



Sicurezza, Hardware e Confidential Computing – Parte 2

A cura di:  Andrea Pasquinucci - Pubblicato il  20 Settembre 2021



Nella [prima parte](#) di questo articolo abbiamo descritto brevemente il ruolo dell'Hardware nel supportare e fornire strumenti di sicurezza informatica, abbiamo mostrato come la catena della fiducia sia cruciale per costruire la sicurezza ed introdotto i concetti di *Confidential Computing*, *Secure Boot* e *Trusted Execution Environment*. Abbiamo poi discusso di come oggi si possono proteggere i dati in transito e a riposo, e visto come la cifratura dei dati in RAM possa proteggere parzialmente le informazioni anche durante "l'uso" dei dati stessi.

In questa seconda parte dell'articolo descriviamo ulteriori modalità per garantire la confidenzialità dei dati "in uso" approfondendo cosa si intende oggi per *Confidential Computing*.

Confidenzialità dei dati "In Uso" – Enclavi Sicure

Cifrare i dati in RAM mette in sicurezza un solo scenario di attacco (lettura dei dati in RAM da parte di un utente privilegiato) e non risolve la problematica generale di eseguire elaborazioni di dati sensibili in sicurezza anche su sistemi terzi come gli ambienti Cloud.

Il passo successivo, che ci porta al *Hardware-Based Trusted Execution Environment* (HW-TEE), è quello della realizzazione all'interno della CPU di una "Enclave Sicura", di cui l'esempio più noto sono le *Software Guard Extensions* (SGX) di Intel [Rif. 10] [1].

Piuttosto che cercare di descrivere una tecnologia specifica o l'approccio tecnologico che è in costante e rapida evoluzione, cercheremo di descrivere ad alto livello le finalità di queste ricerche e sviluppi, e

quello che ci possiamo aspettare diventi realtà nel prossimo futuro.

L'idea di base, e la nostra descrizione è per forza approssimativa, è di creare all'interno della CPU una zona Hardware, appunto una "Enclave Sicura", nella quale possano essere elaborati in maniera sicura dati sensibili anche su sistemi di terzi. Alla base di tutto è il fatto che questa elaborazione sicura è protetta dall'Hardware, ovvero utilizza specifiche caratteristiche dell'Hardware per proteggere i dati; per la nostra discussione non è importante che quanto sia protetto sia una macchina virtuale, un container o direttamente un'applicazione, né come la protezione sia implementata in Hardware.

Indicativamente una Enclave Sicura dovrebbe offrire:

- la creazione e la gestione di chiavi crittografiche
- la cifratura/decifrazione di dati con le chiavi prodotte internamente o ricevute dall'esterno
- l'esecuzione di programmi caricati nell'enclave, anche se non tutte le istruzioni macchina della CPU possono essere disponibili
- che nessun altro programma esterno all'enclave, incluso il sistema operativo, possa accedere ai dati o alle istruzioni eseguite all'interno dell'enclave stessa
- istruzioni macchina che permettono la creazione / utilizzo di un'enclave e lo scambio di dati e istruzioni con l'enclave da parte di un processo utilizzatore dell'enclave stessa.

Una modalità semplificata, e vedremo che è già ben complessa, di utilizzare questo scenario potrebbe essere la seguente. Si assuma di avere a disposizione un sistema presso un fornitore Cloud che mette a disposizione una Enclave Sicura di cui ovviamente ci si fida. Sul sistema in Cloud sono caricati dal sistema on-premises alcuni dati cifrati, vedremo più avanti come, ad esempio in un Database. Sul sistema in Cloud è anche presente un'applicazione che è in grado di elaborare normalmente i dati non cifrati presenti nel database, ma non è in grado di elaborare i dati cifrati. Quando è necessario elaborare dei dati cifrati, l'applicazione è in grado di inviarli all'Enclave Sicura e di caricare nell'Enclave Sicura anche una piccola componente applicativa scritta appositamente per essere eseguita nell'Enclave Sicura e che può elaborare i dati una volta decifrati.

