



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria

Corso di Laurea in Ingegneria e Scienze Informatiche

**Analisi e Mitigazione delle  
Vulnerabilità nel Protocollo BGP:  
Un Approccio al Rafforzamento della  
Sicurezza nelle Reti di  
Telecomunicazioni**

**Relatore:**  
**Chiar.mo Prof.**  
**Andrea Piroddi**

**Presentata da:**  
**Roberto Pisu**

---

**Sessione Ottobre 2025**  
**Anno Accademico 2024/2025**

(DA FARE ALLA FINE)

5 parole chiave per caratterizzare il contenuto della dissertazione:  
(se non ti piacciono così sparse puoi anche semplicemente scriverle su una riga sola)

parola 5

parola 4

parola 3

parola 2

Parola 1

*Alla mia famiglia  
ai miei amici e a chi mi è stato accanto.  
Che questo traguardo sia solo l'inizio.*



# Abstract

Abstract qui (ti consiglio di farlo alla fine)



# Indice

<b>0</b>	<b>INTRODUZIONE</b>	<b>1</b>
<b>1</b>	<b>Autonomous System</b>	<b>5</b>
1.1	Il ruolo dell'AS in Internet . . . . .	5
1.1.1	Cos'è il Regional Internet Registry . . . . .	6
1.1.2	Struttura ASN . . . . .	7
1.2	Classificazione AS . . . . .	7
<b>2</b>	<b>Protocollo di routing BGP 4.0</b>	<b>9</b>
2.1	Cos'è il routing . . . . .	9
2.2	Nascita del protocollo BGP . . . . .	9
2.2.1	Sostituzione di EGP . . . . .	11
2.3	Caratteristiche protocollo BGP . . . . .	11
2.3.1	Modello ISO/OSI e TCP/IP . . . . .	12
2.3.2	Collocazione architetturale e modalità operative di BGP . . .	14
2.3.3	Il ruolo del router e le tabelle di routing . . . . .	15
2.3.4	Le tabelle di routing in BGP . . . . .	16
2.4	Funzionamento BGP . . . . .	16
2.4.1	Tipologie di sessione . . . . .	16
2.4.2	Caratteristiche del protocollo path-vector . . . . .	17
2.4.3	Policy di routing . . . . .	17
2.4.4	Formato dei messaggi . . . . .	18
2.4.5	Gestione delle sessioni (FSM) . . . . .	18
2.4.6	Path attributes . . . . .	20
2.4.7	Processo . . . . .	22
2.4.8	Considerazioni finali . . . . .	23
<b>3</b>	<b>Metodologie di attacco e prevenzione del protocollo BGP</b>	<b>25</b>
3.1	BGP prefix hijacking . . . . .	25
3.1.1	Descrizione dell'attacco. . . . .	25

3.1.2	La regola del Longest Prefix Match . . . . .	26
3.1.3	Tipologie di BGP Prefix Hijacking . . . . .	26
3.2	BGP route leaking . . . . .	27
3.2.1	Descrizione dell'attacco . . . . .	27
3.3	BGP session hijacking . . . . .	28
3.3.1	Descrizione dell'attacco . . . . .	28
3.3.2	Dinamica dell'attacco . . . . .	28
3.4	BGPsec . . . . .	29
3.4.1	Come funziona BGPsec . . . . .	29
3.4.2	Vulnerabilità risolte . . . . .	30
3.5	BGP RPKI . . . . .	31
3.5.1	Come funziona BGP RPKI . . . . .	31
3.5.2	Vulnerabilità risolte . . . . .	32
3.6	Monitoraggio e rilevamento anomalie . . . . .	32
3.6.1	BGPStream (CAIDA) . . . . .	33
3.6.2	RIPE RIS . . . . .	33
3.6.3	Route Views . . . . .	34
<b>4</b>	<b>Implicazioni degli attacchi BGP nel mondo</b>	<b>35</b>
4.1	BGP prefix hijacking Pakistan Telecom 2008 . . . . .	35
4.1.1	Timeline dell'incidente . . . . .	36
4.1.2	Motivazioni dell'attacco . . . . .	36
4.1.3	Analisi tecnica . . . . .	36
4.1.4	Implicazioni e lezioni apprese . . . . .	37
4.2	BGP prefix hijacking di China Telecom 2010 . . . . .	37
4.2.1	Timeline dell'incidente . . . . .	37
4.2.2	Motivazioni e controversie . . . . .	38
4.2.3	Analisi tecnica . . . . .	38
4.2.4	Implicazioni e lezioni apprese . . . . .	38
<b>5</b>	<b>Tecnologie utilizzate</b>	<b>39</b>
5.1	Ambiente di virtualizzazione . . . . .	39
5.1.1	VMware Workstation Pro . . . . .	39
5.1.2	LXC . . . . .	42
5.2	FRRouting . . . . .	45
5.3	GoBGP . . . . .	46
5.4	Topologia rete . . . . .	47
5.4.1	Struttura rete generale (topologia a maglia parziale tra 7 AS)	48
5.4.2	Struttura singolo AS . . . . .	49
<b>6</b>	<b>Sviluppo e implementazione</b>	<b>51</b>



6.1	Creazione di un AS . . . . .	51
6.1.1	Creazione VM con ubuntu server . . . . .	51
6.1.2	Configurazione Ubuntu server . . . . .	56
6.1.3	Installazioni preliminari sulla VM host . . . . .	59
6.1.4	Installazione FRR sui container . . . . .	65
6.1.5	Installazione GoBGP sui container dei Border Router . . . . .	66
6.1.6	Configurazione OSPF nei router . . . . .	67
6.2	Clonazione dei restanti AS . . . . .	68
6.2.1	Cambio nomenclatura e indirizzo ip esterno del Border Router (BR) . . . . .	69
6.2.2	. . . . .	70
<b>7</b>	<b>Prospettive future: SDN e BGP</b>	<b>71</b>
7.1	Cos'è la SDN . . . . .	71
7.1.1	Architettura e principio di separazione control/data plane . .	71
7.1.2	Vantaggi principali: flessibilità, programmazione, automazione	71
7.2	Integrazione tra SDN e BGP . . . . .	71
7.2.1	Routing interdominio gestito centralmente . . . . .	71
7.2.2	Esempi di progetti o framework (es. SDX, BGP-SDN) . . . . .	71
7.3	Prospettive evolutive . . . . .	71
7.3.1	Reti programmabili e scenari futuri . . . . .	71
7.3.2	Possibili impatti su sicurezza e gestione globale . . . . .	71
<b>A</b>	<b>Ricerca su ECDSA</b>	<b>73</b>



# Elenco delle tabelle

2.1	Principali attributi BGP e loro classificazione . . . . .	21
5.1	Confronto tra LXC e Docker <sup>[7]</sup> . . . . .	44
6.1	Indirizzi di rete /24 assegnati alle LAN dei router interni (IR) . . . .	63



# Elenco delle figure

1.1	Mappa dei RIR <sup>[11]</sup> . . . . .	6
2.1	Modello ISO/OSI e modello TCP/IP . . . . .	13
5.1	OS virtualization VS Hardware virtualization . . . . .	41
5.2	bare-metal vs hosted . . . . .	41
5.3	para vs full virtualization . . . . .	42
5.4	Topologia della rete completa . . . . .	48
5.5	Topologia dell'AS . . . . .	49
6.1	Schermata di avvio di workstation pro 17 . . . . .	52
6.2	Selezione file iso . . . . .	53
6.3	Nome e percorso VM . . . . .	54
6.4	Impostazioni VM . . . . .	55
6.5	VM completata . . . . .	56
6.6	Selezione lingua . . . . .	57
6.7	Ubuntu minimized . . . . .	57
6.8	Configurazione mirror ubuntu . . . . .	58
6.9	Configurazione storage . . . . .	58
6.10	Configurazione bridge virtuali . . . . .	60
6.11	Configurazione interfacce . . . . .	61
6.12	Configurazione rete VM host . . . . .	64



# Capitolo 0

## INTRODUZIONE

Al giorno d'oggi, Internet rappresenta una delle infrastrutture più critiche e pervasive della nostra società, in quanto viene utilizzata costantemente in molteplici ambiti della vita quotidiana: dalla comunicazione personale e professionale che ci tiene connessi a livello globale, all'accesso immediato a una quantità crescente di informazioni e contenuti multimediali, fino alla gestione di transazioni economiche, servizi bancari, amministrazione pubblica e logistica internazionale.

Semplificando, Internet può essere definita come la “rete delle reti”: una struttura complessa e gerarchica che consente il collegamento tra milioni di reti locali eterogenee, distribuite in tutto il mondo. A livello architetturale, Internet è composta da un numero elevato, ma finito, di Autonomous System (AS), ciascuno dei quali corrisponde a una rete di proprietà e gestione unificata — come ad esempio un Internet Service Provider (ISP), un'azienda, un'università o un ente governativo — caratterizzata da una propria politica di routing indipendente.

(Si stima l'esistenza di più di 90.000 AS )

Per “politica di routing” si intende l'insieme di regole, preferenze e vincoli che determinano in che modo il traffico di rete viene instradato all'interno dell'AS e verso gli altri sistemi autonomi. Il protocollo di routing incaricato di gestire la propagazione delle informazioni tra AS è il Border Gateway Protocol (BGP), attualmente considerato lo standard de facto per l'interconnessione a livello globale. Il suo com-

pito è annunciare e apprendere rotte <sup>1</sup>, determinando così i percorsi lungo i quali i pacchetti viaggiano da un'estremità all'altra del mondo.

Nonostante la sua centralità e longevità, il protocollo BGP presenta una serie di limiti strutturali, legati al fatto che fu progettato in un'epoca in cui la sicurezza informatica non era ancora una priorità. BGP si basa infatti su un modello di fiducia implicita tra gli operatori di rete e non prevede, nella sua implementazione standard, meccanismi di autenticazione, integrità o verifica delle informazioni propagate. Questa mancanza di sicurezza nativa rende il protocollo vulnerabile a diversi tipi di attacco, tra cui il *prefix hijacking*, il *route leaking* e il *session hijacking*, che possono compromettere seriamente l'affidabilità e la sicurezza della rete Internet.

Alla luce di queste considerazioni, risulta fondamentale analizzare in dettaglio il funzionamento di BGP e le sue vulnerabilità, al fine di individuare le possibili contromisure per prevenire o limitare gli effetti di un eventuale attacco. In un mondo sempre più dipendente dall'utilizzo di Internet, garantire la resilienza e la sicurezza del protocollo di routing interdominio è una priorità non solo tecnica, ma anche strategica e geopolitica.

La tesi si compone di sette capitoli suddivisi come segue:

- **Primo Capitolo - Autonomous System** Nel primo capitolo viene approfondito il ruolo dell'AS, le sue caratteristiche, a cosa serve, come vengono classificati e altre informazioni utili a comprendere il funzionamento del protocollo BGP.
- **Secondo Capitolo - Protocollo di routing BGP 4.0** In questo capitolo viene analizzato in dettaglio il funzionamento del protocollo BGP (Border Gateway Protocol), attualmente lo standard principale per il routing tra sistemi autonomi su Internet. Dopo una panoramica introduttiva sui concetti fondamentali di routing, viene descritto il contesto storico e tecnico che ha portato all'introduzione di BGP, in particolare come evoluzione del precedente Exterior Gateway Protocol (EGP). Viene poi approfondita la natura del protocollo, la sua collocazione nei livelli del modello di rete e le dipendenze da altri protocolli sottostanti. Una sezione centrale del capitolo è dedicata

---

<sup>1</sup>Una rotta è il percorso scelto dal protocollo di routing per raggiungere una rete specifica.



al funzionamento interno di BGP: vengono spiegati il meccanismo di routing basato su path vector, l'applicazione delle politiche di importazione ed esportazione delle rotte, gli attributi utilizzati per determinare il miglior percorso e le strategie per evitare la formazione di cicli. Il capitolo si conclude con una descrizione del formato dei messaggi BGP e delle sessioni di peering tra router (iBGP ed eBGP), per poi presentare un confronto tra le versioni storiche del protocollo e le novità introdotte nell'attuale versione 4.0.

- **Terzo Capitolo - Metodologie di attacco e prevenzione del protocollo BGP** Nel terzo capitolo vengono approfondite le principali vulnerabilità del protocollo BGP nella sua ultima versione 4.0, ne vengono analizzate le cause, le modalità di esecuzione dell'attacco e le possibili conseguenze. Vengono trattati tre scenari d'attacco particolarmente rilevanti: il *prefix hijacking*, in cui un AS annuncia indebitamente prefissi IP altrui; il *route leaking*, che comporta la diffusione impropria di rotte apprese da altri peer; e il *session hijacking*, in cui un attore malevolo intercetta o falsifica la sessione BGP tra due router. Per ciascun attacco vengono analizzati nel dettaglio il funzionamento tecnico, le ripercussioni sul traffico e le vulnerabilità specifiche che vengono sfruttate. Successivamente, il capitolo introduce due tecnologie progettate per aumentare la sicurezza di BGP: Border Gateway Protocol Security (BGPsec), che fornisce autenticazione crittografica delle informazioni di routing tramite firme digitali basate su curve ellittiche (ECDSA), e Resource Public Key Infrastructure (RPKI), un'infrastruttura a chiave pubblica che permette di validare l'autorità di un AS ad annunciare un determinato prefisso IP. Vengono evidenziate le differenze tra le due soluzioni e i rispettivi ambiti di applicazione.

Nella parte finale, viene trattato il tema del monitoraggio delle anomalie BGP attraverso piattaforme pubbliche come *BGPStream* (di Cooperative Association for Internet Data Analysis (CAIDA)), *RIPE RIS* e *Route Views*, strumenti fondamentali per l'analisi forense, la diagnosi di incidenti e l'early warning<sup>2</sup> nel routing globale.

- **Quarto Capitolo - Implicazioni degli attacchi BGP nel mondo** Questo

---

<sup>2</sup>L'early warning è la capacità di rilevare tempestivamente anomalie o potenziali attacchi alla rete.

capitolo analizza due noti casi reali di attacchi al protocollo BGP, con l'obiettivo di mostrare le conseguenze concrete che vulnerabilità teoriche possono avere su scala globale. Il primo caso è quello del BGP hijacking da parte di Pakistan Telecom nel 2008, che causò l'inaccessibilità mondiale di YouTube per diverse ore. Il secondo riguarda China Telecom, coinvolta in episodi di deviazione di traffico internazionale tra il 2010 e il 2018.

- **Quinto Capitolo - Tecnologie utilizzate** Nel quinto capitolo entriamo nel vivo della simulazione, vengono infatti presentate le principali tecnologie adottate per costruire l'ambiente virtuale necessario alla simulazione degli attacchi e delle contromisure al protocollo BGP. L'infrastruttura è basata su Virtual Machine (VM) Ubuntu Server create con VMware Workstation Pro, che sfrutta virtualizzazione hardware di tipo 2 (hosted) con supporto alla full virtualization assistita da hardware.

All'interno di ogni VM vengono eseguiti container LXC, impiegati per simulare i router interni (Internal Router (IR)) in modo efficiente. Per la gestione del routing interno è stato utilizzato Free Range Routing (FRR), mentre per i router di confine (BR) è stato adottato GoBGP, particolarmente adatto alla configurazione delle sessioni BGP e alla sperimentazione di meccanismi come RPKI e BGPsec.

La rete complessiva è strutturata secondo una topologia a maglia parziale tra sette AS, ciascuno dotato di una struttura interna realistica composta da router, switch e VLAN. Tale configurazione permette di testare scenari di routing complessi in un ambiente controllato e riproducibile.

- **Sesto Capitolo - Sviluppo e implementazione**
- **Settimo Capitolo - Prospettive future: SDN e BGP**

# Capitolo 1

## Autonomous System

In questo capitolo, andiamo ad analizzare cos'è un AS, il suo ruolo nel routing globale e la loro classificazione.

### 1.1 Il ruolo dell'AS in Internet

Un AS è definito come un insieme di indirizzi IP e router sotto il controllo di una singola entità amministrativa, che adotta una politica di routing uniforme e coerente verso l'esterno. Secondo l'Request For Comment (RFC) 1930 dell'Internet Engineering Task Force (IETF), un AS è necessario ogniqualvolta un'organizzazione desidera definire delle regole di instradamento proprie e differenziate rispetto ad altri domini di routing, oppure quando intrattiene relazioni di peering con più fornitori di connettività a Internet, spesso attraverso gli Internet Exchange Point (IXP). Gli IXP sono infrastrutture fisiche dove diversi AS si connettono per scambiare traffico di rete tra loro in modo diretto. Questa interconnessione diretta, nota come peering, permette agli AS di scambiarsi dati senza dover passare per le reti di transito di terze parti, riducendo i costi e migliorando la latenza. Ogni AS è identificato univocamente dal Autonomous System Number (ASN), assegnato da uno dei cinque Regional Internet Registry (RIR).<sup>[5][9]</sup>

### 1.1.1 Cos'è il Regional Internet Registry

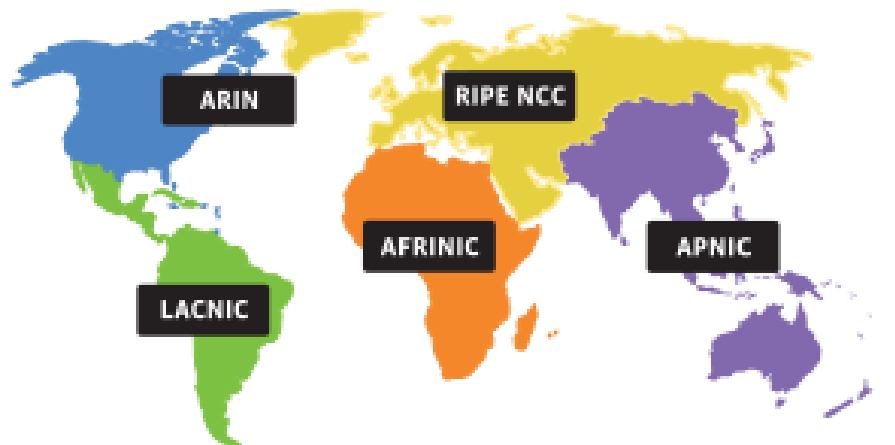
I RIR sono organizzazioni responsabili dell'assegnazione e della registrazione delle risorse numeriche di Internet all'interno di specifiche regioni geografiche del mondo. Le risorse che i RIR assegnano e registrano sono:

- Indirizzi IP sia IPv4 che IPv6.
- ASN usati per identificare gli AS.

Le organizzazioni (come gli ISP, grandi aziende, università, enti governativi, ecc.) che desiderano connettersi a Internet in modo indipendente (ovvero, operare il proprio AS) e/o avere un blocco di indirizzi IP pubblico da gestire direttamente, devono richiedere queste risorse al RIR competente per la loro area geografica.

Attualmente, esistono 5 RIR a livello globale:

- AfriNIC (Africa)
- ARIN (Nord America)
- LACNIC (America Latina e Caraibi)
- APNIC (Asia e Oceania)
- RIPE NCC (Europa, Medio Oriente e parti dell'Asia Centrale)



**Figura 1.1:** Mappa dei RIR<sup>[11]</sup>

Queste 5 RIR collaborano attraverso l'Number Resource Organization (NRO) e sono sotto la supervisione generale dell'ente Internet Assigned Numbers Authority (IANA), che è il coordinatore globale delle risorse numeriche e dei nomi di dominio di Internet.

### 1.1.2 Struttura ASN

La nomenclatura degli ASN segue due formati principali: il tradizionale a 16 bit e quello a 32 bit, introdotto successivamente per rispondere all'aumento della domanda (RFC 4893). Gli ASN a 16 bit vanno da 1 a 65535, con alcune riserve speciali: per esempio, i numeri da 64512 a 65534 sono destinati all'uso privato, mentre l'ASN 23456 viene usato come placeholder nella transizione tra 16 e 32 bit (RFC 6996). Il nuovo spazio a 32 bit estende la numerazione fino a 4.294.967.295 e consente anche l'utilizzo della notazione m:n (es. 1:10), che rappresenta una forma più leggibile del numero intero.

Gli ASN vengono spesso preceduti dal prefisso AS (es: AS13335 per Cloudflare, AS15169 per Google) e sono registrati pubblicamente in database come il WHOIS.

<sup>1</sup> [18]

## 1.2 Classificazione AS

Gli Autonomous System possono essere classificati secondo diversi criteri, in base al loro ruolo funzionale nella topologia globale di Internet, alla natura delle connessioni che intrattengono con altri AS, oppure alle politiche di routing adottate.

Una classificazione comunemente diffusa è quella che distingue gli AS in tre grandi categorie:

- **Stub AS:** AS connesso ad un solo provider (single-homed), che non fornisce connettività ad altri AS. È il caso tipico di una rete aziendale o universitaria.
- **Multihomed AS:** AS connesso a più provider, senza però fornire transito ad altri.

