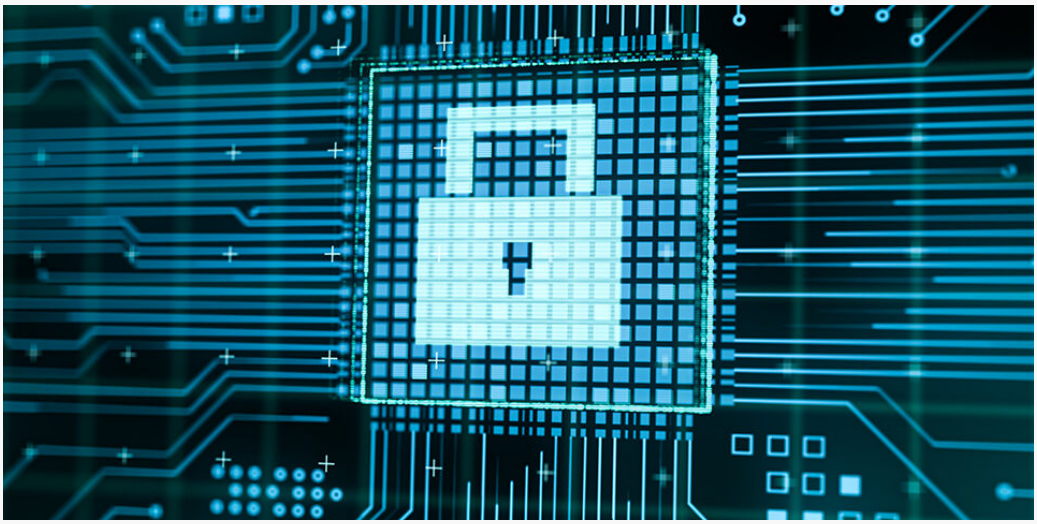
Sicurezza, Hardware e Confidential Computing – Parte 1

A cura di: Andrea Pasquinucci - Pubblicato il 6 Settembre 2021



Negli ultimi anni si è posta molta attenzione agli aspetti di sicurezza informatica basati su componenti Hardware: progetti sono in corso, consorzi sono stati creati (si veda ad esempio [Rif. 1]), annunci e nuovi prodotti sono stati presentati. In questo articolo proviamo a fare una rapida panoramica di alcuni aspetti della sicurezza informatica strettamente basati su componenti Hardware che potrebbero portare ad interessanti sviluppi nel prossimo futuro. Per fare questo però, è utile partire ricordando alcuni dei principali concetti del ruolo dell'Hardware nella sicurezza informatica.

Sicurezza e fiducia (o Trust)

Dobbiamo sempre ricordarci che la sicurezza informatica si basa comunque sempre su una **catena di fiducia** (o all'Inglese, *Chain of Trust*) che ci coinvolge tutti in vari modi.

Quando utilizziamo un software, dai Sistemi Operativi ad applicazioni grandi e piccole, è ovvio, o dovrebbe esserlo, che esplicitamente o implicitamente ognuno di noi si fida del produttore del software, anche se non sempre allo stesso modo. Per questo anche se ci fidiamo, utilizziamo anti-virus, verifichiamo (o dovremmo verificare) le notizie pubbliche su eventuali incidenti di sicurezza delle applicazioni, facciamo eseguire (o dovremmo fa eseguire) Vulnerability Assessment, Penetration Test, Source Code Analysis, Black Box Test ecc. per verificare l'assenza di lacune di sicurezza dai prodotti. Nessuno di noi dovrebbe procedere ad installare un prodotto software per il quale non si ha nessuna fiducia sulla sicurezza, sia basata su evidenze tecniche sia sulla storia ed il buon nome del produttore.

In genere ci fidiamo di più dell'Hardware, ed a ragione, tipicamente perché vi sono meno incidenti di sicurezza informatica dovuti a problemi Hardware, ma anche perché spesso siamo meno consapevoli del suo ruolo cruciale nella sicurezza informatica.

Ma dove comincia la catena della fiducia? Il primo punto di vista è che ci dobbiamo fidare di tutti i produttori, sia Software che Hardware. Il secondo punto di vista è di guardare a come la sicurezza è implementata nei sistemi informatici, ed in questo caso dovrebbe essere ovvio che si parte dall'Hardware e dal suo Firmware.