Una descrizione come sempre approssimata ed ad alto livello di questo processo, è la seguente:

- i sistemi del cliente e l'Enclave Sicura sul sistema Cloud si scambiano le relative chiavi crittografiche pubbliche o dei certificati digitali che le contengono; in questo modo i sistemi del cliente possono cifrare dei dati che possono essere decifrati solo all'interno dell'Enclave Sicura, e viceversa l'Enclave può cifrare dei dati che possono essere decifrati solo sui sistemi del cliente;
- l'applicazione in esecuzione sul sistema Cloud è in grado di comunicare con l'Enclave Sicura sia scambiando dati cifrati, sia inviando in esecuzione del codice con una firma digitale riconosciuta dall'enclave sicura;
- quando l'applicazione in esecuzione sul sistema Cloud deve elaborare dei dati sensibili e cifrati, invia all'Enclave Sicura il codice (firmato) necessario per l'elaborazione ed i dati cifrati;
- l'Enclave Sicura esegue l'elaborazione come segue:
 - verifica la firma sul codice da eseguire;
 - decifra i dati con la propria chiave privata; [2]
 - esegue l'elaborazione;
 - cifra i dati risultato dell'elaborazione con la chiave pubblica del cliente;
 - invia i dati cifrati all'applicazione in esecuzione sul sistema Cloud.

Sin da questa prima descrizione, è chiaro che questo non è un processo molto semplice, in primo luogo perché richiede che le applicazioni siano adattate a poter eseguire proprie componenti (che devono essere scritte appositamente) all'interno di una Enclave Sicura quando devono agire su dati sensibili e cifrati per l'Enclave stessa. In secondo luogo è necessario gestire le chiavi crittografiche e la cifratura dei dati tra i sistemi del cliente e l'ambiente Cloud.

Tutto questo però ci riporta a considerare il concetto di “fiducia”.

La catena della fiducia distribuita

Come appena descritto, oltre al necessario sviluppo applicativo che permette alle applicazioni in esecuzione su sistemi con Enclave Sicura di comunicare ed eseguire elaborazione in essi, è necessario gestire le chiavi crittografiche con cui identificare sia i sistemi del cliente che l'Enclave Sicura e cifrare i dati. Questo risulta, di nuovo, abbastanza complesso.

Consideriamo il provider Cloud: sinora abbiamo ragionato come se l'applicazione in Cloud fosse eseguita sempre e solo su di un specifico server hardware la cui CPU ha una Enclave Sicura, ma questo al giorno d'oggi sicuramente non è vero. La situazione più comune è che l'applicazione è eseguita in una o più macchine virtuali che possono migrare tra macchine fisiche anche in sale macchine e località diverse.

Come facciamo a gestire le relative chiavi crittografiche o anche solo ad essere sicuri che l'applicazione sia in esecuzione su una CPU con una Enclave Sicura reale e non una versione virtuale che permette ad un attaccante di intercettare tutti i dati del cliente?

Per rispondere a questa domanda dobbiamo ripartire dalla catena della fiducia: per prima cosa il cliente si deve fidare del fornitore Cloud che deve assicurare di avere un gruppo di server fisici ognuno con la propria Enclave Sicura. Nel caso di una Enclave Sicura SGX è richiesto che [Rif. 15]:

1. L'Enclave è in esecuzione su un vero processore Intel;
2. La piattaforma è aggiornata al livello di sicurezza più recente;
3. L'identità dell'Enclave è quella asserita;
4. L'Enclave non è stata manomessa.

Per soddisfare queste richieste è necessario che il *Secure Boot* (si veda la prima parte di questo articolo) garantisca l'autenticità ed integrità del sistema Hardware e Software, e fornisca anche le relative versioni di Hardware e Software. Per gestire tutto ciò è necessario che il provider Cloud abbia un servizio di *Remote Attestation* [Rif. 2, 3] in grado di raccogliere le attestazioni di *Secure Boot* di ogni server (firmate digitalmente) contenenti anche l'indicazione delle versioni di Firmware, sistema operativo ecc.

Tramite il servizio di *Remote Attestation* del provider Cloud, un cliente può quindi identificare i server e le CPU le cui caratteristiche Hardware e Software soddisfano i requisiti di sicurezza di cui necessita.

