello di ingresso e il livello di output
- N: la dimensione dello spazio latente ossia il numero di neuroni che compongono il livello interno delle informazioni latenti

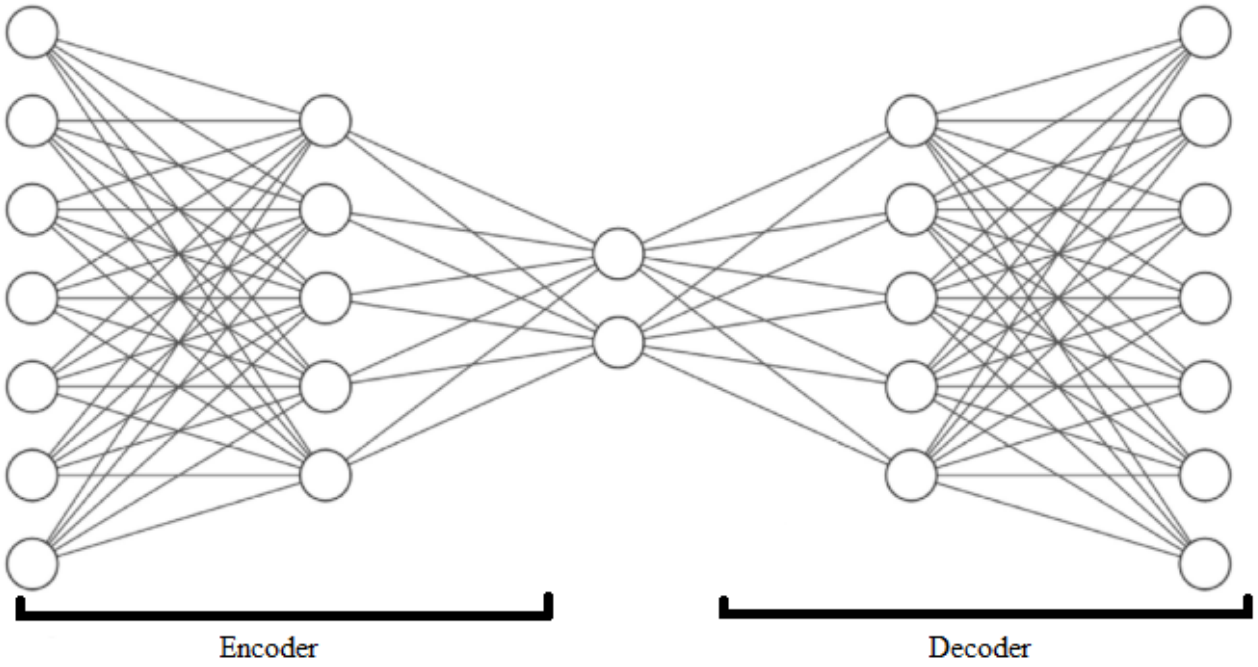

Figura 2.1: Esempio di struttura di un autoencoder a tre layer con L=7 e N=2

Generalmente l'encoder può essere rappresentato come una funzione g che dipenderà da alcuni parametri. Definiamo con h_i l'uscita dell'encoder in risposta all'ingresso x_i

$$h_i = g(x_i). \tag{1}$$

L'uscita dell'encoder h_i corrisponde al livello interno che fornisce una descrizione delle informazioni latenti. Il decoder può essere scritto come una funzione f delle informazioni latenti. Definiamo con $\dot{x}_i$ l'uscita del decoder

$$\dot{x}_i = f(h_i) = f\big(g(x_i)\big) \tag{2}$$

La fase di allenamento dell'autoencoder consiste quindi nel trovare i parametri delle funzioni g e f in modo che minimizzino la funzione di costo come verrà esposto nel Paragrafo 2.3. Se l'autoencoder riesce a minimizzare la funzione al suo minimo globale il risultato non è di particolare interesse difatti non siamo interessati ad avere una copia esatta dell'ingresso ma una sua buona approssimazione e per fare questo si adottano alcune tecniche come applicare un collo di bottiglia che riduca la dimensione dell'uscita dell'encoder rispetto al suo ingresso. Nella pratica l'architettura dell'autoencoder a cui viene applicato un collo di bottiglia è simmetrica e il numero di neuroni che costituisce un livello decresce spostandosi verso il centro dove raggiungerà il numero minimo. Il decoder è infatti in grado di ricostruire una buona approssimazione dell'ingresso iniziale a partire da un numero di informazioni minore rispetto a quelle iniziali.

2.2 Parametri

Essendo l'autoencoder una rete neurale vanno definiti alcuni parametri per il suo funzionamento ovvero la funzione di attivazione e la funzione di perdita. All'interno di una rete neurale lo scopo di ciascun neurone è quello di ricevere in ingresso dei valori e fornire un valore in uscita che viene immesso nel livello successivo. Al momento della ricezione dei valori in ingresso il neurone combina questi ultimi nella loro somma pesata in base ai parametri della rete neurale, il risultato così ottenuto costituirà l'ingresso della funzione di attivazione, che ha lo scopo di verificare se i segnali di ingresso sono uno stimolo sufficientemente grande per propagare l'attività all'interno della rete. Definiamo con y il valore di uscita del neurone, x_m il valore di ingresso trasmesso dal neurone m del livello precedente a quello considerato, w_m il peso del collegamento tra il neurone considerato e il neurone m e ψ la funzione di attivazione, ovvero

$$y = \psi(\textstyle\sum_m w_m x_m). \tag{3}$$

Negli autoencoder le funzioni di attivazioni più largamente usate sono la funzione $ReLu$ $\varsigma(x)$ e la funzione sigmoide $\gamma(x)$:

$$\varsigma(x) = \max(0, x) \tag{4}$$

$$\gamma(x) = \frac{1}{1+e^{-x}} \tag{5}$$

La funzione di attivazione $ReLu$ è adatta ai casi in cui i valori di ingresso assumono un ampio intervallo di valori reali positivi mentre la funzione sigmoide è più adatta ai casi in cui i valori di ingresso assumono solo valori compresi tra 0 e 1.

2.3 Funzione di costo

Un aspetto di fondamentale importanza per la costruzione di una rete neurale è la definizione della funzione di costo o funzione di perdita. La fase di allenamento della rete neurale è infatti volta a scegliere i parametri della rete che minimizzino tale funzione. Negli autoencoder la funzione di costo deve misurare quanto grande è la differenza tra l'ingresso dell'encoder e l'uscita del decoder. La funzione più largamente usata negli autoencoder è quella dell'errore quadratico medio Γ, abbreviato come MSE (Mean Square Error),

$$\Gamma(x) = \frac{1}{L}\sum_{i=1}^{L}|x_i - \dot{\mathrm{x}}_i|^2 \tag{6}$$

La funzione, infatti, può essere utilizzata indipendentemente dalla funzione di attivazione scelta o da come vengono normalizzati i dati in ingresso.

2.4 Applicazioni degli autoencoder

Gli autoencoder ad oggi hanno un'ampia e diversificata gamma di utilizzi, di seguito ne presentiamo alcuni:

- La riduzione delle dimensioni, funzione possibile grazie all'architettura in cui è presente un collo di bottiglia, porta con sé dei benefici rispetto alla riduzione delle dimensioni attuata con altri algoritmi, difatti dal punto di vista computazionale l'autoencoder può gestire un numero molto più grande di dati più efficientemente in quanto il suo allenamento può essere fatto con una parte del totale dei dati.
- La classificazione dei dati, come ad esempio immagini, portata a termine con l'utilizzo di autoencoder sfrutta il livello interno di descrizione delle informazioni latenti e costituisce una alternativa più affidabile per ridurre il tempo di allenamento di molti ordini di grandezza pur incorrendo in una caduta minima nell'accuratezza rispetto ad altri algoritmi.

- Il riconoscimento di anomalie si basa sullo stesso meccanismo utilizzato per la classificazione dei dati: una volta che l'autoencoder è stato allenato sull'intero insieme di dati a disposizione si può infatti assumere generalmente che qualsiasi anomalia verrà identificata da un errore di corretta ricostruzione più alto rispetto alle immagini affini all'insieme su cui la rete è stata allenata.

Capitolo 3

Generazione di chiavi

Al fine di ottenere una metodologia per la generazione di chiavi a partire da un canale radiomobile che faccia uso di autoencoder è necessario che questo venga modificato al fine di ottenere un risultato che risulti quantizzato. Come mostrato nel Paragrafo 1.4 la chiave che deve essere generata deve infatti presentarsi come chiave binaria di dimensione ridotta a quella dell'ingresso.

3.1 Quantizzatore non uniforme e autoencoder

Consideriamo ora un quantizzatore non uniforme. Questo viene usato nel caso in cui il segnale da quantizzare non abbia una funzione di densità di probabilità uniforme e quindi un quantizzatore uniforme risulterebbe subottimale. Un quantizzatore non uniforme aumenterebbe l'errore di quantizzazione per valori meno probabili diminuendo quello per i valori più probabili guadagnandone complessivamente nel rapporto segnale rumore. Esistono più modi di implementare un quantizzatore non uniforme, uno di questi è quello di far seguire a una funzione di compressione un quantizzatore uniforme. Al termine del quantizzatore il risultato deve essere espanso attraverso la funzione inversa della funzione di compressione adottata, per ottenere una versione quantizzata del segnale di ingresso.

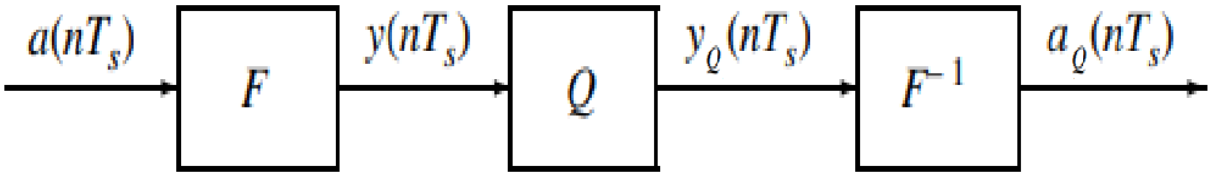

Figura 3.1: Uso della funzione di compressione nella struttura del quantizzatore non uniforme

Definendo a come l'ingresso, F come la funzione di compressione e Q come la funzione di quantizzazione otteniamo dunque:

$$a_Q = F^{-1}[Q[F(a)]] \tag{7}$$

Da questa serie di operazioni a cascata può essere visto chiaramente un parallelismo tra la funzione svolta dal quantizzatore uniforme e quella svolta dall'autoencoder. La funzione di compressione corrisponde a quella svolta dall'encoder e quella di decompressione a quella svolta dal decoder. Si può

concludere dunque che per ottenere una quantizzazione del risultato nell'autoencoder è necessario porre un quantizzatore uniforme in corrispondenza del livello interno che descrive le informazioni latenti.

3.2 Addestramento di un autoencoder

La funzione del quantizzatore è una funzione definita a tratti che presenta dunque punti di non derivabilità che rendono difficoltosa l'operazione di addestramento della rete neurale. L'operazione di addestramento di una rete neurale completamente connessa come l'autoencoder si divide in due fasi: la fase di propagazione in avanti e la fase di propagazione all'indietro. La fase di propagazione in avanti fa parte sia dell'addestramento ma anche dell'impiego della rete e consiste nell'utilizzare la rete per ottenere un primo risultato che sarà diverso da quello voluto. La propagazione all'indietro costituisce invece la fase fondamento dell'operazione di addestramento in cui la rete affina i suoi parametri in modo che vadano a minimizzare la funzione di perdita e raggiungere un risultato finale che sia in linea con quanto desiderato. Per realizzare la propagazione all'indietro si applica un metodo di ottimizzazione che prende il nome di discesa stocastica del gradiente. Per minimizzare la funzione di perdita, che consideriamo essere il Mean Square Error, si necessita di esprimerla in funzione di tutti i parametri della rete neurale. Per minimizzare la funzione se ne calcola il gradiente, ma essendo espressa in funzione delle funzioni di attivazione dei neuroni del livello precedente, a cascata il calcolo del gradiente verrà effettuato su tutte le funzioni di attivazione dei vari livelli. In tal modo si permette di identificare i parametri della rete, che portano a raggiungere il valore di minimo della funzione di perdita il più velocemente possibile. Il procedimento si ripete per ogni dato dell'insieme dei dati per l'addestramento della rete memorizzando per ciascuno di essi la modifica voluta dei parametri per poi farne una media e aggiornare i parametri. Nella pratica la discesa del gradiente così descritta richiede tuttavia tempi molto lunghi per i calcolatori, si divide pertanto l'insieme dei dati per l'addestramento in più gruppi su cui effettuare la discesa del gradiente, in questo modo si dà una buona approssimazione dei parametri. La tecnica descritta consiste nella discesa stocastica del gradiente.

