UNIVERSITÀ POLITECNICA DELLE MARCHE

Dipartimento di Ingegneria dell'Informazione

TESI DI LAUREA MAGISTRALE

Thesis Title

Author:
Davide DE ZUANE

Supervisor:
Dr. Paolo SANTINI

October 5, 2024

?*Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.***?**

Dave Barry

UNIVERSITÀ POLITECNICA DELLE MARCHE

Abstract

Faculty Name
Dipartimento di Ingegneria dell'Informazione

Doctor of Computer Sciences

Thesis Title

by Davide DE ZUANE

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

List of Abbreviations

DH	**D**iffie **H**ellman
KE	**K**ey **E**xchange
PQ	**P**ost **Q**uantum
IKE	**I**nternet **K**ey **E**xchange
KEM	**K**ey **E**ncapsulation **M**echanism
PRF	**P**seudo **R**andom **F**unction
MTU	**M**aximum **T**ransmission **U**nit
ISP	**I**nternet **S**ervice **P**rovider

List of Symbols

$\mid$	concatenazione	
a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

For/Dedicated to/To my…

Chapter 1

Introduction

Introduzione al quantum computer e spiegare il nome, ovvero perchè si basa sui principi fisici. Confronto con il computer classico utilizzato per calcolare la classe di complessità di un problema

L'attuale crittorafia a chiave pubblica è minacciata da due algoritmi pioneri in questo campo ovvero quello di Grover e Shor Negli utilimi anni la minaccia del quantum computing, in particolare la loro potenza di calcolo insieme agli algoritmi di Grover e shor ha ribaltato quelle che sono le carte in tavola, dato che consentono di risolvere in tempo polinomiale i problemi su cui si basano gli schemi crittografici più diffusi tra cui il problema del logaritmi discreto alla base di DH e la fattorizzazione di numeri primi alla base di RSA.

Questo ha spinto a introdurre nuovi schemi di firma basati su problemi matematici più complessi, tra questi abbiamo i lattice-based, hash-based, ecc.. Recentemente tra quelli proposti ne sono stati standardizzati diversi, tra questi abbiamo: kyber, dilithium, classim mceliece

Criticità

- Tradeoff tra aumento della complessità con dimensioni chiave e velocità delle operazioni
- Requisiti della rete e in generale delle applicazioni
- Implementazioni
- Affidabilità e dunque transizione da uno all'alto.

Ora il fatto che questi problemi si siano dimostrati computazionalmente onerosi non li rende ottimali anche per l'utilizzo pratico su tutte quelle che sono le infrastrutture di rete esistenti.

Questo perchè per contrastare l'incredibile potenza di calocolo del quantum computer occorre rendere il problema più complesso che in generale potrebbe

portare e dimensioni delle chiavi molto grandi oppure ad operazioni di keygen, codifica e decoddifica molto lonte.

Dunque l'algoritmo dal punto di vista matematico soddisfa quelli che sono i requisiti tuttavia non è detta che soddisfi quelli che sono i requisiti che lo rendano adatto ad essere applicato a contesti reali come quello dlele reti di computer.

Oltre ad un problema computazionale abbiamo anche un problema di fiducia dei cofronti di questi algoritmi, ovvero dato che sono stati appena introdotti l'implementazione potrebbe peccare da qualche punto di vista Inoltre fare una transizione così drastica risulta molto problematico.

Io la metterei si dal punto di vista della ricerca ma anche dal punto di vista implementativo, ovvero non basta definire solamente nuovi schemi che dal punto di vista teorico possono essere sicuri ma questi devono poi trovare un'utilizzo pratico.

L'utilizzo pratico va in contro a diverse problematiche in particolare ha requisiti più stringenti che al momento della definizione matematiche dello schema non vengono presi in considerazione. Si hanno constraint sia di usabilità che di fiducia nei loro confronti poichè l'approccio standard è quello di aumentare le dimensioni delle chiavi in modo tale da contrapporti all'aumento di capacità computazionale del quantum computer

Nel caso reale l'autemto di dimensione ha effetti significativi sulle prestazioni della rete dato che possono portare a problematiche di frammentazione. E va considerata anche la latenza dovuta alle operazioni di cifratura e altre cose

Contributo Apportato

L'obiettivo di questo lavoro è andare a vedere quali sono gli effetti di applicare primitive di questo tipo nei protocolli maggiormenti diffusi per la sicurezza delle comunicazioni. In particolare considerando il caso specifico di comunicazioni satellitari, che hanno constraint importanti sul numero di pacchetti da scambiare e di conseguenza sulla dimesione di quest'utilimi

Una volta determinati quelli che sono gli effetti, siamo passati a verificare se il protocollo utilizzato rispetto alla sua applicazione, fosse quello ideale. In particolare dalle conclusioni del benchmarcking siamo arrivati ad una prima implementazione, molto spartana, di quella che è una versione minimale di IKE.

Organizazione della Tesi

Il proseguo della tesi sarà strutturato nel seguente modo:

- Capitolo 2: si danno le fondamenta matematica delle sicurezza nelle comunicazioni sicure e di come queste vengono applicte nelle comunicazioni digitali. In particolare prendiamo in esame il caso di IPsec e di un suo protocollo ausiliario utilizzato per negoziarne i parametri di sicurezza.
- Capitolo 3: descrizione di quello che è lo scenario applicativo che si prende in considerazione

Chapter 2

Fondamenti di Comunicazioni Sicure

In questo capitolo esamineremo le problematiche relative alla sicurezza nelle comunicazioni. Inizieremo con un'analisi degli strumenti matematici fondamentali che sono alla base della protezione dei dati, esplorando le tecniche crittografiche e i loro principi teorici. Successivamente, ci concentreremo su come queste tecniche vengono effettivamente applicate per garantire la sicurezza nelle comunicazioni sulle reti di computer.

2.1 Teoria

Dalla crittografia classica, come il cifrario di Cesare, fino alle tecniche più sofisticate del ventesimo secolo, come i sistemi di cifratura a chiave pubblica, la storia della crittografia è caratterizzata da una continua evoluzione e innovazione. Le fondamenta sui cui si basa non sono cambiate, andiamo a vedere strumenti matematici che ne fanno parte.

2.1.1 Hash Function

Una funzione **hash crittografiche** è una funzione matematica che prende in input un messaggio di lunghezza arbitraria e restituisce un output di lunghezza fissa, noto come digest.

$$H : \{0,1\}^* \to \{0,1\}^n \tag{2.1}$$

- $\{0,1\}^*$: l'insieme di tutte le stringhe binarie di lunghezza arbitraria.
- $\{0,1\}^n$: l'insieme delle stringhe binarie di lunghezza fissa n.

Le funzioni di hash crittografiche sono strumenti fondamentali nel campo della sicurezza informatica, progettate per garantire l'integrità e l'autenticità

dei dati. Per questo motivo, a questo tipo di funzioni sono richieste le seguenti proprietà:

- **Resistenza alle collisioni**: devono essere progettate in modo tale che sia computazionalmente impraticabile invertire il processo, ovvero, dato il digest è dificile risalire al messaggio che lo ha prodotto.
- **Proprietà di diffusione**: una leggere variazione dell'input deve produrre un hash completamente diverso, questa è fondamentale per garantire che gli attaccanti non possano prevedere o manipolare il valore di hash a seguito di modifiche all'input.

aggiungere grafico sulle one way function

2.1.2 Schemi crittografici

Uno schema di cifratura è un insieme di algoritmi e funzioni che definisce come trasformare un messaggio in chiaro (plaintext) in un messaggio cifrato (ciphertext) e viceversa, al fine di garantire la confidenzialità e la sicurezza delle comunicazioni. Formalmente possiamo rappresentarlo come una quintupla:

$$(\mathcal{P}, \mathcal{C}, \mathcal{K}, E, D)$$

- $\mathcal{P}$: Insieme dei messaggi in chiaro (plaintext).
- $\mathcal{C}$: Insieme dei messaggi cifrati (ciphertext).
- $\mathcal{K}$: Insieme delle chiavi utilizzate per la cifratura e decifratura, *key space*.
- $E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$: Funzione di cifratura.
- $D : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$: Funzione di decifratura.