---

<sup>1</sup>WHOIS è un archivio pubblico che raccoglie informazioni sulla titolarità e sull'assegnazione di risorse e di rete.

- **Transit AS:** AS che offre connettività ad altri sistemi autonomi, permettendo loro di scambiare traffico. Gli ISP operano tipicamente come transit AS.

Un'altra classificazione si basa sul ruolo gerarchico ricoperto all'interno dell'ecosistema di Internet:

- **Tier 1:** AS che può raggiungere qualsiasi altra rete senza dover acquistare transito da altri AS. Hanno accordi di peering reciproci con gli altri tier di livello 1.
- **Tier 2:** AS che acquista transito (da tier 1), ma può anche offrire servizi a clienti e stabilire peering (offre servizi a tier 3 e fa da peering con altri tier 2).
- **Tier 3:** AS che acquista connettività esclusivamente da altri provider e non fornisce servizi di transito.

Infine, secondo le linee guida dell'IETF (RFC 1930), un Autonomous System è definito anche in base all'autonomia decisionale rispetto alle politiche di routing. Questa indipendenza costituisce uno dei principali motivi per cui un'organizzazione può richiedere un ASN.

# Capitolo 2

## Protocollo di routing BGP 4.0

Con questo capitolo capiremo tutto sul funzionamento e sulle caratteristiche del protocollo di routing esterno BGP.

### 2.1 Cos'è il routing

Il routing è il processo attraverso il quale i router determinano il percorso migliore per inviare pacchetti IP da un host sorgente a uno destinatario. Esistono 2 tipi principali di routing:

- Routing interno (Interior Gateway Protocol (IGP)): gestisce il traffico di rete all'interno di uno stesso AS.
- Routing esterno (EGP): gestisce lo scambio di informazioni tra AS diversi.

BGP è un protocollo di routing EGP di tipo *path-vector*, in cui ogni informazione veicolata include il percorso completo di AS attraversati, utile per evitare loop e applicare regole di policy di instradamento.

### 2.2 Nascita del protocollo BGP

Il protocollo BGP fu sviluppato alla fine degli anni '80 per superare i limiti del precedente protocollo di routing di tipo EGP. La prima versione: BGP-1, fu specificata

tramite RFC 1105<sup>[23]</sup> nel 1989 da Yakov Rekhter (IBM) e Kirk Lougheed (Cisco) e venne implementata già su router Cisco e il backbone NSFNET.<sup>[6]</sup> BGP-1 introdusse il concetto di protocollo *path vector*.

### Versioni intermedie: BGP-2 e BGP-3

Con l'evoluzione di Internet e l'espansione del backbone NSFNET, fu necessario migliorare le funzionalità iniziali di BGP-1.

- **BGP-2** fu descritto nell'RFC 1163<sup>[15]</sup> (1990). Questa versione migliorava la gestione delle sessioni TCP tra i peer BGP, raffinando il meccanismo di trasporto dei messaggi e introducendo una maggiore stabilità nella selezione del percorso.
- **BGP-3** arrivò nel 1991 con l'RFC 1267<sup>[16]</sup>. In questa versione si consolidarono le basi del protocollo, introducendo formalmente i messaggi UPDATE, NOTIFICATION, KEEPALIVE e OPEN nel formato che verrà poi mantenuto anche nella versione successiva. BGP-3 fu largamente utilizzato fino all'introduzione del Classless Inter-Domain Routing (CIDR).

### Il salto a BGP-4: CIDR e scalabilità

La vera rivoluzione arrivò con BGP-4, introdotto inizialmente nell'RFC 1654<sup>[21]</sup> (1994), poi aggiornato da RFC 1771 (1995), e infine formalizzato nell'attuale RFC 4271<sup>[22]</sup> (2006).

La caratteristica distintiva di BGP-4 fu l'introduzione del supporto al CIDR, che permetteva di:

- Aggregare efficacemente i prefissi IP,
- Ottenere una drastica riduzione delle dimensioni delle tabelle di routing globali,
- Assicurare la scalabilità necessaria per Internet, contrastando la crisi imminente degli indirizzi IPv4.



Non solo, BGP-4 offrì anche un meccanismo robusto per la gestione di attributi di routing personalizzati (come LOCAL\_PREF, MED, AS\_PATH), il che diede agli operatori la capacità di attuare un routing basato su policy su scala mondiale.

### 2.2.1 Sostituzione di EGP

Prima dell'introduzione di BGP, l'unico protocollo esterno per l'instradamento tra AS era l'**EGP (Exterior Gateway Protocol)**, formalizzato nella RFC 904<sup>[26]</sup> (1984). EGP era un protocollo di tipo *distance-vector* molto semplice, progettato per operare in una struttura gerarchica a “stella” centrata sull'AS 1 (Autonomous System principale dell'ARPANET/NSFNET). Esso permetteva esclusivamente di segnalare la raggiungibilità delle reti, senza fornire informazioni sui percorsi o sulle politiche da adottare per instradare il traffico.

Con l'espansione di Internet e l'adozione di topologie a maglia, EGP rivelò rapidamente i suoi limiti:

- **Rigidità topologica:** incapace di operare in reti con interconnessioni multiple tra AS diversi.
- **Assenza di path information:** veniva indicata solo la raggiungibilità, senza dettagli sul percorso.
- **Scarsa scalabilità:** non supportava CIDR, né aggregazione dei prefissi IP.

In risposta a queste esigenze, nel 1989 fu introdotto BGP-1, che adottava un meccanismo path-vector con l'attributo AS-PATH per evitare loop e consentire l'applicazione di policy amministrative.

Il passaggio da EGP a BGP fu graduale tra il 1989 e il 1995, periodo durante il quale i principali backbone — incluso NSFNET — migrarono verso BGP-3 e poi BGP-4.

## 2.3 Caratteristiche protocollo BGP

In questa sezione andiamo a vedere alcune caratteristiche del protocollo BGP, che tipo di protocollo è, dove opera e in che modalità, per poi nella sezione successiva

analizzare effettivamente come avviene il suo funzionamento. Per comprendere appieno le caratteristiche del protocollo BGP è utile richiamare i principi dei modelli di riferimento ISO/OSI e TCP/IP, che ne costituiscono il contesto architetturale.

### 2.3.1 Modello ISO/OSI e TCP/IP

La progettazione dei protocolli di rete si basa su una struttura a strati, concettualizzata nei modelli di riferimento **ISO/OSI** e **TCP/IP**. Ogni livello di questi modelli svolge un compito specifico e si interfaccia direttamente con i livelli adiacenti, attraverso un meccanismo di scambio di dati chiamato *incapsulamento*.

Durante la trasmissione, ogni volta che un messaggio attraversa un livello verso il basso, viene arricchito con un *header* (intestazione) contenente le informazioni necessarie per il funzionamento del protocollo di quel livello. L'unica eccezione è rappresentata dal primo e dall'ultimo livello. In fase di ricezione, il messaggio risale la pila protocollare: ogni livello rimuove il proprio header e interpreta i dati contenuti, secondo la logica del *decapsulamento*.

Ogni livello può supportare più protocolli, ma ciascun protocollo appartiene esclusivamente al livello in cui opera. L'**entità** è l'unità funzionale all'interno di un livello che implementa uno specifico protocollo. Di conseguenza, il numero di entità presenti in un livello coincide con il numero di protocolli gestiti. Durante la comunicazione, è sempre una sola entità per livello ad aggiungere o rimuovere l'header del proprio protocollo.

Le entità omologhe situate agli stessi livelli dei due nodi comunicanti prendono il nome di *peer entities*, mentre i livelli stessi sono detti *peer levels*. Le informazioni scambiate tra livelli adiacenti sono chiamate Protocol Data Unit (PDU), che includono i dati e l'eventuale header del livello corrente.

Ogni entità è identificata univocamente da un indirizzo detto Service Access Point (SAP), che consente la comunicazione tra livelli differenti.

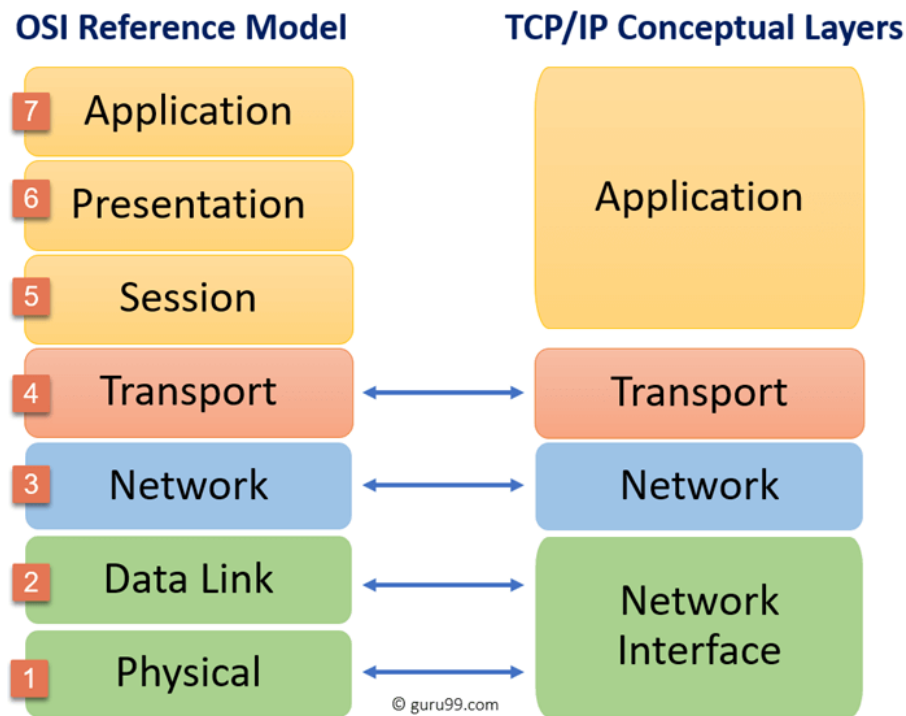
I protocolli di rete si distinguono inoltre in:

- **Connectionless**, che privilegiano la velocità di trasmissione e non instaurano un canale persistente (es. UDP);

- **Connection-oriented**, che instaurano una connessione stabile per garantire affidabilità (es. TCP).

Dal punto di vista architetturale, il modello **TCP/IP** rappresenta lo standard *de facto*, utilizzato nella rete Internet fin dalle sue origini. Il modello **ISO/OSI**, invece, costituisce lo standard *de jure* proposto dall'ISO.<sup>[8]</sup>

La struttura e la corrispondenza tra i due modelli sono illustrate nella figura seguente, che evidenzia i livelli funzionali e la loro equivalenza logica:



**Figura 2.1:** Modello ISO/OSI e modello TCP/IP

Di seguito si riporta una breve descrizione di cosa si occupa ciascun livello:

- **Physical:** definisce le caratteristiche fisiche del mezzo di trasmissione (cavi, segnali elettrici/ottici, connettori) e la codifica dei bit.
- **Data Link:** gestisce l'incapsulamento dei dati in frame, il controllo degli errori a livello di collegamento e l'indirizzamento fisico (MAC address).

- **Network:** si occupa dell'instradamento dei pacchetti tra reti differenti, includendo il calcolo del percorso e l'indirizzamento logico (es. indirizzi IP).
- **Transport:** garantisce il trasferimento dei dati end-to-end, controllando affidabilità, segmentazione, riordinamento e gestione della congestione (es. TCP e UDP).
- **Session** (solo modello OSI): gestisce l'instaurazione, il mantenimento e la terminazione delle sessioni di comunicazione tra applicazioni.
- **Presentation** (solo modello OSI): si occupa della sintassi e della semantica delle informazioni trasmesse, includendo cifratura, compressione e traduzione dei dati.
- **Application:** fornisce servizi di rete direttamente alle applicazioni utente (es. HTTP, FTP, SMTP, DNS).

Per quanto riguarda i dispositivi fisici associati ai vari livelli:

- La **scheda di rete** (NIC) è un apparato di livello 1 (Physical).
- Uno **switch** opera a livello 2 (Data Link).
- Un **router** lavora a livello 3 (Network).
- Un **host**, come un computer personale, implementa tutti i livelli, fino al livello 7 (Application) nel modello OSI.

### 2.3.2 Collocazione architetturale e modalità operative di BGP

Dopo aver analizzato i modelli di riferimento ISO/OSI e TCP/IP, è possibile inquadrare il BGP nel contesto architetturale delle reti e descriverne le principali caratteristiche operative. BGP è un protocollo di routing esterno di tipo *path-vector*, progettato per lo scambio di informazioni di raggiungibilità tra AS differenti. Opera al livello **Application** sia nel modello ISO/OSI sia in quello TCP/IP e utilizza una modalità di comunicazione **connection-oriented**, basata su una connessione **TCP** stabile (porta 179) per garantire affidabilità, ordinamento e controllo degli errori

nella consegna dei messaggi. A differenza dei protocolli di routing interno, BGP applica criteri di instradamento non solo tecnici (costo, distanza, latenza, banda), ma anche amministrativi (le policy), consentendo agli amministratori di un AS di stabilire con precisione quali rotte accettare, rifiutare o preferire, in base a politiche economiche, di sicurezza o di accordi di peering.<sup>[17]</sup>

### 2.3.3 Il ruolo del router e le tabelle di routing

Il router è il dispositivo di rete incaricato di instradare i pacchetti IP tra reti differenti. Per svolgere questa funzione, mantiene al suo interno strutture dati chiamate **tabelle di routing**, che contengono le informazioni necessarie a determinare il *next hop*, ossia il prossimo nodo a cui inoltrare un pacchetto destinato a una certa rete.

Una voce tipica di tabella di routing comprende:

- l' **IP di rete** (es. 192.168.0.0/24), che identifica l'insieme di indirizzi raggiungibili;
- la **subnet mask**, che specifica la dimensione del prefisso;
- il **next hop**, ovvero l'indirizzo IP del router di transito successivo;
- l'**interfaccia di uscita** locale, usata per inoltrare il pacchetto verso il next hop;
- eventuali **metriche o attributi** che descrivono la qualità o la preferenza della rotta.

È importante distinguere tra due concetti:

- **Tabella di routing** (*Routing Information Base (RIB)*): contiene tutte le rotte conosciute dal router, apprese sia tramite protocolli di routing (es. OSPF, BGP) sia tramite configurazioni statiche.
- **Tabella di forwarding** (*Forward Information Base (FIB)*): è una versione ottimizzata della tabella di routing, mantenuta spesso direttamente nell'hardware per garantire l'inoltro ad alta velocità. Mentre la tabella di routing può contenere più alternative per la stessa destinazione, la FIB conserva solo la rotta effettivamente selezionata (la *best path*).

In questo modo, i protocolli di routing come BGP non instradano i pacchetti direttamente, ma aggiornano le tabelle di routing del router. Sarà poi il meccanismo di forwarding a occuparsi dell'inoltro vero e proprio dei pacchetti, basandosi sulla FIB.

### 2.3.4 Le tabelle di routing in BGP

Per descrivere in modo chiaro il funzionamento di BGP, è utile distinguere tre insiemi logici di tabelle di routing che vengono fatti dal protocollo:

- **Adj-RIBs-In:** contengono tutte le rotte apprese dai peer BGP tramite i messaggi UPDATE, ancora prima che vengano applicati filtri o politiche di selezione;
- **Loc-RIB:** rappresenta la tabella BGP locale, in cui confluiscono le rotte selezionate come “migliori” dal Decision Process;
- **Adj-RIBs-Out:** raccolgono le rotte che, dopo l'applicazione delle policy di esportazione, sono pronte per essere annunciate ai peer.

Queste strutture non corrispondono a implementazioni fisiche obbligatorie nei router, ma costituiscono un modello logico standardizzato, utilizzato per descrivere come le informazioni passano da un peer all'altro. In questo modo è più semplice comprendere su quali insiemi di rotte agiscono il processo decisionale e le politiche di filtro o manipolazione.

## 2.4 Funzionamento BGP

In questa sezione analizziamo nel dettaglio come il protocollo BGP adempie al suo compito principale: permettere il routing tra i diversi AS, selezionando il percorso più vantaggioso secondo criteri sia tecnici sia amministrativi.<sup>[22]</sup>

### 2.4.1 Tipologie di sessione

BGP stabilisce connessioni affidabili tra router tramite **TCP** (porta 179), demandando al livello transport le funzioni di controllo degli errori, ordinamento e ritrasmissione. Le connessioni BGP, chiamate *sessioni*, possono essere di due tipi:

- **eBGP (External BGP)**: instaurate tra router appartenenti a AS differenti. Vengono utilizzate per scambiare informazioni di raggiungibilità IP in ambito interdominio, secondo lo schema CIDR.
- **iBGP (Internal BGP)**: instaurate tra router dello stesso AS. Servono a distribuire internamente le informazioni acquisite dai peer esterni.

### 2.4.2 Caratteristiche del protocollo path-vector

BGP è un protocollo di routing esterno di tipo **path-vector**, evoluzione dei protocolli distance-vector. Invece di propagare solo una distanza numerica, BGP include in ogni annuncio l'intera sequenza di AS (**AS\_PATH**) da attraversare per raggiungere una destinazione.

Questo approccio presenta due vantaggi principali:

- **Prevenzione dei cicli**: se un router riceve un annuncio che contiene già il proprio AS nel campo **AS\_PATH**, quell'annuncio viene scartato, evitando loop di instradamento.
- **Supporto alle policy**: la lista di AS permette di applicare criteri non solo tecnici ma anche amministrativi, in base a rapporti economici o di peering.

### 2.4.3 Policy di routing

BGP è un protocollo **policy-based**: le rotte non vengono accettate o pubblicizzate solo in base a metriche di costo, ma in base a policy locali stabilite dall'amministratore.

- **Export policy**: un AS decide quali reti IP (identificate in notazione CIDR) rendere note ai vicini. Ad esempio, una rete aziendale (Stub AS) può annunciare solo i propri prefissi interni senza offrire transito.
- **Import policy**: un AS può rifiutare rotte che passano da sistemi indesiderati (es. per motivi di sicurezza o commerciali) oppure preferire rotte provenienti da determinati peer.

Questo rende BGP molto flessibile ma introduce anche maggiore complessità nella gestione delle tabelle di routing e un maggior consumo di memoria nei router, poiché è necessario mantenere molteplici informazioni di percorso.

#### 2.4.4 Formato dei messaggi

Tutti i messaggi BGP hanno una struttura comune, composta da:

- **Marker** (16 byte): usato originariamente per autenticazione, oggi spesso a valore fisso.
- **Length** (2 byte): indica la lunghezza totale del messaggio (da 19 a 4096 byte).
- **Type** (1 byte): specifica il tipo di messaggio, tra i seguenti:
  1. **OPEN**: il primo messaggio scambiato, utilizzato per instaurare la sessione. Contiene numero di AS, Hold Time, BGP Identifier ed eventuali parametri opzionali.
  2. **UPDATE**: trasporta annunci e ritiri di Network Layer Reachability Information (NLRI) (ovvero le informazioni di raggiungibilità delle reti IP identificate da prefissi in notazione CIDR) insieme ai relativi attributi di percorso.
  3. **NOTIFICATION**: segnala errori e causa l'immediata chiusura della sessione.
  4. **KEEPALIVE**: inviato periodicamente per confermare la validità della sessione in assenza di aggiornamenti.

#### 2.4.5 Gestione delle sessioni (FSM)

L'instaurazione e la gestione di una sessione BGP sono regolate da una **macchina a stati finiti** (Finite State Machine, FSM) definita nell'RFC 4271. Essa descrive il comportamento di un router BGP in risposta a eventi interni (scadenza di timer) o esterni (ricezione di messaggi, esito delle connessioni TCP).

Gli stati principali sono sei:

- **Idle**: Stato iniziale, in cui il router non ha ancora connessioni attive. La sessione BGP può essere avviata:



- tramite *Manual Start*, ossia un comando esplicito dell'amministratore di rete;
- tramite *Automatic Start*, cioè l'avvio automatico del processo BGP, ad esempio al riavvio del router.