Protezione Hardware di base e catena della fiducia

Che l'Hardware sia il punto di partenza (o *Root of Trust*) della catena della fiducia della sicurezza operativa è fatto ben noto e risale agli albori dell'informatica. L'approccio attuale alla sicurezza Hardware ebbe le prime implementazioni negli anni '60 e storicamente si fa riferimento a Multics (1964), il papà di Unix. All'interno delle CPU ogni programma è eseguito a un livello di protezione[1] (*Protection Ring*) controllato dall'Hardware e che ha un set predefinito di privilegi, ovvero nel quale i programmi possono eseguire solo un certo insieme di istruzioni macchina. Tipicamente i programmi sono eseguiti nel livello meno privilegiato nel quale non hanno accesso diretto ad alcuna risorsa, dai file alle periferiche ecc. Nel livello più privilegiato è eseguito invece il Kernel del Sistema Operativo. Ad esempio, quando un programma deve leggere o scrivere dei dati, prepara i dati da leggere o scrivere in apposite locazioni di memoria e poi attiva un particolare bit Hardware (detto *Hardware Gate*) tramite una chiamata detta *System Call*. La CPU a questo punto interrompe l'esecuzione del programma ed attiva il Kernel del Sistema Operativo che legge la richiesta del programma, verifica che sia lecita, ovvero ad esempio che non voglia leggere o scrivere dati di altri utenti o su dispositivi ove l'utente o il programma non ha il permesso di accesso, e procede a leggere o scrivere i dati per conto del programma.

Qualsiasi funzionalità di sicurezza informatica nell'esecuzione degli eseguibili è basata su questa architettura Hardware che fa si che solo il Kernel del Sistema Operativo abbia diretto accesso alle periferiche e all'Hardware in generale. Il Kernel del Sistema Operativo ha quindi il compito di implementare le politiche di sicurezza operative.

Il Kernel di un Sistema Operativo è caricato nella CPU all'avvio del sistema e tipicamente non dovrebbe essere modificato durante l'esecuzione [2]. Ma come possiamo essere sicuri che il Sistema Operativo non sia stato manomesso al momento dell'avvio del sistema o durante la sua esecuzione? Entrambi gli scenari sono possibili: il primo ad esempio tramite un aggiornamento fraudolento del Sistema Operativo anche tramite un accesso fisico alla macchina, il secondo ad esempio sfruttando una vulnerabilità del Sistema Operativo stesso durante la sua esecuzione.

Per garantire **l'integrità** del Sistema Operativo è necessario risalire al momento dell'avviamento del sistema quando viene eseguito il Firmware presente in Hardware e poi i programmi BIOS/UEFI necessari all'avvio del Sistema Operativo. La catena della fiducia deve quindi risalire all'Hardware al momento dell'avvio del sistema: ci fidiamo del produttore dell'Hardware che a sua volta si fida del Firmware, del BIOS/UEFI, del Sistema Operativo e così via. Ad ogni passo è necessario verificare che il componente successivo sia **autentico**, ovvero di un produttore di cui ci si fida, e **integro**, ovvero non modificato. E le verifiche devono essere fatte periodicamente per individuare eventuali modifiche non autorizzate a *run-time*. Il processo è implementato utilizzando tecniche crittografiche calcolando il *Hash* degli eseguibili (integrità) e apponendo firme digitali (autenticità) tramite moduli crittografici Hardware dedicati o inclusi nelle CPU. Questo processo è tipicamente chiamato *Secure Boot* ed è ormai implementato nella maggior parte dei dispositivi informatici e supportato da quasi tutti i Sistemi Operativi.

Integrità, Autenticità e Confidenzialità

Prima di procedere è necessario distinguere due problematiche che nella pratica sono interconnesse ma hanno scopi e finalità ben diversi, come riassunto in Fig. 1.

Fig. 1 Approcci e alcuni esempi di tecnologie di sicurezza delle elaborazioni

La prima, quella appena descritta, ha come scopo di garantire che gli strumenti Hardware e Software che utilizziamo per le elaborazioni informatiche siano quelli in cui riponiamo la nostra fiducia e che vogliamo utilizzare. Inoltre vogliamo essere immediatamente informati nel caso questi siano modificati in maniera non autorizzata ad esempio da qualche *malware*. In generale le soluzioni a questi requisiti vanno sotto il nome di *Trusted Execution Environment* (TEE). Ovviamente in questo caso le proprietà che ci interessano sono l'Integrità e l'Autenticità di Hardware e Software.