Deve esistere una relazione inversa tra le operazioni di cifratura e decifratura:

$$D(k, E(k, m)) = m \quad \forall m \in \mathcal{P}, k \in \mathcal{K} \tag{2.2}$$

Lo schema in *Fig.* 2.1, mostra il funzionamento generale di uno schema crittografico. Tuttavia andando a caratterizzare le chiavi utilizzate nelle operazioni di cifratura e decifratura possiamo dare una prima classificazione:

- se $K_1 = K_2$ allora si parla di un schema di crittografia *simmetrico*.
- se $K_1 \neq K_2$ allora si parla di schema di crittografia *asimmetrico*.

Colorare con colori giusti lo schema

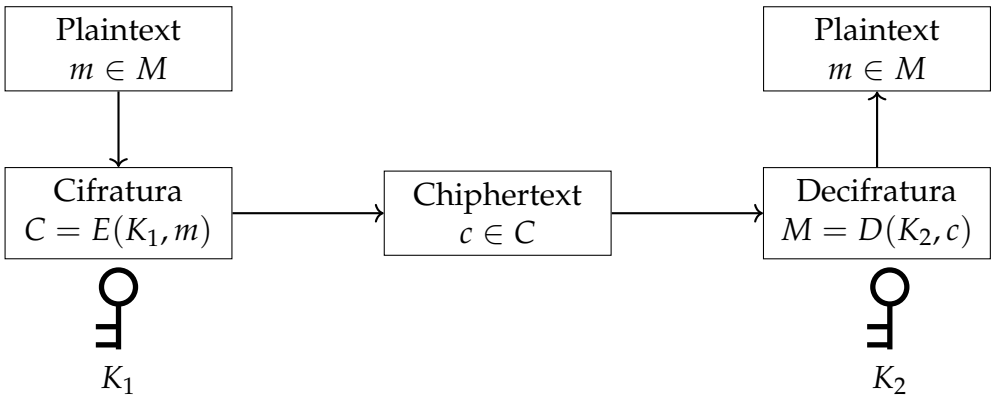

FIGURE 2.1: Funzionamento di uno schema crittografico

Simmetrici

Come descritto precedentemente, in uno schema simmetrico si utilizza la stessa chiave sia per le operazioni di cifratura che di decifratura. Ciò implica che le due parti coinvolte nella comunicazione debbano possedere la medesima chiave segreta, nota anche come chiave pre-condivisa (PSK - Pre-Shared Key). Questa caratteristica fondamentale rende gli schemi di crittografia simmetrica particolarmente veloci ed efficienti, esempi ne sono AES e DES.
Le loro caratteristiche li rendono ideali per:

- *Cifratura* di Dati: proteggere file e database memorizzati su disco, garantendo che le informazioni sensibili rimangano riservate anche in caso di accesso non autorizzato. Sia proteggere i dati mentre vengono trasmessi su reti.
- *HMAC* (Hash-based Message Authentication Code): combinati con funzioni di hash, gli algoritmi simmetrici possono generare codici HMAC, che forniscono autenticità e integrità ai messaggi. Fondamentale per garantire che i dati non vengano manomessi durante la trasmissione.

Asimmetrici

In uno schema di cifratura asimmetrica, detto anche a **chiave pubblica**, lo spazio delle chiavi $\mathcal{K}$ è costituito da una coppia di chiavi $(k_{\text{pub}}, k_{\text{priv}})$, dove:

- La chiave pubblica k_{pub} viene condivisa liberamente e utilizzata da chiunque per cifrare messaggi destinati al proprietario della chiave.
- La chiave privata k_{priv} è mantenuta segreta dal proprietario e viene utilizzata per decifrare i messaggi cifrati con la corrispondente chiave pubblica.

Quindi le due funzioni si riscrivono come:

$$E : \mathcal{K}_{\text{pub}} \times \mathcal{P} \to \mathcal{C} \tag{2.3}$$

$$D : \mathcal{K}_{\text{priv}} \times \mathcal{C} \to \mathcal{P} \tag{2.4}$$

Le due chiavi sono matematicamente legate, ma è computazionalmente difficile
ottenere la chiave privata a partire da quella pubblica. Il funzionamento si basa sul concetto di **trapdoor** che rende possibile una funzione (come la cifratura o la decifratura) semplice per chi conosce un segreto (la chiave privata) ma estremamente difficile per chi non lo conosce.

Uno dei principali utilizzi della crittografia asimmetrica è il **Key-Exchange**, il quale consente di scambiarsi un'informazione segreta su un canale pubblico. La procedura più conosciuta è quella proposta da *Diffie-Hellman* ed è mostrata in *Fig.* 2.2

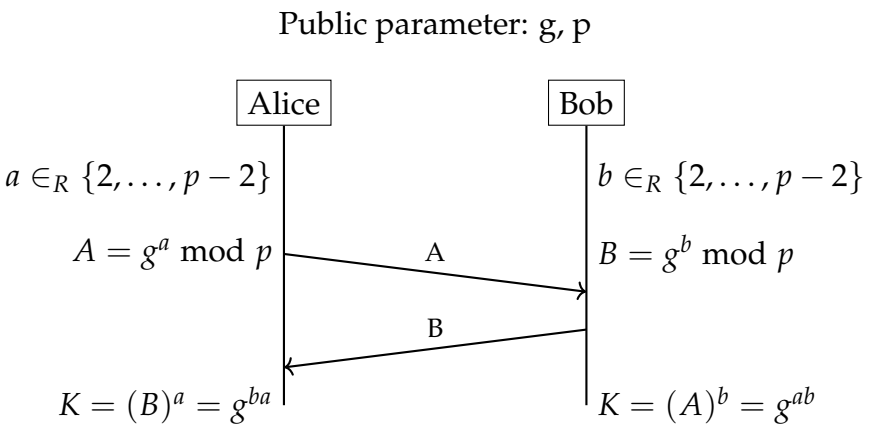

FIGURE 2.2: Scambio di Chiave Difie-Hellman

aggiungere una spiegazione più approfondita sui KE

Le **firme digitali**, sono un meccanismo chiave per garantire l'autenticità e l'integrità dei messaggi. In questo caso si fornisce sia il messaggio che un digest del messaggio firmato, in questo modo chi lo riceve può utilizzare la chiave pubblica per verificare che l'hash firmato equivalga a quello calcolato. Questa pratica si utilizza per garantire integrità e autenticità.

aggiungere una spiegazione più approfondita sulle firme digitali

Key Distribution

L'assunto che si è fatto in entrambi le tipologie di schema è che l'altra parte della comunicazione avesse ottenuto in qualche modo la chiave. Tuttavia la

distribuzione delle chiavi è un problema importante per il crittosistema.

Le distribuzione delle chiavi per crittosistemi simmetrici deve avvenire tramite un canale segreto, per questo motivo si utilizzano le seguenti modalità:

- *Manuale*: vengono installate manualmente coppie di chiavi per ogni nodo che si vuol far comunicare, se si vogliono far comunicare n nodi sono necessarie $n(n-1)/2$ chiavi. Questo approccio è robusto, essendo decentralizzato, ma risulta impraticabile per reti di grandi dimensioni come mostrato in *Fig.* 2.3.
- *Key Distribution Center (KDC)*: da un'approccio decentralizzato si passa ad uno centralizzato, in cui è presente un server che fa da intermediario fidato per la distribuzione delle chiavi.