A questo punto è necessario fare una osservazione che riguarda le macchine virtuali: la configurazione standard dei sistemi di virtualizzazione rende disponibile alle macchine virtuali solo Hardware virtuale e non permette l'accesso all'Hardware reale. Diversamente i Container accedono all'Hardware reale (non virtuale) anche se il sistema operativo limita fortemente le loro possibilità di accesso. Quindi se si usano macchine virtuali è necessario accertarsi che queste possano accedere a Enclavi Sicure fisiche e non virtualizzate (almeno per quanto si discute in questo articolo), mentre questo problema non si pone per i Container.

Ritorniamo ora al problema dell'Attestazione delle Enclavi Sicure considerando come esempio il caso di Container. L'Orchestratore che gestisce i Container (ad esempio Kubernetes o OpenStack) deve pertanto essere a conoscenza dell'Attestazione di ogni server e deve poter gestire una classificazione di ogni Container secondo i requisiti di sicurezza richiesti dal cliente, che permette di selezionare i server e le CPU sui quali il Container può essere eseguito in sicurezza ed in presenza di un Enclave Sicura “attestata”. Ovviamente il provider Cloud può raggruppare i propri server in classi di sicurezza che tengono conto di vari parametri, dall'Attestazione Hardware e Software alla località o regione in cui il server è situato.

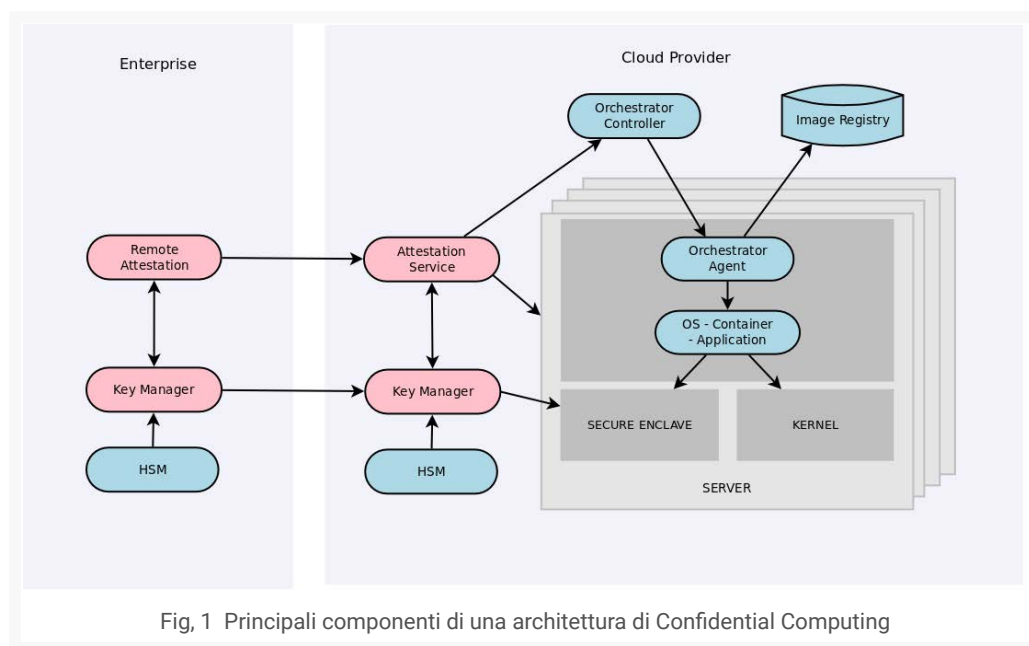
Questo sistema permette quindi al cliente di selezionare i server con Enclave Sicura sui quali può essere eseguito il proprio applicativo capace di utilizzarla.

Il successivo passo è quello di scambiare le chiavi crittografiche necessarie a cifrare e decifrare i dati. Ovviamente anche questa procedura deve essere automatizzata, ma deve avere anch'essa una propria radice della catena della fiducia (o *Root of Trust*) sia presso il provider Cloud che presso il cliente.