In entrambi i casi, il router inizializza le risorse necessarie alla sessione: azzerava i contatori (*ConnectRetryCounter*), attiva i timer (*ConnectRetryTimer*) e alloca le strutture di controllo interne. Successivamente:

- se configurato come **peer attivo**, tenta di instaurare una connessione TCP verso il vicino e passa allo stato **Connect**;
  - se configurato come **peer passivo**, rimane in ascolto di connessioni in ingresso e passa allo stato **Active**.
- **Connect**: Il router ha avviato un tentativo di connessione TCP verso il peer e attende il completamento del 3-way handshake. Se la connessione fallisce o scade il *ConnectRetryTimer*, viene avviato un nuovo tentativo: a seconda della configurazione e degli eventi, il router può rimanere in **Connect** o transitare in **Active**. In caso di successo della connessione TCP, si procede con l'invio del messaggio *OPEN* (se l'opzione *DelayOpen* è disabilitata) oppure con l'attesa dello scadere del *DelayOpenTimer*.
  - **Active**: Stato in cui il router non ha ancora stabilito una connessione TCP valida. In questa fase rimane in ascolto di connessioni in ingresso dal peer. Se entro lo scadere del *ConnectRetryTimer* non arriva alcuna connessione, il router torna in **Connect** per avviare un nuovo tentativo attivo di connessione.
  - **OpenSent**: in questo stato il router ha inviato un messaggio *OPEN* e attende l'*OPEN* dal peer. Se il messaggio ricevuto è valido, vengono negoziati i parametri della sessione (in particolare l'*HoldTimer* e il *BGP Identifier*) e si passa a *OpenConfirm*. In caso di errore o incompatibilità, viene inviato un *NOTIFICATION* e la sessione viene chiusa.
  - **OpenConfirm**: i peer attendono un *KEEPALIVE* di conferma. Se il messaggio arriva entro l'*HoldTimer*, si passa allo stato finale *Established*. Se invece il

timer scade prima, viene inviato un *NOTIFICATION* con codice *Hold Timer Expired*.

- **Established:** la sessione è attiva e operativa. In questo stato vengono scambiati messaggi *UPDATE*, *KEEPALIVE* e *NOTIFICATION*. La ricezione di *KEEPALIVE* o *UPDATE* resetta l'*HoldTimer*, mantenendo viva la connessione. Qualsiasi errore o scadenza dei timer comporta la chiusura della sessione e il ritorno allo stato *Idle*.

Questa struttura a stati consente a BGP di garantire un comportamento prevedibile e robusto in ogni fase della connessione, riducendo ambiguità e rendendo il protocollo resistente a errori e disconnessioni temporanee.

## 2.4.6 Path attributes

Ogni annuncio BGP (UPDATE) include attributi che specificano le proprietà della rotta. Essi si classificano in:

- **Well-known mandatory:** sempre presenti (es. *ORIGIN*, *AS\_PATH*, *NEXT\_HOP*);
- **Well-known discretionary:** riconosciuti da tutti ma non obbligatori (es. *LOCAL\_PREF*);
- **Optional transitive:** opzionali, ma se non riconosciuti vengono comunque propagati (es. *COMMUNITY*);
- **Optional non-transitive:** opzionali e non propagati se non riconosciuti (es. *MED*);
- **Partial:** optional-transitive che non sono stati riconosciuti da un router e vengono inoltrati con un marcatore speciale.

Tra gli attributi più importanti:

Attributo	Classificazione	Descrizione
ORIGIN	Well-known mandatory	Specifica l'origine della rotta: <i>IGP</i> se appresa internamente, <i>EGP</i> se tramite un protocollo esterno, <i>INCOMPLETE</i> se l'origine non è determinata (es. redistribuzione da altre fonti).
AS_PATH	Well-known mandatory	Elenca in sequenza gli Autonomous System attraversati dalla rotta. È usato sia per la prevenzione dei loop (se un AS vede se stesso nella lista, scarta l'update) sia come criterio di selezione preferendo i percorsi con meno AS.
NEXT_HOP	Well-known mandatory	Indirizzo IP del router BGP da utilizzare come prossimo destinatario per raggiungere la destinazione finale. Determina concretamente a chi inoltrare il traffico.
LOCAL_PREF	Well-known discretionary	Valore propagato solo all'interno dello stesso AS (iBGP). Indica la preferenza per le rotte in uscita: percorsi con valore più alto sono scelti come prioritari.
ATOMIC_AGGREGATE	Well-known discretionary	Segnala che la rotta è stata ottenuta tramite un processo di aggregazione, e quindi alcune informazioni più specifiche (es. dettagli di prefissi originari) possono essere andate perse.
MED	Optional non-transitive	Multi Exit Discriminator: viene usato per suggerire quale collegamento preferire quando esistono più punti di ingresso verso lo stesso AS adiacente. Valori più bassi sono considerati migliori, ma l'attributo non è sempre rispettato da tutti i peer.
COMMUNITY	Optional transitive	Etichette logiche che consentono di raggruppare insieme di rotte e applicare politiche comuni (es. blocco, preferenza, redistribuzione). Sono molto usate per semplificare la gestione del routing.
AGGREGATOR	Optional transitive	Identifica l'AS e il router che hanno effettuato un'aggregazione di rotte. È utile per tracciare l'origine del processo di aggregazione e garantire trasparenza nell'instradamento.

**Tabella 2.1:** Principali attributi BGP e loro classificazione

### 2.4.7 Processo

Il **Decision Process** costituisce il cuore di BGP ed è responsabile della selezione della rotta “migliore” verso una determinata destinazione, quando ne esistono più di una. A differenza dei protocolli interni (IGP), che si basano su metriche uniche come costo o latenza, BGP adotta un approccio multilivello, in cui criteri amministrativi e tecnici sono combinati in modo gerarchico.<sup>[22]</sup>

Il processo decisionale si applica a ogni informazione di raggiungibilità (NLRI) ricevuta tramite messaggi **UPDATE**. Quando un router BGP riceve per la prima volta un **UPDATE** da peer esterni o interni (quindi da router con cui non aveva ancora una conoscenza diretta delle reti raggiungibili) inserisce tutte le rotte candidate nella propria *Adj-RIBs-In*. Da questo insieme viene poi selezionata una sola rotta preferita, che sarà utilizzata per l'instradamento effettivo e potenzialmente annunciata ad altri peer.

Il processo si articola in tre fasi principali:

1. **Assegnazione della preferenza locale:** Ogni rotta viene inizialmente valutata in base a criteri interni all'AS. In questa fase è fondamentale l'attributo **LOCAL\_PREF**, che consente all'amministratore di indicare quale uscita preferire in modo indipendente dai parametri tecnici. Questo garantisce che, anche in presenza di molte alternative, la scelta rifletta le politiche di instradamento locali (es. preferire collegamenti verso clienti anziché verso peer o provider).
2. **Selezione della rotta migliore:** Dopo l'applicazione delle policy, le rotte rimaste vengono confrontate seguendo un ordine di priorità ben definito dallo standard. In particolare:
  - (a) scegliere la rotta con **LOCAL\_PREF** più alto;
  - (b) a parità, preferire quella con **AS\_PATH** più corto;
  - (c) a parità, preferire il valore di **ORIGIN** più favorevole (IGP < EGP < INCOMPLETE);
  - (d) se le rotte provengono dallo stesso AS adiacente, scegliere quella con **MED** più basso;

- (e) preferire rotte apprese tramite **eBGP** rispetto a quelle iBGP;
- (f) scegliere la rotta il cui **NEXT\_HOP** è raggiungibile con il **costo IGP minore**;
- (g) in caso di ulteriore parità, selezionare la rotta proveniente dal router con **BGP Identifier** più basso;
- (h) come criterio finale, utilizzare l'indirizzo IP più basso del peer.

Questo meccanismo garantisce che, anche quando più router sconosciuti iniziano a scambiarsi informazioni, ognuno di essi arrivi a determinare una sola rotta coerente e stabile per ogni destinazione.

3. **Disseminazione della rotta:** La rotta vincente viene installata nella *Loc-RIB* (tabella BGP locale). In base alle export policies, può poi essere inserita nelle *Adj-RIBs-Out* e annunciata ai peer tramite messaggi **UPDATE**. In questa fase possono essere applicate tecniche di manipolazione come l'aggregazione dei prefissi IP o l'*AS\_PATH prepending*, strumenti utili per influenzare le scelte di instradamento dei router vicini.

In questo modo, BGP non opera come un classico protocollo di “shortest path”, ma come un sistema flessibile e distribuito di negoziazione delle rotte, in cui le politiche commerciali (cliente, provider, peer) contano quanto i parametri tecnici. La convergenza emerge dal fatto che ogni router, anche senza conoscenze pregresse della topologia globale, applica localmente lo stesso processo decisionale definito dallo standard, ottenendo così una rete interdominio coerente e priva di cicli.

## 2.4.8 Considerazioni finali

Grazie alla combinazione di meccanismi tecnici (*AS\_PATH*, *NEXT\_HOP*, *ORIGIN*), controlli di robustezza (*FSM*, *HoldTimer*, *NOTIFICATION*) e criteri amministrativi (*import/export policies*), BGP rappresenta il cuore del routing interdominio su Internet. È un protocollo pensato non per la rapidità di convergenza, ma per la stabilità e la possibilità di esprimere politiche complesse, che riflettono le relazioni economiche e tecniche tra i numerosi AS.



# Capitolo 3

## Metodologie di attacco e prevenzione del protocollo BGP

In questo capitolo andiamo a vedere quali vulnerabilità del protocollo BGP possono essere sfruttate, come e soprattutto come prevenire che ciò avvenga.

### 3.1 BGP prefix hijacking

Una delle vulnerabilità più note e critiche di BGP è il cosiddetto **Prefix Hijacking**, ovvero il dirottamento di blocchi di indirizzi IP (prefissi) da parte di un AS malevolo o mal configurato. Questo attacco sfrutta una debolezza intrinseca del protocollo: l'assenza di un meccanismo nativo di autenticazione dell'origine dei prefissi annunciati<sup>[31]</sup>.

#### 3.1.1 Descrizione dell'attacco.

Nel funzionamento ordinario, un router BGP annuncia ai propri peer i prefissi IP che può raggiungere, insieme agli attributi di percorso (**AS\_PATH**, **NEXT\_HOP**, ecc.). Nel caso di prefix hijacking, un AS trasmette intenzionalmente (o a seguito di un errore) un **UPDATE** che include come raggiungibile un prefisso IP non legittimamente assegnato. Gli altri AS, non disponendo di strumenti di validazione, accettano e

propagano l'annuncio, inserendolo nelle rispettive *Adj-RIBs-In*. Se, in base al processo decisionale, la rotta malevola risulta preferibile a quella legittima (ad esempio per un **AS\_PATH** più corto o per l'annuncio di un sottoprefisso), essa verrà selezionata come *best path* e diffusa ulteriormente.

### 3.1.2 La regola del Longest Prefix Match

Per comprendere il *Subprefix hijack* è necessario richiamare la regola del **Longest Prefix Match (LPM)**. Un router, quando deve decidere come instradare un pacchetto IP, sceglie la rotta più specifica nella propria tabella di routing, ossia quella con la subnet mask più alta.

- La rotta 192.0.2.0/23 copre 512 indirizzi (da 192.0.2.0 a 192.0.3.255).
- La rotta 192.0.2.0/24 copre solo 256 indirizzi (da 192.0.2.0 a 192.0.2.255), quindi è più specifica.

Se nella tabella sono presenti entrambe, il router sceglierà sempre /24 per un pacchetto destinato a 192.0.2.55, anche se /23 include comunque quell'indirizzo. Questa proprietà, normalmente utile per ottimizzare l'instradamento, diventa l'elemento chiave sfruttato negli attacchi di tipo Subprefix hijack.

### 3.1.3 Tipologie di BGP Prefix Hijacking

Si distinguono due tipologie principali di hijack:

- **Regular prefix hijack:** l'AS attaccante annuncia lo stesso prefisso dell'AS legittimo, presentandosi come origin AS. La propagazione del falso annuncio provoca una parziale deviazione del traffico, dipendente dalle politiche di routing adottate dai diversi AS.

**Esempio pratico:** Immaginiamo di essere un AS, l'AS100, che possiede il blocco IP 203.0.113.0/24. Normalmente, i suoi router annunciano questo prefisso ai peer BGP, così il mondo intero sa che per raggiungere 203.0.113.x deve passare da AS100. Ora entra in gioco un attaccante, AS666, che anche se non possiede il blocco IP 203.0.113.0/24, può inviare un UPDATE in cui



annuncia di raggiungerlo. Se il percorso verso AS666 è considerato più conveniente (per esempio grazie a un **AS\_PATH** più corto), alcuni peer inizieranno a instradare il traffico verso di lui, causando una deviazione parziale.

- **Subprefix hijack**: l'attaccante annuncia un prefisso più specifico rispetto a quello legittimo (ad esempio, 192.0.2.0/24 invece di 192.0.2.0/23). A causa della regola del *Longest Prefix Match*, quasi tutto il traffico destinato a 192.0.2.x verrà instradato verso l'attaccante, mentre il legittimo AS continuerà a ricevere solo il traffico diretto a 192.0.3.x. Questa variante è la più pericolosa, in quanto consente un dirottamento pressoché completo.

## 3.2 BGP route leaking

La seconda vulnerabilità di BGP che andremo ad analizzare è detta **BGP Route Leaking**. A differenza del *prefix hijacking*, in cui un AS annuncia prefissi non posseduti, nel *route leak* i prefissi sono legittimi ma vengono diffusi in modo improprio, raggiungendo domini di instradamento ai quali non erano destinati. Questa anomalia può deviare grandi quantità di traffico su cammini non ottimali o addirittura inaffidabili, causando disservizi difficili da diagnosticare e potenzialmente con impatto globale<sup>[27;10]</sup>.

### 3.2.1 Descrizione dell'attacco

Un *BGP route leak* si verifica quando un AS inoltra rotte ricevute da un provider o da un peer verso un altro provider o peer, invece di limitarle ai propri clienti. Questo comportamento viola il principio del *valley-free routing*, che impone che:

- le rotte ricevute da **provider** o **peer** vengano propagate **solo verso clienti**;
- le rotte ricevute da **clienti** possono essere propagate verso qualsiasi altro AS (provider, peer o altri clienti).

In termini economici, questo modello assicura che un AS non faccia involontariamente da punto di transito non compensato.

L'RFC 7908 classifica sette tipologie di route leaks, tra cui le più comuni sono:

- **Provider → Provider:** un AS riceve rotte da un provider e le pubblicizza a un altro provider;
- **Provider → Peer:** un AS inoltra a un peer rotte apprese da un provider;
- **Peer → Peer:** un AS diffonde a un peer rotte ricevute da un altro peer;
- **Peer → Provider:** un AS pubblicizza a un provider rotte che aveva appreso da un altro peer.

### 3.3 BGP session hijacking

Terzo e ultimo attacco al protocollo che tratteremo è il *BGP session hijacking*. Il BGP session hijacking rappresenta una minaccia che agisce direttamente sulla connessione tra due peer BGP, compromettendo la sessione TCP sottostante e mettendo a rischio la stabilità e la sicurezza del protocollo.<sup>[20]</sup>

#### 3.3.1 Descrizione dell'attacco

Il *BGP session hijacking* è una delle vulnerabilità più critiche del BGP. Questo tipo di attacco si verifica quando un attore malevolo riesce a prendere il controllo di una sessione TCP già stabilita tra due router BGP, sfruttando la mancanza di meccanismi di autenticazione robusti nel protocollo. Poiché BGP si basa su TCP (porta 179), un attaccante che riesce a predire o intercettare i numeri di sequenza TCP può inserire pacchetti falsi nella connessione, con la possibilità di iniettare o manipolare messaggi BGP legittimi.

#### 3.3.2 Dinamica dell'attacco

L'attacco si articola generalmente in tre fasi principali:

1. **Identificazione della sessione:** l'attaccante individua i due router coinvolti in una connessione BGP attiva.

2. **Predizione o intercettazione dei numeri di sequenza TCP:** attraverso sniffing del traffico o tentativi di brute force, l'attaccante ottiene i valori corretti per inserirsi nella comunicazione.
3. **Iniezione di pacchetti malevoli:** una volta presa la sincronizzazione, l'attaccante può iniettare messaggi BGP alterati, forzando i peer a modificare la loro tabella di routing.

Un attacco di questo tipo può portare a:

- interruzione della connessione BGP tra i router, con conseguente instabilità della rete;
- manipolazione delle rotte, con possibilità di deviare il traffico verso destinazioni non legittime;
- esposizione del traffico intercettato a tecniche di analisi o modifica.

## 3.4 BGPsec

Il protocollo **BGPsec** è un'estensione del protocollo BGP sviluppata dall'IETF con l'obiettivo di migliorare la sicurezza nella propagazione delle informazioni di routing. A differenza di BGP tradizionale, che si limita a propagare gli annunci senza alcuna forma di autenticazione crittografica, BGPsec introduce un sistema di validazione del percorso basato su firme digitali, permettendo così la verifica crittografica di ogni hop all'interno dell'AS-PATH<sup>[14]</sup>.

### 3.4.1 Come funziona BGPsec

Dal punto di vista tecnico, BGPsec sostituisce il tradizionale attributo **AS\_PATH** con una nuova struttura, chiamata **BGPsec\_Path**, che contiene due componenti principali:

- **Secure\_Path Segment:** registra l'identità (ASN) di ciascun AS che ha propagato l'annuncio.
- **Signature Segment:** contiene la firma digitale generata da quell'AS sull'intero annuncio ricevuto, calcolata con l'algoritmo Elliptic Curve Digital Signa-

ture Algorithm (ECDSA) su curva P-256 e hash SHA-256, come specificato in RFC 8608<sup>[28]</sup>

Per un approfondimento teorico e matematico sull'algoritmo ECDSA si rimanda all'Appendice A.

Il flusso operativo è il seguente:

1. L'AS originator di un prefisso genera un messaggio **UPDATE** e lo firma con la propria chiave privata, associata a un certificato valido della RPKI.
2. Quando un AS riceve l'annuncio:
  - (a) verifica tutte le firme presenti nei segmenti precedenti utilizzando le chiavi pubbliche distribuite tramite RPKI;
  - (b) se l'annuncio è valido, aggiunge il proprio ASN in un nuovo **Secure\_Path Segment**;
  - (c) calcola e allega la propria firma digitale in un **Signature Segment**.
3. L'annuncio così aggiornato viene propagato al peer successivo, che ripete lo stesso processo.

In questo modo ogni router non solo apprende il percorso di instradamento, ma può anche verificare crittograficamente che:

1. l'annuncio sia stato originato da un AS legittimo (validazione dell'origination);
2. l'AS-PATH non sia stato manomesso lungo il percorso (path validation).

Tale meccanismo riduce drasticamente la possibilità che un attaccante possa inserire ASN falsi o alterare il path, poiché qualsiasi modifica non autorizzata invaliderebbe la catena di firme.

### **3.4.2 Vulnerabilità risolte**

BGPsec nasce come risposta alle principali vulnerabilità storiche di BGP, in particolare:

- **Prefix hijacking:** grazie alla validazione crittografica, diventa più difficile per un AS non autorizzato annunciare prefissi IP di cui non è legittimo proprietario.
- **Session hijacking:** sebbene questo attacco agisca a livello di trasporto (TCP), BGPsec mitiga gli effetti legati alla manipolazione degli annunci, poiché un attaccante non può modificare la catena di firme senza invalidarla.
- **Route leaks e man-in-the-middle:** la verifica delle firme lungo il path riduce la possibilità che un AS introduca rotte false o modifichi l'AS-PATH senza essere rilevato.

Nonostante ciò, BGPsec non elimina tutte le debolezze di BGP: rimangono ad esempio problematiche legate alla disponibilità (DoS) e all'adozione pratica su larga scala.

## 3.5 BGP RPKI

La **RPKI** è una tecnologia sviluppata dall'IETF per rafforzare la sicurezza del routing interdominio. Il suo obiettivo principale è contrastare attacchi che sfruttano l'assenza di autenticazione in BGP, come il *prefix hijacking*, fornendo un meccanismo crittografico che permette di stabilire quali AS siano legittimati ad annunciare determinati prefissi IP. RPKI rappresenta dunque il fondamento della cosiddetta *origin validation* e costituisce la base crittografica anche per estensioni più avanzate come BGPsec.<sup>[13]</sup>

### 3.5.1 Come funziona BGP RPKI

La **RPKI** è un'infrastruttura a chiave pubblica che associa le risorse di rete (prefissi IP e ASN) ai rispettivi detentori legittimi. Ogni allocazione di indirizzi IP e ASN viene certificata mediante certificati X.509 conformi allo standard PKIX<sup>1</sup>, emessi dalle autorità di registrazione regionali (RIR).

---

<sup>1</sup>PKIX (*Public Key Infrastructure X.509*) è uno standard IETF che definisce l'uso dei certificati digitali X.509 per costruire infrastrutture a chiave pubblica (PKI). Specifica formati, algoritmi e procedure di validazione per garantire autenticità e integrità nelle comunicazioni sicure.