La seconda, che di recente sta assumendo sempre maggior interesse, ha come scopo di garantire la Confidenzialità delle elaborazioni anche nei confronti degli amministratori del sistema e del Sistema Operativo stesso e non solo degli altri utenti non amministrativi. Questa esigenza nasce anche dall'utilizzo da parte di più utenti, completamente estranei, dello stesso sistema Hardware/Software in particolare negli ambienti Cloud.

Questo aspetto è usualmente chiamato *Confidential Computing* (Rif. [1-5]). Prima però di discutere le relazioni tra *Confidential Computing*, *Secure Boot* e *Trusted Execution Environment*, è necessario approfondire le principali tematiche del *Confidential Computing*.

Confidential Computing

Consideriamo subito il caso più complicato ma sicuramente di maggior interesse: l'utilizzo di servizi Cloud erogati in modalità SaaS, PaaS o IaaS. Lo scenario ben noto è quello di una piattaforma Hardware e Software gestita dal fornitore ed utilizzata in maniera condivisa e self-service dai clienti. I clienti condividono le risorse Hardware e Software, inclusi Sistemi Operativi, Middleware e applicazioni. D'altra parte ogni cliente richiede la confidenzialità dei propri dati, sicuramente rispetto agli altri clienti e in molti casi anche rispetto al fornitore Cloud stesso.

La confidenzialità può essere ottenuta principalmente utilizzando due approcci:

1. sostituendo i dati sensibili con dati non significativi tramite processi quali la Tokenizzazione, il Data Masking o la pseudo-anonimizzazione;
2. cifrando tutti i dati o soli i dati sensibili.

In entrambi i casi, se solo parte dei dati sono protetti, si pone il problema di garantire che non sia possibile estrarre informazioni sensibili, via inferenza, dai dati non protetti. Il problema dell'inferenza e dei canali paralleli (*side channel*) è un problema reale e spesso di difficile soluzione, se non quella di ridurre al massimo la quantità di dati non protetti e rendere assolutamente inintelligibili tutti gli altri.

I processi di mascheramento o sostituzione dei dati sensibili sono particolarmente esposti ai problemi d'inferenza e analisi. Inoltre sono di norma completamente in carico all'utente, che deve preparare correttamente i dati prima d'inviarli al fornitore Cloud. Infine non è possibile adottare queste soluzioni ad esempio quando le elaborazioni devono essere fatte proprio e direttamente sui dati sensibili.

Sembrerebbe quindi che la cifratura di tutti i dati possa essere l'approccio vincente e più semplice per garantire la loro confidenzialità, ma vanno considerati i tre possibili stati dei dati (si veda anche la Tabella 1):

1. la cifratura dei dati in transito, ovvero quando trasmessi sulle reti di telecomunicazione
2. la cifratura dei dati a riposo, ovvero quando archiviati su un Filesystem, Storage, Database ecc.
3. La cifratura dei dati in uso, ovvero durante l'elaborazione.

In transito	Cifratura comunicazioni		
A riposo	Cifratura disco / FileSystem per furto HW o accesso fisico	Cifratura dati (es. Tabella / Colonna in DB) con chiave dell'utente, *Confidential Computing*	Tokenizzazione, Data Masking, Pseudonimizzazione
In uso	*Confidential Computing*	Crittografia omomorfa	Tokenizzazione, Data Masking, Pseudonimizzazione

Tabella 1: alcune tecniche per la confidenzialità nei tre possibili stati dei dati

La cifratura di tutti i dati in transito è ormai la norma e non dovrebbe costituire un problema per nessuno.

La cifratura dei dati a riposo è ormai abbastanza comune ma è necessario fare alcune precisazioni. La cifratura di interi dischi o FileSystem garantisce la confidenzialità dei dati sicuramente rispetto a scenari quali l'accesso fisico od il furto del supporto Hardware, ma non è utile per garantire la confidenzialità rispetto gli accessi degli amministratori di sistema o quando l'applicazione è condivisa da più utenti e utilizza lo stesso Storage per tutti gli utenti. Al giorno d'oggi questa però è la situazione più comune.