3.3 Quantizzatore differenziabile

Per far fronte alla non derivabilità della funzione di quantizzazione uniforme è stato sviluppato il DSQ (Differentiable Soft Quantization). Il DSQ è una approssimazione continua e derivabile del quantizzatore uniforme, adatto dunque al processo di addestramento delle reti neurali. Per definire il

DSQ è necessario definire i parametri che ne permettono la realizzazione: l'intervallo di quantizzazione $[t, u]$ su cui opera e il numero di livelli L in cui si vuole dividere l'intervallo. A partire da questi dati, di seguito, possiamo indicare con $P_i, i \in \{0,1,\dots,2^b-1\}$ l'i-esimo intervallo di quantizzazione essendo b il numero di bit necessario a rappresentare L livelli, $L = 2^b$, e definire la dimensione di tali livelli denominata come passo di quantizzazione Δ:

$$\Delta = \frac{u-t}{2^b-1} \tag{8}$$

Poiché la derivata della funzione di quantizzazione uniforme corrisponde a zero in quasi tutti i punti del dominio è necessario introdurre una funzione derivabile che bene approssimi quella del quantizzatore uniforme. La funzione è definita come

$$\varphi(x) = s \tanh\big(k(x - m_i)\big) \;\; se\; x \in P_i \tag{9}$$

con

$$m_i = l + (i + 0.5)\Delta \tag{10}$$

$$s = \frac{1}{\tanh(0.5k\Delta)} \tag{11}$$

Il parametro s garantisce che le funzioni di tangente iperbolica delle funzioni ϕ di livelli adiacenti siano connesse garantendo quindi la continuità e la derivabilità della funzione. Il coefficiente k determina la forma della funzione, indicativamente più k sarà grande e più la funzione ϕ si avvicinerà alla funzione di quantizzazione uniforme. La funzione di quantizzazione DSQ è quindi la seguente:

$$Q(x) = \begin{cases} t & x < t \\ u & x > u \\ t + \Delta\left(i + \frac{\varphi(x)+1}{2}\right) & x \in P_i \end{cases} \tag{12}$$

Tramite la funzione DSQ è facilmente identificabile un sostituto derivabile per l'operazione di quantizzazione all'interno delle reti neurali, tuttavia, la qualità dell'approssimazione può influenzare di molto la qualità del modello su cui si opera. L'utilizzo di un quantizzatore DSQ è da questo punto di vista vantaggioso in quanto permette durante la fase di addestramento della rete di variare il parametro k della funzione φ per ottenere progressivamente una sempre più buona approssimazione del quantizzatore uniforme.

3.4 Generazione di chiavi con autoencoder

La generazione di chiavi private attraverso l'ausilio di un autoencoder si basa sugli stessi principi di variazione temporale del canale, reciprocità del canale e decorrelazione spaziale che stanno alla base della metodologia tradizionale come descritto al Paragrafo 1.3. A questi, inoltre, per semplificare le condizioni di partenza aggiungiamo l'ipotesi che i canali stimati alle due estremità del canale abbiano la stessa densità di probabilità. Siano quindi x e y le due stime con

$$p(x) = p(y). \tag{13}$$

Partendo da tali principi è possibile quindi definire il comportamento dell'autoencoder. Se nell'autoencoder classico la fase di addestramento consisteva nel minimizzare la funzione di perdita, che consideriamo essere il Mean Square Error tra i dati in ingresso e i dati in uscita, l'autoencoder considerato vuole minimizzare nella sua fase di addestramento il Mean Square Error tra y, i dati trasmessi all'altro autoencoder coinvolto nella comunicazione, e $g(x)$, i dati di uscita.

$$\Gamma = \frac{1}{L}\sum_{i=1}^{L}|y_i - \dot{x}_i|^2 = \frac{1}{L}\sum_{i=1}^{L}\left|y_i - f\left(g(x_i)\right)\right|^2 \tag{14}$$

In questo modo le due reti neurali che saranno uguali per entrambi gli utenti coinvolti nella comunicazione, grazie all'ipotesi aggiuntiva (13) genereranno al livello interno di descrizione delle informazioni latenti, al quale è applicata una funzione DSQ, due chiavi binarie, rispettivamente b_x e b_y a partire dai segnali x e y, di dimensione ridotta rispetto ai segnali d'ingresso e uguali tra loro.

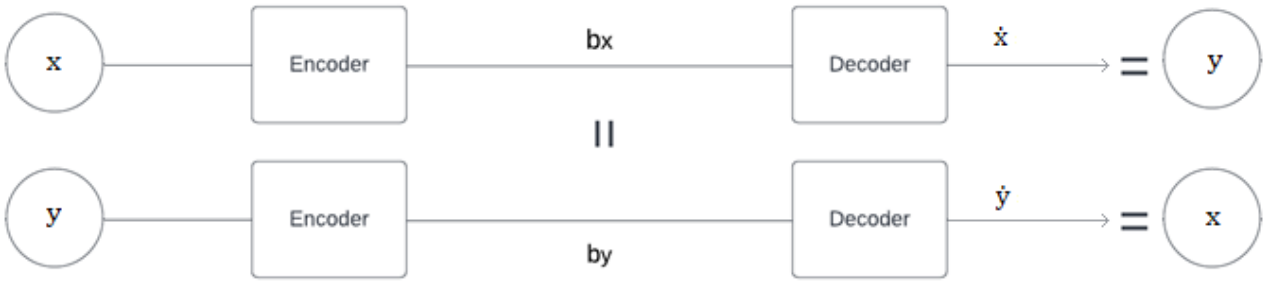

Figura 3.2: Struttura dei due autoencoder presi in considerazione ai terminali del canale di comunicazione

3.5 Procedura di attacco

Per verificare la sicurezza delle chiavi generate introduciamo la figura dell'attaccante ovvero un terzo dispositivo E estraneo alla comunicazione che vuole ricostruire a sua volta la chiave mediante anch'esso l'utilizzo di un autoencoder partendo però da un canale a lui visibile che risulta essere differente da quello visibile ai dispositivi A e B come è stato definito al Paragrafo 1.1. Per simulare il dispositivo E consideriamo un autoencoder che vuole minimizzare nella sua fase di addestramento la funzione di Mean Square Error tra x, uno dei segnali trasmessi tra i due autoencoder coinvolti nella comunicazione, e $g(z)$, i dati di uscita dell'autoencoder a partire da un ingresso z consistente nel campionamento del canale visibile all'attaccante.

$$\Gamma = \frac{1}{L}\sum_{i=1}^{L}\left|x_i - f\left(g(z_i)\right)\right|^2 \tag{15}$$

Questo metodo di generazione delle chiavi da parte dell'attaccante risulta essere valido se i vettori di ingresso per gli autoencoder x e z presentano tra di loro un certo grado di correlazione. Se così non fosse risulta essere maggiormente valido sostituire l'autoencoder dell'attaccante con una rete neurale costituita solo da un encoder, come rappresentato in Figura 3.3, che vuole minimizzare la funzione di perdita tra la sua uscita e la chiave generata dall'autoencoder di uno degli utenti coinvolti nella comunicazione

$$\Gamma = \frac{1}{M}\sum_{i=1}^{M}|b_{xi} - g(z_i)|^2 \tag{16}$$

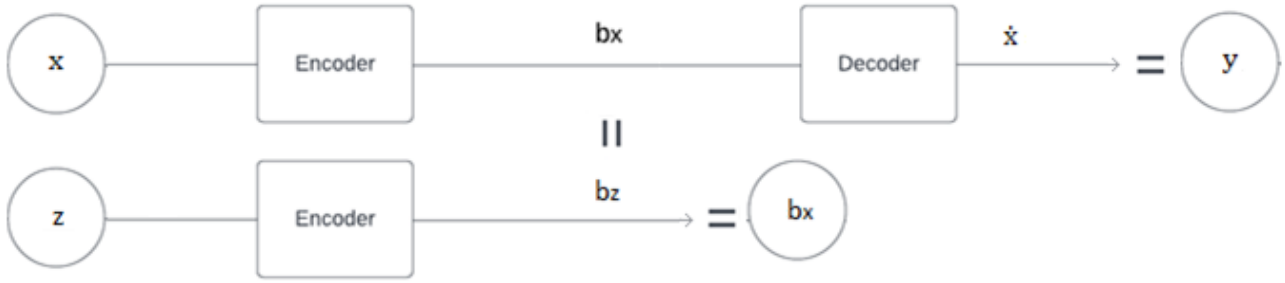

Figura 3.3: Struttura della rete neurale per l'attaccante in relazione all'autoencoder posseduto da un utente coinvolto nella comunicazione

3.6 Confronto con lo stato dell'arte

L'utilizzo di reti neurali deep-learning per la generazione di chiavi private nell'ambito della comunicazione radiomobile è ancora oggi soggetto di studio per valutarne vantaggi e svantaggi rispetto al campionamento del canale con successiva conciliazione delle informazioni. In [7] gli autori partono da un autoencoder come quello da noi considerato che tenta di ricostruire la stima del canale fatta da B a partire da quella fatta da A e viceversa portando lo scopo dell'autoencoder considerato ad essere quello di estrarre le informazioni in comune ad A e B. La funzione di attivazione utilizzata è tuttavia quella della Leaky ReLu definita, considerando ε un parametro della funzione compreso tra 0 e 1, come

$$\varsigma(x) = \begin{cases} x, x > 0 \\ \varepsilon x, x < 0 \end{cases} \tag{17}$$

L'autoencoder proposto fornisce solamente le informazioni in comune ad A e B da cui derivare una chiave, lasciando l'operazione di quantizzazione di tali informazioni per la generazione di una chiave a terzi. Dalle simulazioni effettuate sullo schema proposto viene messo in evidenza come non solo sia possibile ma come l'utilizzo di reti neurali sia migliore rispetto alla metodologia classica in termini di mutua informazione e di MSE della metodologia principale di analisi delle componenti.

Allo stesso modo in [8] gli autori costruiscono una rete volta ad imparare la reciprocità del canale che definiscono come *CRLNet* (Channel Reciprocity Learning Net). L'uso di questa rete non viene pensato tuttavia per sostituire la metodologia già discussa di campionamento, quantizzazione e conciliazione ma va a semplificare significativamente la fase di campionamento generando subito la componente in comune dei canali degli utenti coinvolti. La rete fa uso di due moduli per minimizzare il rumore presente nei segnali di ingresso, due encoder diversi per i diversi segnali e un decoder comune. La funzione di perdita utilizzata è una funzione ibrida che coinvolge i diversi moduli in diverse combinazioni dei loro parametri. La rete proposta risulta essere significativamente più efficace per altri metodi di estrazione delle componenti in comune ai canali visti da A e B compreso quello del semplice utilizzo di un autoencoder come viene utilizzato nell'esempio precedente e nel caso da noi preso in esame e di conseguenza dopo aver svolto la quantizzazione della componente estratta risulta avere anche una più bassa probabilità di errore nella generazione delle chiavi.

La seguente tesi si distingue pertanto dalla letteratura già presente in quanto vuole conciliare l'operazione di quantizzazione all'interno della rete neurale utilizzata per estrarre la componente in comune ai canali visti da A e B. Viene inoltre riportato un primo approccio alle modalità con cui un

utente terzo estraneo alla comunicazione possa generare anche egli la chiave concordata da A e B partendo ancora da un tentativo di ricostruzione delle componenti in comune.

Capitolo 4

Risultati delle simulazioni

A partire da quanto descritto finora abbiamo implementato in linguaggio Python, servendoci del framework PyTorch, diversi tipi di autoencoder per verificare la validità della soluzione teorizzata in precedenza.