Per fare questo dobbiamo ricordarci che ogni Enclave Sicura ha le proprie chiavi crittografiche, e che è necessario gestire anche le chiavi crittografiche pubbliche (o certificati digitali) del cliente. Per questo è in pratica necessario utilizzare un *Key Manager*, ovvero un'applicazione dedicata alla gestione delle chiavi crittografiche e che può quindi raccogliere e rendere disponibili le chiavi crittografiche necessarie. Un *Key Manager* di norma utilizza un *Hardware Security Module* (HSM) come *Root of Trust*, ovvero la chiave privata che identifica il *Key Manager* è creata e gestita unicamente all'interno dell'HSM, protetta in Hardware. Se anche il cliente utilizza un *Key Manager*, basta che ogni *Key Manager* riconosca la chiave crittografica *Root of Trust* dell'altro per stabilire la fiducia relativa tra i due ambienti. Il processo è simile idealmente a quello delle *Certification Authorities* (CA) per la navigazione Web: una volta riconosciuta ed accettata la CA superiore (in questo caso la chiave crittografica *Root of Trust* dell'altro *Key Manager*), tutti i certificati digitali e le chiavi crittografiche firmate da questa, anche in cascata, sono accettati.

Infine la comunicazione ed il trasferimento di chiavi crittografiche tra *Key Manager*, HSM e server è possibile ad esempio utilizzando il "*Key Management Interoperability Protocol*" (KMIP) [Rif. 16, 17], protocollo sviluppato apposta da OASIS a questo scopo. [3]

In Figura 1 abbiamo provato a riassumere i principali componenti di questa architettura.



E' opportuno notare che il processo appena descritto di gestione delle chiavi crittografiche del cliente da parte di un provider Cloud, si sta diffondendo e molte aziende lo hanno o stanno adottando sotto il nome di *Bring Your Own Key* (BYOK). L'idea è che una applicazione o servizio del provider Cloud o presso il provider Cloud, sia essa uno Storage, un Database o dei dati cifrati a livello applicativo, per cifrare e decifrare i dati utilizza delle chiavi crittografiche non create, gestite e archiviate nei sistemi del provider Cloud stesso, ma nei sistemi del cliente. Le chiavi crittografiche sono fornite all'applicazione o servizio solo quando sono necessarie per la cifratura/decifrazione dei dati e direttamente dai sistemi del cliente tramite i *Key Manager* e protocolli quali KMIP. Le chiavi crittografiche non sono mai archiviate o salvate sui sistemi del provider Cloud ma mantenute in memoria sicura solo per il tempo strettamente necessario alla cifratura/decifrazione dei dati.

Tornando al Confidential Computing, dovrebbe ormai essere chiaro che la sua implementazione è abbastanza complessa sia a livello tecnico che di gestione. Per raggiungere e garantire la confidenzialità (e integrità) completa della elaborazione di informazioni su di una piattaforma di terzi, dobbiamo comunque partire dalla fiducia di base nell'Hardware e Firmware stesso dei sistemi utilizzati dalla terza parte e dobbiamo dotarci di applicazioni e servizi che permettano di:

- identificare e verificare l'identità e integrità dei sistemi sui cui eseguire le elaborazioni in sicurezza (Attestazione Remota);
- gestire le chiavi di cifratura e la cifratura dei dati in modo che i dati siano disponibili in chiaro solo per le elaborazioni nelle Enclavi Sicure sui sistemi Hardware scelti;
- adattare le applicazioni in esecuzione sulla piattaforma del Provider in modo che possano comunicare ed eseguire le componenti di codice necessarie nelle Enclavi Sicure.

Sicurezza e HW-TEE

L'architettura che abbiamo appena descritta molto velocemente ed ad alto livello, dovrebbe permettere l'elaborazione di dati sensibili su sistemi di terze parti garantendo la massima confidenzialità ed integrità dei dati. Infatti non vi dovrebbe essere alcun modo in cui la terza parte sui cui sistemi avvengono le elaborazioni, abbia la possibilità di accedere ai dati in chiaro.