Per garantire la confidenzialità dei dati a riposo anche rispetto agli amministratori dei sistemi Cloud ed agli altri utenti della stessa applicazione, è necessario che i dati siano cifrati con chiavi crittografiche gestite e note solo all'utente. Un caso ormai comune è fornito dai servizi di Storage Web in Cloud, ovvero i servizi di tipo "*Cloud Drive*" accessibili tipicamente tramite Browser Web e App dedicata. In questi servizi, i dati sono cifrati dal Browser dell'utente sul proprio dispositivo informatico prima dell'invio al servizio di Storage in maniera trasparente all'utente. Lo stesso avviene per servizi Cloud di backup e di puro Storage, ove i dati sono cifrati / decifrati da una applicazione sui sistemi interni dell'utente. Questo da un lato garantisce la confidenzialità dei dati ma dall'altro rende impossibile qualunque elaborazione dei dati sui sistemi in Cloud visto che la chiave di cifratura è loro ignota.

Ma come poter garantire al contempo la confidenzialità dei dati a riposo, ovvero sullo Storage dei servizi Cloud, e la possibilità per le applicazioni Cloud di elaborare i dati?

Questo è il principale problema che il *Confidential Computing* si propone di risolvere. La soluzione è basata sull'utilizzo di moduli Hardware crittografici da affiancare od integrare in tutti i processori (si veda ad esempio [Rif. 6]), che permettano solo all'utente padrone dei dati di cifrarli e decifrarli sui server Cloud. Questi moduli sono chiamati *Hardware-Based Trusted Execution Environment* (o HW-TEE). Proveremo più avanti a dare una descrizione ad alto livello di come questo possa funzionare, il punto principale è che le chiavi segrete e le elaborazioni sensibili, quali la de-cifratura, sono svolte

esclusivamente in Hardware dedicato e non accessibile, in modo da impedire a chiunque non autorizzato, inclusi gli amministratori dei sistemi, di accedervi. Da questo punto di vista gli HW-TEE sono da considerare come un'evoluzione degli *Hardware Security Module* (HSM) e dei *Trusted Platform Module* (TPM): i primi sono sistemi HW per gestire in sicurezza le chiavi e le operazioni crittografiche, i secondi sono micro-chip presenti ormai su quasi tutti i computer anche personali, che gestiscono in sicurezza chiavi crittografiche (si veda anche la Fig. 2).

	HW TEE	Homomorphic Encryption	Secure element e.g., TPM
Data integrity	Y	Y (subject to code integrity)	Keys only
Data confidentiality	Y	Y	Keys only
Code integrity	Y	No	Y
Code confidentiality	Y (may require work)	No	Y
Authenticated Launch	Varies	No	No
Programmability	Y	Partial ("circuits")	No
Attestability	Y	No	Y
Recoverability	Y	No	Y

Fig. 2 Caratteristiche dei tre principali approcci crittografici per la confidenzialità [Sorgente Rif. 4]

Un esempio semplice in ambito Cloud IaaS può essere utile a chiarire questo scenario. Si supponga di creare in un ambiente Cloud una macchina virtuale con un suo storage virtuale dedicato. Si cifra però lo storage con una chiave di cifratura nota solo all'utente che utilizza la macchina virtuale. La maniera tradizionale e poco funzionale di fare ciò, è di inserire manualmente la chiave di cifratura dello storage al momento in cui il relativo FileSystem viene montato sulla macchina virtuale. Questo ovviamente richiede una gestione manuale delle chiavi, e la presenza di una persona che deve digitarle al momento dell'avvio (e riavvio) della macchina virtuale. L'approccio corrente è invece totalmente automatizzato: alla macchina virtuale sono associate delle chiavi crittografiche che permettono di identificarla esattamente e, tramite queste chiavi, solo la macchina virtuale può accedere alla chiave di cifratura del FileSystem. Le chiavi crittografiche possono essere gestite in sistemi Hardware (TPM, HSM, HW-TEE) ma più comunemente sono gestite tramite applicazioni dedicate di *Key Management*, sistemi sui cui torneremo più avanti in questo articolo quando descriveremo brevemente un altro approccio detto comunemente *Bring Your Own Key (BYOK)*. Il risultato finale è che solo gli utenti della macchina virtuale possono accedere ai dati presenti sullo Storage virtuale ad essa collegato, gli utenti delle altre macchine virtuali e gli amministratori del servizio Cloud non hanno alcun accesso in chiaro ai dati sullo storage.