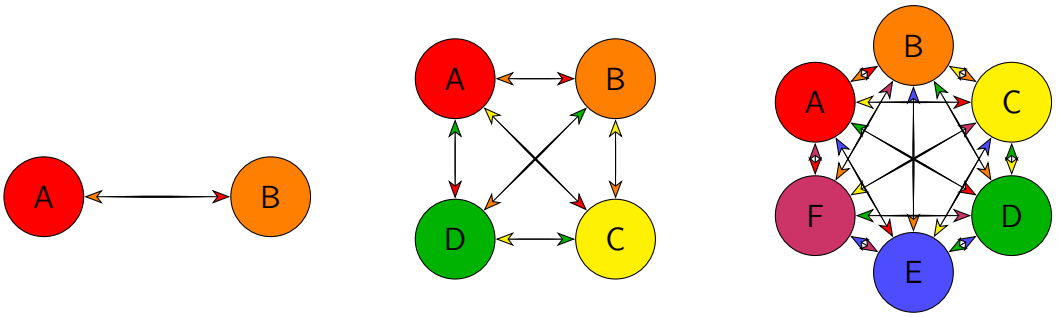

FIGURE 2.3: Esplosione combinatoria

cambaire i colori ai nodi

Per loro natura le chiavi pubbliche sono liberamente distribuite, non è dunque necessario un canale segreto. Il problema che si hanno con questa tipologia di chiavi è quello dell'autenticità, ovvero che la chiave provenga realmente dalla fonte dichiarata. Questa tipologia di schemi infatti per loro natura è vulnerabile ad attacchi MITM. Per questo nasce la *Public Key Infrastructure (PKI)*, che tramite l'utilizzo dei certificati X.509 consente la distribuzione sicura di chiavi pubbliche.

aggiungere immagine della PKI

Questo approccio consente una gestione scalabile delle chiavi.

Attacchi Quantum

Gli schemi asimmetrici sono vulnerabili alla minaccia del quantum computer Parlare degli algoritmi di Shor e ....

Per rispondere a questa minaccia emergente, si stanno sviluppando algoritmi post-quantum (PQC), progettati per resistere agli attacchi di potenti computer quantistici.

Formalizzare per bene questa parte in particolare gli effetti di

chiedere a Paolo come scrivere bene questa parte

2.1.3 Sicurezza

Il **security level** è una misura della forza che una primitiva crittografica raggiunge rispetto ad attacchi. Solitamente viene espresso come un numero di "bit di sicurezza", dove n-bit di sicurezza significa che l'attaccante dovrebbe eseguire 2^n operazioni per romperlo.

- Per i cifrari simmetrici il livello di sicurezza è pari alla dimensione del key-space. Equivale ad un attacco a forza bruta.
- La sicurezza dei cifrari simmetrici si basa su problemi matematici noti. Tuttavia, gli attacchi contro gli attuali sistemi a chiave pubblica sono sempre più veloci della ricerca a forza bruta dello spazio delle chiavi.

Il NIST (National Institute of Standards and Technology) ha introdotto livelli di sicurezza, definiti in *Tabella 2.1* per gli algoritmi di cifratura asimmetrica e post-quantistica come parte della sua iniziativa per standardizzare algoritmi che resistano anche ai computer quantistici.

Security Level	Descrizione
Livello 1	Sicurezza equivalente alla cifratura simmetrica con chiavi da 128 bit, come AES-128.
Livello 2	Sicurezza equivalente ad attacchi contro SHA-256, con complessità circa pari a 128 bit. Leggermente più sicuro del livello 1.
Livello 3	Sicurezza equivalente alla cifratura simmetrica con chiavi da 192 bit, come AES-192.
Livello 4	Sicurezza equivalente ad attacchi contro SHA-384. Leggermente più sicuro del Livello 3.
Livello 5	Sicurezza equivalente alla cifratura simmetrica con chiavi da 256 bit, come AES-256.

TABLE 2.1: Security Levels definiti dal NIST

2.2 Applicazioni

Le reti sono un mezzo di comunicazione intrinsecamente insicuro, soprattutto quando operano in modalità broadcast. In questo contesto, la crittografia riveste un ruolo cruciale nel garantire la sicurezza dei dati scambiati tra entità remote. I crittosistemi, ossia le applicazioni crittografiche, integrano algoritmi di cifratura, autenticazione e gestione delle chiavi per fare in modo che vengano rispettati i requisiti di sicurezza per le informazioni trasmesse.

Il modello di riferimento per la comunicazione su Internet è il modello **TCP/IP**, il quale suddivide il processo di trasmissione dei dati in vari livelli, ciascuno con delle funzioni specifiche che non si sovrappongono con quelle degli altri livelli. Come mostrato in *Figura 2.4*, è possibile applicare la sicurezza ai vari livelli della pila e di lato sono riportati i protocolli che vengono utilizzati.

- SSH (Secure Shell): Protegge l'accesso remoto e il trasferimento di file, fornendo autenticazione e cifratura.
- TLS (Transport Layer Security): permette di instaurare una connessione TCP sicura. Viene utilizzato per cifrare e autenticare i dati tra client e server.
- IPsec (Internet Protocol Security): Protegge i pacchetti IP scambiati tra due nodi, fornendo autenticazione, integrità e cifratura.

finire di spiegare anche gli altri protocolli di sicurezza e colorare la pila

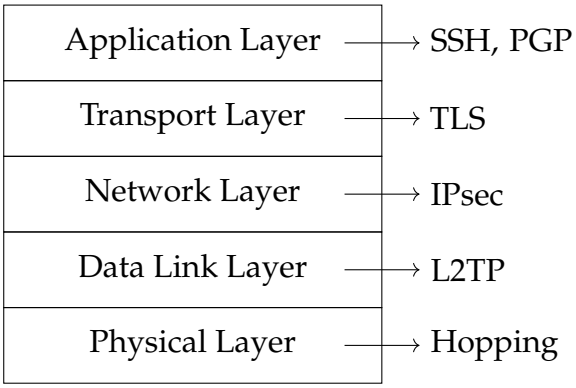

FIGURE 2.4: Stack TCP/IP

Ogni protocollo avrà le proprie caratteristiche, tuttavia fare sicurezza a L3 dello stack TCP/IP offre un vantaggio significativo: poiché tutti gli strati superiori dipendono da esso per la trasmissione dei dati, non è necessario apportare modifiche ai singoli protocolli o applicazioni che si basano su di

esso. Questo consente di implementare soluzioni di sicurezza centralizzate e trasparenti, senza dover intervenire su ciascun servizio o applicazione a livello più alto.

2.3 IPsec

IPsec (Internet Protocol Security) è un insieme di protocolli standard, suddivisi tra core e ausiliari in *Figura 2.5*, utilizzati in modo tale da fornire meccanismi per l'autenticazione, la cifratura e l'integrità dei dati trasmessi tra due o più dispositivi, proteggendo così le comunicazioni IP da intercettazioni e manomissioni.

2.3.1 Architettura

Come definito dall'RFC 1825, l'architettura di IPsec è composta dai seguenti componenti:

- **AH (Authentication Header)**: si tratta di un protocollo di sicurezza che fornisce autenticazione e integrità dei dati, garantendo che i pacchetti non vengano modificati durante la trasmissione. Non offre cifratura, quindi i dati rimangono in chiaro.
- **ESP (Encapsulating Security Payload)**: protocolo che fornisce cifratura per garantire la riservatezza dei dati, oltre a integrità e autenticazione opzionale. ESP è il protocollo più utilizzato per garantire sia sicurezza che riservatezza.
- **SA (Security Association)**: un insieme di parametri che definisce come i dati devono essere protetti durante la comunicazione tra due entità su una rete. Ogni SA contiene le informazioni necessarie per stabilire e mantenere una connessione sicura.
- **IKE (Internet Key Exchange)**: protocollo che consente di negoziare, autenticare e distribuire dinamicamente le chiavi crittografiche che vengono poi impiegate dai protocolli di sicurezza per proteggere le comunicazioni.
- **Algoritmi**: gli algoritmi crittografici e di hashing utilizzati per ottenere sicurezza.