Un elemento centrale di RPKI è il *Route Origin Authorization (ROA)*, un oggetto firmato digitalmente che specifica quali ASN sono autorizzati ad annunciare un determinato prefisso. Quando un router riceve un annuncio BGP, può consultare la RPKI per verificare se l'AS originator è autorizzato ad annunciare quel prefisso:

- **Valid:** il prefisso è autorizzato dall'AS specificato nel ROA.
- **Invalid:** l'annuncio non è coerente con i ROA (potenziale hijack).
- **NotFound:** non esiste alcun ROA per il prefisso.

### 3.5.2 Vulnerabilità risolte

RPKI nasce come soluzione per mitigare attacchi legati all'**origin validation**, in particolare:

- **Prefix hijacking:** impedisce a un AS malevolo di annunciare prefissi IP di cui non possiede i diritti.
- **Route leaks:** riduce la probabilità che un annuncio non autorizzato venga accettato globalmente.

Tuttavia, RPKI non valida l'intero percorso (AS-PATH), quindi non impedisce modifiche intermedie o manipolazioni del path; per questo è considerata complementare a BGPsec.

## 3.6 Monitoraggio e rilevamento anomalie

Il protocollo BGP, nella sua versione base, è privo di meccanismi di sicurezza intrinseci, e quindi esposto ad attacchi come session hijacking, route leaking e prefix hijacking. Per questo motivo, osservare il piano di controllo di Internet, cioè i meccanismi con cui i router definiscono e aggiornano le rotte, è essenziale per individuare anomalie e preservare l'affidabilità della rete.

Negli ultimi decenni sono stati sviluppati diversi progetti e piattaforme che raccolgono e distribuiscono dati BGP su scala globale, consentendo di:

- rilevare tempestivamente anomalie nella propagazione degli annunci;

- analizzare eventi di sicurezza come attacchi di hijacking o manipolazioni del path;
- studiare l'evoluzione a lungo termine della stabilità del routing interdominio.

Tra i principali strumenti e dataset disponibili si trovano **BGPStream**, il **CAIDA**, il **Routing Information Service (RIS)** di RIPE NCC e il progetto **Route Views** dell'Università dell'Oregon. Ognuno di essi fornisce dati complementari e metodologie differenti, permettendo così un'analisi più completa e accurata delle dinamiche BGP a livello globale.

### 3.6.1 BGPStream (CAIDA)

**BGPStream** è una piattaforma open-source sviluppata da CAIDA per l'analisi di grandi dataset BGP. Non fornisce dati propri, ma mette a disposizione un framework software che permette di accedere e analizzare, in tempo reale o in modalità storica, i flussi di messaggi BGP (UPDATE e RIB dumps) provenienti da fonti come RIPE RIS e Route Views. Grazie a questa astrazione, i ricercatori possono integrare e confrontare dataset eterogenei con strumenti uniformi, facilitando il rilevamento di anomalie, lo studio di eventi di hijacking e il monitoraggio della stabilità del piano di controllo globale.<sup>[19]</sup>

### 3.6.2 RIPE RIS

Il **RIS** di RIPE NCC stabilisce sessioni BGP con centinaia di router (peer) distribuiti globalmente, chiamati *Route Collectors*. Da queste sessioni vengono generati due insiemi principali di dati:

- **UPDATE streams**: sequenze di messaggi BGP in tempo reale, che mostrano modifiche e anomalie nella propagazione dei prefissi.
- **RIB dumps**: snapshot periodici delle tabelle di routing complete osservate da ciascun collector.

Questi dataset, disponibili pubblicamente dal 2001, costituiscono una risorsa fondamentale per analizzare la diffusione di prefissi, rilevare fenomeni di hijacking e condurre studi longitudinali sul comportamento di BGP.<sup>[25]</sup>

### 3.6.3 Route Views

Il progetto **Route Views**, ospitato presso la University of Oregon, opera in maniera analoga a RIPE RIS, mantenendo sessioni BGP con un ampio insieme di Autonomous System a livello globale. I dati resi disponibili comprendono:

- **BGP UPDATE messages:** archiviati con granularità temporale fine, utili per ricostruire eventi di routing e anomalie.
- **RIB snapshots:** dump regolari delle tabelle di routing complete, che forniscono una visione storica dettagliata dell'evoluzione della connettività inter-AS.

Avviato nel 1997, Route Views rappresenta uno dei dataset storici più longevi e utilizzati per studi accademici e per la rilevazione di anomalie di instradamento su Internet.<sup>[29]</sup>



## Capitolo 4

# Implicazioni degli attacchi BGP nel mondo

Gli episodi di hijacking hanno mostrato come le vulnerabilità del piano di controllo possano produrre effetti ben oltre il contesto locale, con ricadute tecniche, economiche e geopolitiche. Alcuni casi sono diventati emblematici: tra questi, l'hijacking del 2008 da parte di Pakistan Telecom, che rese YouTube inaccessibile a livello globale, e il caso attribuito a China Telecom, spesso richiamato per le implicazioni in termini di sicurezza e sorveglianza. L'analisi di tali incidenti consente di comprendere concretamente l'impatto delle debolezze del maggior protocollo EGP.

### 4.1 BGP prefix hijacking Pakistan Telecom 2008

Il 24 febbraio 2008 si verificò uno degli episodi di hijacking più noti della storia di Internet, quando AS17557 (Pakistan Telecom) annunciò in modo non autorizzato il prefisso 208.65.153.0/24, parte dello spazio di indirizzi appartenente a YouTube (AS36561). L'azione era motivata dalla richiesta del governo pakistano di bloccare l'accesso al sito a livello nazionale, ma l'annuncio fu propagato al provider upstream AS3491 (PCCW Global), che a sua volta lo diffuse all'intera rete globale<sup>[24]</sup>.

### 4.1.1 Timeline dell'incidente

Secondo l'analisi condotta dal RIPE NCC, l'evento ebbe inizio alle 18:47 UTC, quando il prefisso venne originato da AS17557. Nel giro di pochi minuti, gran parte degli AS mondiali preferì la nuova rotta, poiché più specifica rispetto al prefisso legittimo annunciato da YouTube (208.65.152.0/22). Alle 20:07 UTC YouTube tentò di contrastare l'hijack annunciando anch'essa lo stesso /24. Successivamente, dalle 20:18 UTC, pubblicò due prefissi /25 per sfruttare la regola del *longest prefix match* e riprendere il controllo del traffico. Infine, alle 21:01 UTC, PCCW ritirò gli annunci di Pakistan Telecom, riportando la situazione alla normalità<sup>[24]</sup>.

### 4.1.2 Motivazioni dell'attacco

L'iniziativa di Pakistan Telecom non nacque come un attacco deliberato contro YouTube a livello globale, bensì come misura di censura interna. Su richiesta del governo pakistano, l'operatore tentò di rendere inaccessibile il sito all'interno del Paese, annunciando un prefisso più specifico per deviare il traffico nazionale verso un *blackhole*<sup>1</sup>. Tuttavia, l'annuncio non venne confinato al solo ambito domestico: propagato all'upstream provider PCCW, si diffuse rapidamente a livello internazionale. Il risultato fu che il tentativo di censura locale si trasformò in un blackout globale, con effetti non previsti e fuori dal controllo delle autorità pakistane.

### 4.1.3 Analisi tecnica

L'incidente fu un tipico caso di *BGP prefix hijacking*. L'origine non autorizzata di un prefisso più specifico (/24) rese l'annuncio di Pakistan Telecom più attraente rispetto al legittimo /22. Questo comportamento mise in evidenza due criticità:

- la mancanza di meccanismi di validazione sull'origine degli annunci;
- l'assenza di filtri da parte degli upstream provider, che avrebbe impedito la propagazione globale.

---

<sup>1</sup>Nel contesto del routing, un *blackhole* è una destinazione di rete fittizia in cui il traffico viene instradato e poi scartato, senza raggiungere la destinazione finale.

Il caso divenne uno studio di riferimento per l'utilizzo di strumenti di monitoraggio come **RISwhois**, **BGPlay** e **BGPath**, che consentirono di ricostruire in dettaglio la propagazione e la successiva mitigazione dell'incidente.

#### 4.1.4 Implicazioni e lezioni apprese

L'episodio dimostrò come un'azione mirata a livello locale potesse avere effetti imprevisti e su scala globale. Per circa due ore YouTube risultò irraggiungibile in gran parte del mondo, con un impatto immediato su milioni di utenti e sull'affidabilità percepita dei servizi online. Da questo evento emerse con forza la necessità di adottare pratiche di filtraggio più rigorose, nonché lo sviluppo di soluzioni come RPKI e sistemi di rilevamento in tempo reale (es. ARTEMIS) per mitigare futuri episodi di hijacking.

## 4.2 BGP prefix hijacking di China Telecom 2010

Il 8 aprile 2010 si verificò un episodio che coinvolse AS4134 (China Telecom), durante il quale furono annunciati in maniera anomala circa 37.000 prefissi IP appartenenti ad altre reti sparse nel mondo. L'incidente ebbe una durata di circa 15 minuti e generò notevoli preoccupazioni a livello internazionale, non tanto per interruzioni di servizio immediate, quanto per il rischio potenziale di intercettazione e manipolazione del traffico globale<sup>[30]</sup>.

### 4.2.1 Timeline dell'incidente

Secondo le analisi condotte dalla comunità di ricerca e riprese nel rapporto della *US-China Economic and Security Review Commission*, l'hijacking iniziò alle 15:54 UTC, quando China Telecom originò una grande quantità di prefissi non appartenenti al proprio spazio di indirizzamento. Nel giro di pochi minuti, diversi AS al di fuori della Cina instradarono parte del proprio traffico attraverso AS4134. Dopo circa un quarto d'ora gli annunci anomali cessarono, e la situazione tornò progressivamente alla normalità. Nonostante la breve durata, l'evento attirò ampia attenzione mediatica e politica per le sue implicazioni in termini di sicurezza.

## 4.2.2 Motivazioni e controversie

Le reali motivazioni dell'incidente non furono mai chiarite con certezza. Una parte della comunità tecnica lo interpretò come un errore di configurazione o una perdita di controllo del routing. Altri analisti ipotizzarono invece la possibilità di un'azione deliberata finalizzata a deviare traffico sensibile attraverso la Cina. Questa ambiguità alimentò un acceso dibattito internazionale: l'episodio fu citato come esempio della vulnerabilità sistemica del BGP e del rischio che un singolo operatore possa, volontariamente o meno, alterare il flusso globale dei dati.

## 4.2.3 Analisi tecnica

Si trattò di un classico caso di *BGP prefix hijacking*, in cui un AS annuncia prefissi non di sua proprietà. L'ampia scala dell'evento (oltre 37.000 prefissi) evidenziò come anche un'alterazione di breve durata potesse avere impatti significativi sulla topologia del routing interdominio. Dal punto di vista tecnico, l'incidente mise in luce due aspetti critici:

- la propagazione incontrollata di prefissi senza alcuna validazione di origine;
- la possibilità che il traffico venga non solo interrotto, ma anche temporaneamente instradato attraverso percorsi non previsti, aprendo scenari di potenziale intercettazione.

## 4.2.4 Implicazioni e lezioni apprese

L'evento non causò un'interruzione massiva di servizi, ma sollevò preoccupazioni rilevanti in termini geopolitici e di sicurezza. Il fatto che una quantità considerevole di traffico internazionale potesse transitare attraverso la Cina, anche solo per pochi minuti, fu interpretato come un campanello d'allarme sulla fragilità dell'ecosistema di routing. Il caso China Telecom 2010 contribuì a rafforzare il dibattito sulla necessità di meccanismi di validazione come RPKI e di pratiche di filtraggio più rigorose da parte degli operatori, al fine di ridurre il rischio che episodi simili possano ripetersi.

# Capitolo 5

## Tecnologie utilizzate

In questo capitolo, esamineremo le tecnologie scelte e il loro ruolo nella costruzione del progetto.

### 5.1 Ambiente di virtualizzazione

Data l'impossibilità di utilizzare, gratuitamente o a basso costo, dei router in grado di supportare BGPsec e RPKI, utilizzeremo delle VM con sistema operativo Ubuntu server 24.04.2. Su queste VM, monteremo poi dei software (FRRouting per i IR e GoBGP per i BR) per rendere le VM dei router effettivi.

#### 5.1.1 VMware Workstation Pro

Come software di virtualizzazione desktop per contenere le VM utilizziamo VMware Workstation Pro nella sua versione 17.6.3. VMware Workstation Pro, nel 2024 è diventato gratuito nella sua versione per uso personale, e ciò ha contribuito ulteriormente nella sua diffusione, a oggi esso è infatti uno dei software di virtualizzazione desktop più utilizzato.<sup>[12]</sup> Quando creiamo delle VM con VMware Workstation Pro, usiamo la virtualizzazione hardware di tipo 2 (hosted) con full virtualization assistita da hardware.

## Terminologia della virtualizzazione

- **Sistema host:** è la macchina fisica su esegue il sistema operativo principale e che poi ospiterà le macchine virtuali.
- **Sistema guest:** è l'insieme delle risorse hardware e SO che viene eseguito "sopra" il sistema host.
- **Hypervisor o Virtual Machine Monitor (VMM):** è il componente software che crea e manda in esecuzione le VM. Ha il compito di astrarre e rendere disponibili le risorse hardware e svolge i compiti di monitoraggio e sicurezza.

## Virtualizzazione Desktop

La virtualizzazione desktop, è un tipo di virtualizzazione che:

- Consente di utilizzare una VM che esegue sul PC dell'utente, in questo modo la macchina virtuale sfrutta le periferiche della macchina fisica (mouse, tastiera, schermo...), per consentire all'utente l'interazione con il sistema operativo guest.
- Viene realizzata mediante l'uso di particolari software che permettono di eseguire altri sistemi operativi "sopra" il sistema operativo host.
- È utile per far funzionare un software non compatibile con il sistema operativo dell'host o un software che si vuole mantenere separato dal sistema operativo host.

## Virtualizzazione livello hardware

Nella virtualizzazione livello hardware (macchine virtuali), all'utente del sistema di virtualizzazione viene presentata un'interfaccia su cui installare un sistema operativo, quindi una CPU virtuale (ma dello stesso tipo della CPU fisica) e risorse hardware virtuali. Invece, nel caso di virtualizzazione livello di sistema operativo (container), all'utente viene presentata una partizione del sistema operativo corrente, su cui installare ed eseguire applicazioni che rimangono isolate nella partizione, pur accedendo ai servizi di uno stesso sistema operativo.

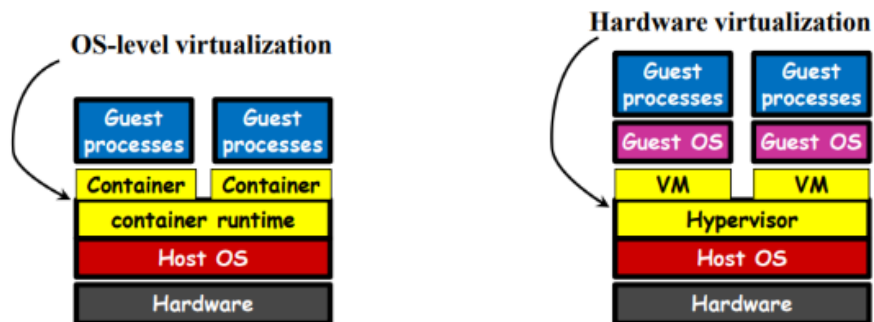


Figura 5.1: OS virtualization VS Hardware virtualization

### Virtualizzazione di tipo 2 (hosted)

Si definisce virtualizzazione di tipo 2 o **hosted** un sistema di virtualizzazione nel quale l'hypervisor è un normale processo utente sul sistema operativo host. Mentre si definisce virtualizzazione di tipo 1 o **bare-metal** un sistema di virtualizzazione nel quale il sistema operativo host è assente e le sue funzioni vengono sostituite dall'hypervisor.

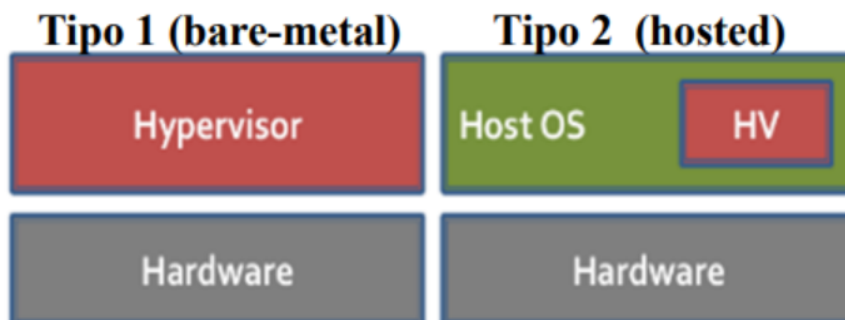
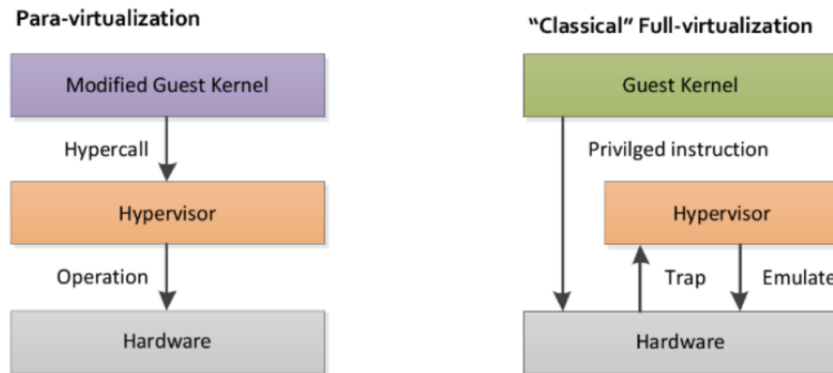


Figura 5.2: bare-metal vs hosted

### Virtualizzazione completa

Nella virtualizzazione completa (o full virtualization), vengono fornite VM che hanno la medesima interfaccia di una macchina fisica. Idealmente, il sistema operativo guest non può identificare che si trova su una macchina virtuale (in realtà di solito è sempre possibile dedurlo attraverso opportune tecniche). Invece, nella para

virtualization la VM presenta un'interfaccia diversa confrontata a una macchina fisica. Questo comporta dover modificare il sistema operativo guest per consentire l'esecuzione all'interno della macchina virtuale stessa.



**Figura 5.3:** para vs full virtualization

### 5.1.2 LXC

Nel progetto, oltre all'utilizzo di macchine virtuali complete tramite VMware, si è resa necessaria una soluzione più leggera e scalabile per simulare un numero elevato di router appartenenti ai vari AS.

#### Cos'è LXC

LXC (Linux Containers) è una tecnologia che permette di creare container Linux con un sistema operativo completo, isolati tra loro ma in grado di condividere lo stesso kernel Linux dell'host. A differenza delle VM, i container LXC non virtualizzano l'intero hardware, ma sfruttano le funzionalità native del kernel come `cgroups` e `namespaces` per ottenere isolamento e gestione delle risorse.

Ogni container LXC può essere considerato una "mini-macchina Linux", capace di eseguire più processi, avviare demoni di sistema (come `systemd`) e simulare realisticamente il comportamento di un router software. Per questo motivo, è particolarmente adatto all'uso con **FRRouting**, in modo da trasformare ciascun container in un router.



## **Motivazioni della scelta**

L'utilizzo esclusivo di VM per ogni router avrebbe comportato la creazione e gestione di oltre 20 istanze complete di Ubuntu Server, con un notevole consumo di risorse hardware. LXC offre una soluzione molto più efficiente: consente di simulare decine di router su un'unica VM host o macchina fisica, mantenendo un livello di realismo sufficiente per lo studio dei protocolli di routing. In questo modo, andremo a creare 7 VM Ubuntu server, e su ognuna 4 Linux Container (LXC).

## LXC vs Docker

Caratteristica	LXC	Docker
Tipo di container	Container di sistema (full OS)	Container per singola applicazione
Sistema operativo	Completo, con supporto per <b>systemd</b> , più processi, demoni	Minimo, pensato per un processo alla volta; <b>systemd</b> non supportato nativamente
Architettura	Accesso diretto a kernel, cgroups, namespaces; somiglia a una VM leggera	Strato intermedio (Docker Engine); isolamento a livello applicativo
Facilità di configurazione	Più complesso, orientato a sysadmin	Semplice, pensato per sviluppatori e DevOps
Gestione dei processi	Più processi supportati nativamente	Singolo processo per container (multi-processo solo con workaround)
Rete	Networking avanzato con configurazioni personalizzabili (bridge, veth, ecc.)	Networking gestito dal motore Docker; meno flessibile
Supporto a demoni e init system	Completo ( <b>systemd</b> , <b>init</b> , <b>cron</b> )	Limitato, solo con immagini o configurazioni speciali
Caso d'uso ideale	Simulare sistemi Linux completi (es: router, server reali)	Deploy di microservizi, app web, pipeline CI/CD
Ecosistema	Più ridotto, ma potente e vicino al kernel Linux	Ampissimo: Docker Hub, Docker Compose, Kubernetes

**Tabella 5.1:** Confronto tra LXC e Docker<sup>[7]</sup>

## Integrazione con VMware

In questo progetto, LXC è stato utilizzato all'interno di una macchina virtuale host Ubuntu Server, eseguita su VMware. In questo modo si ottiene il vantaggio di una struttura modulare: le VM rappresentano i confini tra gli AS, mentre i container LXC al loro interno permettono di simulare in modo efficiente i router interni. I

container sono interconnessi tramite bridge virtuali creati con **bridge-utils**, che permettono di emulare le reti locali tra gli IR e gli BR di ciascun AS.