4.1 Dataset

Per simulare i due segnali x e y così che fossero indipendenti ma avessero uguale densità di probabilità siamo partiti da un primo dataset iniziale α costituito da 40000 vettori di 16 elementi ciascuno a cui abbiamo sommato per andare a simulare il rumore dei due segnali x e y un vettore ciascuno, ω_1e ω_2, anche esso di 16 elementi consistenti in variabili aleatorie gaussiane a media nulla e deviazione standard 10^{-3}, ovvero

$$\begin{aligned} x &= \alpha + \omega_1, \\ y &= \alpha + \omega_2. \end{aligned} \tag{18}$$

Così si ottengono due dataset di 40000 elementi ciascuno dove i primi 30000 elementi dei due dataset vengono usati per l'addestramento della rete e i restanti 10000 per la fase di test. I vettori di α sono costituiti in un primo caso in vettori di 16 elementi dove ogni elemento consisteva in una variabile aleatoria gaussiana avente media nulla e varianza unitaria e in un secondo caso da variabili aleatorie gaussiane tra loro correlate attraverso la matrice di Fourier o matrice *DFT* (Discrete Fourier Trasnform). In particolare, in questo ultimo caso siamo partiti da una variabile gaussiana β a media nulla e varianza unitaria, che è stata moltiplicata per il vettore υ corrispondente alla seconda colonna della matrice *DFT* di dimensione 16x16.

$$\alpha = \beta \upsilon \, . \tag{19}$$

L'elemento della matrice di Fourier di dimensione Q alla riga r e alla colonna c è definito come:

$$e^{-\frac{2j\pi rc}{Q}} \quad r, c = 0, \ldots, Q-1. \tag{20}$$

4.2 Addestramento

Gli autoencoder sono addestrati con i dataset descritti nella sezione precedente, per minimizzare la funzione di perdita (14). Nell'addestramento sono stati usati i seguenti parametri per il quantizzatore differenziale e gli autoencoder:

- Il parametro k del quantizzatore è stato variato nell'intervallo $[1,100]$ incrementandolo di 1 ogni 15 epoche, per un totale di $\xi = 1500$ epoche di addestramento.
- I parametri u e t del quantizzatore sono stati fissati rispettivamente a -1 e 1.
- Il learning rate dell'autoencoder è stato fissato a 10^{-3}

Il numero di bit di quantizzazione utilizzati per ogni autoencoder è stato fatto variare da 1 a 10. Avendo input gaussiano, inoltre, tutti i livelli di tutte le reti neurali testate utilizzano la funzione tangente iperbolica come funzione di attivazione.

4.3 Modelli utilizzati

Abbiamo utilizzato tre diverse tipologie di autoencoder per verificare i risultati: autoencoder a un livello, autoencoder a tre livelli e autoencoder a cinque livelli. Di seguito nelle Figure 4.1, 4.2 e 4.3, riportiamo una rappresentazione grafica delle reti utilizzate dove in blu vengono colorati i neuroni di ingresso, in giallo il neurone di quantizzazione e in verde i neuroni di uscita. Abbiamo rappresentato soltanto i casi in cui $M = 1$ dove con M indichiamo il numero di bit di quantizzazione.

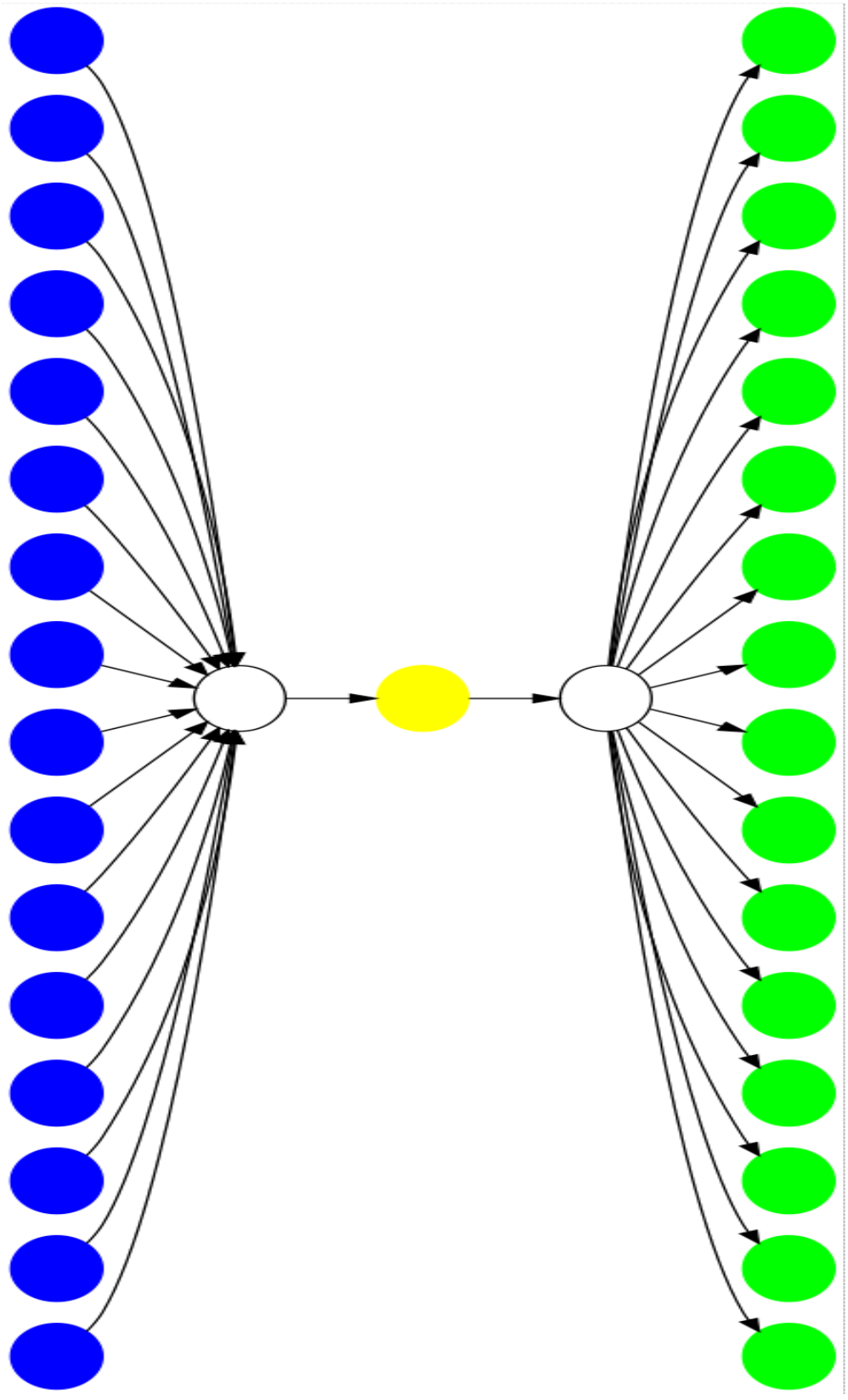

Figura 4.1: Topologia dell'autoencoder avente l'encoder con un layer con L=16 e M=N=1

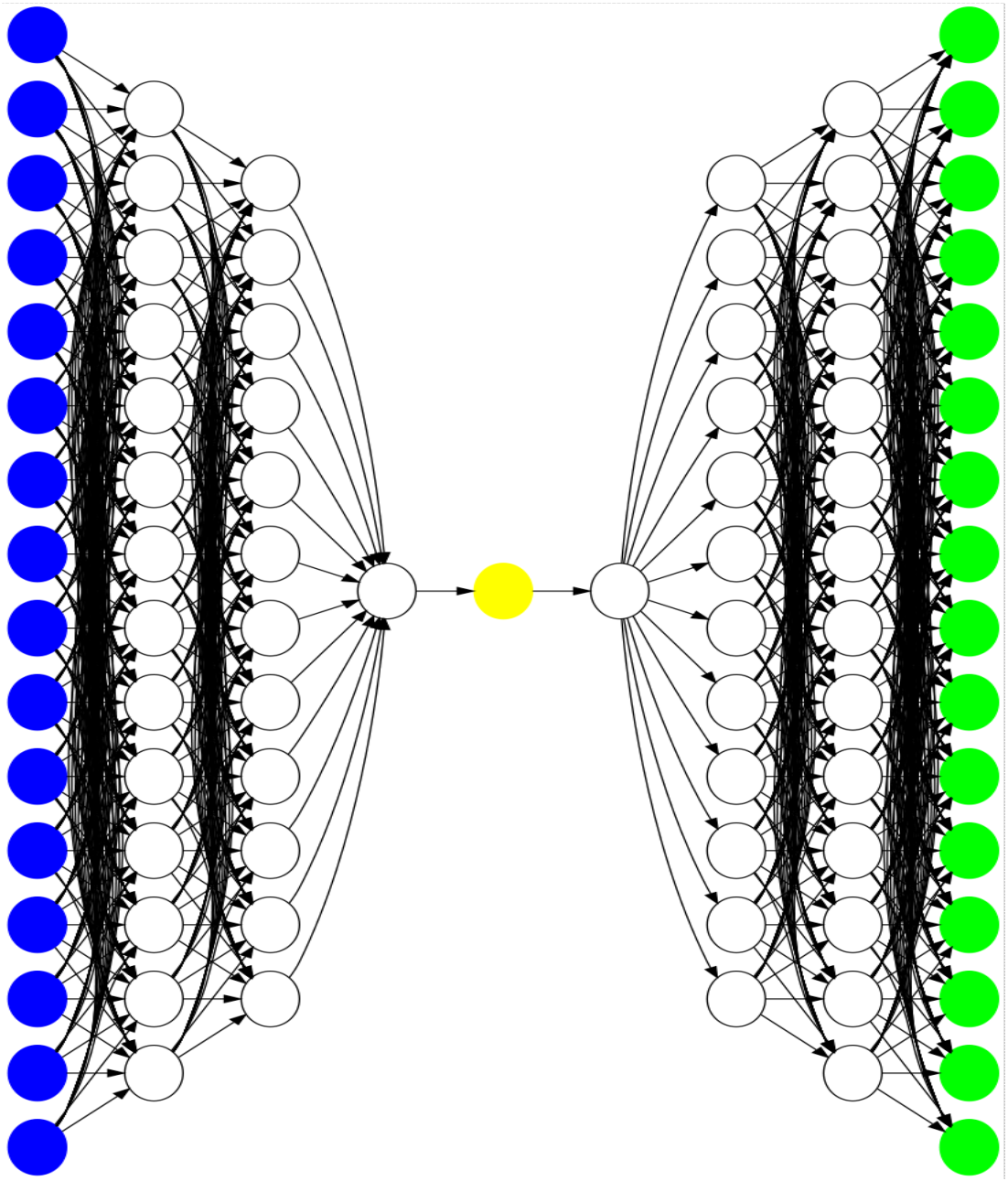

Figura 4.2: Topologia dell'autoencoder avente l'encoder con tre layer con L=16 e M=N=1