La domanda che sorge spontanea è quanto sia sicura in se stessa questa architettura. Vi sono vari aspetti da considerare:

- la complessità dell'architettura stessa ed il coinvolgimento di diversi protocolli può esporre a vulnerabilità nei protocolli stessi o dovute all'interazione tra i protocolli;
- come sempre, è possibile che vi siano vulnerabilità di sicurezza nel codice di Firmware, sistemi operativi ed applicazioni, ad esempio anche quelle utilizzate per la gestione delle chiavi crittografiche;
- similmente è possibile che l'implementazione in Software degli algoritmi crittografici abbia una vulnerabilità mentre, a meno dell'arrivo di un elaboratore quantistico, gli algoritmi crittografici in uso sono oggi considerati sicuri;
- infine non va sottovalutato l'aspetto di possibili vulnerabilità dell'Hardware stesso.

E' opportuno in particolare approfondire l'ultimo punto. Tutta l'architettura che abbiamo descritto si basa sulla presenza di una Enclave Sicura in Hardware. Come ben sappiamo anche solo dalla recente esperienza con le vulnerabilità Hardware delle CPU quali *Meltdown* e *Spectre*, la reale soluzione di vulnerabilità Hardware richiede la sostituzione dell'Hardware stesso, anche se in alcuni casi è possibile mitigare i rischi introducendo specifici controlli nel Firmware.

Quindi una vulnerabilità nell'Hardware di una Enclave Sicura minerebbe alla base la sicurezza delle informazioni gestite con il Confidential Computing e HW-TEE. Purtroppo le prime generazioni di HW-TEE non sono immuni da vulnerabilità di sicurezza in Hardware (si veda [Rif. 9] per un elenco esemplificativo) anche se alcune di queste sono state mitigate con appositi controlli Firmware.

E' importante notare che le Enclavi Sicure Hardware, quali Intel SGX, disponibili al momento sul mercato, sono state progettate senza considerare come scenario di sicurezza gli attacchi di inferenza "*Side Channel*", quali gli appena citati *Meltdown* e *Spectre*. Pertanto sono stati dimostrati, in condizioni di laboratorio, alcuni attacchi Side Channel in grado di estrarre dati sensibili da Enclavi Sicure Hardware. Ovviamente future generazioni di Enclavi Sicure risolveranno queste vulnerabilità, ma la soluzione sarà più costosa e complessa rispetto ad un aggiornamento software.

Questo rende ancora più importante il processo di Attestazione Remota dei sistemi, in questo caso per l'Hardware, per cui un cliente potrebbe individuare i singoli modelli di HW-TEE del fornitore su cui eseguire le proprie elaborazioni basandosi sulle vulnerabilità Hardware note ed i possibili rischi alle proprie informazioni che ne potrebbero derivare.

Infine va citato un aspetto, per certi versi controverso, degli HW-TEE, ovvero la possibilità di un loro abuso od uso per fini non leciti. I dati elaborati in un HW-TEE non sono accessibili a nessuno se non il legittimo proprietario, per cui ad esempio anche applicazioni quali anti-virus/malware ecc. non possono verificare la presenza di codice maligno in esecuzione in un HW-TEE (ma potrebbero essere identificati gli effetti sui sistemi esterni all'HW-TEE), e non è possibile verificare che non vengano eseguite transazioni all'interno di HW-TEE non lecite per motivi legali o contrattuali. Comunque al momento non è del tutto chiaro quanto questi possano essere dei rischi reali.

Crittografia Omomorfica

Un approccio complementare o alternativo al Confidential Computing è quello di utilizzare la Crittografia Omomorfica [Rif. 11]. Sin dal 1978 nel contesto dello studio degli algoritmi crittografici asimmetrici quali RSA, Rivest, Adleman e Dertouzos proposero il concetto di Crittografia Omomorfica, ovvero di algoritmi crittografici in qualche modo commutativi rispetto alle operazioni matematiche. Si può descrivere questo concetto con il seguente esempio: dati due numeri M e N ed una operazione $O[,]$ tra di loro (somma, prodotto ecc.), un algoritmo crittografico omomorfico $C()$ permette di cifrare i dati in $C(M)$ e $C(N)$ ed ha una operazione $P[,]$ sui dati cifrati tale che