### Vantaggi principali

- **Efficienza:** ogni container consuma pochi megabyte di RAM.
- **Realismo:** ambiente Linux completo, con supporto per FRRouting e configurazioni di rete avanzate.
- **Scalabilità:** facilità nel creare e clonare container per simulare topologie complesse.
- **Automazione:** possibilità di creare script per lanciare e configurare decine di router containerizzati.

## 5.2 FRRouting

FRR è un software open-source che implementa numerosi protocolli di routing dinamico, tra cui BGP, Open Shortest Path First (OSPF), Routing Information Protocol (RIP) e altri. Nasce come fork di Quagga nel 2016, con l'obiettivo di offrire una suite più aggiornata, stabile e orientata alla produzione in ambienti carrier-grade, data center e reti ISP.<sup>[1]</sup>

FRRouting è scritto in C e si basa su una struttura modulare: ogni protocollo è gestito da un demone<sup>1</sup> separato (es. **ospfd**, **bgpd**), coordinati dal demone **zebra**, responsabile dell'interazione con la tabella di routing del kernel Linux. La comunicazione tra i demoni avviene tramite un socket interno, e la configurazione può essere gestita tramite l'interfaccia a riga di comando **vtysh**. Utilizzeremo l'ultima versione aggiornata a luglio 2025: la 10.3.1.

Nel progetto, FRRouting è utilizzato per trasformare i container LXC in veri e propri IR interni all'AS. In particolare, viene impiegato il protocollo OSPF per abilitare il routing interno all'AS tra i router interni e il router di confine (BR). Alcuni motivi per cui è stato scelto FRR sono:

---

<sup>1</sup>dall'inglese *daemon*, è un processo in background che opera in modo autonomo.

- pieno supporto al protocollo OSPF;
- leggerezza e compatibilità con ambienti containerizzati (es. LXC);
- documentazione aggiornata e ampia community.

FRRouting è usato da numerosi operatori e progetti di rete, tra cui Cumulus Linux, VyOS, Google e altri attori del settore delle telecomunicazioni e dei data center<sup>[2]</sup>.

## 5.3 GoBGP

GoBGP è un'implementazione software del protocollo BGP interamente sviluppata in Go. È stato progettato per essere un'applicazione BGP moderna e flessibile, ideale per ambienti che richiedono scalabilità e automazione, come le infrastrutture Software Defined Networking (SDN) e per testare funzionalità avanzate quali RPKI e BGPsec.<sup>[4]</sup>

GoBGP è un progetto open-source mantenuto dalla sua community, con il supporto attivo di contributori da NTT Communications e altri operatori. A differenza di molti altri router software, GoBGP adotta un'architettura "daemon-less": si tratta di un unico binario che include tutte le funzionalità. Questo lo rende gestibile in modo flessibile tramite API gRPC o file di configurazione in formato YAML o TOML<sup>[3]</sup>

Nel progetto, GoBGP viene utilizzato per configurare i router di confine (BR) di ciascun AS (in particolare utilizzeremo l'ultima versione aggiornata a luglio 2025: la 3.37.0). A differenza di FRRouting, GoBGP non implementa protocolli IGP come OSPF o RIP, ma offre supporto avanzato per:

- validazione RPKI tramite RTR (RFC 6810/8210);
- supporto a BGPsec (in parte sperimentale);
- API per interazioni dinamiche e test automatizzati;
- capacità di annunciare/ritirare prefissi dinamicamente.

Questo rende GoBGP ideale per la simulazione di attacchi alla sicurezza di BGP (es. prefix hijacking, route leaking) e per testare le contromisure basate su RPKI.

## 5.4 Topologia rete

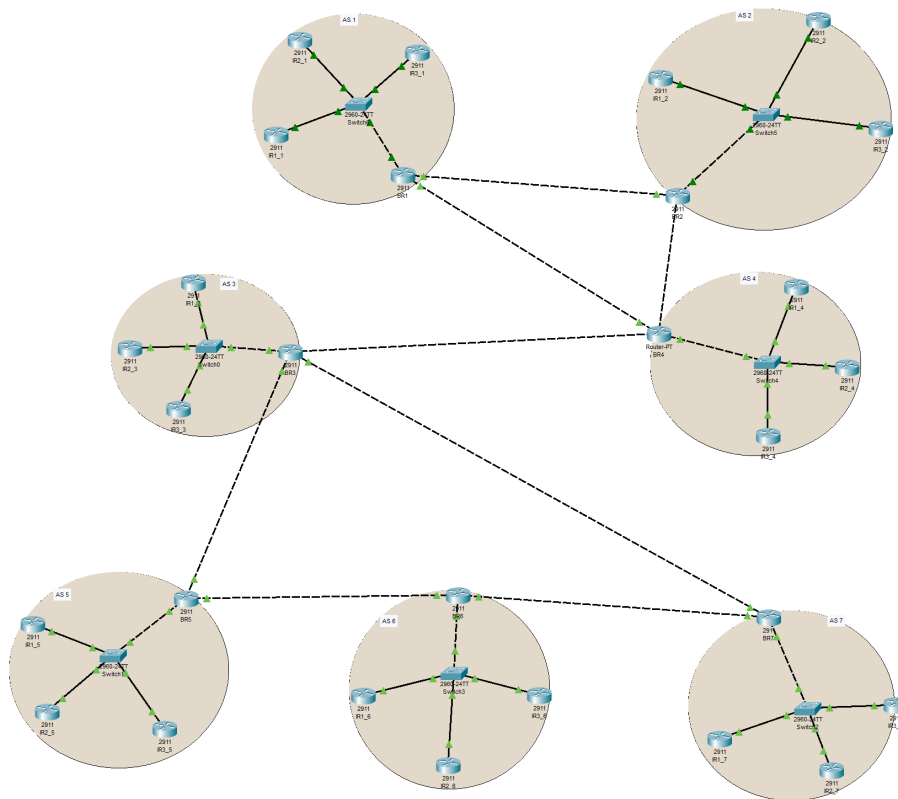
Una topologia di rete descrive come i nodi e le connessioni sono organizzati, sia fisicamente che logicamente, all'interno di una rete. Per il progetto è stata adottata una topologia fisica a maglia parziale, al fine di avvicinarci maggiormente a ciò che avviene nella realtà. Mentre nella topologia a maglia completa tutti i nodi sono collegati tra loro, nella topologia a maglia parziale non tutti i nodi sono collegati. Nella realtà, la topologia ha una struttura gerarchico-scalabile, in cui i grandi provider sono fortemente connessi tra loro, mentre i più piccoli hanno connessioni limitate verso l'alto. Vediamo ora alcune caratteristiche di una topologia a maglia parziale:

- ha una buona scalabilità <sup>2</sup>
- non tutti i nodi sono collegati tra loro, ma molti dispongono di connessioni multiple verso determinati nodi, ciò garantisce percorsi alternativi in caso di malfunzionamenti
- rispetto a una topologia a maglia completa, quella parziale più semplice da realizzare e meno costosa (banalmente perché richiede meno link fisici e meno hardware).

---

<sup>2</sup>La scalabilità è la caratteristica di una rete di aggiungere o rimuovere nodi facilmente.

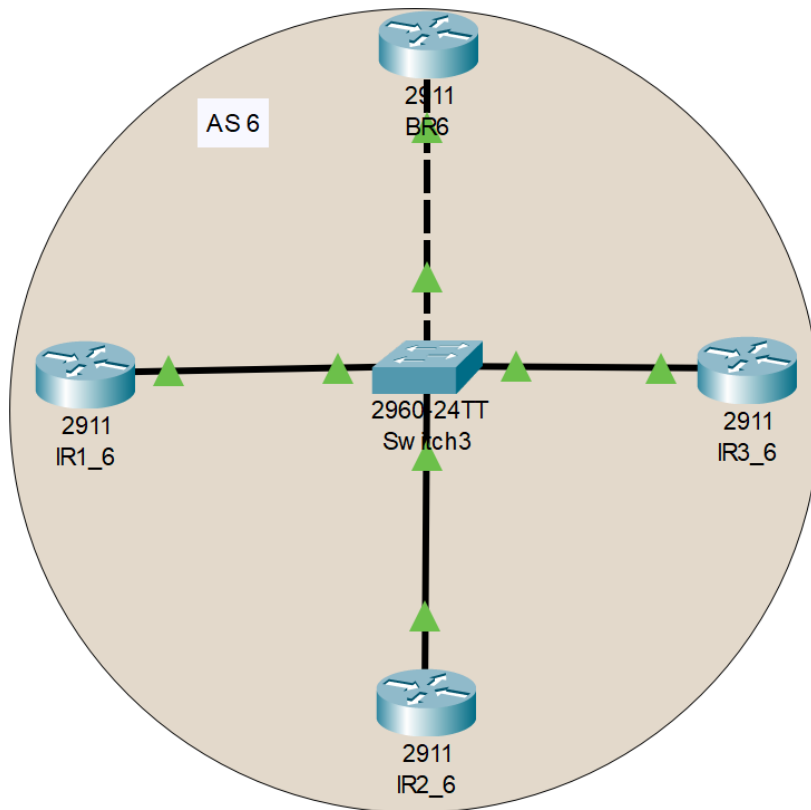
### 5.4.1 Struttura rete generale (topologia a maglia parziale tra 7 AS)



**Figura 5.4:** Topologia della rete completa

La rete è composta da sette AS, ciascuno dotato di un proprio sistema interno di routing e di un BR (router di confine). I BR sono connessi tra loro secondo una topologia logica a maglia parziale, in cui ogni router di confine stabilisce connessioni BGP soltanto con un sottoinsieme degli altri AS. I link BGP tra BR sono stati quindi progettati per formare una rete ridondata ma non simmetrica, garantendo percorsi alternativi senza replicare la connettività tra tutti i nodi. Un approccio a simulazioni con questa topologia consente di osservare il comportamento del protocollo BGP in scenari realistici, in cui le decisioni di instradamento devono tener conto di topologie non perfettamente connesse e della propagazione selettiva delle rotte.

### 5.4.2 Struttura singolo AS



**Figura 5.5:** Topologia dell'AS

Ogni AS è progettato con una struttura interna composta da moduli funzionali distinti (router interni, router di confine, switch e VLAN), pensata per simulare in modo realistico l'architettura di una rete aziendale. La composizione di ogni AS è la seguente:

- **1 BR:** router di confine che comunica con gli altri BR tramite il protocollo BGP. È responsabile dell'interscambio delle rotte con l'esterno e dell'inoltro del traffico verso la rete interna dell'AS.
- **3 IR:** router interni all'AS, collegati tra loro tramite un protocollo IGP, in questo caso OSPF. Gli IR non comunicano direttamente con l'esterno, ma solo con il BR e tra loro.

- **1 Switch Layer 2** configurato con **4 VLAN**:
  - **3 VLAN dedicate**, ognuna per il collegamento tra un IR e lo switch;
  - **1 VLAN condivisa**, che collega i tre IR e il BR, consentendo il funzionamento del routing interno (OSPF) tra tutti e quattro i router.



# Capitolo 6

## Sviluppo e implementazione

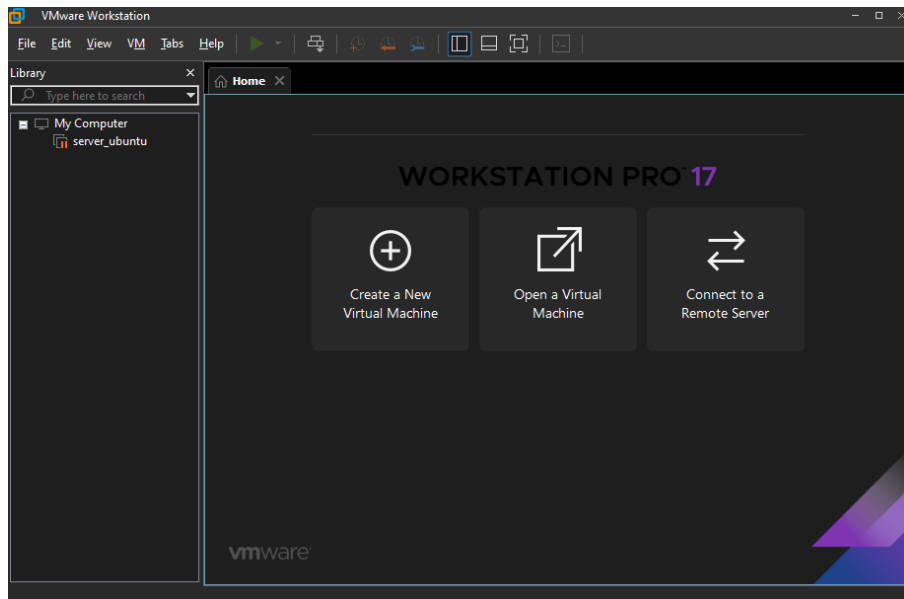
In questo capitolo, vediamo tutti i passaggi pratici messi in pratica per realizzare l'ambiente di simulazione e i vari test alla sicurezza del protocollo BGP.

### 6.1 Creazione di un AS

Ora vediamo i passi dettagliati per la creazione di un AS, che ricordiamo essere composto da 3 IR e 1 BR. Inoltre abbiamo anche uno switch con 4 Virtual Local Area Network (VLAN): 3 VLAN per ogni IR e una VLAN che permette ai 4 router di essere collegati tra loro e di gestire il routing tramite il protocollo OSPF.

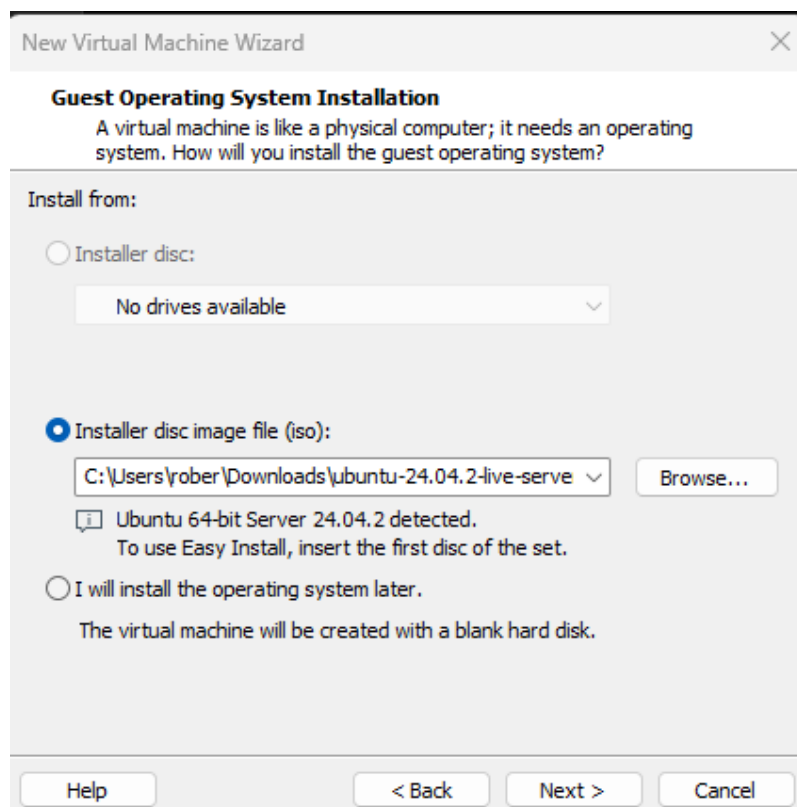
#### 6.1.1 Creazione VM con ubuntu server

1. Dal sito ufficiale di ubuntu: [www.ubuntu.com](http://www.ubuntu.com), andiamo a installare l'ultima versione della iso del sistema operativo ubuntu server. Nel mio caso l'ultima versione in data Giugno 2025 è: "ubuntu-24.04.2-live-server-amd64.iso".
2. Avviamo VMware Workstation Pro e clicchiamo su "Create a New Virtual Machine".



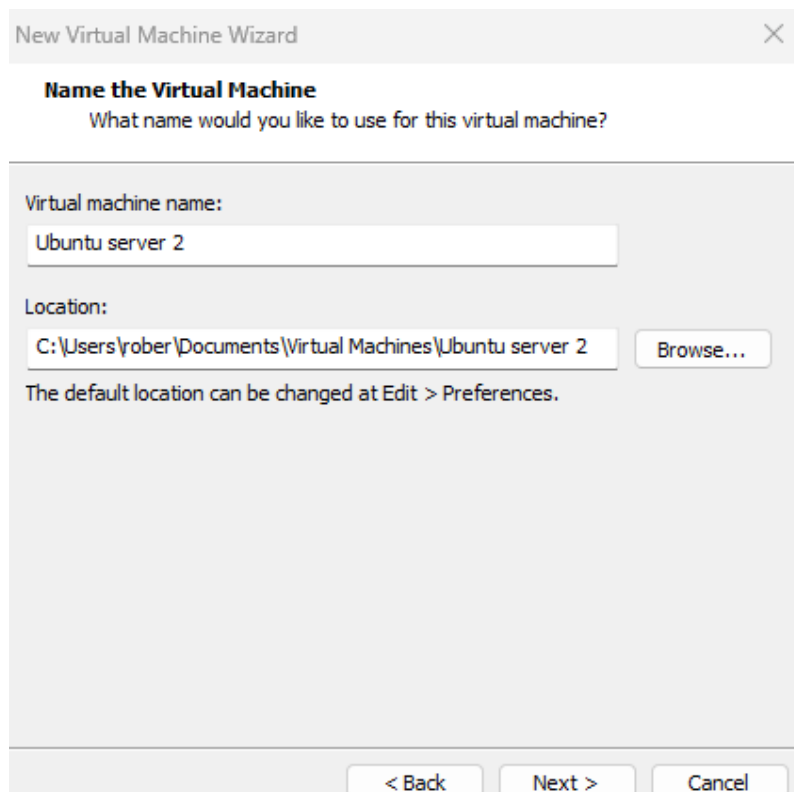
**Figura 6.1:** Schermata di avvio di workstation pro 17