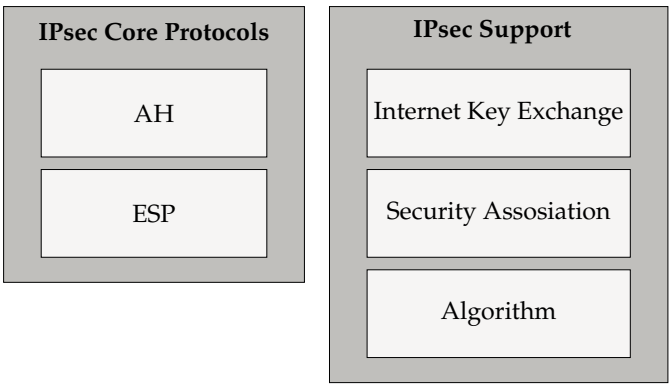

FIGURE 2.5: IPsec Protolocol Suite

2.3.2 Security Association

IP è un protocollo *stateless*, ovvero non mantiene informazioni o stato relativo alle connessioni o ai pacchetti che gestisce. Tuttavia affinchè IPsec possa garantire la sicurezza è necessario che mantenga il contesto di ogni connessione, le principali motivazioni sono:

- *Replay Protection*: per evitare attacchi di tipo reply, IPsec tiene traccia dei numeri di sequenza dei pacchetti, un'inforamzione di stato che va mantenuta per ogni connessioni e che IP non fa nativamente.
- *Connessioni Multiple*: in uno scenario di rete complesso, un singolo dispositivo potrebbe avere più connessioni sicure in corso simultaneamente, ognuna delle quali ha i propri parametri di sicurezza. IPsec deve tenere traccia di queste informazioni per sapere come trattare i pacchetti in entrata e uscita in base alla connessione a cui appartengono.
- *Protezione*: i protocolli di sicurezza AH e ESP richiedono di conoscere le chiavi crittografiche corrette e gli algoritmi utilizzati per cifrare e decifrare i pacchetti.

IP diventa in grado di mantenere un'insieme di informazioni di stato grazie al concetto di *Security Association* (SA). Più precisamente si tratta di un'insieme di parametri che servono per associare a ciascun canale uno stato condiviso tra le entità coinvolte nello comunicazione, tra questi abbiamo:

- *Security Parameter Index* (SPI): un'identificatore della SA.
- *Destination Address*: serve all'host per determinare quale SA utilizzare.
- *Lifetime*: il tempo di vita della SA, si obbliga a refresh periodici.

- *Protocol Identifier*: determina il tipo di protezione da applicare ai pacchetti, dunque anche chiavi e algoritmi associati.
- Altri parametri opzionali, per una lista completa fare riferimento all'RFC.

La SA è caratterizzata dall'essere un canale *simplex*, dunque al fine di stabilire un canale di comunicazione bidirezionale IPsec tra due entità occorrono due SA unidirezionali di verso opposto. La *Figura 2.6* mostra il tunnerl virtuale in esecuzione tra i due host.

aggiungere al lato di un host la configurazione dell'SA

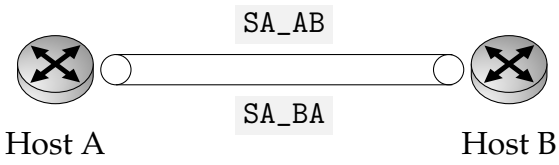

FIGURE 2.6: Security Association bidirezionali

2.3.3 Negoziazione SA

In un contesto come quello delle reti potremmo avere che lo stesso nodo ha connessini IPsec multiple, le SA consentono di distinguere e identificare in modo univoco la configurazioni di sicurezza da applicare alla comunicazione. Tuttavia queste SA come si configurano?. IPsec prevede tecniche di negoziazione delle SA di tipo:

- *Manuale*: occorre configurire manualmente le chiavi e le impostazioni di sicurezza per ciascun dispositivo o punto finale di comunicazione.
- *Automatico*: si utilizzanoprotocolli per stabilire automaticamente le chiavi di crittografia e le politiche di sicurezza senza intervento umano diretto.

L'utilizzo di tecniche di negoziazione automatica offre un approccio sicuro, flessibile e scalabile alla gestione delle Security Association, un esempio di questo è IKE. Andiamo a vedere nel dettaglio IKE nella prossima sezione.

2.4 IKE

Questo protocollo definisce una serie di scambi, mostrati in *Figura 2.7*, al termine del quale i due peer avranno negioziati i parametri di sicurezza e le chiavi crittografiche per una SA.

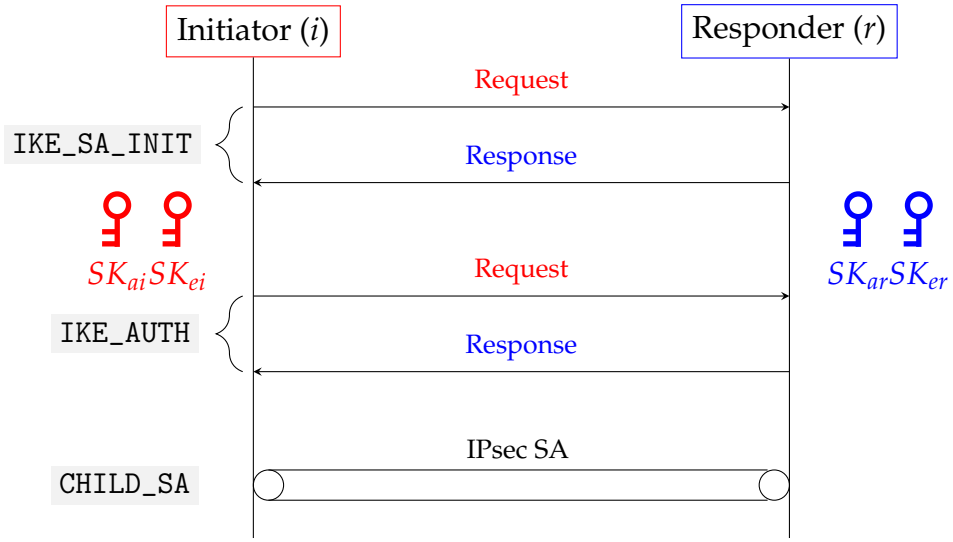

FIGURE 2.7: Fasi di Negoziazione del Protocollo IKEv2

2.4.1 IKE_SA_INIT

Lo scopo di questa prima fase è quello di creare una **IKE SA**, che consenta di rendere sicure i successivi scambi di dati al fine di realizzare una **IPsec SA**. Dunque funge da apripista al fine di stabilire quelli che sono i parametri di sicurezza al fine di avere una comunicazione sicura. Per questo motivo in questo scambio i peer si scambiano le seguenti informazioni:

TABLE 2.2: Tabella dei parametri e delle descrizioni

Parametro	**Descrizione**
SA	Security Association, vengono negoziati i parametri per la SA
KE	Key Exchange, e nel caso classico è l'esponente DH
N	Nonce

Al termine di questo scambio i due peer ottengono il *DH Shared Secret* (indicato con g^{ir}), il quale insisme ai nonce, consentirà di ottenere quelli che sono i parametri di sicurezza della $IKESA$ al fine di instauare un canale sicuro, per approfondimenti in appendice.

2.4.2 IKE_AUTH

Il risultato della fase precedente è un canale sicuro su cui comunicare, in quanto è cifrato e utenticato. Si questo hanno luogo gli scambi per instaurare la IPsec SA. In questa fase i nodi si autenticano mutuamente:

TABLE 2.3: Tabella dei parametri e delle descrizioni

Parametro	**Descrizione**
AUTH	Payload che deve essere firmato affinchè ci sia autenticazione
CERT	Si allega il certificato digitale per la chiave pubblica
CERTQ	Si fa richiesta al peer di fornire il certificato

Tutto il contenuto appena descritto è protetto mediante le chiavi segrete di quella direzione. Ciò è indicato mediante la notazione $SK\{...\}$ La modalità di autenticazione può essere: PSK, EAP oppure mediante chiave pubblica.