$$P[C(M), C(N)] = C(O[M,N])$$

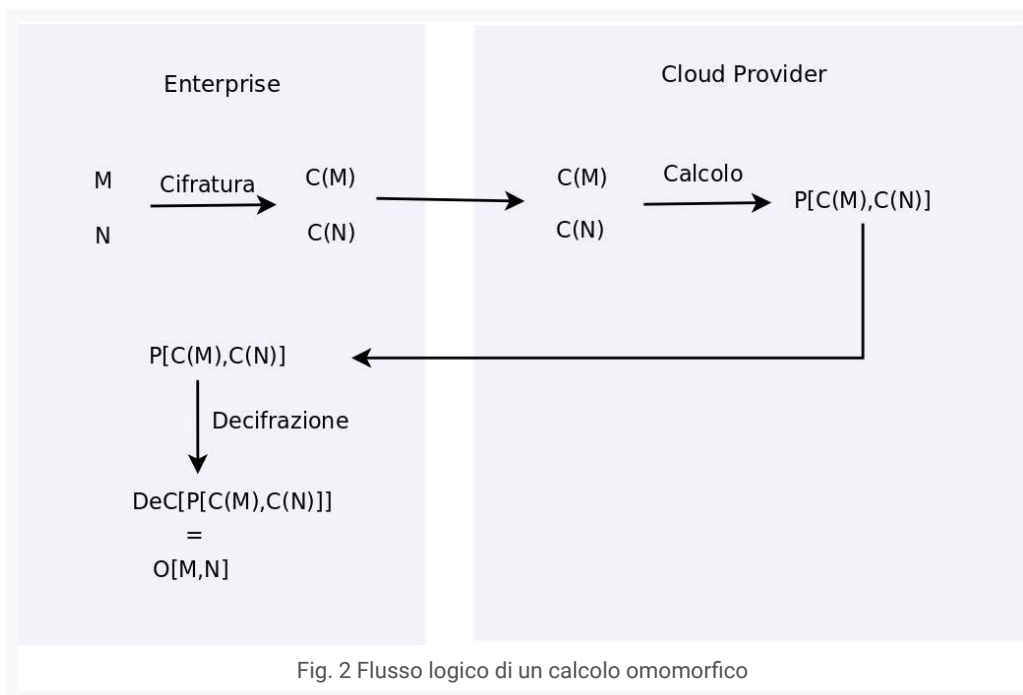
Ad esempio supponendo che O sia la somma e che P sia il prodotto[4]:

$$C(M) * C(N) = C(M + N)$$

In altre parole, se si cifrano i due numeri, si moltiplicano i due numeri cifrati e infine di decifra il risultato del prodotto, si ottiene la somma dei due numeri originali:

$$M + N = C^{-1}(C(M) * C(N))$$

Assumiamo quindi che il nostro fornitore Cloud offra sistemi che supportano il calcolo omomorfico di questo tipo, possiamo quindi cifrare *on-premises* tutti i dati numerici prima di inviarli all'applicazione Cloud, far eseguire il prodotto dei numeri cifrati dall'applicazione Cloud, scaricare dall'applicazione Cloud il risultato del prodotto dei numeri e decifrarlo *on-premises* ottenendo la somma dei numeri originali. In tutto ciò il sistema Cloud gestisce solo dati cifrati e nessuna chiave di cifratura/decifrazione, e quindi non ha nessuna possibilità di accedere ai dati in chiaro.



La crittografia omomorfica è da sempre considerata come la soluzione definitiva per la confidenzialità dei dati in quanto permette al proprietario dei dati di mantenere la confidenzialità degli stessi (solo il proprietario può decifrarli) ed al contempo permette ad altri di elaborare i dati per conto del proprietario senza però venire a conoscenza delle informazioni rappresentate dai dati stessi.