3. Selezioniamo la iso precedentemente scaricata:



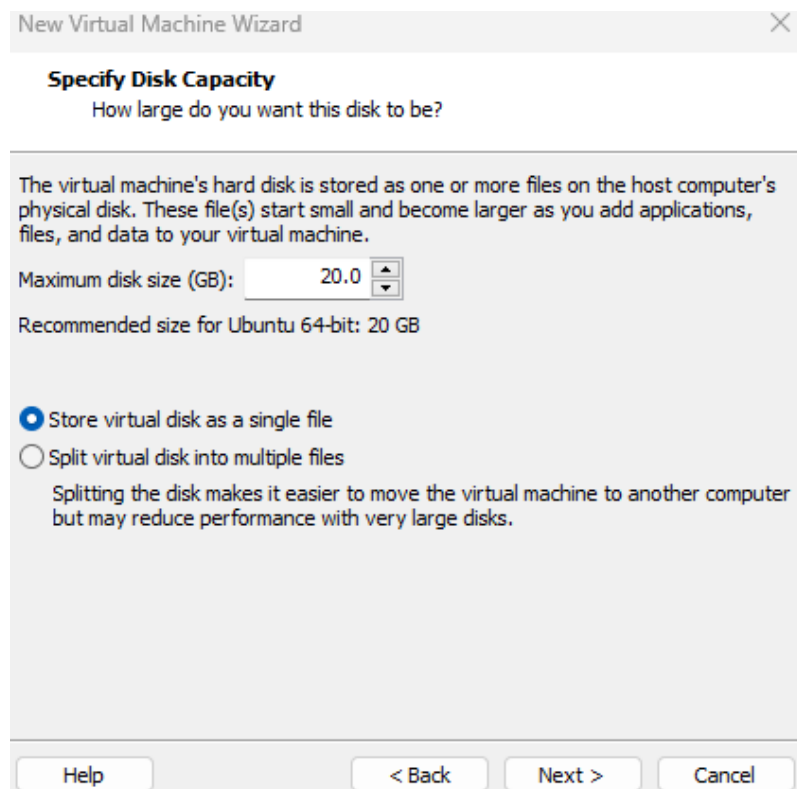
**Figura 6.2:** Selezione file iso

4. Gli diamo un nome e un percorso dove salvarla nel file system:



**Figura 6.3:** Nome e percorso VM

5. Specifichiamo 20 GB di archiviazione interna e l'opzione "Store virtual disk as a single file"



**Figura 6.4:** Impostazioni VM

6. A questo punto la configurazione è completa e si può avviare la macchina virtuale:

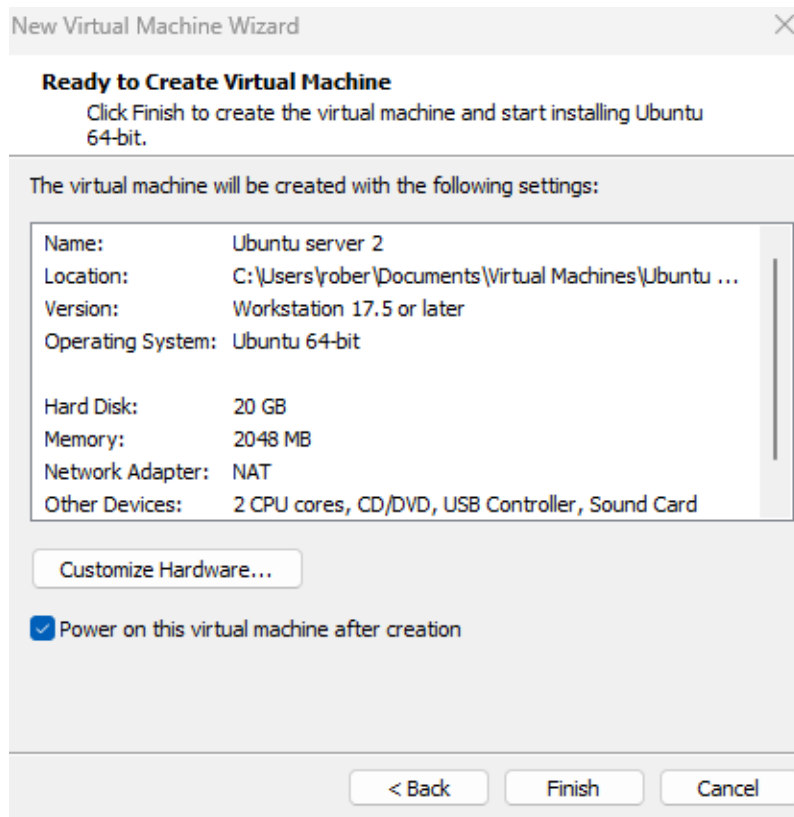
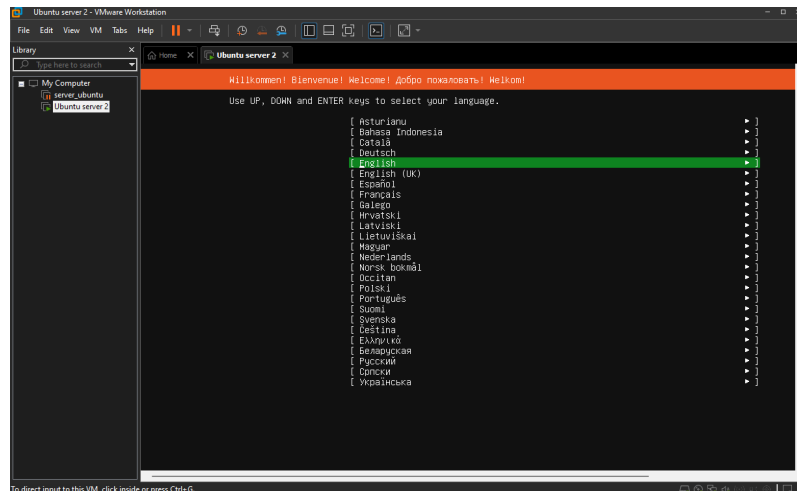


Figura 6.5: VM completata

### 6.1.2 Configurazione Ubuntu server

Al primo avvio della macchina virtuale, il sistema operativo Ubuntu server si configurerà in parte in automatico, mentre una parte della configurazione la dobbiamo completare noi manualmente:

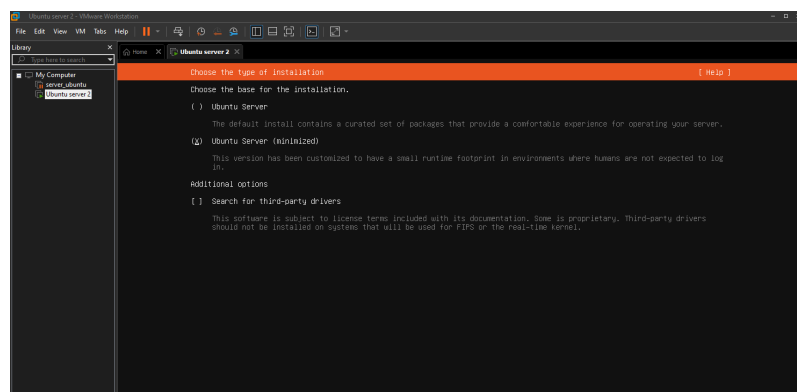
1. Selezioniamo la lingua di sistema e quella della tastiera:



**Figura 6.6:** Selezione lingua

2. Scegliamo la modalità di installazione "minimized". La modalità minimized di una distribuzione Linux è una modalità più leggera rispetto a quella standard. È ideale al nostro contesto perché:

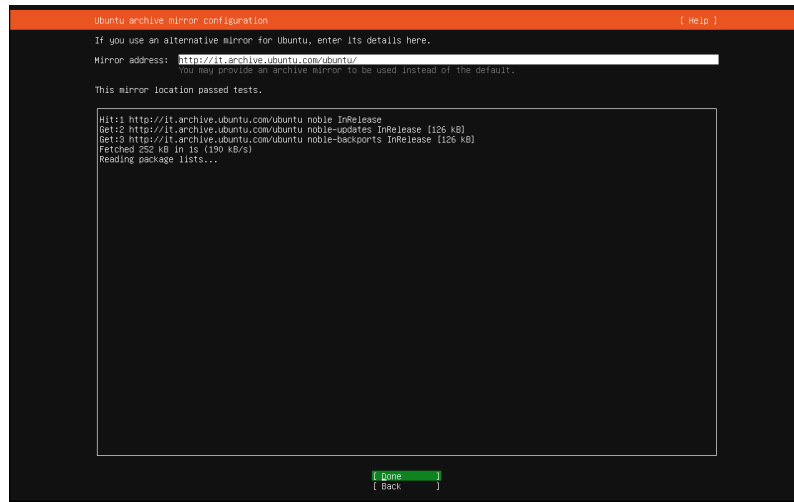
- riduce l'utilizzo di risorse della VM (disco, RAM);
- è più semplice da configurare;
- evita software non necessario.



**Figura 6.7:** Ubuntu minimized

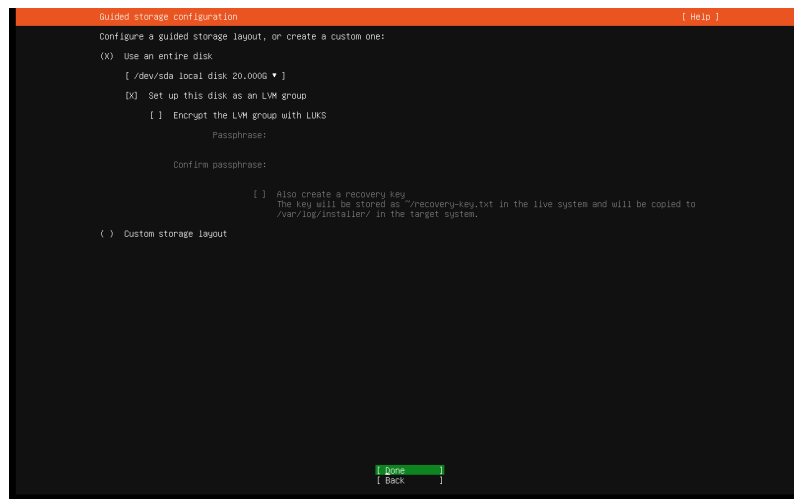
3. La configurazione della rete per ora la lasciamo così, la andremo a modificare in un secondo momento.

4. La configurazione del proxy la lasciamo vuota.
5. A questo punto ci viene chiesto quale mirror usare, lasciamo quello di default e andiamo avanti. Un mirror è un server che contiene una copia dell'archivio ufficiale di Ubuntu e serve principalmente a distribuire pacchetti del sistema operativo e dei software.



**Figura 6.8:** Configurazione mirror ubuntu

6. Per la configurazione storage lasciamo così com'è.



**Figura 6.9:** Configurazione storage



7. Ora ci viene chiesto di inserire username, nome del server, la password e il nome della macchina. Compiliamo i dati e andiamo avanti.
8. Infine ci verrà chiesto di installare qualche software tra i più utilizzati e altre impostazioni che possiamo ignorare (lasciare tutto com'è di default e andare avanti). A questo punto, Ubuntu server minimized è installato sulla nostra VM e pronto all'uso.

### 6.1.3 Installazioni preliminari sulla VM host

#### Installazione LXC e dipendenze

```
$ sudo apt update
$ sudo apt install lxc bridge-utils net-tools -y
```

#### Verifica se LXC è configurato correttamente

```
$ lxc-checkconfig
```

Se lo è, darà in output tutte flag verdi con scritto "Enabled".

#### Creazione container

Ora creiamo i container che conterranno i router virtuali del nostro AS. Avranno tutti sistema operativo Ubuntu 20.04 LTS a 64 bit.

```
$ sudo lxc-create -t download -n ir1a -- -d ubuntu -r focal -
a amd64
$ sudo lxc-copy -n ir1a -N ir1b
$ sudo lxc-copy -n ir1a -N ir1c
$ sudo lxc-copy -n ir1a -N br1
```

Verifichiamo che i container siano visibili:

```
$ lxc-ls -f
```

Se lo sono, ci darà come output l'elenco dei nomi dei container con alcune caratteristiche.

## Creazione di bridge virtuali sulla VM host

Per le 4 VLAN interne, andiamo a creare 4 rispettivi bridge virtuali modificando il file `”/etc/netplan/50-cloud-init.yaml”` in questo modo:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens33:
      dhcp4: true

  bridges:
    vlan1:
      dhcp4: no
      parameters:
        stp: true
        forward-delay: 0

    vlan2:
      dhcp4: no
      parameters:
        stp: true
        forward-delay: 0

    vlan3:
      dhcp4: no
      parameters:
        stp: true
        forward-delay: 0

    vlan4:
      dhcp4: no
      parameters:
        stp: true
        forward-delay: 0

    br-bgp:
      dhcp4: no
      parameters:
        stp: true
        forward-delay: 0

    br-external:
      interfaces: [ens33]
      dhcp4: true
      parameters:
        stp: true
        forward-delay: 0
```

**Figura 6.10:** Configurazione bridge virtuali

e applichiamo la configurazione:

```
$ sudo netplan apply
```

Una volta creati, li attiviamo:

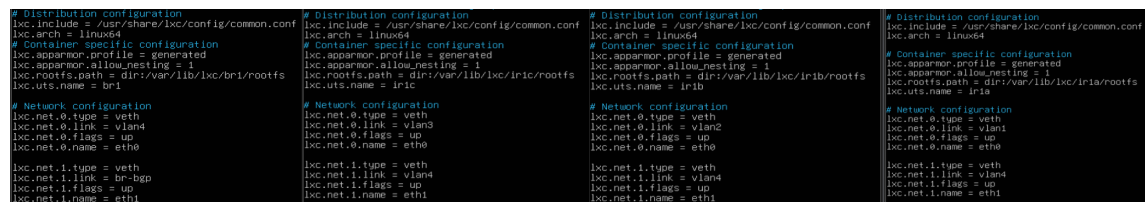
```
$ sudo ip link set vlan1 up
$ sudo ip link set vlan2 up
$ sudo ip link set vlan3 up
```

```
$ sudo ip link set vlan4 up
$ sudo ip link set br-bgp up
```

e assegniamo le interfacce dei container ai bridge. Per ogni container eseguiamo il comando:

```
$ sudo nano /var/lib/lxc/<nome-container>/config
```

e li configuriamo come nell'immagine sottostante:



The image shows four side-by-side screenshots of LXC configuration files. Each file contains the following configuration:

```
# Distribution configuration
lxc.include = /usr/share/lxc/config/common.conf
lxc.arch = linux64
# Container specific configuration
lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
lxc.rootfs.path = dir:/var/lib/lxc/br1/rootfs
lxc.uts.name = br1
# Network configuration
lxc.net.0.type = veth
lxc.net.0.link = vian3
lxc.net.0.flags = up
lxc.net.0.name = eth0
lxc.net.1.type = veth
lxc.net.1.link = br-bgp
lxc.net.1.flags = up
lxc.net.1.name = eth1
```

The four screenshots show variations of these configurations, likely for different containers or network setups.

Figura 6.11: Configurazione interfacce

## Avvio dei container

Per ogni container andiamo a eseguire:

```
$ sudo lxc-start -n <nome-container>
```

E per controllare che tutti siano partiti correttamente:

```
$ sudo lxc-ls -f
```

## Configurazione della rete nei router

Per prima cosa, entriamo nel container che rappresenta il nostro router (ad esempio: ir1a)

```
$ sudo lxc-attach -n ir1a
```

Poi andiamo a modificare il file "10-lxc.yaml" di netplan <sup>1</sup>. Questo file conterrà la configurazione di rete per i bridge virtuali che verranno usati dai container LXC.

<sup>1</sup>Netplan è un utility di configurazione di rete.

```
$ sudo nano /etc/netplan/10-lxc.yaml
```

E lo modifichiamo in questo modo:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses: [172.16.32.1/19]
    eth1:
      addresses: [172.16.128.2/19]
    eth2:
      dhcp4: yes
```

In eth0, ci mettiamo l'indirizzo ip del default gateway della rete interna di quel rispettivo IR, ovvero l'ip successivo a quello di rete, che sono rispettivamente per ogni IR:

Rete	CIDR	Router
172.16.1.0	/24	IR1 - AS1
172.16.2.0	/24	IR2 - AS1
172.16.3.0	/24	IR3 - AS1
172.16.10.0	/24	IR1 - AS2
172.16.11.0	/24	IR2 - AS2
172.16.12.0	/24	IR3 - AS2
172.16.20.0	/24	IR1 - AS3
172.16.21.0	/24	IR2 - AS3
172.16.22.0	/24	IR3 - AS3
172.16.30.0	/24	IR1 - AS4
172.16.31.0	/24	IR2 - AS4
172.16.32.0	/24	IR3 - AS4
172.16.40.0	/24	IR1 - AS5
172.16.41.0	/24	IR2 - AS5
172.16.42.0	/24	IR3 - AS5
172.16.50.0	/24	IR1 - AS6
172.16.51.0	/24	IR2 - AS6
172.16.52.0	/24	IR3 - AS6
172.16.60.0	/24	IR1 - AS7
172.16.61.0	/24	IR2 - AS7
172.16.62.0	/24	IR3 - AS7

**Tabella 6.1:** Indirizzi di rete /24 assegnati alle LAN dei router interni (IR)

Mentre l'ip delle interfacce eht1 degli altri router saranno rispettivamente:

- **ir1b:** 172.16.128.3
- **ir1c:** 172.16.128.4
- **br1:** 10.0.0.2/27 (esterna per BGP), 172.16.128.5 (interna)

### Abilitiamo la connessione a internet a tutti i container

Per scaricare i software necessari per rendere i nostri container dei router virtuali, è necessario che essi riescano ad accedere a internet, per cui:

- Nella VM host, apriamo il file `"/etc/netplan/50-cloud-init.yaml"` e creiamo il bridge `"br-external"`:

```

network:
  version: 2
  renderer: networkd
  ethernets:
    ens33:
      dhcp4: true

  bridges:
    br-external:
      interfaces: [ens33]
      dhcp4: true
      parameters:
        stp: true
        forward-delay: 0

```

**Figura 6.12:** Configurazione rete VM host

Poi salviamo la configurazione:

```
sudo netplan apply
```

- Abilitiamo il NAT sulla VM host:

```
sudo nano /etc/sysctl.conf
```

e decommentiamo "net.ipv4.ip\_forward=1"

- Creiamo una regola NAT:

```
sudo iptables -t nat -A POSTROUTING -o br-external -j
MASQUERADE
```

e facciamo in modo che si salvi anche al reboot:

```
sudo apt install iptables-persistent
sudo netfilter-persistent save
```

- Aggiungiamo il bridge "br-external" nella configurazione di rete del container. Apriamo il file "/var/lib/lxc/<nome-container>/config" e aggiungiamo:

```
lxc.net.2.type = veth
lxc.net.2.link = br-external
lxc.net.2.flags = up
lxc.net.2.name = eth2
```

e poi riavviamo il container

```
lxc-stop -n ir1a
lxc-start -n ir1a
```

- Accediamo al container e diamo una configurazione dhcp all'interfaccia:

```
lxc-attach -n ir1a
dhclient eth2
```

Ora il container ir1a è abilitato ad accedere a internet (possiamo testare con un semplice *ping google.com*), ripetiamo questi passaggi per tutti gli altri container e possiamo andare avanti.

### 6.1.4 Installazione FRR sui container

Ora vediamo come installare FRR sui container ai router:

- Aggiorniamo l'indice dei pacchetti disponibili sui repository:

```
sudo apt update
```

- Installiamo 3 tool che ci saranno utili: curl <sup>2</sup>, gnupg2<sup>3</sup> e lsb-release<sup>4</sup>

```
sudo apt install -y curl gnupg2 lsb-release
```

- Ora, siccome FRR non è in un repository predefinito di linux, dobbiamo:

- scaricare una chiave pubblica GPG di FRR con curl

---

<sup>2</sup>Curl è un tool utile per scaricare file o interagire con API web da riga di comando.

<sup>3</sup>Gnupg2 fornisce un implementazione del software di crittografia OpenPGP, usato per verificare l'autenticità dei pacchetti.

<sup>4</sup>Tool che fornisce informazioni sulla distribuzione Linux in uso.

- rendere questa chiave utilizzabile dal sistema Linux con "gpg --dearmor"
- installarla in un posto sicuro dove apt (il gestore dei pacchetti Ubuntu) la cercherà

```
curl -s https://deb.frrouting.org/frr/keys.asc | sudo gpg
--dearmor -o /usr/share/keyrings/frr.gpg
```

- Ora aggiungiamo una nuova voce di repository per FRR alla configurazione del gestore di pacchetti apt.

```
echo "deb [signed-by=/usr/share/keyrings/frr.gpg] https
://deb.frrouting.org/frr/ $(lsb_release -cs) frr-
stable" | sudo tee /etc/apt/sources.list.d/frr.list
```

- Aggiorniamo nuovamente l'indice dei pacchetti disponibili sui repository:

```
sudo apt update
```

- E finalmente installiamo FRR e i suoi strumenti python:

```
sudo apt install frr frr-pythontools -y
```

- Per testare se tutto l'installazione è avvenuta correttamente, il seguente comando deve mostrare la versione di FRR installata:

```
vttysh -c "show version"
```

### 6.1.5 Installazione GoBGP sui container dei Border Router

Invece, per installare GoBGP sui container dedicati ai BR:

- Aggiorniamo l'indice dei pacchetti disponibili sui repository:

```
sudo apt update
```

- Installiamo i pacchetti wget<sup>5</sup> e unzip<sup>6</sup>:

---

<sup>5</sup>Tool da riga di comando per scaricare file da internet

<sup>6</sup>Tool da riga di comando per decomprimere file compressi in formato .zip.



```
sudo apt install wget unzip -y
```

- Scarichiamo con wget l'ultima versione dell'archivio zip contenente l'eseguibile di GoBGP, prendendolo dal repository Github ufficiale del progetto.

```
wget https://github.com/osrg/gobgp/releases/download/v3.37.0/gobgp_3.37.0_linux_amd64.tar.gz
```

- Decomprimiamo il file .tar.gz:

```
tar -xzf gobgp_3.37.0_linux_amd64.tar.gz
```

- Spostiamo gli eseguibili "gobgp" e "gobgpd" estratti dall .tar.gz nella directory "/usr/local/bin/", in modo da poterli eseguire da qualsiasi posizione del terminale:

```
sudo mv gobgp gobgpd /usr/local/bin/
```

- Infine controlliamo se abbiamo fatto tutto correttamente, controllando la versione di GoBGP:

```
gobgp --version
```

### 6.1.6 Configurazione OSPF nei router

Per ogni router appartenente all'AS, andiamo ora a configurare OSPF in questo modo:

- Avviamo il container:

```
lxc-start -n <nome-router>
```

- Accediamo al container:

```
lxc-attach -n <nome-router>
```

- Accediamo alla console del router:

```
vttysh
```

- Entriamo nella modalità di configurazione avanzata:

```
conf t
```

- Abilitiamo il processo OSPF:

```
router ospf
```

- Selezioniamo l'interfaccia preposta per la vla4:

```
interface eth1
```

- Colleghiamo l'interfaccia di rete alla backbone OSPF (area 0):

```
ip ospf area 0
```

- Usciamo dalla modalità di configurazione avanzata:

```
exit
```

- E salviamo la configurazione:

```
write
```

- Per vedere se i vicini ospf sono corretti:

```
show ip ospf neighbor
```

- Per vedere se i percorsi ospf sono corretti:

```
show ip route ospf
```

## 6.2 Clonazione dei restanti AS

Una volta creato una macchina virtuale che ci fungerà da AS1, per creare i restanti 6 utilizzeremo la clonazione completa della macchina virtuale, in modo da evitare di dover ripetere tutte le installazioni e configurazioni varie.