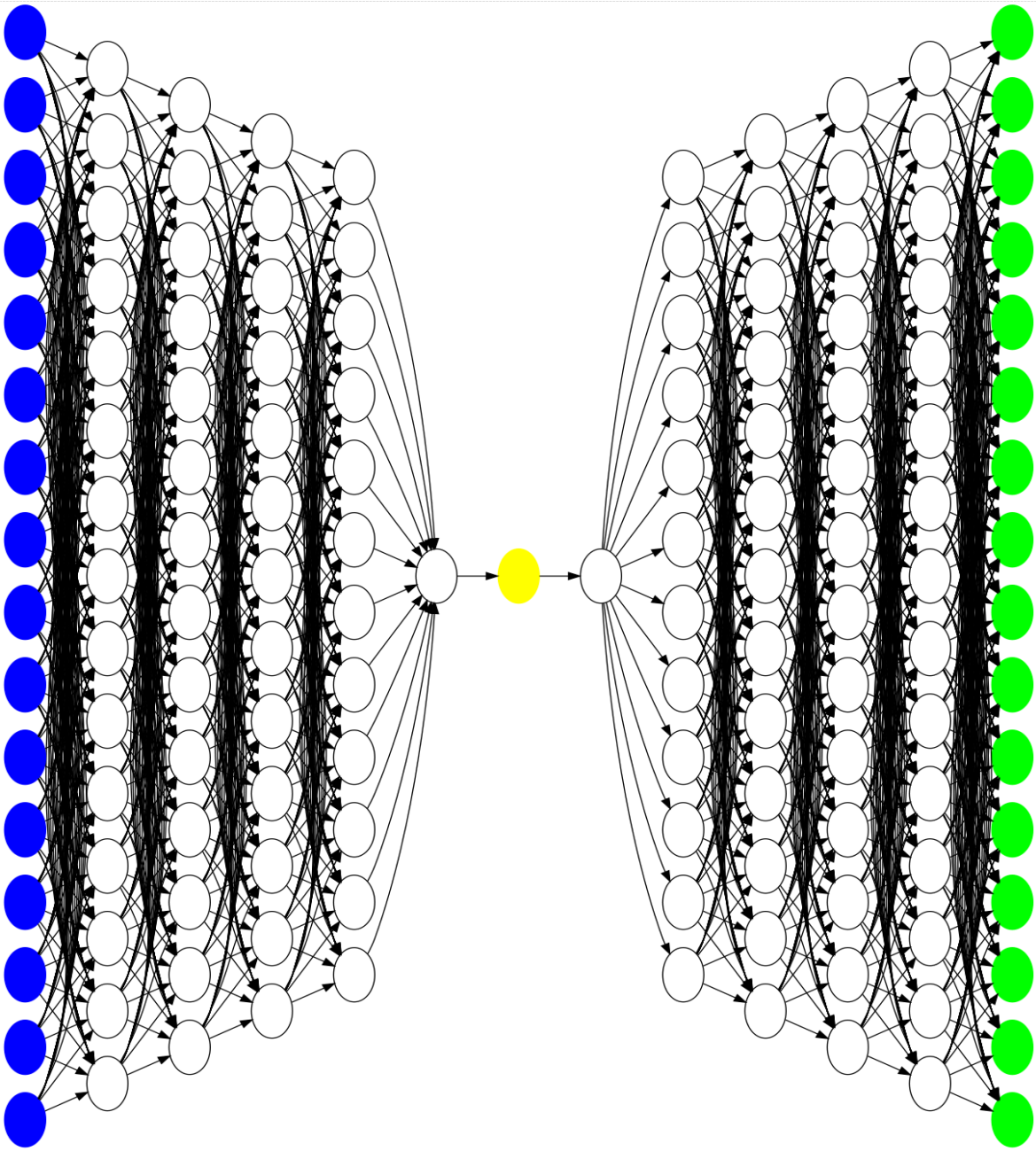

Figura 4.3: Topologia dell'autoencoder avente l'encoder con cinque layer con L=16 e M=N=1

4.4 Probabilità di errore

La metrica di valutazione di interesse è la probabilità di errore $P(E)$ nella corrispondenza delle chiavi generate dagli ingressi nei vari autoencoder. Indichiamo con n_{tot} il numero delle coppie di chiavi generate dai due autoencoder coinvolti nella comunicazione a partire dagli ingressi di x e y e con n_e il numero di casi in cui le chiavi componenti tali coppie non corrispondono. Calcoliamo quindi la probabilità di errore come

$$P(E) = \frac{n_e}{n_{tot}} \ . \tag{21}$$

I grafici seguenti mostrano la probabilità di errore degli autoencoder prima nel caso in cui α sia costituito da vettori di variabili aleatorie gaussiane normali indipendenti e poi da variabili aleatorie gaussiane correlate.

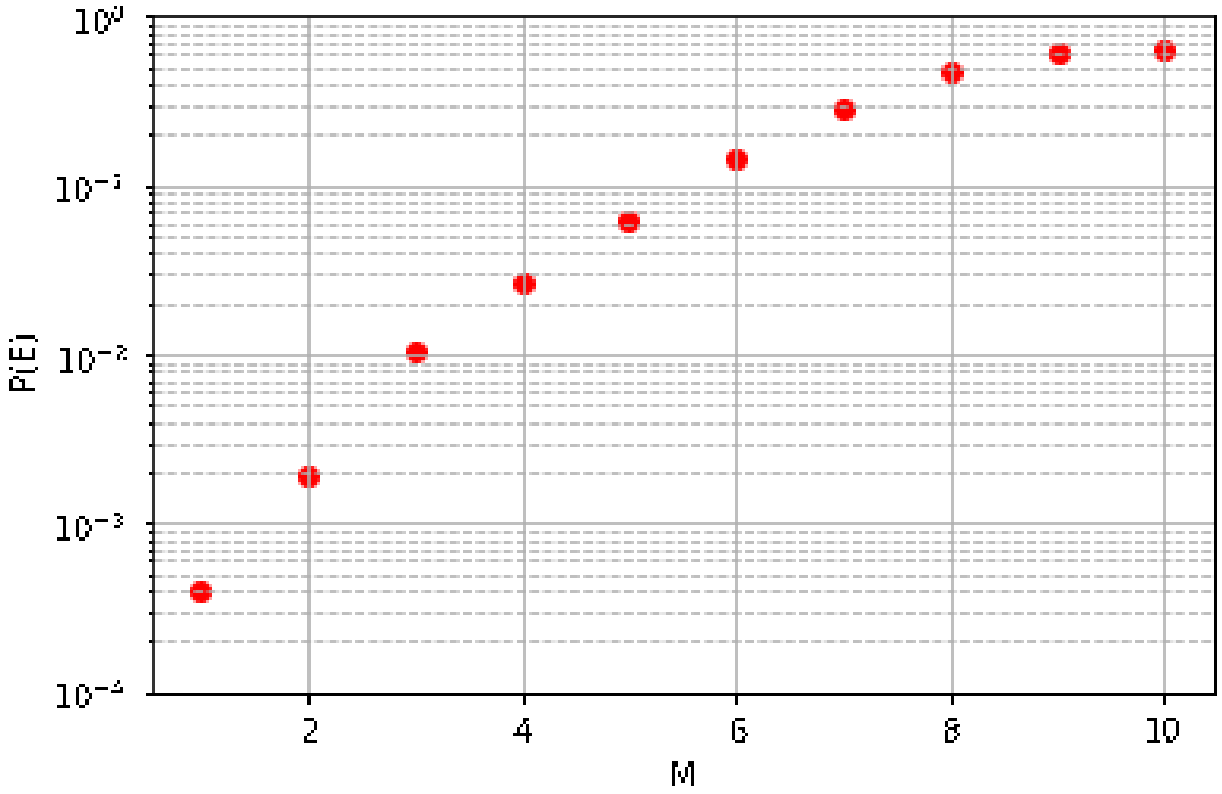

Figura 4.4: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con un layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane indipendenti.

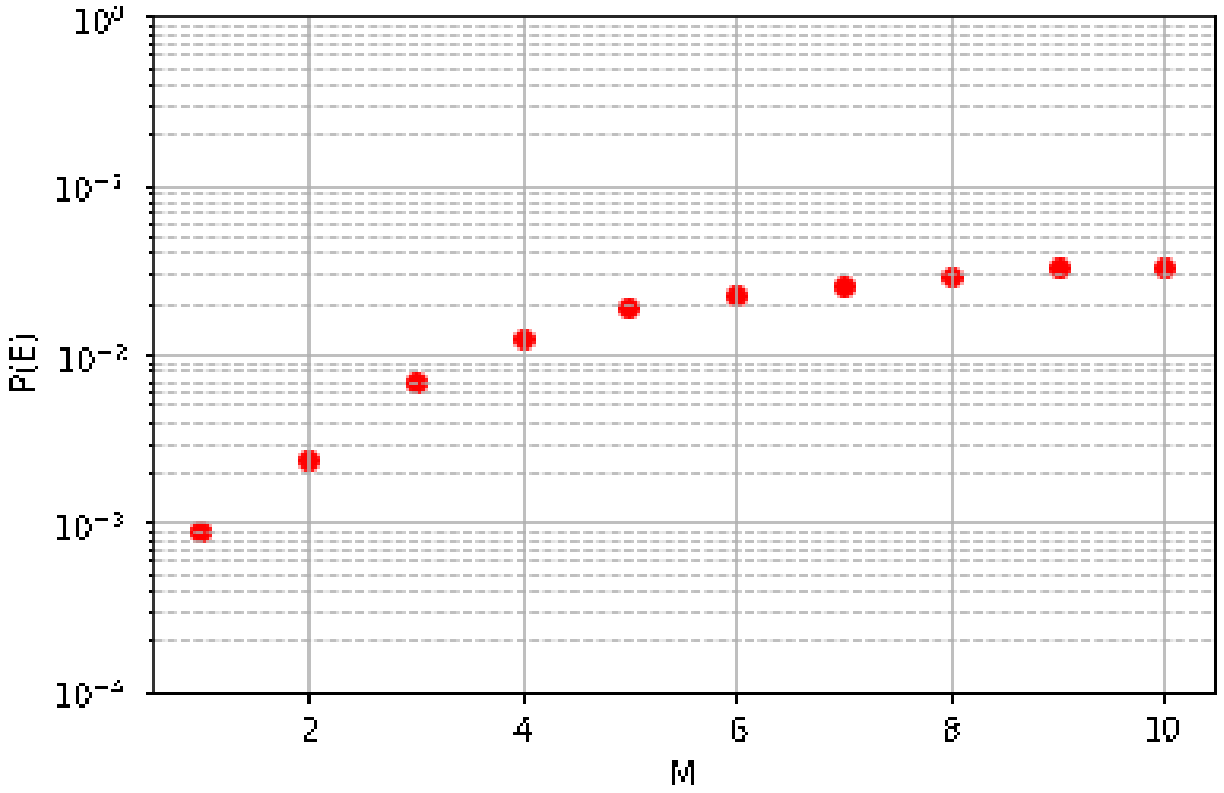

Figura 4.5: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con tre layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane indipendenti.

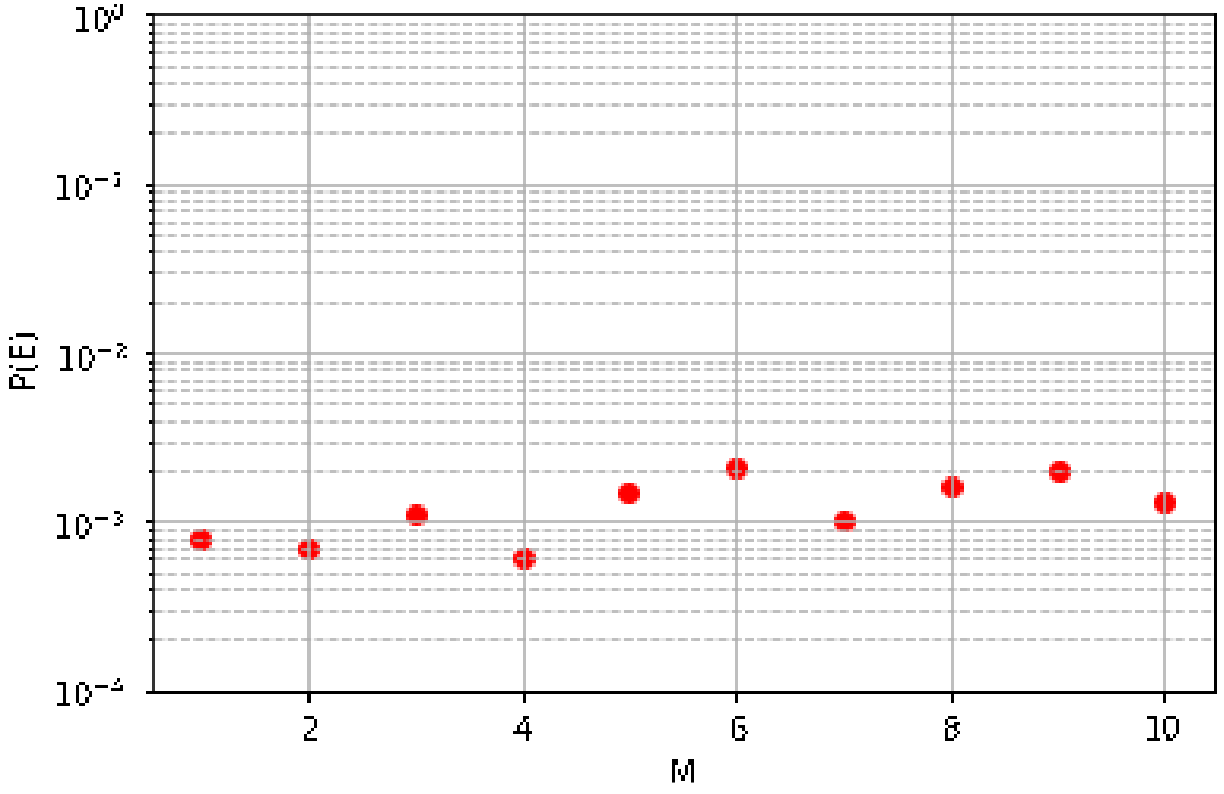

Figura 4.6: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con cinque layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane indipendenti.