Purtroppo, come al solito, non tutto è così semplice come si desidererebbe (altrimenti utilizzeremmo già da decenni la crittografia omomorfica). Prima di tutto, dall'enunciazione dell'idea di crittografia omomorfica (1978) sino al primo modello teorico reale sono passati ben 31 anni, ovvero sino alla tesi di dottorato del 2009 di Craig Gentry [Rif. 12]. Da allora sono stati identificati ben 4 tipi di "homomorphic encryption scheme" (HE o FHE) [5], ed i lavori dei crittografi continuano. Non solo, nel frattempo si è anche aggiunta la richiesta di sicurezza di produrre algoritmi resistenti ad eventuali futuri arrivi dei Computer Quantistici [Rif. 13] che richiede ulteriori valutazioni e, in parte, lo sviluppo di nuovi algoritmi.

Ad oggi sono già disponibili librerie sperimentali che permettono di utilizzare la crittografia omomorfica ed il consorzio in [Rif. 11] ha proprio lo scopo di definire degli standard in modo che soprattutto i fornitori di servizi Cloud la possano offrire ai propri clienti.

I problemi principali sono dovuti alla lentezza e dimensione degli algoritmi e dei dati, ed alle reali garanzie di sicurezza. Benché gli ultimi algoritmi siano ordini di grandezza più veloci e leggeri dei primi proposti, sono ancora molto lenti rispetto al calcolo svolto sui dati in chiaro e la procedura di cifratura rende i dati molto più grandi, occupando più spazio a riposo, in transito ed in uso.

Gli aspetti di sicurezza riguardano da una parte la sicurezza intrinseca degli algoritmi e dei protocolli stessi, che è ovviamente in corso di valutazione, e dall'altra le modalità di uso e applicative anche per evitare attacchi di inferenza e "side channel".

Uno dei principali problemi degli attuali algoritmi omomorfici è che di norma permettono di effettuare solo alcune operazioni aritmetiche, ad esempio solo la somma e la moltiplicazione. Questo rende difficile, se non impossibile, utilizzarli con applicazioni complesse che gestiscono molti dati, come un DataWareHouse od un modello di Machine Learning. Ovviamente si spera che queste siano solo difficoltà tecniche che in futuro possano essere superate.

Inoltre, finché ci si limita a effettuare somme e prodotti di numeri è difficile immaginare attacchi di inferenza sui dati, ovvero dedurre delle informazioni direttamente dai dati cifrati. Supponiamo invece come semplice esempio, di avere una tabella con gli stipendi dei dipendenti cifrata con crittografia omomorfica e che sia possibile effettuare un test logico su due dati cifrati che ha come risultato di sapere se il primo dato reale è maggiore del secondo (questo è lo stesso che introdurre un meccanismo per ordinare i dati reali). Visto che i dati sono cifrati, guardando questa operazione da sola (atomicamente) non ne otteniamo quasi alcuna informazione sui dati reali. Ma se prendiamo un gruppo di dati, possiamo incominciare ad estrarne delle informazioni: per prima cosa il loro ordinamento reale; inoltre se due dati (cifrati) hanno lo stesso numero di dati più grandi di loro, vuol dire che i dati reali sono uguali. Quindi la possibilità di ordinare i dati cifrati secondo l'ordinamento numerico dei dati reali (ignoti a chi esegue l'ordinamento) permette di ottenere alcune informazioni sui dati reali stessi.

Questo è ovviamente un esempio banale, ma ci deve far riflettere se e come la crittografia omomorfica sarà da sola in grado di garantire la confidenzialità delle informazioni anche durante l'elaborazione in un ambiente potenzialmente ostile, o se più probabilmente sarà un altro strumento a nostra disposizione per la protezione delle nostre informazioni da utilizzare insieme agli altri, a partire dal Confidential Computing.

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-8-5x11-1.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: ArsTechnica "Intel promises Full Memory Encryption in upcoming CPUs",
<https://arstechnica.com/gadgets/2020/02/intel-promises-full-memory-encryption-in-upcoming-cpus/>

Rif. 9: si vedano ad esempio i seguenti recenti articoli:

- ArsTechnica "Hackers can use just-fixed Intel bugs to install malicious firmware on PCs",
<https://arstechnica.com/information-technology/2020/11/intel-patches-high-severity-bugs-protecting-lost-stolen-or-confiscated-pcs/> ;
- The Register "Intel's SGX cloud-server security defeated by \$30 chip, electrical shenanigans",
https://www.theregister.com/2020/11/14/intel_sgx_physical_security/ ;
- Nilsson, P.S. Bideh, J. Brorsson "A Survey of Published Attacks on Intel SGX", giugno 2020,
<https://arxiv.org/abs/2006.13598> ;
- Steel "How Ledger Hacked an HSM", giugno 2019, <https://cryptosense.com/blog/how-ledger-hacked-an-hsm> ;
- Wikipedia "Software Guard Extensions", https://en.wikipedia.org/wiki/Software_Guard_Extensions ;
- JP Aumasson, Luis Merino, "SGX Secure Enclaves in Practice: Security and Crypto Review", BlackHat 2016, <https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf>

Rif. 10: per Intel SGX si veda ad esempio <https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html>

Rif. 11: per riferimenti si veda ad esempio consorzio "Homomorphic Encryption Standardization",
<https://homomorphicencryption.org/>

Rif. 12: Craig Gentry "Fully Homomorphic Encryption Using Ideal Lattices". In the 41st ACM Symposium on Theory of Computing (STOC), 2009, <http://portal.acm.org/citation.cfm?id=1536414.1536440>

Rif. 13: si veda ad esempio lo standard NIST in sviluppo "Post-Quantum Cryptography",
<https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>

Rif. 14:

- Microsoft "Azure confidential computing virtual machines (VMs) overview",
<https://docs.microsoft.com/en-us/azure/confidential-computing/confidential-computing-enclaves> ;
- Microsoft "Always Encrypted with secure enclaves", <https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves?view=sql-server-ver15> ;
- Amazon "AWS Nitro Enclaves", <https://aws.amazon.com/it/ec2/nitro/nitro-enclaves/> ;
- Google "Confidential Computing", <https://cloud.google.com/confidential-computing>

Rif. 15: Intel Corporation "Strengthen Enclave Trust with Attestation", 2020,
<https://software.intel.com/en-us/sgx/attestation-services>




Rif. 16: "OASIS Key Management Interoperability Protocol (KMIP)", https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip

Rif. 17: NIST Special Publication 800-57 1.rev5 "Recommendation for Key Management",
<https://doi.org/10.6028/NIST.SP.800-57pt1r5>

Note

- [1] Alcune implementazioni di enclavi sono riportate in [Rif. 14].
- [2] Tipicamente con la chiave asimmetrica (componente pubblica), viene cifrata una chiave simmetrica (ad esempio per AES) utilizzata per cifrare i dati; le chiavi asimmetriche non vengono normalmente utilizzate per cifrare i dati direttamente.
- [3] Oltre a KMIP vi sono altri protocolli / Programming Interface / API (alcuni proprietari) quali MSCAPI, MS-EKM, PKCS#11 ecc., che permettono il trasferimento di chiavi crittografiche.
- [4] Esistono algoritmi omomorfici che implementano esattamente questa proprietà; invece RSA ha la caratteristica di essere commutativo rispetto alla moltiplicazione, ovvero nel nostro esempio sia O che P sono la moltiplicazione.
- [5] Le quattro generazioni di “Homomorphic Encryption Scheme” sono:
1. *Partially Homomorphic Encryption*: schemi di cifratura che permettono l'esecuzione di un solo tipo di operazione, ad esempio solo la somma o il prodotto
 2. *Somewhat Homomorphic Encryption*: schemi di cifratura che permettono l'esecuzione di due tipi di operazioni, ad esempio somma e prodotto, ma solo per un numero limitato di composizioni tra le operazioni (detto anche “sottoinsieme di circuiti”)
 3. *Leveled Fully Homomorphic*: schemi di cifratura che permettono l'esecuzione di molti tipi di operazioni, ad esempio somma, prodotto ecc., ma solo per un numero limitato di iterazioni (detto anche “circuiti a profondità limitati”) senza limitazioni sul numero di composizioni
 4. *Fully Homomorphic*: schemi di cifratura che permettono l'esecuzione di molti tipi di operazioni senza limitazioni sul numero di iterazioni né di composizioni.

Articolo a cura di **Andrea Pasquinucci**

 Autore	 Bio
	<p>Andrea Pasquinucci PhD CISA CISSP</p>