### 6.2.1 Cambio nomenclatura e indirizzo ip esterno del BR

In questa fase, andiamo a modificare la nomenclatura delle varie macchine interne all'AS.

#### Cambio username macchina host AS

Per prima cosa, andiamo a modificare l'hostname della macchina host che contiene i 4 router, da AS1 a ASN, dove N è il numero dell'AS che stiamo modificando, apriamo quindi i 2 file che contengono l'hostname e sostituiamo il vecchio con il nuovo:

```
sudo vi /etc/hostname
```

```
sudo vi /etc/hosts
```

#### Cambio nomi dei container

Ora andiamo a rinominare le cartelle dei container, quindi eseguiamo il seguente comando per i 4 container:

```
sudo mv /var/lib/lxc/<vecchio-nome> /var/lib/lxc/<nuovo-nome>
```

Sempre per ogni container, vado a cambiare i riferimenti da <nome-vecchio-container> a <nome-nuovo-container> nel file di configurazione:

```
sudo vi /var/lib/lxc/<nome-container>/config
```

#### Cambio hostname dei container

A questo punto, cambiamo l'hostname di ogni container, quindi per ognuno di essi:

```
# avviamo il container
sudo lxc-start -n <nome-container>

# entriamo nella sua shell
sudo lxc-attach -n <nome-container>
```

```
# sostituiamo le occorrenze del vecchio hostname con il nuovo
   nei seguenti 2 file:
sudo nano /etc/hostname
sudo nano /etc/hosts
```

## Cambio ip interfaccia esterna del BR

Per il solo BR, andiamo a modificare anche l'indirizzo ip dell'interfaccia eth1, modificando il file:

```
sudo nano /etc/netplan/10-lxc.yaml
```

E poi applichiamo il seguente comando per salvare le modifiche:

```
sudo netplan apply
```

## Riavvio del sistema

Infine, per vedere tutti i vari nomi modificati, andiamo a fare un riavvio del sistema:

```
sudo reboot
```

## 6.2.2

# Capitolo 7

## Prospettive future: SDN e BGP

### 7.1 Cos'è la SDN

7.1.1 Architettura e principio di separazione control/data plane

7.1.2 Vantaggi principali: flessibilità, programmazione, automazione

### 7.2 Integrazione tra SDN e BGP

7.2.1 Routing interdominio gestito centralmente

7.2.2 Esempi di progetti o framework (es. SDX, BGP-SDN)

### 7.3 Prospettive evolutive

7.3.1 Reti programmabili e scenari futuri

7.3.2 Possibili impatti su sicurezza e gestione globale



# Appendice A

## Ricerca su ECDSA

# Capitolo 2

## Firma digitale con curve ellittiche: ECDSA

### 2.1 Nascita di ECDSA

L'ECDSA nasce come estensione dell'algoritmo Digital Signature Algorithm (DSA), sviluppato agli inizi degli anni '90 dal National Institute of Standards and Technology (NIST) degli Stati Uniti come standard federale per la firma digitale (Federal Information Processing Standard (FIPS) 186 nel 1991).

Successivamente, grazie agli studi di due matematici americani, Neal Koblitz e Victor Miller, fu dimostrato che le curve ellittiche potevano essere sfruttate per costruire sistemi crittografici a chiave pubblica con una sicurezza equivalente a quella di algoritmi preesistenti (come RSA o lo stesso DSA), ma con chiavi significativamente più piccole e, di conseguenza, operazioni computazionali più rapide ed efficienti.

Fu Scott Vanstone, matematico canadese e co-fondatore dell'azienda Certicom, a proporre specificamente l'uso delle curve ellittiche all'interno dello schema DSA. La sua proposta, formulata nei primi anni '90, portò alla definizione dell'ECDSA, che fu infine standardizzato nel 1999 dallo Institute of Electrical and Electronics Engineers (IEEE) (nello standard P1363) e dall'American National Standards Institute (ANSI) (nello standard X9.62), segnando l'ingresso ufficiale di ECDSA nel panorama della crittografia moderna.

### 2.2 Firma digitale

ECDSA è un tipo particolare di firma digitale che utilizza le curve ellittiche. Quindi, prima di addentrarci nel comprendere il suo funzionamento, è necessario parlare di firme digitali e delle hash function (utilizzate per la creazione delle firme).

#### 2.2.1 Hash Function

Le hash function sono particolari funzioni che prendono un input di lunghezza variabile (l'input può avere anche più bit di quelli fissi dell'output), e restituisce un output di lunghezza fissa in base al tipo di funzione di hash. L'output di una hash function prende il nome di **message digest**, **digest** o **impronta**.



# Hashing



Figura 2.1: Hash function

## Caratteristiche hash function

- è **one way function** → dato l'output, non è possibile risalire all'input
- è **deterministica** → dato lo stesso input, per la medesima funzione, ottengo sempre lo stesso output
- gode del cosiddetto "**effetto valanga**" → se in input cambia anche solo di 1 bit, cambia completamente l'output.

## 2.2.2 Come avviene la firma digitale e cosa garantisce

### Mittente

1. Si parte dal **messaggio in chiaro** (plaintext), indicato con  $M$ .
2. Il mittente calcola il **digest** del messaggio tramite una funzione di hash sicura:  $h = \text{Hash}(M)$ .
3. Il digest  $h$  viene **firmato con la chiave privata del mittente**  $SK_A$  (private key), ottenendo la **firma digitale**:  $\sigma = \text{Sign}_{SK_A}(h)$ .
4. Il mittente invia al destinatario la coppia:  $(M, \sigma)$ .

### Destinatario

1. Il destinatario riceve  $(M, \sigma)$  e calcola nuovamente il digest:  $h' = \text{Hash}(M)$ .
2. Il destinatario verifica la firma usando la **chiave pubblica del mittente**  $PK_A$ :  $h = \text{Verify}_{PK_A}(\sigma)$ .
3. La firma è valida se e solo se:  $h = h'$ .

Se  $h = h'$  allora è garantita l'integrità del messaggio (grazie all'hash function) e l'autenticazione del mittente, oltre che il non ripudio (ovvero il mittente non può negare di aver firmato il messaggio, dato che solo lui possiede la chiave privata).

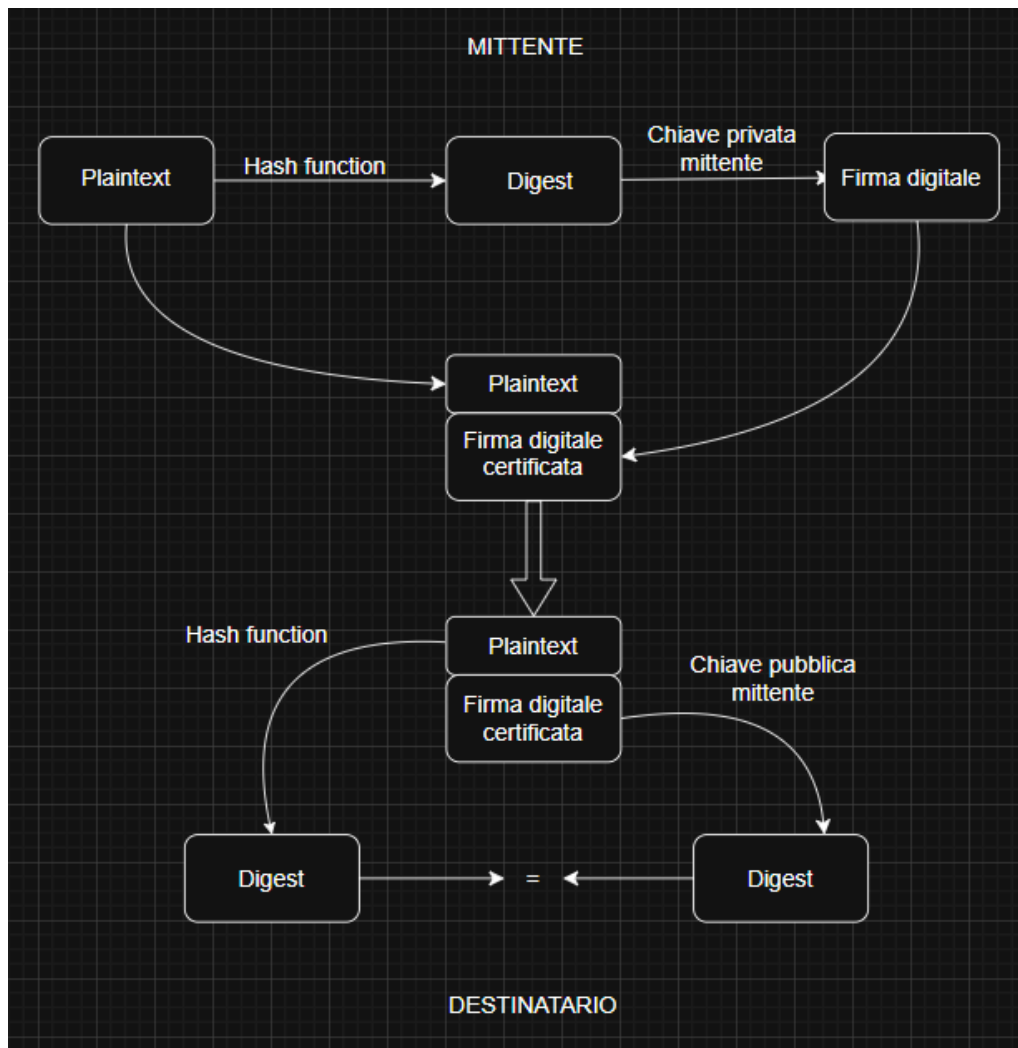


Figura 2.2: Firma digitale

## 2.3 Come funziona ECDSA

La firma digitale tramite ECDSA si articola in 3 fasi che andremo ad analizzare una per una: Inizializzazione dei parametri, generazione della firma da parte del mittente e verifica della firma da parte del destinatario.

### 2.3.1 Inizializzazione dei parametri

Questi parametri sono pubblici e devono essere concordati da tutte le parti coinvolte nella comunicazione. Non vengono scelti da ogni singolo utente ma sono scelti da standard predefiniti come quelli NIST (es. curve P-192, P-256, P-384, P-521) o Standard for Efficient Cryptography Group (SECG) (es. secp256k1, usata in Bitcoin). Le curve sono definite su campi finiti di grandi dimensioni, come  $\mathbb{F}_p$  con  $p$  primo da 256 o più bit.

- Un campo finito primo  $\mathbb{F}_p$ , con  $p > 3$  primo.
- Una curva ellittica  $E$  definita su  $\mathbb{F}_p$ , espressa in forma ridotta di Weierstrass:

$$E : y^2 = x^3 + ax + b \mod p$$

con  $a, b \in \mathbb{F}_p$ , tali che il discriminante  $\Delta = 4a^3 + 27b^2 \pmod{p} \neq 0$ , per garantire la non-singularità.

- Un punto generatore  $G \in E(\mathbb{F}_p)$  di ordine primo  $n$ , tale che:

$$nG = \mathcal{O}$$

dove  $\mathcal{O}$  è il punto all'infinito.

- Una funzione di hash crittografica  $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ , come SHA-256. L'output dell'hash viene utilizzato per derivare un valore numerico  $e$  (spesso  $z$  o  $H_M$ ) che è poi utilizzato nel calcolo della firma. Per l'ECDSA, la lunghezza dell'output dell'hash deve essere almeno pari alla dimensione in bit di  $n$ . Tipicamente, il valore hash è convertito in un intero e poi, se necessario, troncato o ridotto modulo  $n$ .

## Chiavi

Ogni utente genera le proprie chiavi basandosi sui parametri di dominio pubblico stabiliti:

- **Chiave Privata ( $d$ ):** Un intero  $d$  scelto casualmente e in modo sicuro dall'intervallo  $d \in_R \{1, \dots, n-1\}$ . Questa chiave deve rimanere segreta.
- **Chiave Pubblica ( $Q$ ):** Il punto  $Q = dG \in E(\mathbb{F}_p)$ , calcolato moltiplicando la chiave privata  $d$  per il punto generatore  $G$ . La chiave pubblica  $Q$  può essere distribuita liberamente.

### 2.3.2 Generazione della firma

Supponiamo che l'utente A (mittente) voglia firmare un messaggio  $m$  utilizzando la sua chiave privata  $d$ . I passaggi della generazione della firma, secondo lo standard ECDSA, sono i seguenti:

1. **Calcolo del Digest del Messaggio:** Si calcola il digest crittografico del messaggio  $m$  usando la funzione di hash  $H$ :

$$e = H(m)$$

Questo valore  $e$  viene poi interpretato come un intero. Il valore  $z$  utilizzato nel calcolo della firma è derivato da  $e$ . Se la lunghezza in bit di  $e$  supera la lunghezza in bit dell'ordine  $n$ , si utilizzano i bit più significativi di  $e$  per formare  $z$ , in modo che la sua lunghezza in bit sia pari a quella di  $n$  (ovvero  $z$  è l'intero rappresentato dai  $\lfloor \log_2 n \rfloor$  bit più significativi di  $e$ , o semplicemente  $e$  se  $e < n$ ).

2. **Scelta del Nonce <sup>1</sup> Crittografico:** Si sceglie un numero intero casuale e segreto  $k \in_R \{1, \dots, n-1\}$ . È **fondamentale** che  $k$  sia generato in modo crittograficamente sicuro per ogni singola firma e non venga mai, per nessuna ragione, riutilizzato per firme diverse. La compromissione o la riutilizzazione di  $k$  esporrebbe immediatamente la chiave privata  $d$ .

---

<sup>1</sup>Il termine "nonce" sta per "number used once".

3. **Calcolo del Punto sulla Curva:** Si calcola il punto  $(x_1, y_1) = [k]G$  eseguendo una *moltiplicazione scalare* del punto generatore  $G$  per l'intero  $k$ .
4. **Calcolo di  $r$ :** Si calcola la prima componente della firma,  $r$ , interpretando la coordinata  $x_1$  come un intero:

$$r = x_1 \mod n$$

Se  $r = 0$ , si deve scartare il valore  $k$  corrente e ripetere il processo a partire dal passaggio 2 con un nuovo  $k$ .

5. **Calcolo di  $s$ :** Si calcola l'inverso moltiplicativo di  $k$  modulo  $n$ , denotato  $k^{-1}$  (cioè  $k \cdot k^{-1} \equiv 1 \pmod{n}$ ). Successivamente, si calcola la seconda componente della firma,  $s$ :

$$s = k^{-1}(z + dr) \mod n$$

Se  $s = 0$ , anche in questo caso si scarta il valore  $k$  corrente e si ripete il processo a partire dal passaggio 2 con un nuovo  $k$ .

**Output:** La firma è la coppia ordinata  $(r, s)$ . Entrambi i valori sono elementi di  $\mathbb{Z}_n^*$  (il gruppo moltiplicativo degli interi modulo  $n$ ) e insieme rappresentano la firma digitale di  $m$ .

Al destinatario vengono spediti sia  $m$  che la corrispondente firma digitale  $(r, s)$ .

### 2.3.3 Verifica della firma

Supponiamo che il destinatario riceva un messaggio  $m$  in chiaro insieme alla firma digitale  $(r, s)$ . Per verificare l'autenticità e l'integrità del messaggio (e che sia stato effettivamente firmato dal mittente), il destinatario esegue i seguenti passaggi. Questo processo richiede la conoscenza della **chiave pubblica del mittente**  $Q$ .

1. **Controllo dei Valori della Firma:** Si verifica innanzitutto che le componenti della firma  $r$  e  $s$  siano entrambe valide, ovvero che appartengano all'intervallo  $[1, n - 1]$ . Se  $r < 1$  o  $r \geq n$ , oppure se  $s < 1$  o  $s \geq n$ , la firma è considerata **invalida** e il processo di verifica si interrompe.
2. **Calcolo del Digest del Messaggio:** Il destinatario calcola il digest crittografico del messaggio ricevuto  $m$  utilizzando la stessa funzione di hash  $H$  impiegata dal mittente:

$$e = H(m)$$

Come nel processo di generazione della firma, questo valore  $e$  viene interpretato come un intero. Il valore  $z$  utilizzato nei calcoli successivi è derivato da  $e$ . Se la lunghezza in bit di  $e$  supera la lunghezza in bit dell'ordine  $n$ , si utilizzano i bit più significativi di  $e$  per formare  $z$ , in modo che la sua lunghezza in bit sia pari a quella di  $n$ .

3. **Calcolo dell'Inverso Modulare di  $s$ :** Si calcola l'inverso moltiplicativo di  $s$  modulo  $n$ , denotato  $w$ . Questo valore soddisfa la relazione  $s \cdot w \equiv 1 \pmod{n}$ . Tale calcolo viene tipicamente eseguito utilizzando l'algoritmo di Euclide Esteso.

$$w = s^{-1} \mod n$$

4. **Calcolo dei Coefficienti Intermedi:** Si calcolano due coefficienti ausiliari,  $u_1$  e  $u_2$ , che saranno usati nel prossimo passaggio per ricostruire un punto sulla curva:

$$u_1 = z \cdot w \mod n$$

$$u_2 = r \cdot w \mod n$$

5. **Calcolo del Punto di Verifica sulla Curva:** Utilizzando i coefficienti  $u_1$  e  $u_2$ , il punto generatore  $G$ , e la chiave pubblica del mittente  $Q$ , si calcola un punto  $(x_V, y_V)$  sulla curva. Questa operazione coinvolge due **moltiplicazioni scalari di punti** ( $[u_1]G$  e  $[u_2]Q$ ) e una successiva **addizione di punti** sulla curva ellittica:

$$(x_V, y_V) = [u_1]G + [u_2]Q$$

Se il risultato di questa operazione è il punto all'infinito  $\mathcal{O}$ , la firma è considerata invalida.<sup>2</sup>

6. **Verifica Finale:** Infine, si confronta la coordinata  $x_V$  del punto calcolato nel passaggio precedente con il valore  $r$  fornito nella firma. La verifica ha successo se e solo se la coordinata  $x_V$ , interpretata come un intero e ridotta modulo  $n$ , è uguale a  $r$ :

$$r \equiv x_V \mod n$$

**Conclusione:** Se l'uguaglianza nel passaggio 6 è verificata, la firma è considerata **valida**. Questo successo implica che:

- Il messaggio non è stato modificato dopo la firma (garanzia di **integrità**).
- Il messaggio è stato firmato con la chiave privata corrispondente alla chiave pubblica  $Q$  fornita (garanzia di **autenticità** del mittente).
- Il mittente non può negare di aver firmato il messaggio, poiché solo lui possedeva la chiave privata  $d$  necessaria a generare  $r$  e  $s$  in modo congruente con  $Q$  (garanzia di **non ripudio**).

### 2.3.4 Esempio generazione e verifica firma

Per comprendere meglio il funzionamento dell'algoritmo, vediamo un esempio semplificato con numeri piccoli<sup>3</sup>.

#### Parametri pubblici condivisi

- Campo finito:  $\mathbb{F}_{17}$
- Curva ellittica:  $E : y^2 = x^3 + 2x + 2 \mod 17$
- Punto generatore:  $G = (5, 1)$ , con ordine primo  $n = 19$
- Funzione di hash:  $H(m) = 9$  (supponiamo che  $H(\text{"ciao"}) = 9$ )

<sup>2</sup>Questo caso è estremamente raro se  $k$  è scelto correttamente

<sup>3</sup>Questi numeri non garantiscono alcuna sicurezza crittografica ma servono a capire meglio il funzionamento dell'algoritmo.