Riassumendo, rispetto alle due situazioni descritte inizialmente, è possibile creare una configurazione intermedia in cui i dati non sono cifrati dallo Storage stesso né dall'utente sui propri sistemi remoti, ma a livello utente sui sistemi o applicazioni in Cloud con chiavi legate all'utente o all'istanza dell'applicazione (o macchina virtuale). In questo modo la cifratura dei dati garantisce la confidenzialità degli stessi a riposo a livello utente, ma permette di decifrarli e poterli elaborare anche sui sistemi remoti.

Questo approccio può essere ripetuto (anche se non ancora in tutti i casi) per la protezione dei dati a riposo in database o direttamente in istanze di applicazioni.

Confidenzialità dei dati "In Uso" – Cifratura RAM

Purtroppo la cifratura dei dati a riposo è il problema semplice e la cui soluzione sta diventando sempre più standard anche nei servizi Cloud. Il problema difficile e non ancora risolto è quello di proteggere i dati "in uso".

Prima di tutto dobbiamo chiarire cosa si intende per dati "in uso": secondo l'accezione generale sono i dati quando non in trasferimento su reti dati e non archiviati in memorie permanenti. I due principali punti di "uso" dei dati sono ovviamente la CPU, che li elabora, e la RAM, che li memorizza temporaneamente. I dati "in uso" sono presenti poi in tutti i registri, cache, bus eccetera che compongono l'architettura hardware degli elaboratori, ed in particolare della scheda madre.

Nella CPU e nei circuiti Hardware che ne formano l'architettura, i dati devono essere in chiaro, ovvero non cifrati, altrimenti nessun tipo di elaborazione è possibile a meno di utilizzare la Crittografia Omomorfica che sarà brevemente considerata più avanti. Inoltre, a meno di attacchi di inferenza ("*side channel*") quali ad esempio Spectre e Meltdown o vulnerabilità Hardware, la CPU ed i circuiti Hardware sono i punti ove i dati sono a minor pericolo per la confidenzialità in quanto protetti dall'Hardware stesso. Analogo discorso non vale per la RAM in quanto un utente o programma con privilegi di amministratore di sistema (o amministratore di Host per i sistemi virtualizzati), può leggere tutti i dati in essa contenuti.

AMD propose nel 2016 [Rif. 7] la realizzazione di moduli RAM nei quali i dati sono cifrati. Questa tecnologia è ormai disponibile da parte di AMD con il nome di "AMD Memory Guard" (inizialmente si chiamava "Secure Memory Encryption") e Intel ha annunciato una simile soluzione con il nome "Total Memory Encryption" (TME) [Rif. 8]. L'implementazione della cifratura dei dati in RAM è, da un punto di vista logico, abbastanza semplice, come riassunto in Figura 3.

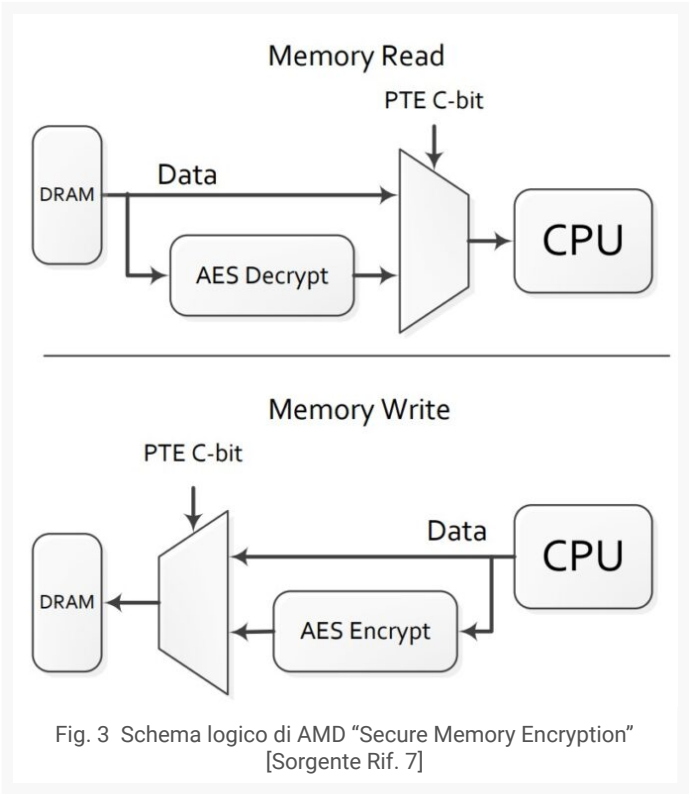


Fig. 3 Schema logico di AMD "Secure Memory Encryption" [Sorgente Rif. 7]