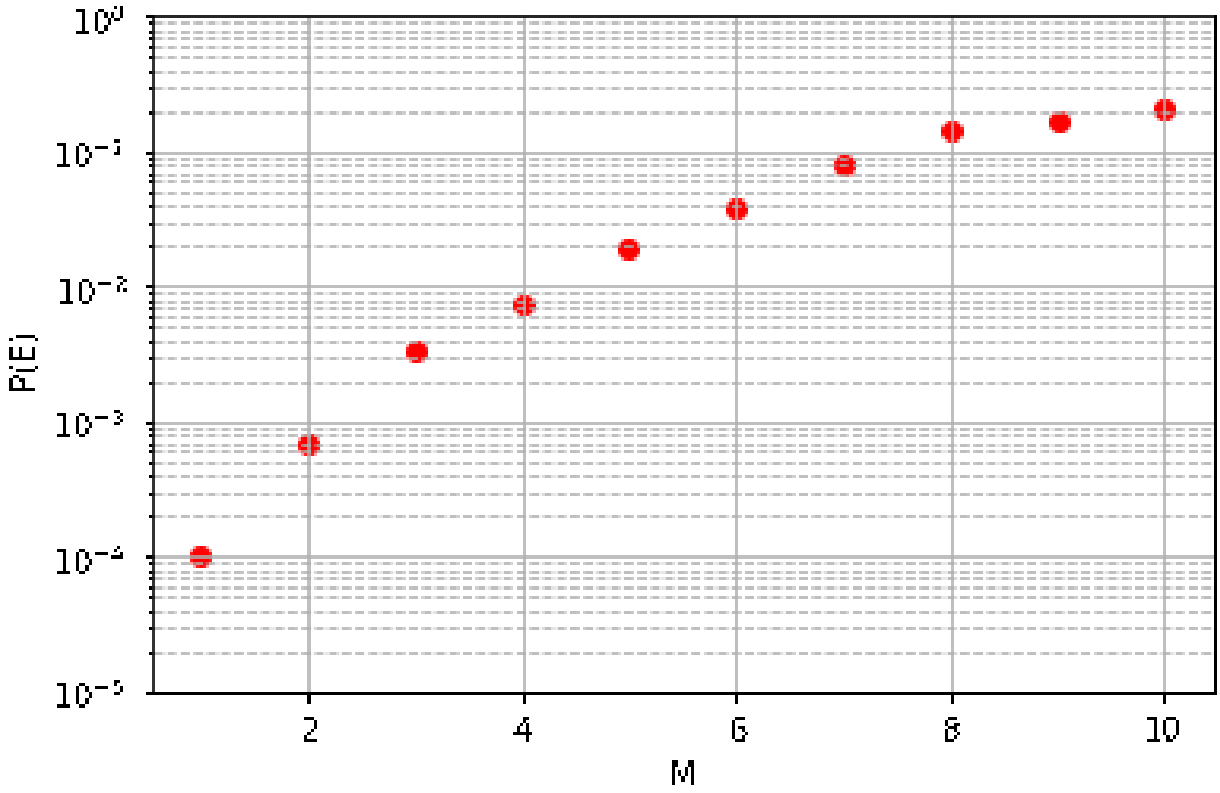

Figura 4.7: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con un layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane correlate.

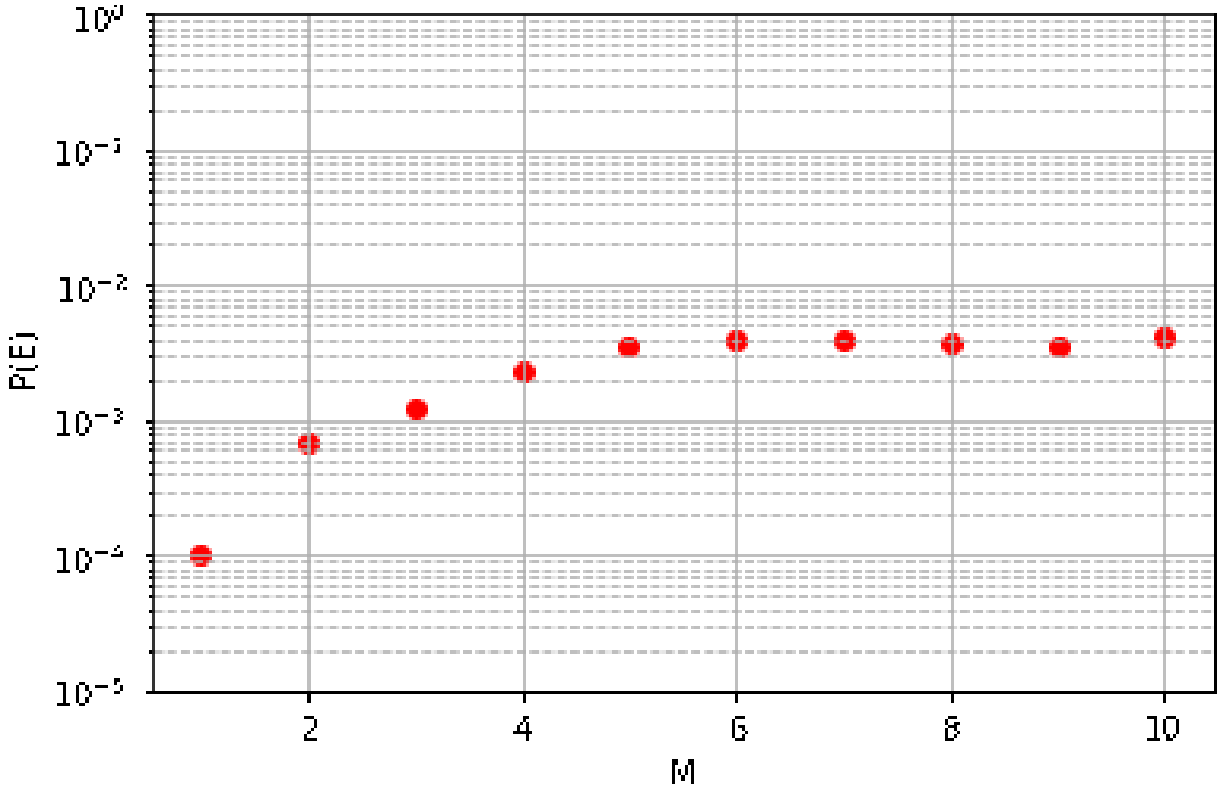

Figura 4.8: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con tre layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane correlate.

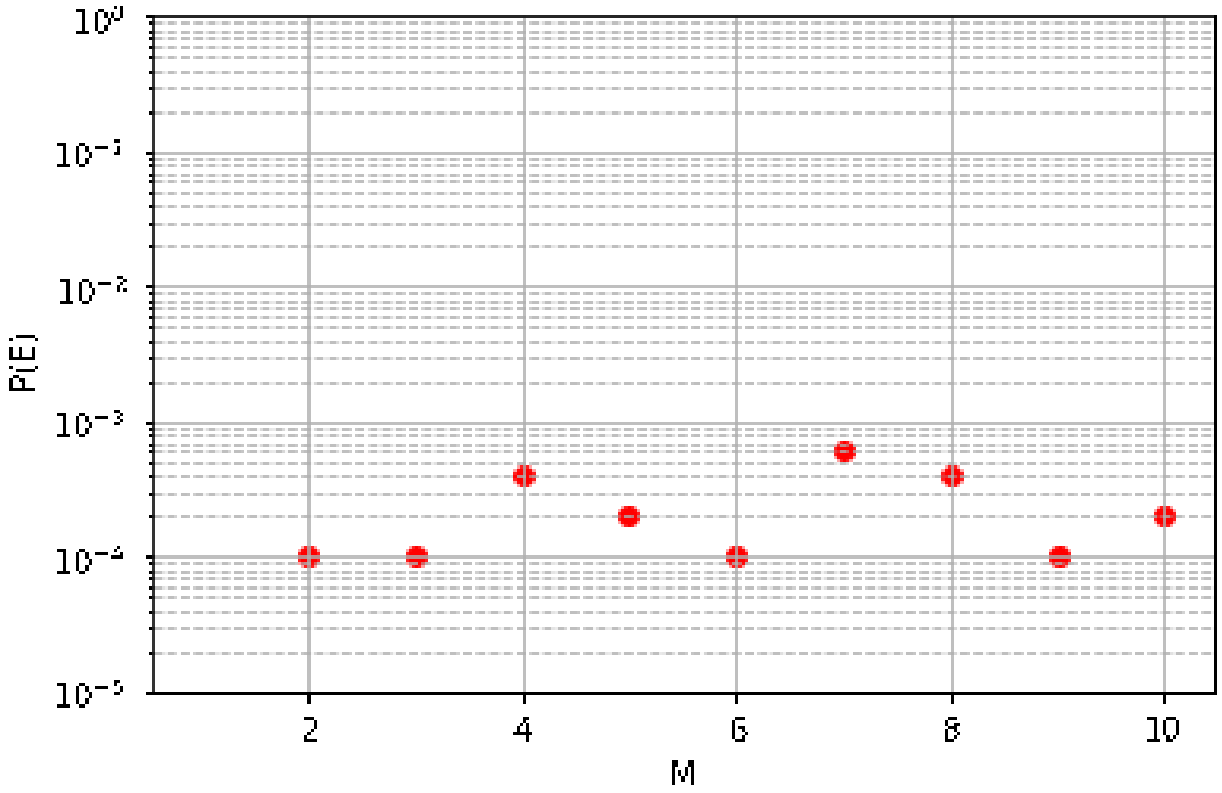

Figura 4.9: Probabilità di errore per numero di bit di quantizzazione nell'autoencoder con cinque layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane correlate.

Nei grafici riportati notiamo come la probabilità di errore si abbassa al crescere del numero di layer utilizzati dall'autoencoder generante le chiavi, mostrando come un numero maggiore di livelli in un autoencoder porti a una ricostruzione più efficace dell'ingresso. Notiamo inoltre come la probabilità di errore si abbassa considerevolmente passando da un ingresso costituito da variabili indipendenti a uno costituito da variabili correlate in quanto risulta più semplice ricostruire y a partire da x se questi hanno un certo grado di correlazione.

4.5 Informazione mutua tra le chiavi generate

La probabilità di errore non va a indicare tuttavia l'efficienza effettiva delle chiavi generate, difatti se gli autoencoder degli utenti coinvolti nella comunicazione dessero origine sempre alla stessa chiave l'utilità di quest'ultima sarebbe minima in quanto renderebbe la comunicazione più esposta ad attacchi esterni. Vogliamo pertanto verificare l'informazione mutua tra le due chiavi b_x e b_y generate a partire dai segnali x e y per verificare il numero di bit che le chiavi hanno in comune e quindi anche quando convenga realmente aumentare il numero di bit di quantizzazione delle chiavi. Indichiamo con $H(s)$ l'entropia della variabile aleatoria s

$$I(b_x; b_y) = H(b_x) + H(b_y) - H(b_x, b_y) \tag{22}$$

Per il calcolo delle entropie semplici e congiunte abbiamo utilizzato una matrice quadrata J in cui ad ogni riga r viene mappata una possibile chiave generata dall'autoencoder avente come ingresso x e ad ogni colonna c viene mappata una possibile chiave generata dall'autoencoder avente come ingresso y. Gli elementi della matrice contengono la probabilità campionaria che nei due autoencoder vengano generate le due chiavi corrispondenti a riga e colonna. Indichiamo con j_{rc} l'elemento alla riga r e alla colonna c della matrice J e con n_{rc} il numero di casi in cui la chiave generata a partire da x corrisponde a r espresso in base 2 e la chiave generata a partire da y corrisponde a c espresso in base 2

$$j_{rc} = \frac{n_{rc}}{n_{tot}}\,. \tag{23}$$

A partire da questa matrice è possibile il calcolo delle entropie singole e di quella congiunta delle chiavi. Prendiamo in esame il caso dell'autoencoder a tre livelli avente in ingresso un vettore di 16 elementi correlati tra loro dove la probabilità di errore si assesta a partire da 4 bit di quantizzazione. In Figura 4.10 riportiamo l'informazione mutua tra le chiavi per il numero di bit di quantizzazione dell'autoencoder. Prendiamo in esame il caso dell'autoencoder a tre livelli avente in ingresso un vettore di 16 elementi correlati tra loro dove la probabilità di errore si assesta a partire da 4 bit di quantizzazione.