## Chiavi

- Chiave privata  $d = 7$
- Chiave pubblica  $Q = dG = 7G$  (supponiamo  $Q = (6, 3)$ )

## Generazione della firma

1. Calcolo dell'hash del messaggio:

$$e = H(m) = 9 \Rightarrow z = 9$$

2. Scelta del nonce segreto:

$$k = 3$$

3. Calcolo del punto  $kG$ :

$$kG = 3G = (10, 6) \Rightarrow x_1 = 10$$

4. Calcolo di  $r$ :

$$r = x_1 \mod n = 10 \mod 19 = 10$$

5. Calcolo di  $s$ :

Calcoliamo l'inverso di  $k \mod n$ :  $k^{-1} = 13 \mod 19$ , poiché  $3 \cdot 13 = 39 \equiv 1 \mod 19$ .

$$s = k^{-1}(z + dr) \mod n = 13(9 + 7 \cdot 10) \mod 19 = 13 \cdot 79 \mod 19 = 1027 \mod 19 = 1$$

**Firma generata:**  $(r, s) = (10, 1)$

## Verifica della firma

1. Verifica che  $r, s \in [1, n - 1]$ : OK

2. Calcolo dell'hash:

$$z = H(m) = 9$$

3. Calcolo dell'inverso di  $s$ :

$$w = s^{-1} \mod n = 1^{-1} \mod 19 = 1$$

4. Calcolo dei coefficienti:

$$u_1 = z \cdot w \mod n = 9 \cdot 1 \mod 19 = 9$$

$$u_2 = r \cdot w \mod n = 10 \cdot 1 \mod 19 = 10$$

5. Calcolo del punto:

$$(x_V, y_V) = [u_1]G + [u_2]Q = 9G + 10Q$$

Supponiamo che  $9G = (3, 1)$ ,  $10Q = (7, 14)$  e che la somma valga  $(10, 6)$

6. Verifica finale:

$$x_V \mod n = 10 \mod 19 = 10 = r \Rightarrow \text{firma valida}$$

**Conclusione:** La firma  $(10, 1)$  è corretta e verificabile, confermando l'integrità e l'autenticità del messaggio firmato.

## 2.4 Analisi della sicurezza di ECDSA

In questa sezione, esploreremo l'importanza della casualità di  $k$ , vedremo alcuni attacchi noti a ECDSA e le best practices per evitarli. Infine, andremo ad analizzare come ECDSA resiste o meno in uno scenario con attaccanti dotati di computer ed algoritmi quantistici.

### 2.4.1 Importanza della casualità di $k$

Nel processo di firma ECDSA, il valore  $k$  è un numero intero scelto in modo casuale per ogni firma. La sicurezza dell'intero schema dipende fortemente dalla **non ripetibilità** e **non prevedibilità** di questo valore. La sua gestione scorretta compromette direttamente la **riservatezza della chiave privata**  $d$ .

#### Riutilizzo di $k$

Se lo stesso valore  $k$  viene usato per firmare due messaggi distinti  $m_1$  e  $m_2$ , generando due firme  $(r, s_1)$  e  $(r, s_2)$ , allora un attaccante può calcolare  $k$  nel seguente modo:

$$k = \frac{z_1 - z_2}{s_1 - s_2} \mod n$$

dove  $z_1 = H(m_1)$ ,  $z_2 = H(m_2)$ , e l'inverso è calcolato modulo  $n$ . Una volta noto  $k$ , si può derivare la chiave privata  $d$  da una delle firme:

$$d = \frac{s \cdot k - z}{r} \mod n$$

#### Esempio pratico: l'attacco alla PlayStation 3 (2010)

Un attacco emblematico alla sicurezza di ECDSA si è verificato nel 2010 con la compromissione della *Sony PlayStation 3*. In quell'occasione, i ricercatori scoprirono che il firmware della console impiegava un generatore di numeri pseudo-casuali mal progettato per calcolare il nonce  $k$  usato nelle firme ECDSA. Invece di generare un valore di  $k$  completamente casuale e imprevedibile, come richiesto per la sicurezza dell'algoritmo, il sistema produceva valori *identici o altamente prevedibili* ogni volta che veniva generata una firma.

Ciò ha portato a una rottura completa del sistema di sicurezza della console: gli hacker (ovvero il gruppo fail0verflow e George Hotz) sono stati in grado di firmare codice come se fossero Sony stessa, eludendo le protezioni contro software non autorizzato.

#### Predizione parziale di $k$

Anche la **conoscenza parziale dei bit di  $k$** —ad esempio ottenuta tramite attacchi side-channel—può essere sufficiente per risalire a  $d$ , utilizzando tecniche di riduzione reticolare (es. attacchi Lattice basati su LLL <sup>4</sup>). Ciò accade perché la relazione che lega  $s$ ,  $k$ ,  $r$ ,  $d$  e  $z$  può essere trasformata in un'istanza di un problema di approssimazione di vettori corti (SVP), risolvibile se abbastanza bit di  $k$  sono noti.

---

<sup>4</sup>L'algoritmo LLL (Lenstra–Lenstra–Lovász) permette di risolvere problemi di reticoli ed è impiegato in crittoanalisi con conoscenza parziale di bit.

## Contromisure

Per mitigare questi attacchi è fondamentale che  $k$  sia:

- generato con una **sorgente di entropia crittograficamente sicura**;
- **unico** per ogni messaggio firmato;
- **imprevedibile**, anche parzialmente.

Lo standard RFC 6979 propone una variante detta *Deterministic ECDSA*, in cui  $k$  è generato **in modo deterministico** a partire dal messaggio  $m$  (più precisamente dal suo hash  $z = H(m)$ ) e dalla chiave privata  $d$ , tramite una funzione pseudo-casuale crittograficamente sicura come HMAC-DRBG. Questo significa che, per lo stesso messaggio e la stessa chiave privata, il valore di  $k$  sarà sempre lo stesso, ma **inaccessibile a un attaccante** che non conosce  $d$ . In questo modo si elimina completamente la dipendenza da generatori casuali esterni, riducendo drasticamente il rischio di errori implementativi.

### 2.4.2 Attacchi noti a ECDSA

La robustezza teorica di ECDSA non protegge automaticamente da tutte le minacce: molte vulnerabilità risiedono a livello implementativo, dove un attaccante può sfruttare informazioni ausiliarie o manipolazioni hardware per compromettere la segretezza delle chiavi. In questa sezione vengono illustrati alcuni attacchi pratici noti.

#### Side-Channel Attacks

Questi attacchi si basano sull'analisi delle informazioni fisiche emesse durante l'esecuzione dell'algoritmo. Non mirano alla rottura matematica di ECDSA, ma alla deduzione di bit sensibili osservando fenomeni collaterali:

- **Timing attacks:** nelle implementazioni di ECDSA, la computazione che coinvolge le operazioni matematiche possono richiedere tempi leggermente diversi a seconda dei bit 0 o 1 del nonce  $k$ . Se l'implementazione non è a tempo costante, ovvero se non impiega sempre lo stesso tempo indipendentemente dai dati, l'attaccante può misurare il tempo che il dispositivo impiega per firmare. Basandosi su queste minuscole differenze di tempo, può così dedurre quali bit erano 0 e quali 1, ricostruendo una parte o l'intero valore di  $k$ .
- **Simple Power Analysis (SPA):** questa tecnica sfrutta la correlazione diretta tra le operazioni e il loro consumo di energia. Durante la generazione della firma ECDSA, una delle operazioni fondamentali è il calcolo del punto  $R = [k]G$ , ottenuto tramite una moltiplicazione scalare del punto generatore  $G$  per l'intero segreto  $k$ . Questa operazione viene tipicamente realizzata con l'algoritmo *double-and-add* (vedi sezione 1.2.2). All'interno dell'algoritmo, le operazioni di *doubling* e *addition* hanno consumi energetici distinti, osservabili con un oscilloscopio o un'analisi delle tracce di potenza. Un attaccante può quindi ricostruire la sequenza dei bit di  $k$  osservando il pattern di consumo energetico del dispositivo durante l'esecuzione dell'algoritmo. Se  $k$  viene compromesso in questo modo, anche la chiave privata  $d$  può essere ricostruita.



- **Differential Power Analysis (DPA):** la DPA non si basa sull'osservazione di una singola operazione o di un singolo pattern evidente, ma sfrutta le *piccole differenze statistiche* nel consumo energetico che emergono quando il dispositivo elabora dati diversi.

Il processo tipico della DPA prevede i seguenti passaggi:

- **Raccolta di molte tracce di potenza:** Vengono registrate centinaia o migliaia di tracce di consumo energetico mentre il dispositivo esegue la stessa operazione (es. una cifratura), ma con *dati di input diversi* (e spesso anche la stessa chiave segreta).
- **Predizione del consumo:** L'attaccante formula delle ipotesi sui valori intermedi che un dispositivo dovrebbe assumere durante un'operazione specifica, basandosi su una *chiave parziale ipotizzata*. Ad esempio, in una cifratura AES, si può ipotizzare una parte della chiave e calcolare il valore del bit più significativo dell'output di uno S-box per ogni traccia.
- **Divisione delle tracce:** Le tracce di potenza raccolte vengono divise in due gruppi distinti, basandosi sul valore ipotizzato del bit intermedio (es. gruppo A se il bit è 0, gruppo B se il bit è 1).
- **Calcolo della differenza:** Viene calcolata la differenza media tra i profili energetici dei due gruppi (A e B) per ogni punto temporale.
- **Identificazione della chiave corretta:** Se l'ipotesi sulla chiave parziale è corretta, la differenza media mostrerà un "picco" significativo in corrispondenza del momento in cui l'operazione che dipende da quel bit viene eseguita. Se l'ipotesi è sbagliata, la differenza sarà prossima allo zero. Ripetendo questo processo per tutte le possibili chiavi parziali, l'attaccante può identificare la chiave segreta.

### 2.4.3 Best practices per un uso sicuro di ECDSA

Per garantire un uso sicuro di ECDSA, è fondamentale adottare alcune best practices tra cui la generazione sicura e non riutilizzabile di  $k$  (es. RFC 6979), l'uso di curve standard e la verifica dei punti sulla curva. Inoltre, vediamo nel dettaglio come prevenire i vari side-attacks descritti nella sezione superiore.

- **Time attacks:** per prevenire questo tipo di attacchi, la cosa migliore è utilizzare implementazioni a tempo costante di tutte le operazioni, indipendentemente dai bit di  $k$ .
- **SPA:** le contromisure a questo attacco prevedono di rendere indistinguibili le operazioni o di mascherare le informazioni reali su  $k$ .
  - **Offuscamento algoritmico:** anziché calcolare  $[k]G$  direttamente, si calcola  $[k + r \cdot n]G$  dove  $k$  è lo scalare segreto,  $r$  è un numero intero casuale generato al momento e  $n$  è l'ordine della curva ellittica (è pubblico). In questo modo, dato che  $nG = O$  (il punto all'infinito), allora  $r \cdot nG = O$ . Di conseguenza,  $[k + r \cdot n]G = [k]G + [r \cdot n]G = [k]G + O = [k]G$ . Il risultato finale quindi non cambia, ma cambia la sequenza di operazioni eseguite, che in questo modo dipende da  $r$  che varia ad ogni esecuzione, rendendo SPA inutilizzabile.

- **Utilizzo di algoritmi uniformi:** un'alternativa all'offuscamento algoritmico, è l'utilizzo di alcuni algoritmi che sono progettati per eseguire una sequenza di operazioni che è indipendente dal valore dei bit di  $k$ .
  - \* **Scala di Montgomery:** durante il calcolo della moltiplicazione scalare  $[k]G$ , l'algoritmo lavora con due punti diversi in parallelo invece di uno solo, tipicamente  $P_0$  e  $P_1$ . Per ogni bit di  $k$ , vengono sempre eseguite sia un'operazione di doubling che un'operazione di addition, indipendentemente dal valore del bit. La selezione dei punti su cui operare e la riassegnazione dei risultati avvengono tramite meccanismi a tempo costante (es. swap condizionali), eliminando le ramificazioni dipendenti dal dato segreto. Questo impedisce all'attaccante di distinguere i bit di  $k$  basandosi su variazioni nella traccia di potenza.
  - \* **Double-and-Add Always:** questo algoritmo prevede l'esecuzione di entrambe le operazioni di *doubling* e *addition* dell'algoritmo *double-and-add* per ogni bit di  $k$ , sia che il bit sia 0 che 1. In questo modo viene eliminata la dipendenza della traccia energetica dal valore del bit. (C'è del lavoro inutile alla computazione dell'algoritmo).
- **DPA:** le contromisure a questo tipo di attacco mirano a rompere la correlazione tra i dati intermedi (che dipendono dalla chiave segreta) e il consumo energetico osservabile. Le tecniche principali includono:
  - **Randomizzazione dei dati intermedi:** consiste nell'introdurre variabili casuali nei calcoli interni (es. maschere additive o moltiplicative su  $k$  e sui punti della curva), in modo che i valori intermedi cambino a ogni firma, pur mantenendo invariato il risultato finale.
  - **Coordinate randomizzate:** invece di usare le coordinate affini  $(x, y)$ , si possono usare coordinate proiettive o Jacobiane con randomizzazione. Ad esempio, si moltiplica il punto iniziale  $G$  per uno scalare casuale prima della moltiplicazione scalare.
  - **Point blinding:** si calcola  $[k]G$  come  $[k](G + [r]G) - [k][r]G$ , dove  $r$  è un intero casuale scelto ogni volta. Questa tecnica “nasconde” le operazioni reali rendendo più difficile isolare il contributo diretto di  $k$ .
  - **Scalar blinding:** anziché usare direttamente  $k$ , si usa  $k' = k + rn$ , dove  $r$  è casuale e  $n$  è l'ordine del punto  $G$ . Poiché  $nG = \mathcal{O}$ , il risultato rimane  $[k']G = [k]G$ , ma la sequenza di operazioni cambia, rendendo inefficace l'analisi statistica.

# Riferimenti bibliografici

- [1] Frrouting official documentation, 2025. Accessed: 2025-07-04. URL: <https://docs.frrouting.org/>.
- [2] Frrouting project - overview, 2025. Accessed: 2025-07-04. URL: <https://frrouting.org/>.
- [3] Gobgp api and configuration reference, 2025. Accessed: 2025-07-04. URL: <https://osrg.github.io/gobgp/docs/reference/>.
- [4] Gobgp documentation, 2025. Accessed: 2025-07-04. URL: <https://osrg.github.io/gobgp/>.
- [5] Cloudflare, Inc. What is an autonomous system?, n.d. Accessed: 2025-07-05. URL: <https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/>.
- [6] DataPath.io. The history of border gateway protocol, 2016. Accessed: 2025-07-26. URL: [https://medium.com/@datapath\\_io/the-history-of-border-gateway-protocol-a212b7ee6208](https://medium.com/@datapath_io/the-history-of-border-gateway-protocol-a212b7ee6208).
- [7] Docker. Lxc vs docker, 2013. Accessed: 2025-07-06. URL: <https://www.docker.com/blog/lxc-vs-docker/>.
- [8] Fortinet. Tcp/ip model vs. osi model, 2024. URL: <https://www.fortinet.com/resources/cyberglossary/tcp-ip-model-vs-osi-model>.
- [9] John Hawkinson and Tony Bates. Guidelines for creation, selection, and registration of an autonomous system, 1996. RFC 1930, IETF. URL: <https://datatracker.ietf.org/doc/html/rfc1930>.

- [10] Nils Höger, Nils Rodday, and Gabi Dreo Rodosek. Mitigating bgp route leaks with attributes and communities: A stopgap solution for path plausibility, 2025. Accessed: 2025-08-23. URL: <https://onlinelibrary.wiley.com/doi/10.1002/nem.70002>, doi:10.1002/nem.70002.
- [11] Internet Assigned Numbers Authority. IANA number resources, 2024. Accessed July 5, 2025. URL: <https://www.iana.org/numbers>.
- [12] IONOS. Software di virtualizzazione a confronto: i migliori strumenti per creare macchine virtuali, 2024. Ultimo accesso: 5 luglio 2025. URL: <https://www.ionos.it/digitalguide/server/configurazione/software-di-virtualizzazione-a-confronto/>.
- [13] S. Kent, C. Lynn, and K. Seo. An Infrastructure to Support Secure Internet Routing. RFC 6480, 2012. Accessed: 2025-08-26. URL: <https://www.rfc-editor.org/rfc/rfc6480>, doi:10.17487/RFC6480.
- [14] M. Lepinski and K. Sriram. BGPsec Protocol Specification. RFC 8205, 2017. Accessed: 2025-08-24. URL: <https://www.rfc-editor.org/rfc/rfc8205>, doi:10.17487/RFC8205.
- [15] Kirk Lougheed and Yakov Rekhter. RFC 1163: A Border Gateway Protocol (BGP). <https://datatracker.ietf.org/doc/html/rfc1163>, 1990. Accessed: 2025-07-26.
- [16] Kirk Lougheed and Yakov Rekhter. RFC 1267: A Border Gateway Protocol 3 (BGP-3). <https://datatracker.ietf.org/doc/html/rfc1267>, 1991. Accessed: 2025-07-26.
- [17] ManageEngine. Border gateway protocol (bgp), 2024. URL: <https://www.manageengine.com/it-operations-management/border-gateway-protocol.html>.
- [18] J. Mitchell and J. G. Scudder. Autonomous system (as) reservation for private use, 2011. RFC 6996, IETF. URL: <https://datatracker.ietf.org/doc/html/rfc6996>.

- [19] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. Bgpstream: A software framework for live and historical bgp data analysis. In *Proceedings of the 2016 Internet Measurement Conference (IMC)*, pages 437–444. ACM, 2016. URL: <https://dl.acm.org/doi/10.1145/2987443.2987482>, doi:10.1145/2987443.2987482.
- [20] Samuel Brako Oti and Joseph Hayfron-Acquah. Practical security approaches against border gateway protocol (bgp) session hijacking attacks between autonomous systems. *Communications and Network*, 6(3):167–176, 2014. URL: <https://www.scirp.org/journal/paperinformation.aspx?paperid=46857>, doi:10.4236/cn.2014.63019.
- [21] Yakov Rekhter and Tony Li. RFC 1654: A Border Gateway Protocol 4 (BGP-4). <https://datatracker.ietf.org/doc/html/rfc1654>, 1994. Accessed: 2025-07-26.
- [22] Yakov Rekhter, Tony Li, and Susan Hares. RFC 4271: A Border Gateway Protocol 4 (BGP-4). <https://datatracker.ietf.org/doc/html/rfc4271>, 2006. Accessed: 2025-07-26.
- [23] Yakov Rekhter and Kirk Lougheed. RFC 1105: Border Gateway Protocol (BGP). <https://datatracker.ietf.org/doc/html/rfc1105>, 1989. Accessed: 2025-07-26.
- [24] RIPE NCC. Youtube hijacking: A ripe ncc ris case study. Technical report, RIPE Network Coordination Centre, March 2008. Accessed: 2025-08-27. URL: <https://www.ripe.net/about-us/news/youtube-hijacking-a-ripe-ncc-ris-case-study>.
- [25] RIPE NCC. Routing information service (ris), 2025. Accessed: 2025-08-26. URL: <https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris>.
- [26] Eric C. Rosen. RFC 904: Exterior Gateway Protocol (EGP). <https://datatracker.ietf.org/doc/html/rfc904>, 1984. Accessed: 2025-07-26.

- [27] Kotikalapudi Sriram, Doug Montgomery, Danny McPherson, Eric Osterweil, and Marla Azinger. RFC 7908: Problem Definition and Classification of BGP Route Leaks. <https://datatracker.ietf.org/doc/html/rfc7908>, 2016. RFC 7908, IETF, Accessed: 2025-08-22.
- [28] Sean Turner and Oliver Borchert. BGPsec Algorithms, Key Formats, and Signature Formats. RFC 8608, 2019. URL: <https://www.rfc-editor.org/rfc/rfc8608>, doi:10.17487/RFC8608.
- [29] University of Oregon. University of oregon route views project, 2025. Accessed: 2025-08-26. URL: <http://www.routeviews.org/>.
- [30] US–China Economic and Security Review Commission. 2010 report to congress of the u.s.-china economic and security review commission. Technical report, U.S. Government Printing Office, November 2010. Accessed: 2025-08-27. URL: <https://www.uscc.gov/annual-report/2010-annual-report-congress>.
- [31] Zheng Zhang, Z. Morley Mao, Ming Zhang, Bo Gao, Ben Y. Zhao, and Anthony D. Joseph. Practical defenses against bgp prefix hijacking, 2007. Proceedings of ACM CoNEXT 2007, Accessed: 2025-08-22. URL: <https://web.eecs.umich.edu/~zmiao/Papers/conextDefendHijack07.pdf>.







# Ringraziamenti

Grazie a tutti