2.4.3 CHILD_SA

2.5 Problemi

IKEv2 utilizza come porotocollo a livello trasporto UDP per per inoltrare i propri messaggi. La maggior parte dei messaggi che i peer si scambiano hanno dimensioni relativamente piccole e quindi che non eccedono l'**MTU** di un pacchetto IP, tuttavia abbiamo degli scambi che richiedono un trasferimento di dati abbastanza grandi.

Per esempio nel caso di autenticazione tramite pubkey nella fase di `IKE_AUTH` è necessario trasferire il proprio certificato che in base allo schema di firma utilizzato può arrivare anche a diversi Kbyte di dimensione. In questi casi si verifica la frammentazione a livello IP.

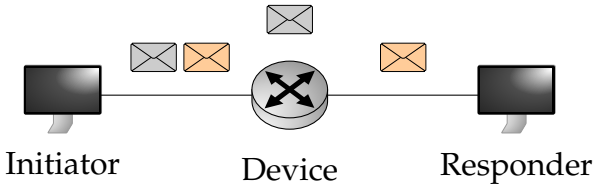

FIGURE 2.8: Drop Pacchetti

Diversi test hanno mostrato che nel caso in cui i peer si trovino in presenza di CGNAT potrebbero non istausarsi le SA. Questo è dovuto al fatto che i device degli ISP non consentono ai frammenti IP di passare attravers di loro, ovvero scartano i pacchetti e di conseguenza bloccano le comunicazioni IKE. Questo è riportato schematicamente in Fig. 2.8. Questo drop dei pacchetti avviene perchè esistono numerosi vettori di attacco che fanno affidamento sulla frammentazione IP, per questo motivo gli ISP operano un filtro su questa tipologia di pacchetti. Ance se in teoria uno dei requisiti del CGNAT definito dagli RFC è proprio consentire la frammentazione.

Per risolvere questa problematica e dunque consentire il passaggio dei messaggi attraverso i dispositivi di rete che non consentono il passaggio degli IP fragment attraverso di loro nell' RFC 7283 viene introdotta la IKEv2 *Message Fragmentation*. In cui la frammentazione dei messaggi è gestita direttamente da parte di chi implementa IKEv2

Per evitare che nel trasferimento di grandi dati ciò avvenga viene introdotto uno scambio aggiuntivo. Questo scambio è introdotto per quei casi in cui la dimensione dei dati da trasferire ecceda la dimensione massima che causerebbe la frammentazione IP. Questo scambio va fatto dopo la `IKE_INIT_SA` e prima della `IKE_AUTH` in questo modo è sia autenticato che cifrato tramite le chiavi negoziate dal primo scambio.

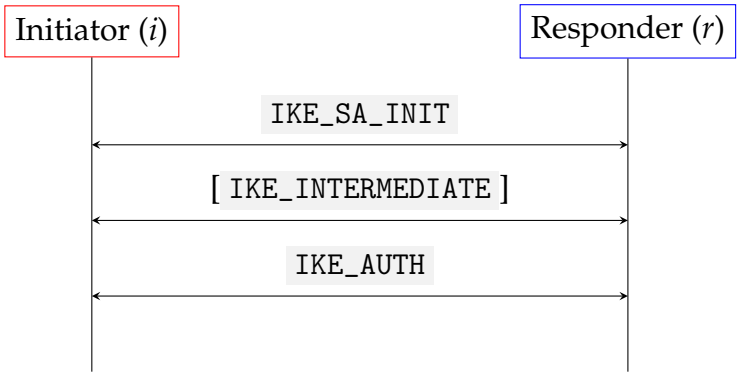

FIGURE 2.9: Scambio nuovo

Questo scambio è posizionato qui in quanto nella `IKE_SA_INIT` per motivi di sicurezza non è possibile applicare la frammentazione. Di solito i messaggi sono piccoli abbastanza da non causare la frammentazione IP, tuttavia questo potrebbe cambiare se si utilizzano scambi di chiave QC-resistant; in quanto hanno chiavi pubbliche larghe e che quindi causerebbero frammentazione IP.

Per questo viene aggiunto questo scambio che viene utilizzato per trasferire grandi quantità di dati.

L'utilizzo principale di questo scambio è quello di trasferire le chiavi pubbliche QC-resistant, tuttavia in generale può essere utilizzzato per trasferire qualsiasi tipologia di dato. Quindi il principale utilizzo è quello di fare un **enforcing** delle chiavi negoziate tramite DH al fine di renderle QC-resistant. Infatti se durante questo scambio si scambiano altre chiavi allora le coppie $\{SK_{a[i/r]}, SK_{e[i/r]}\}$ vengono aggiornate.

Permette di realizzare Multiple Key Exchange Gli scambi di chiave aggiuntivi vengono aggiunti alla proposal tramite `PQ_KEM_1`

Lo scambio `IKE_FOLLOWUP_KE` è introdotto specificatamente per trasferire dati sulla chiavi addizionali da realizzare in una CHILD SA. In questo caso le chiavi aggiuntive vengono utilizzate per aggiornare il KEYMAT

- flag `IKE_FRAGMENTATION_SUPPORT` : il peer dice di supportare la frammentazione IKEv2, affinchè venga utilizzata entrambi i peer devono supportarla.
- flag `INTERMEDIATE_EXCHANGE_SUPPORT` : il peer dice di supportare gli scambi intermedi

Una volta terminati gli scambi, per proteggere lo scambio `IKE_AUTH` e gli scambi successivi vengno utilizzata le ultime chiavi calcolate Dato che i dati trasferiti in questi scambi aggiuntivi vanno autenticati si aggiungono all' `AUTH` payload che poi andrà

Il supporto per lo scambio aggiuntivo viene comunicato aggiungendo all'interno dell `IKE_SA_INIT` il flag `IKE_INT_SUP` (che sta per Intermediate Exchange Support). Se anche il responder lo supporta lo includerà nel messaggio di risposta dello scambio.

Considerazioni, L'IKE fragmentation viene introdotta a causa del NAT tuttavia nel nostro caso di satelliti non ha senso utilizzarla in quanto non credo che si utilizzi il NAT soprattutto perchè introduce ritardi dovuti alla traduzione degli indirizzi

Chapter 3

Scenario

In questo capitolo facciamo un introduzione su quelloo he è lo scenario che stiamo considerando per il nostro lavoro, ovvero quello delle comunicazioni satellitare e di quali sono le attuali tecnologie utilizzate per realizzarle.

Riformula meglio

3.1 Comunicazioni Satelliari

Le comunicazioni satellitari sono fondamentali per le infrastrutture moderne, poichè abilitano una vasta gamma di servizi. Negli ultimi decenni, con l'aumento della domanda di connettività globale e l'espansione delle reti di comunicazione, i satelliti sono diventati strumenti essenziali per garantire una copertura estesa. L'emergere delle costellazioni di satelliti in orbita bassa (LEO - Low Earth Orbit) sta cambiando il paradigma delle comunicazioni satellitari, offrendo vantaggi significativi rispetto ai satelliti geostazionari (GEO), un confronto tra la differenza di distanze è mostrato in *Figura 3.1*. Questo cambio di paradigma insieme al quantum computer hanno portato diversi enti, tra cui l'Agenzia Spaziale Europea (ESA), ad affrontare nuove sfide.

Scrivere meglio questa parte finale

3.1.1 Limitazioni

L'ambiente spaziale, caratterizzato da radiazioni intense, temperature estreme e lunghi periodi senza manutenzione, pone sfide significative in termini di progettazione e operatività dell'hardware e del software. Tra i principali vincoli per l'hardware satellitare troviamo:

- *Resistenza alle radiazioni*: i componenti elettronici sono progettati per resistere all'esposizione costante alle radiazioni spaziali.