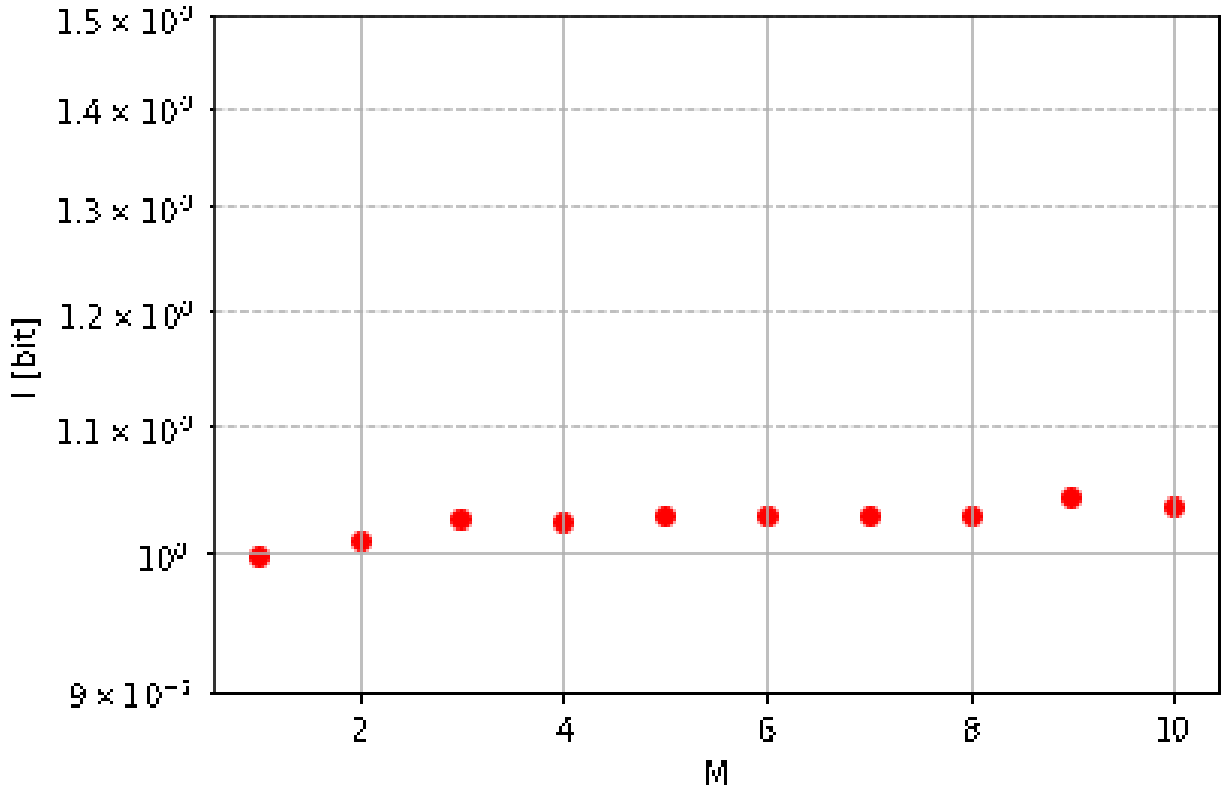

Figura 4.10: Informazione mutua per numero di bit di quantizzazione tra le chiavi generate negli autoencoder con tre layer nell'encoder per ingressi x e y aventi α costituito da un vettore di variabili aleatorie gaussiane correlate.

Dal grafico si può notare come anche la mutua informazione tra le chiavi tende a rimanere costante per l'aumentare del numero di bit di quantizzazione, assestandosi in un intorno del valore di 1 bit. Da questi risultati si può dedurre come l'utilizzo di un numero sempre maggiore di bit che costituiscono la chiave risulta essere inefficiente in quanto non va a diminuire la probabilità di errore né tantomeno ad aumentare la quantità di informazione che essa porta con sé, risulterà infatti sempre più probabile la generazione della stessa chiave e pertanto inefficace. Per verificare se il fenomeno è dovuto a una differenza del rumore tra i due segnali eccessivo o alla modalità di ricostruzione adottata dall'autoencoder andiamo a misurare l'informazione mutua tra le chiavi generate all'aumentare della deviazione standard σ del rumore dei segnali considerati. Le chiavi sono generate per i seguenti casi specifici:

- un autoencoder avente tre livelli e un bit di quantizzazione per neurone per quattro neuroni nel livello interno per un ingresso avente per base α variabili aleatorie correlate a partire da una variabile aleatoria gaussiana (19);
- un autoencoder avente tre livelli e tre bit di quantizzazione per neurone per quattro neuroni nel livello interno per un ingresso avente per base α variabili aleatorie correlate a partire da una variabile aleatoria gaussiana (19);

- un autoencoder avente tre livelli e un bit di quantizzazione per neurone per quattro neuroni nel livello interno per un ingresso avente per base α variabili aleatorie correlate a partire da due variabili aleatorie gaussiane. Indichiamo con β_2 un vettore costituito da due variabile gaussiane a media nulla e varianza unitaria indipendenti tra loro e con Y la matrice costituita dalla seconda e terza colonna della matrice *DFT* di dimensione 16x16.

$$\alpha = Y\beta_2 \tag{24}$$

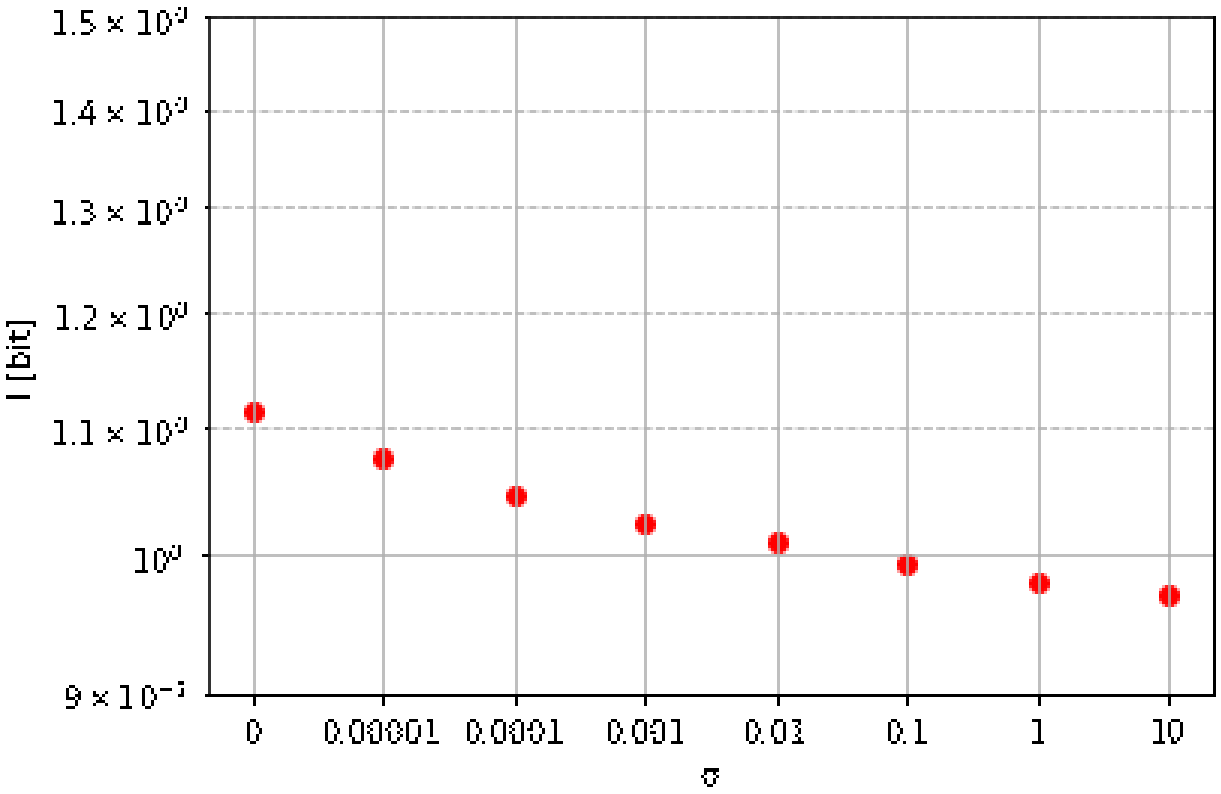

Figura 4.11: Informazione mutua per deviazione standard delle variabili ω_1e ω_2 tra le chiavi generate negli autoencoder con tre layer nell'encoder e M=N=4 per ingressi x e y aventi α costituito da un vettore di variabili aleatorie correlate, a partire da una variabile aleatoria gaussiana.

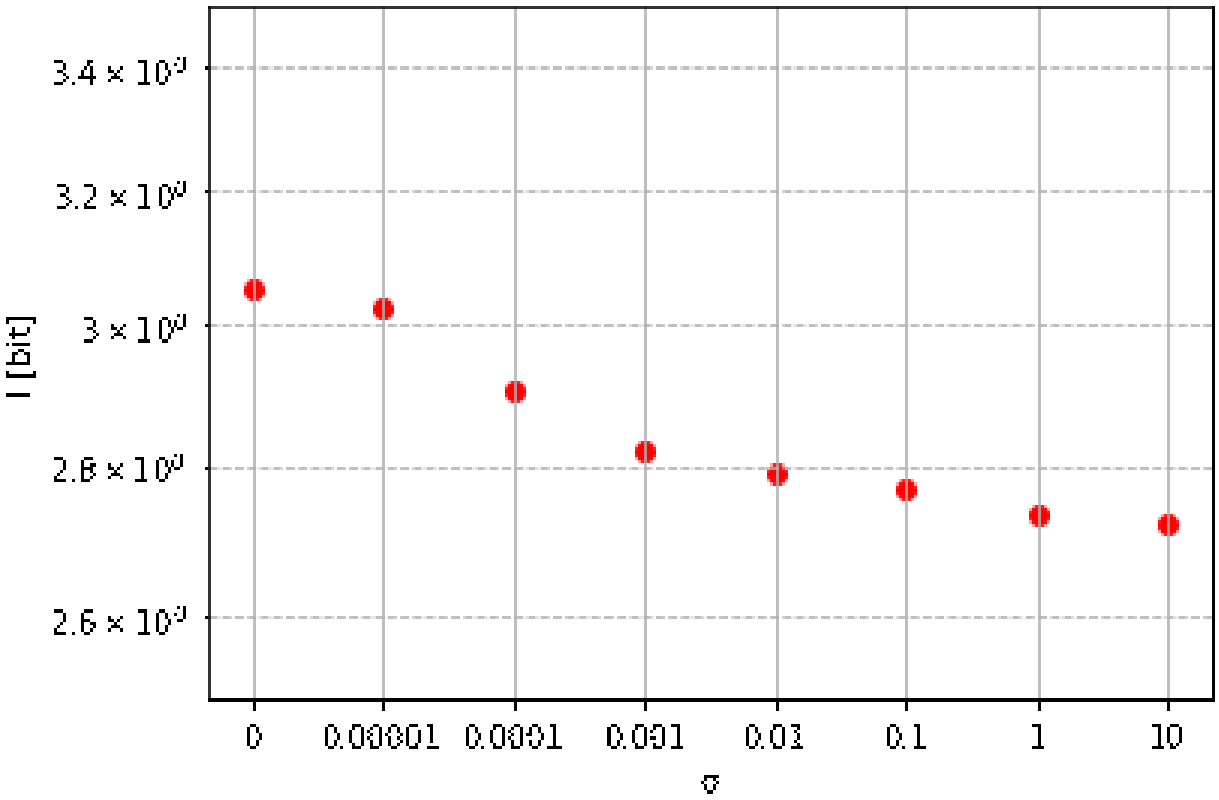

Figura 4.12: Informazione mutua per deviazione standard delle variabili ω_1e ω_2 tra le chiavi generate negli autoencoder con tre layer nell'encoder e M=12 e N=4 per ingressi x e y aventi α costituito da un vettore di variabili aleatorie correlate, a partire da una variabile aleatoria gaussiana.