L'idea di base è di inserire dei circuiti dedicati tra la CPU e la RAM che cifrano i dati con AES (128bit) quando questi sono scritti in memoria dalla CPU, e li decifrano quando sono trasferiti dalla memoria alla CPU. In questo modo i dati sono sempre cifrati nella RAM e sempre in chiaro nella CPU. Nella tabella "*Page Table Entry*" (PTE) che indirizza i dati nella memoria, un bit ("*C-bit*") è dedicato a indicare se nella locazione di memoria i dati sono cifrati o in chiaro. Quindi a seconda del valore del "C-bit" nel trasferimento tra CPU e RAM i dati vengono (de-) cifrati o meno. Le chiavi di cifratura sono generate da un generatore di numeri pseudo-casuali Hardware e mantenute in registri Hardware dedicati, temporanei e non accessibili se non al Chip di cifratura. Le chiavi di cifratura sono tipicamente create ad ogni avvio del sistema. AMD-SME permette di cifrare con una singola chiave segreta tutta la RAM o solo i dati gestiti da un particolare processo, ad esempio l'Hypervisor e quindi tutte le macchine virtuali. In questo secondo caso, l'amministratore di sistema dell'Host di virtualizzazione non ha accesso ai dati delle macchine virtuali in RAM. Inoltre in modalità "*Secure Encrypted Virtualization*" (AMD-SEV) viene creata una chiave di cifratura dedicata per ogni macchina virtuale. Questo permette di proteggere i dati in RAM di una macchina virtuale non solo rispetto all'Host ma anche rispetto a tutte le altre macchine virtuali.

Nella seconda parte di questo articolo proseguiremo la presentazione di soluzioni per la confidenzialità dei dati in "uso", per arrivare ai *Trusted Execution Environment*, ai Key Management System e

concludere brevemente con un accenno alla crittografia omomorfica.

Riferimenti Bibliografici

Rif. 1: Confidential Computing Consortium, https://confidentialcomputing.io/

Rif. 2: NIST "Hardware-Enabled Security for Server Platforms: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases", Draft 28 aprile 2020, https://doi.org/10.6028/NIST.CSWP.04282020-draft

Rif. 3: NIST "Hardware-Enabled Security: Container Platform Security Prototype", Draft dicembre 2020, https://doi.org/10.6028/NIST.IR.8320A-draft

Rif. 4: Confidential Computing Consortium "Confidential Computing Deep Dive v1.0", ottobre 2020, https://confidentialcomputing.io/wp-content/uploads/sites/85/2020/10/Confidential-Computing-Deep-Dive-white-paper.pdf

Rif. 5: Confidential Computing Consortium "Hardware-Based Trusted Execution for Applications and Data", luglio 2020, https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/01/confidentialcomputing_outreach_whitepaper-8-5×11-1.pdf

Rif. 6: si vedano ad esempio i seguenti recenti articoli: The Register "Microsoft brings Trusted Platform Module functionality directly to CPUs under securo-silicon architecture Pluton", https://www.theregister.com/2020/11/17/microsoft_pluton_cpu_hardware_security/ ; The Hacker News "Intel Adds Hardware-Enabled Ransomware Detection to 11th Gen vPro Chips", https://thehackernews.com/2021/01/intel-adds-hardware-enabled-ransomware.html

Rif. 7: "AMD Memory Encryption white paper", aprile 2016, https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf ; AMD Memory Guard white paper", marzo 2020, https://www.amd.com/system/files/documents/amd-memory-guard-white-paper.pdf

Rif. 8: ArsTecnica "Intel promises Full Memory Encryption in upcoming CPUs", https://arstechnica.com/gadgets/2020/02/intel-promises-full-memory-encryption-in-upcoming-cpus/

Note

[1] Le CPU attuali hanno usualmente 4 o 5 livelli, anche se tipicamente ne sono usati solo 2, o 3 nel caso di supporto Hardware per gli Hypervisor di macchine virtuali; Multics aveva ben 8 livelli.

[2] Per limitarne le dimensioni, i Kernel sono tipicamente modulari ed i moduli sono caricati ed eseguiti a *run-time* solo al momento del reale utilizzo.

Articolo a cura di **Andrea Pasquinucci**

Autore | Bio

Andrea Pasquinucci

PhD CISA CISSP