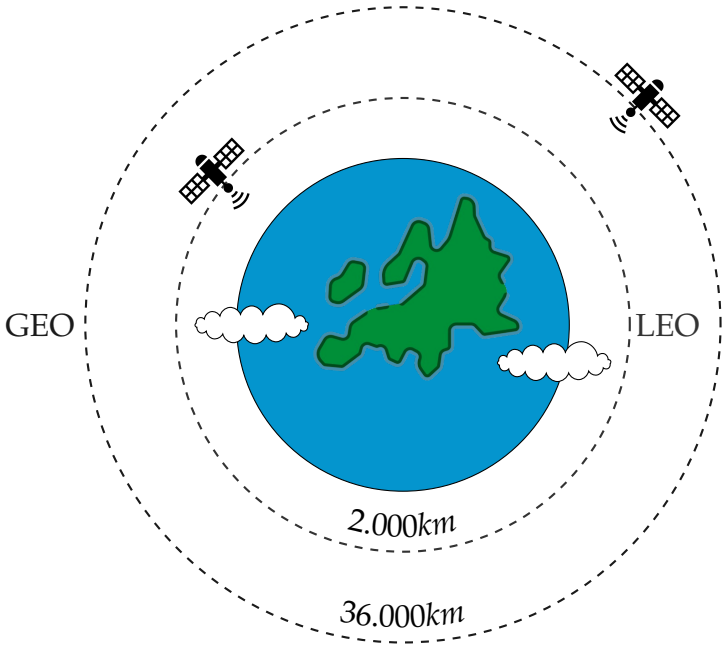

FIGURE 3.1: Orbite dei satelliti

- *Basso consumo energetico*: l'energia disponibile per le operazioni risulta limitata, quindi si usano processori a basso consumo e ad alta efficienza energetica, sacrificando potenza di calcolo.
- *Elaborazione in tempo reale*: per alcune tipologie di servizi è necessario che il processamento avvenga in tempo reale.
- *Compattezza*: a causa dello spazio limitato a bordo di un satellite, i componenti hardware devono essere progettati in modo estremamente compatto.

L'hardware limitato ha un impatto diretto sullo sviluppo del software per i satelliti. Rispetto al un contesto terrestre, dove le risorse computazionali sono abbondanti, il software per i satelliti deve essere ottimizzato per funzionare su processori con bassa potenza di calcolo, limitato parallelismo e memoria ridotta. Le principali sfide per gli sviluppatori sono:

- *Semplicità e ottimizzazione*: gli algoritmi devono essere semplici e ottimizzati per funzionare su hardware con risorse limitate.
- *Parallelismo limitato*: le operazioni devono essere eseguite in modo lineare o con limitato parallelismo, aumentando la complessità della progettazione.

- *Affidabilità assoluta*: il software deve essere robusto, sicuro e testato ampiamente in modo tale che possibile errori non abbiano conseguenza catastrofiche.

Particolare attenzione va posta sulle implementazinoni crittografiche, dato che in questo contesto è essenziale riuscire a bilanciare sicurezza e prestazioni Occorre ridurre al minimo l'impatto sulle risorse in modo tale che gli algoritmi di crittografia non possano compromettere l'efficienza operativa del satellite.

3.1.2 Stato Attuale

Per capire quale è la differenza in termini di hardware e software rispetto a quelli a cui siamo abiutati introduciamo quelli che sono gli attuali standard impiegati in questo settore.

- Tra i processori utilizzati abbiamo *LEON3*, un processore open-source basato sull'architettura SPARC, progettato dall'ESA.
- ESA Power Interface Standard (ECSS-E-ST-20C): definisce come deve avvenire la distribuituzione dell'alimentazione elettrica all'interno dei satelliti.
- Cubesat Standard: definisce dimensioni compatte modulari (10x10x10 cm per 1U) per ridurre i costi e semplificare il lancio e la costruzione dei satelliti.
- Triple Modular Redundancy (TMR): nei sistemi critici spaziali si utilizza la ridondanza tripla modulare, per garantire l'affidabilità dei risultati tramite sistemi di voting.

Il sistema operativo utilizzato in queste applicazioni è RTEMS (Real-Time Executive for Multiprocessor Systems) che le caratteristiche di essere open-source, real-time e basato su GNU/Linux.

se si trova riportare qualche riferimento

3.1.3 Sfide

Per rendere le comunicazioni satellitari sicure ad attacchi Qauntum occorre adottare algoritmi più complessi. Questi utilimi, tuttavia, oltre a richiedere più memoria per la gestione delle chiavi, comportano un incremento significativo del carico di calcolo. Le sfide da affrontare sono:

Scrivere bene quelle che sono le problematiche e come intendiamo affrontarle

- Aumento complessità computazionale: l'incremento delle dimensioni delle chiavi e della complessità degli algoritmi portano a un aumento del carico. computazionale. Questo impone vincoli importanti su sistemi già limitati in termini di potenza computazionale, come i satelliti.
- Incremento della larghezza di banda necessaria: oltre all'aumento di carico si ha anche magiore utilizzo della larghezza di banda, che in comunicazioni satellitari risuta già limitata.
- Compatibilità retroattiva: compatibilità con i sistemi esistenti. Tramite l'utiizzo di soluzioni ibride, dove algoritmi classici convivono con quelli post-quantum.

3.2 Benchmarking

Per determinare se è possibile applicare algoritmi di PQC nel contesto satellitare occorre capire quale è il loro impatto sui protocolli di comunicazione. Prima lo facciamo nel caso desktop rispetto a quella di quelli classici per vederne le differenze, se queste sono molto evidenti già nel caso simulato allora non ha neanche senso andarle a considerare in un'ambiente ancora più limitato.

3.2.1 Ambiente

L'ambiente di test, descritto in *Tabella 3.1*, è stato progettato in modo da isolare i singoli servizi all'interno di container, facilitando l'analisi del carico e delle risorse consumate da ciascuno di questi.
La libreria `liboqs` fa parte del progetto open-source *OpenQuantumSafe* (OQS). Il quale a rendere disponibile l'infrastruttura crittografica necessaria per proteggere i sistemi informatici dall'avvento dei computer quantistici. L'obiettivo del progetto è quello di fornire:

- API standardizzate per supportare lo scambio chiavi e la firma digitale.
- Modularità, facilitando l'aggiunta di nuovi algoritmi.
- Prestazioni ottimizzate per diverse architetture hardware.

Si tratta di una raccolta di implementazioni di algoritmi crittografici post-quantum, KEM e SIG, e strumenti per integrarli in protocolli di sicurezza esistenti per sperimentare e testare il loro impatto. La versione di Strongswan

Componente	Descrizione
Hardware	
CPU	Ryzen 7-5825U (8 core, 3.8 GHz)
RAM	24 GB DDR4
Storage	1024 GB SSD NVMe
OS	
Distribuzione	Arch Linux
Kernel	Linux 6.10.10-arch1-1
Software	
Linguaggio	Bash Script
Strongswan	6.0.0beta Post-Quantum IKEv2 Daemon
liboqs	Version 0.9.2
Docker	Version 24.0.6

TABLE 3.1: Descrizione dell'ambiente di test virtualizzato

che andiamo considerare ha integrato al suo interno la libreria e quindi ci consente di vedere gli effetti di quesi algoritmi sul protocollo IKEv2.