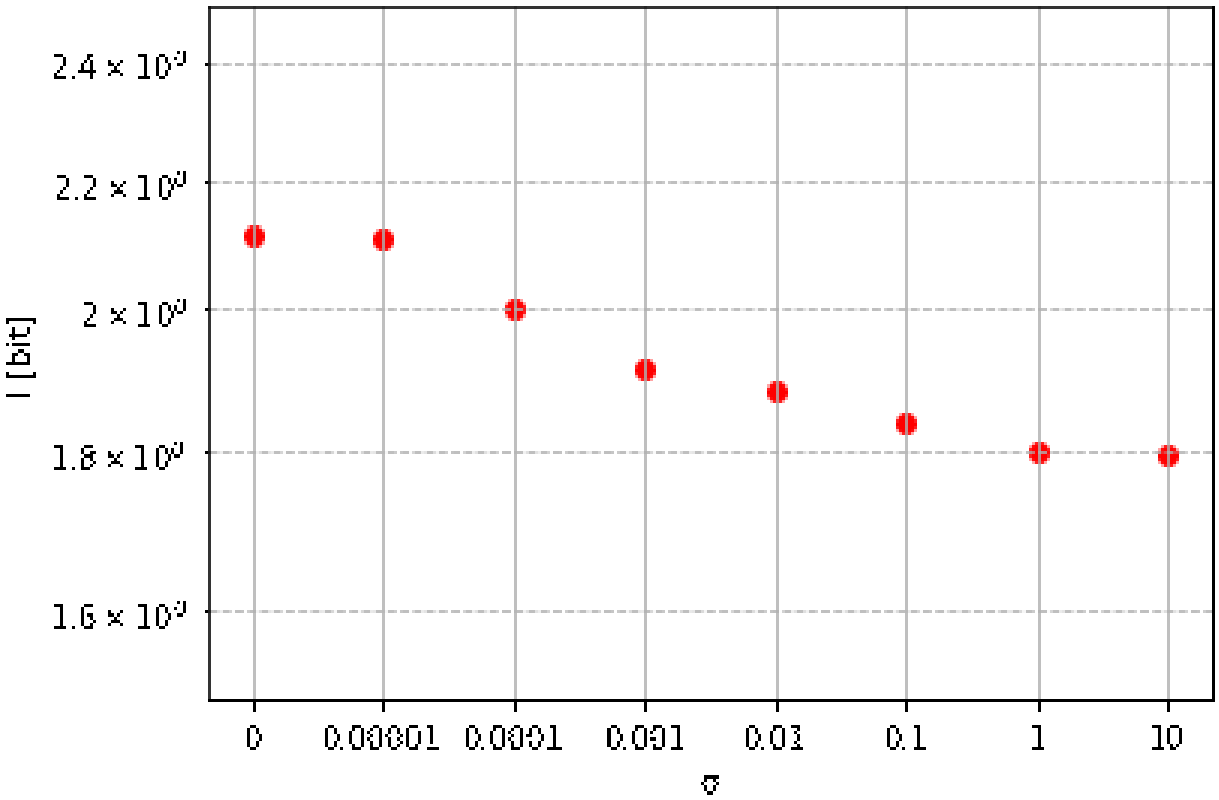

Figura 4.13: Informazione mutua per deviazione standard delle variabili ω_1e ω_2 tra le chiavi generate negli autoencoder con tre layer nell'encoder e M=N=4 per ingressi x e y aventi α costituito da un vettore di variabili aleatorie correlate, a partire da due variabili aleatorie gaussiane.

Dai grafici notiamo come l'autoencoder va a ricostruire il segnale di uscita andando ad utilizzare nel livello interno un numero di neuroni pari al numero delle variabili indipendenti che originano il vettore di ingresso, pertanto, il numero di bit significativi di una chiave generata tramite l'uso di autoencoder dipenderà principalmente dal segnale di ingresso.

4.6 Probabilità di errore di un attaccante

Per la simulazione della figura dell'attaccante andiamo a simulare il segnale z in modo che risulti differente da x e y. Consideriamo α' come un vettore di 16 elementi generati a partire da una variabile aleatoria gaussiana a media nulla e varianza unitaria β, la quale è stata moltiplicata per la terza colonna della matrice $DFT\ \ v'$.

$$\alpha' = \beta v' \tag{25}$$

Il rumore ω_3 consiste invece in un vettore di 16 variabili aleatorie gaussiane aventi varianza doppia rispetto alla varianza dei rumori ω_1 e ω_2 che vanno a costruire i vettori x e y considerati come mostrato in (18).

$$z = \alpha' + \omega_3 \tag{26}$$

Nei grafici seguenti riportiamo la probabilità di errore, calcolata come descritto al Paragrafo 4.4 dell'attaccante nel ricostruire le chiavi dei dispositivi coinvolti nella comunicazione nei casi dei tre autoencoder presentati al Paragrafo 4.3 considerando che A, B e E usino lo stesso autoencoder.

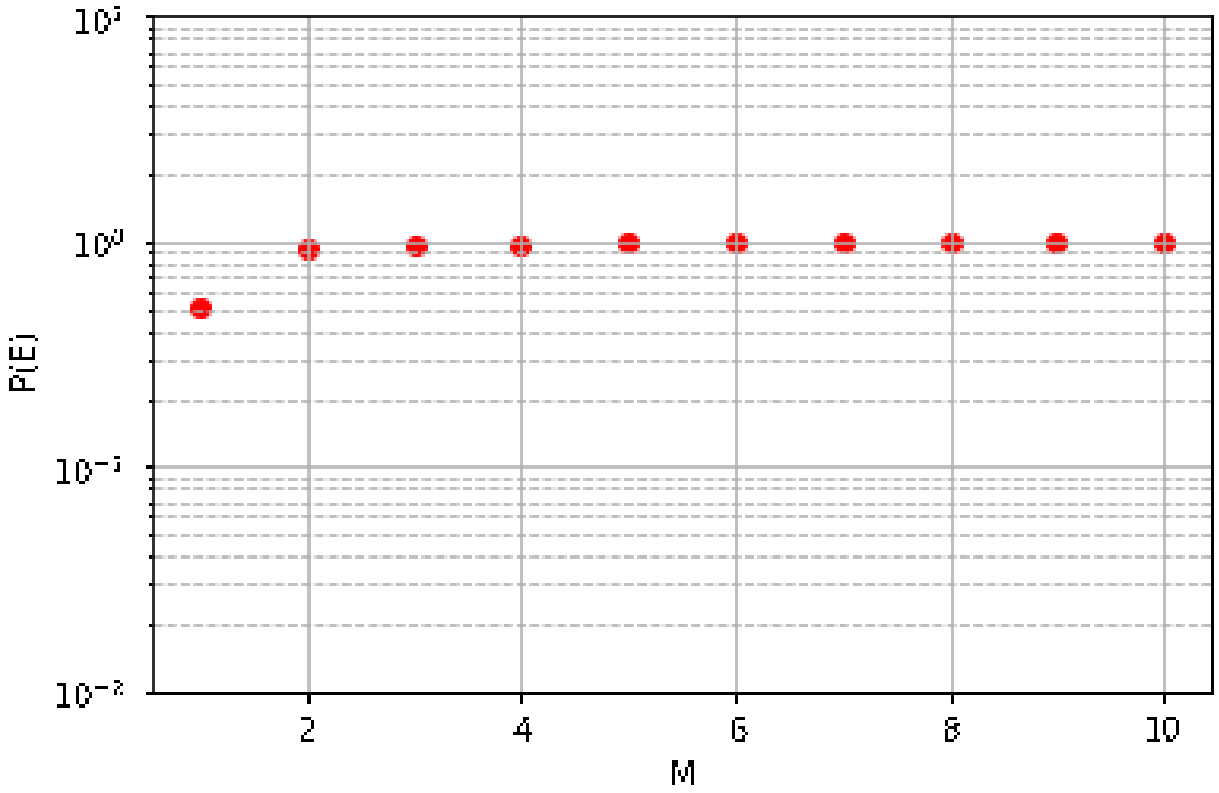

Figura 4.14: Probabilità di errore dell'attaccante per numero di bit di quantizzazione nell'autoencoder con un layer nell'encoder.

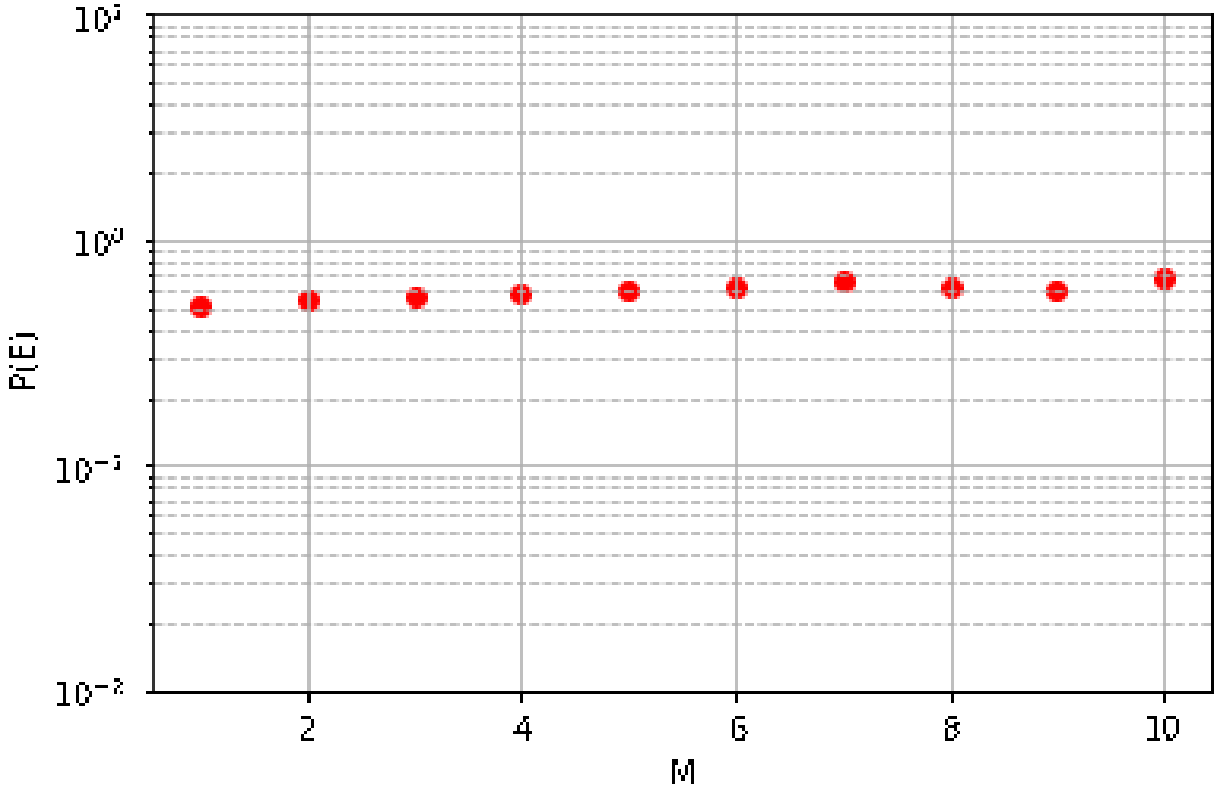

Figura 4.15: Probabilità di errore dell'attaccante per numero di bit di quantizzazione nell'autoencoder con tre layer nell'encoder.

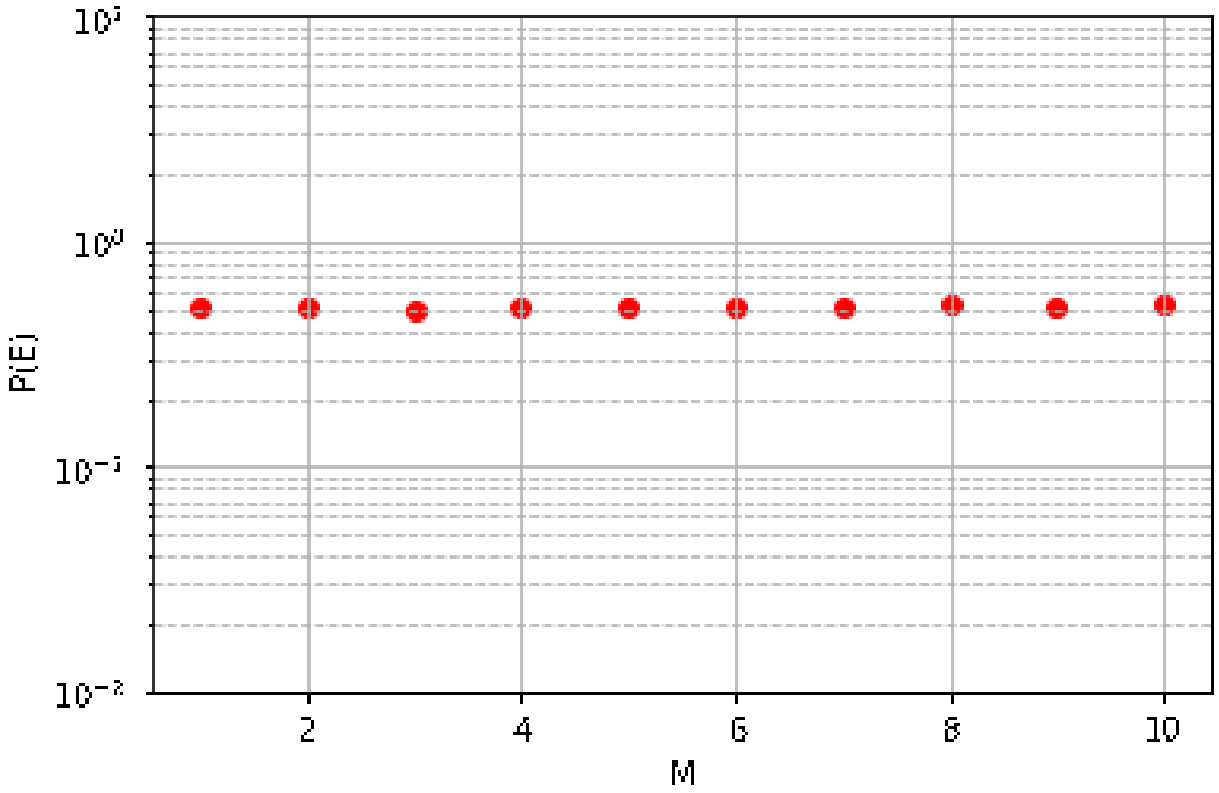

Figura 4.16: Probabilità di errore dell'attaccante per numero di bit di quantizzazione nell'autoencoder con cinque layer nell'encoder.

Dai grafici si nota come l'utilizzo di un autoencoder con un solo layer nell'encoder risulta essere per l'attaccante un metodo di ricostruzione della chiave molto peggiore di una ricostruzione casuale della chiave. L'uso invece di autoencoder a tre e a cinque layer nell'encoder abbassa notevolmente la probabilità di errore nel ricostruire la chiave utilizzata dagli utenti per la comunicazione.

Per testare la metodologia descritta al Paragrafo 3.5 in cui x e z non presentano un alto grado di correlazione, siamo partiti da alcuni dataset artificialmente generati in accordo con la statistica di alcune misurazioni di un canale di comunicazione sottomarino come descritto in [9]. Tali dataset inoltre fornivano un certo grado di correlazione C delle feature. Di questi dataset il 60% dei dati viene utilizzato per l'addestramento delle reti e il restante 40% per la fase di test. Poiché i dataset forniscono dei vettori di ingresso a quattro variabili utilizziamo degli autoencoder a 3 livelli nell'encoder aventi N=M=2. Di seguito riportiamo il grafico della probabilità di errore, in funzione del grado di correlazione tra i dataset C, dell'attaccante. Al fine di valutare la probabilità di errore dell'attaccante riportiamo anche il grafico della probabilità di errore nella concordanza delle chiavi generate tra i due utenti A e B.

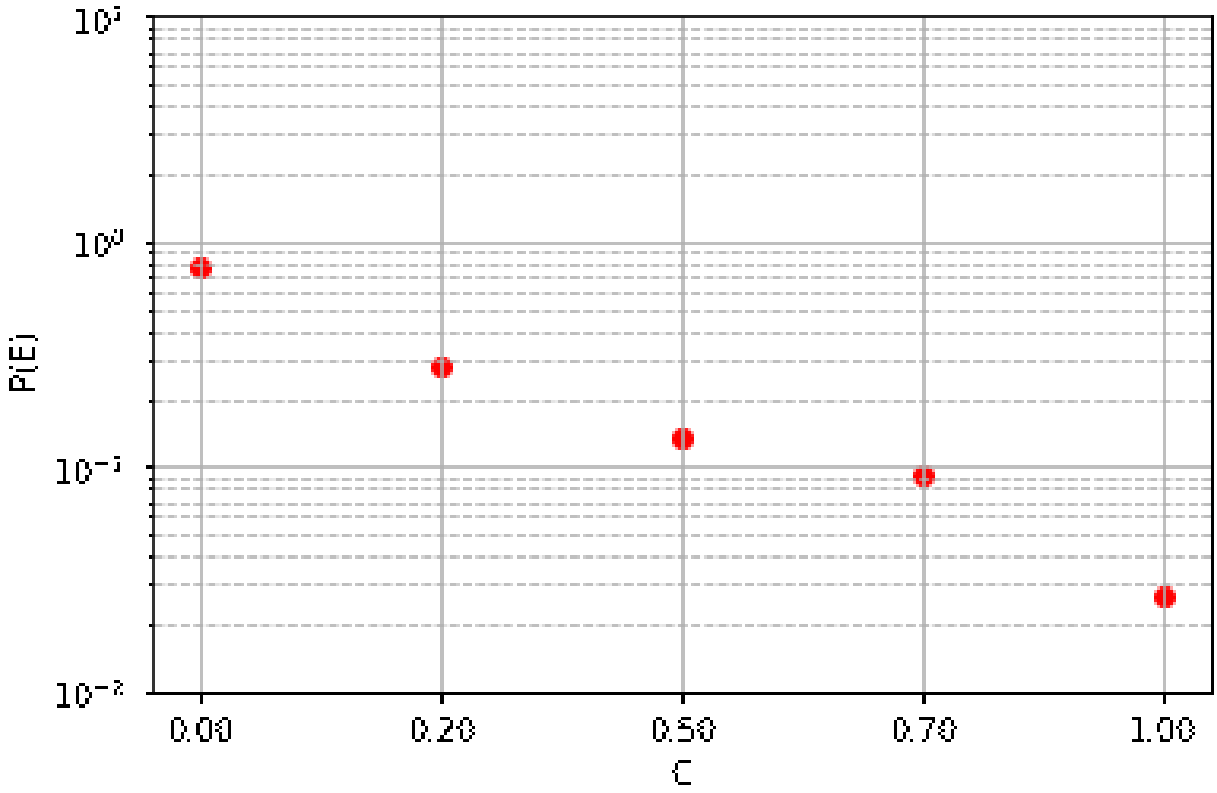

Grafico 4.17: Probabilità di errore dell'attaccante per grado di correlazione del dataset di A con quello dell'attaccante.

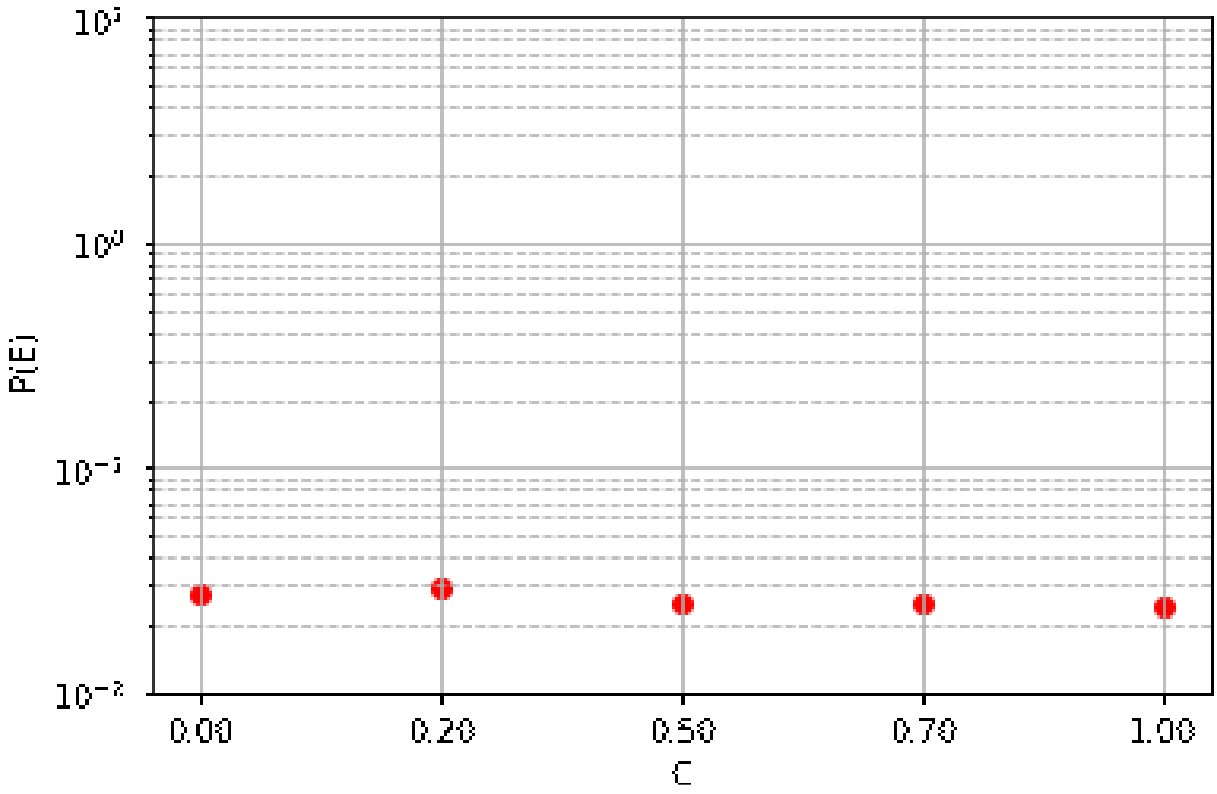

Grafico 4.18: Probabilità di errore nella concordanza delle chiavi generate dagli utenti A e B, per dataset utilizzato di A.

Dai grafici possiamo notare come nel caso in cui i due dataset non presentino alcuna correlazione tra di loro la generazione di chiavi da parte dell'attaccante equivale a un attacco di forza bruta mentre aumentando la correlazione tra di essi la probabilità di errore diminuisce sempre più fino ad essere confrontabile con quella che si ottiene nella generazione di chiavi tra i soggetti coinvolti nella comunicazione, per una correlazione di 1.00 tra i due dataset.

Capitolo 5

Conclusioni

Dai risultati presentati nel Capitolo 4 risulta evidente come l'uso di autoencoder aventi un quantizzatore differenziabile al livello interno delle informazioni latenti, ricostruisce tanto meglio una chiave per i dispositivi A e B quanto più sono i livelli che vanno a costituire l'encoder e quanto più i vettori di ingresso presentano una correlazione tra i loro elementi in quanto questo parametro come visto dai grafici del Paragrafo 4.5 influenza l'informazione mutua tra le due chiavi. L'attaccante che utilizza una rete neurale per la costruzione della chiave risulta poco efficiente per un numero basso di bit di quantizzazione e di layer nell'encoder tanto da essere confrontabile con un attacco di forza bruta e dipende soprattutto dal grado di correlazione tra il canale visto dagli utenti coinvolti nella comunicazione e quello visto dall'attaccante.

Bibliografia

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. “Learning internal representations by error propagation In Parallel Distributed Processing”. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[2] U. Michelucci, “An Introduction to Autoencoders,” TOELT.AI., arXiv:2201.03898, Sci. Rep, 12 Jan. 2022.

[3] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Feedforward Network,” in Deep Learning, www.deeplearningbook.org, MIT Press, 2016.

[4] I. Goodfellow, Y. Bengio, and A. Courville, “Autoencoders,” in Deep Learning, www.deeplearningbook.org, MIT Press, 2016.

[5] “Sources of Digital Information” in Principles of Communications Networks and Systems, N. Benvenuto and M. Zorzi eds, John Wiley & Sons, 2011.

[6] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu and J. Yan, “Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks”; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 4852-4861.

[7] J. Han, X. Zeng, X. Xue and J. Ma, "Physical Layer Secret Key Generation Based on Autoencoder for Weakly Correlated Channels", 2020 IEEE/CIC International Conference on Communications in China (ICCC), pp. 1220-1225.

[8] H. He, Y. Chen, X. Huang, M. Xing, Y. Li, B. Xing, L. Chen, "Deep Learning-Based Channel Reciprocity Learning for Physical Layer Secret Key Generation", Security and Communication Networks, vol. 2022.

[9] F. Ardizzon, S. Tomasin, R. Diamant and P. Casari “Machine Learning-Based Distributed Authentication of UWAN Nodes With Limited Shared Information”, UCOMM 2022, University of Padova and CNIT, Italy.