3.2.2 Metodologia

Per ragioni progettuali e di portabilità del codice, il servizio Strongswan è stato containerizzato tramite l'utilizzo di Docker. Ciò consente di avere due istanze dello stesso servizio in esecuzione contemporaneamente che comunicano attaverso un'interfaccia virtuale. La gestione di queste istanze avviene tramite `docker-compose`, una rappresentazione logica è la seguente.

```
+-------+                          +--------+
| carol | === Virtual Interf. === |  moon  |
+-------+                          +--------+
```

In secondo luogo siamo andati a definire quali sono le configurazioni da confrontare, riportate in *Tabella 3.2*, ognuna delle quali è caratterizzata da:

- **Chiper suite**: un'insieme di algoritmi crittografici che determinano la sicurezza di una connessione in un protocollo di rete. Le cipher suite sono generalmente denominate seguendo una convenzione di naming standardizzata che riflette i componenti inclusi nella suite.

```
<ENCR>-<INTEG>-<KEM>
```

- **Authentication Method**: dato che stiamo considerando algoritmi post-quantum di firma utilizziamo quella mediante certificati. Quindi per ogni schema di firma andiamo a definire un root certificate ed uno per ogni entità.

Nella nostra analisi, abbiamo scelto tre configurazioni distinte per le cipher suite, ognuna progettata per affrontare specifici aspetti delle tecnologie crittografiche attuali e future.
La prima configurazione utilizza esclusivamente di *primitive classiche*. Questa scelta rappresenta il benchmark attuale delle tecnologie di crittografia e fornisce una base solida per confrontare le altre configurazioni. La seconda configurazione è composta da *primitive post-quantum*, e consente di analizzare le prestazioni e l'efficacia delle soluzioni crittografiche post-quantum nel contesto del protocollo. Infine, abbiamo implementato una *onfigurazione Ibrida*, la quale combina elementi delle primitive classiche e post-quantum. Questa scelta è progettata per garantire la transizione graduale verso l'utilizzo esclusivo di tecnologie quantistiche.

gli altri algoritmi tipo hqc, bike,...

Name	Chiper Suites	Firma Digitale
A1	aes128ctr-sha256-ecp256	ECDSA
B1	aes128ctr-sha256-kyber1	dilithium2
C1	aes128ctr-sha256-ecp256-ke1_kyber1	falcon512
A3	aes192ctr-sha384-ecp384	ECDSA
B3	aes192ctr-sha384-kyber3	dilithium3
C3	aes192ctr-sha384-ecp384-ke1_kyber3	falcon1024
A5	aes256ctr-sha512-ecp521	ECDSA
B5	aes256ctr-sha512-kyber5	dilithium5
C5	aes256ctr-sha512-ecp521-ke1_kyber5	falcon1024

TABLE 3.2: Cipher Suites suddivise per Livello di Sicurezza

A questo punto devo andare a parlare della parte dello script

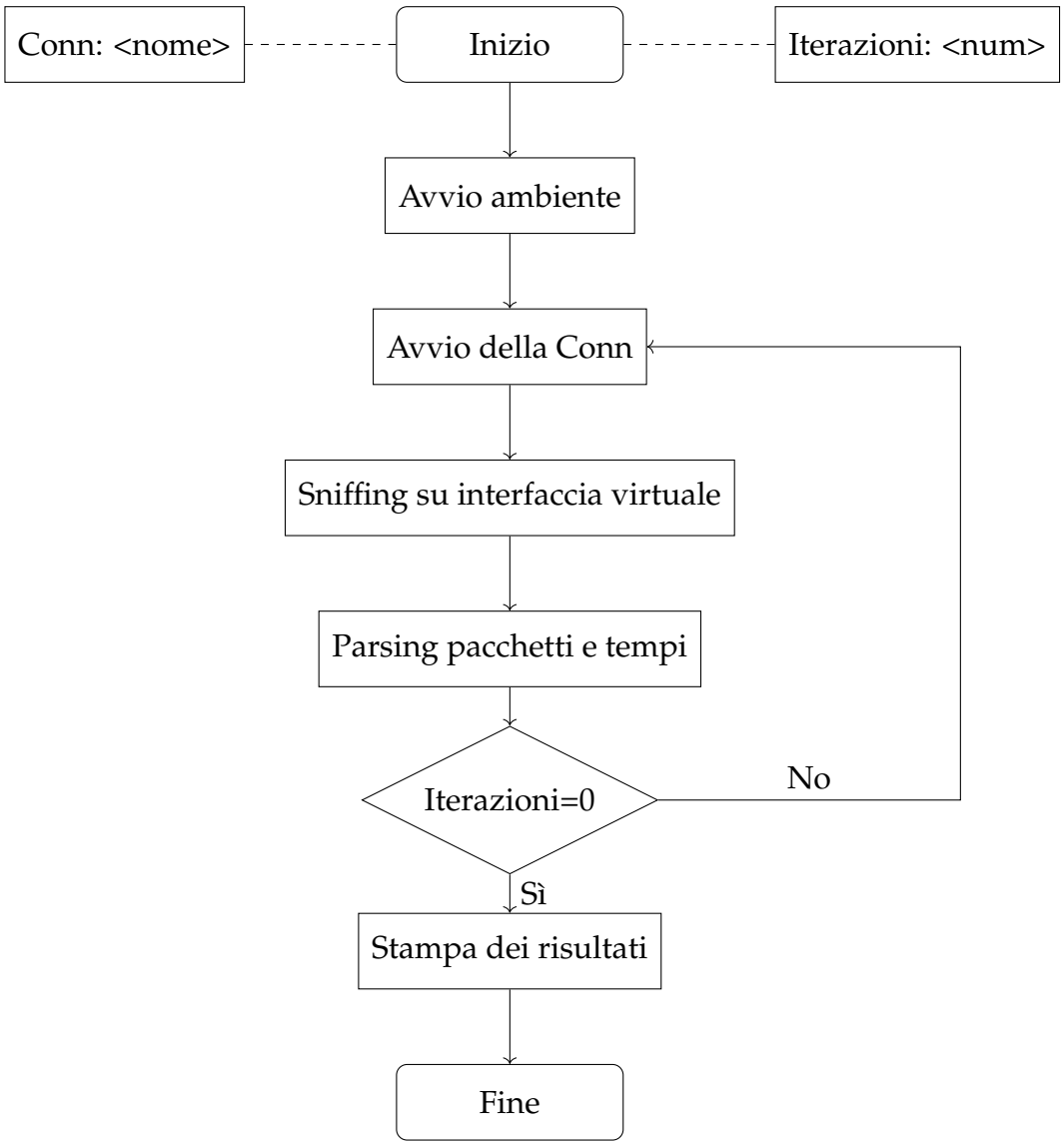

FIGURE 3.2: Diagramma di flusso dello script

3.2.3 Risutlati

Le metriche di maggiore interesse per il nostro studio sono quelle che riguardano, il tempo complessivo per stabilire la SA. Non ci interessa il tempo di elaborazione del singolo algoritmo, poichè queste informazioni sono già fornite in maniera esaustiva dalla libreria openquantumsafe e inoltre questi tempi sono trascurabili rispetto all'RTT.

Confrontare nella fase auth l'utilizzo di schemi classici rispetto a quelli quantum

E il numero di byte che occorre scambiarsi per generare la SA.

grafico riassuntivo che riporta il confronto tra le varie configurazioni, uno per i tempi e uno per i

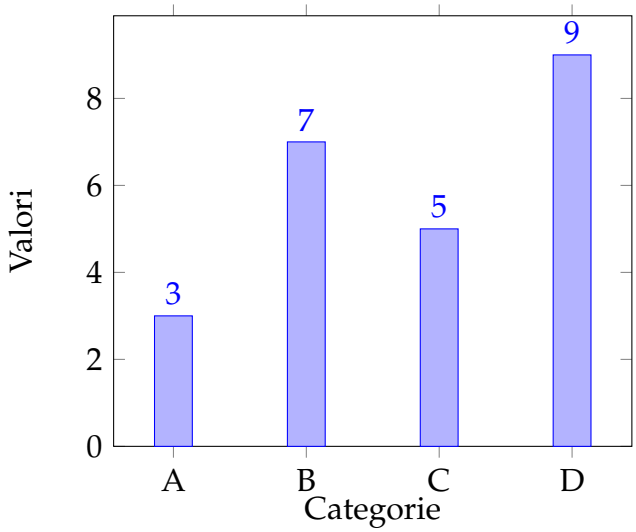

FIGURE 3.3: Esempio di istogramma con dati fittizi.

3.3 Tuttavia

Notiamo che non ci sono enormi differenze in termini di tempi ma è presente un notevole aumento in termini della dimensione in particolare per quanto riguarda l'autenticazione. E in contesto di questo tipo non è auspicabile

Cercando di trovare possibili soluzioni, ci siamo imbattuti su quello che è minimal IKE ovvero una versione di IKE applicabile in scenari soggetti a constraint di risorse simili a quelli presenti nello spazio.

Riguardo a questo non esistono implementazioni, per questo motivo siamo passati a provare a dare un'implementazione di quest'ultimo In modo tale che rappresenti un punto di inizio per questo scenario

Di questo trattiamo nel prossimo capitolo

Chapter 4

Hummingbird

4.1 Progettazione

4.1.1 Requisiti

Nella parte di progettazione portare quelli che sono i requisiti che deve rispettare l'implementazione sia funzionali che non uno tra tutto met

4.1.2 Architettura

Archietettura sia delle directory che a livello dei moduli Il C richiede una chiara strutturazione per gestire la complessità del codice in modo efficace

Moduli

Per mantenere la separazione dei compiti

Strutture Dati

4.2 Implementazione

4.2.1 Strumenti

Librerie utilizzate e cose varie, tra queste quelle utilizzate sono:

- libcjson: per fare il parsing del file di configurazione scritto in formato json
- libcrypto: fornisce le implementazione dei principali schemi crittografici, di hashing e gestione delle chiavi

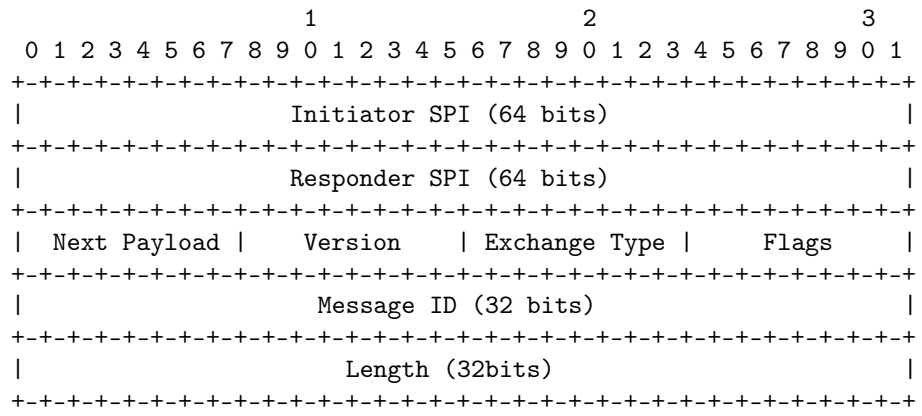

FIGURE 4.1: Formato IKE Header

4.2.2 Codice

Il formato dell'header IKE, presente in *Figura 4.1*, nel codice `C` possiamo tradurlo come una struct

```
1    #include <stdio.h>
2    int main() {
3        printf("Hello, World!\n");
4        return 0;
5    }
```

Nel parsing della risposta, è stata creata una lookup table. Giustificazione dovuta al fatto che la struttura di un pacchetto ike può essere vista come una lista semplicemente puntata quindi si presta bene ad approcci iterativi. Per questo motivo invece di andare a creare tanti buffer quanti sono i payload si è adottato un approccio diverso

a partire dal buffer del pacchetto si è creata una funzione ricorsiva il cui criterio di stop è quello del next payload nullo (fine della lista), che ad ogni iterazione si va a "mangiare un pezzo del pacchetto", nel senso che invece che riallocarlo si gioca con i puntatori una sorta di pacman ma in questo caso il buffer che non consideriamo è ancora esistente, tuttavia questa modalità evita ogni volta di andare a creare e distruggere dei buffer che sarebbe molto oneroso .

Inoltre è possibile adottare una strategia di buffering pool in cui

4.2.3 Sfide

4.3 Analisi

Appendix A

IKEv2 Notation

Riportare i vari approfondimenti riguardanti IKE

per esempio come vengono generate le varie chiavi e il significato delle informazioni presenti tra i messaggi

A.0.1 Authentication

L'autenticazione dei peer avviene effettuando il sign (o calcolando il MAC) di un payload che dipende dagli scambi precedenti. In particolare questo payload è cmposto da un ottetto che viene autenticato in base alla modalità di aunteticazione scelta:

- Nel caso di *PubKey* questo viene firmato con la chiave privata del peer e ne viene allegato il certificato della chiave pubblica
- Nel caso di *PSK* l'AUTH payload viene generato a partire dalla chiave condivisa a cui viene aggiunge della unpredicability tramite del padding e una prf

A.1 Key Derivation

A.1.1 IKE SA

Le chiavi in una IKE SA vengono derivate a partire dagli attributi dei dirrenti scambi. In particolare al termine del primo scambio viene calcolato il:

$$SKEYSEED = PRF(N_i|N_r, g^{ir})$$

A partire da questo sidder vengono generati i prametri di sicurezza da utililizzare per la IKE SA, questi sono derivati nel seguente modo:

$$\{SK_d|SK_{ai}|SK_{ar}|SK_{ei}|SK_{er}|SK_{pi}|SK_{pr}\} = PRF + (SKEYSEED, N_i|N_r, SPI_i, SPI_r)$$

Chiave	Descrizione
SK_d	Utilizzata per generare il keymaterial per le CHILD_SA
SK_a	Chiavi per autenticare gli scambi successivi, una per direzione
SK_e	Chiavi per cifrare gli scambi successivi, una per direzione
SK_p	Chiavi utilizzata per generare l'AUTH Payload, una per direzione

TABLE A.1: Chiavi e loro utilizzo

A.1.2 IPsec SA

Nel caso di una SA questa può essere generata automaticamente dopo l'auth oppure attraverso l'apposito scambio di questo tipo il keymaterial a partire dal quale vengono derivati i parametri di sicurezza è ottenuto nel seguente modo:

$$KEYMAT = prf + (SK_d, N_i|N_r)$$

Nel caso in cui invece si utilizza lo scambio apposito il key material è ottenuto nel seguente modo

A.2 Security Association Payload

Il Security Association Payload denotato con SA è utlilizzatoper negoziare gli attributi di una Secuiry Association. Dunque può contenere molteplici proposte, le quali devono essere ordinate per preferenza, ogni proposal contiene i seguenti algoritmi crittografici:

- Encryption Algorithm (ENCR)
- Preudorandom Function (PRF)
- Integrity Algorithm (INTEG)
- Diffie-Hellman Group (KE)
- PQ KEM

```
services:
    moon:
        build: ./
        container\_name: moon
        cap\_add:
            - NET\_ADMIN
            - SYS\_ADMIN
            - SYS\_MODULE
        stdin\_open: true
        tty: true
        volumes:
            - ./moon:/etc/swanctl
            - ./strongswan.conf:/etc/strongswan.conf
        networks:
            internet:
                ipv4\_address: 192.168.0.2
            intranet:
                ipv4\_address: 10.1.0.2

    carol:
        build: ./
        container\_name: carol
        depends\_on:
            - moon
        cap\_add:
            - NET\_ADMIN
            - SYS\_ADMIN
            - SYS\_MODULE
        stdin\_open: true
        tty: true
        volumes:
            - ./carol:/etc/swanctl
            - ./strongswan.conf:/etc/strongswan.conf
        networks:
            internet:
                ipv4\_address: 192.168.0.3

networks: internet: ipam: driver: default config: -
   subnet: 192.168.0.0/24
intranet: ipam: driver: default config: - subnet:
   10.1.0.0/16
```