Dettagli della vulnerabilità: CVE-2023-4807

Riepilogo del problema: l'implementazione POLY1305 MAC (codice di autenticazione del messaggio) contiene un bug che potrebbe danneggiare lo stato interno delle applicazioni sulla piattaforma Windows 64 quando vengono eseguite su processori X86_64 più recenti che supportano le istruzioni AVX512-IFMA. Riepilogo dell'impatto: se in un'applicazione che utilizza la libreria OpenSSL un utente malintenzionato può influenzare l'utilizzo dell'algoritmo MAC POLY1305, lo stato dell'applicazione potrebbe essere danneggiato con varie conseguenze dipendenti dall'applicazione. L'implementazione POLY1305 MAC (codice di autenticazione del messaggio) in OpenSSL non salva il contenuto dei registri XMM non volatili sulla piattaforma Windows 64 quando si calcola il MAC dei dati più grandi di 64 byte. Prima di ritornare al chiamante tutti i registri XMM vengono azzerati anziché ripristinare il contenuto precedente. Il codice vulnerabile viene utilizzato solo sui processori x86_64 più recenti che supportano le istruzioni AVX512-IFMA. Le conseguenze di questo tipo di corruzione dello stato interno dell'applicazione possono essere varie: da nessuna conseguenza, se l'applicazione chiamante non dipende affatto dal contenuto dei registri XMM non volatili, alle conseguenze peggiori, in cui l'aggressore potrebbe ottenere il controllo completo di il processo di candidatura. Tuttavia, dato che il contenuto dei registri viene semplicemente azzerato in modo che l'aggressore non possa inserire valori arbitrari all'interno, la conseguenza più probabile, se presente, sarebbe un risultato errato di alcuni calcoli dipendenti dall'applicazione o un arresto anomalo che porta a una negazione del servizio. L'algoritmo POLY1305 MAC viene utilizzato più frequentemente come parte dell'algoritmo CHACHA20-POLY1305 AEAD (crittografia autenticata con dati associati). L'utilizzo più comune di questo codice AEAD è con le versioni 1.2 e 1.3 del protocollo TLS e un client dannoso può influenzare l'utilizzo di questo codice AEAD da parte del server. Ciò implica che le applicazioni server che utilizzano OpenSSL possono essere potenzialmente influenzate. Tuttavia, al momento non siamo a conoscenza di alcuna applicazione concreta che potrebbe essere interessata da questo problema, pertanto lo consideriamo un problema di sicurezza di bassa gravità. Come soluzione alternativa, il supporto delle istruzioni AVX512-IFMA può essere disabilitato in fase di esecuzione impostando la variabile di ambiente OPENSSL_ia32cap: OPENSSL_ia32cap=:~0x200000 Il provider FIPS non è interessato da questo problema.

Utilizza una libreria crittografica affidabile : Evita di implements Poly1305 da zero. Utilizza invece una libreria crittografica ben consolidata, come libsodium (una libreria crittografica moderna e facile da usare) o OpenSSL.

- **Genere chiavi robuste** : Assicurati di generare chiavi di autenticazione robuste e sic
- **Proteggi le chiavi** : Mantieni le chiavi di autenticazione al
- **Evita di rivelare le chiavi** : Mai esporre le chiavi di autenticazione nei tuoi codici sorgente o in log non protetti.
- **Valida i dati di input** : Assic
- **Proteggi le comunicazioni** : Se stai utilizzando Poly1305 per autenticare dati scambiati
- **Monitoraggio e registrazioni** : configura un sistema di monitoraggio
- **Aggiorna regolarmente** : Mantieni aggiornati le librerie crittografiche e gli algoritmi
- **Seguire le linee guida di sicurezza** : Seg

Dettagli della vulnerabilità: CVE-2023-3817

Riepilogo del problema: il controllo di chiavi o parametri DH eccessivamente lunghi potrebbe essere molto lento. Riepilogo dell'impatto: le applicazioni che utilizzano le funzioni DH_check(), DH_check_ex() o EVP_PKEY_param_check() per verificare una chiave DH o parametri DH potrebbero riscontrare lunghi ritardi. Laddove la chiave o i parametri da controllare siano stati ottenuti da una fonte non attendibile, ciò potrebbe portare a un rifiuto di servizio. La funzione DH_check() esegue vari controlli sui parametri DH. Dopo aver corretto CVE-2023-3446 si è scoperto che un valore elevato del parametro q può anche attivare un calcolo eccessivamente lungo durante alcuni di questi controlli. Un valore q corretto, se presente, non può essere maggiore del parametro modulo p, quindi non è necessario eseguire questi controlli se q è maggiore di p. Un'applicazione che richiama DH_check() e fornisce una chiave o parametri ottenuti da una fonte non attendibile potrebbe essere vulnerabile a un attacco Denial of Service. La funzione DH_check() è essa stessa chiamata da una serie di altre funzioni OpenSSL. Un'applicazione che richiama una qualsiasi di queste altre funzioni potrebbe essere influenzata in modo simile. Le altre funzioni interessate da questo sono DH_check_ex() e EVP_PKEY_param_check(). Sono vulnerabili anche le applicazioni della riga di comando OpenSSL dhparam e pkeyparam quando si utilizza l'opzione "-check". L'implementazione SSL/TLS di OpenSSL non è interessata da questo problema. I provider FIPS OpenSSL 3.0 e 3.1 non sono interessati da questo problema.

Lo *scambio di chiavi Diffie-Hellman* (**DH**,**Diffie-Hellman key exchange**) è un protocollo crittografico attraverso cui due entità sono in grado di condividere le proprie chiavi segrete su un canale di comunicazione pubblico, quindi considerato insicuro.

La verifica di una chiave DH coinvolge la conferma

Ecco i passi generali per verificare una chiave DH:

- **Ricezione della chiave pubblica** : Assicurati di ricevere la chiave pubblica da una fonte attendibile o da una comunicazione sicura.
- **Verifica dei parametri DH** : Controlla che i parametri Diffie
- **Verifica della firma digitale (opzionale)** : Se hai concordato di utilizzare una firma digitale per autenticare la chiave pubblica DH, verifica che la firma sia valida utilizzando la chiave pubblica del mittente.
- **Scambio di chiavi segrete** : Utilizza la chiave pubblica ricevuta per generare la chiave

CVE-2023-0466 :

La funzione X509_VERIFY_PARAM_add0_policy() è documentata per abilitare implicitamente il controllo della politica di certificato durante la verifica del certificato. Tuttavia l'implementazione della funzione non abilita il controllo che consente ai certificati con policy non valide o errate di superare la verifica del certificato. Poiché l'attivazione improvvisa del controllo della policy potrebbe interrompere le distribuzioni esistenti, si è deciso di mantenere il comportamento esistente della funzione X509_VERIFY_PARAM_add0_policy(). Invece le applicazioni che richiedono OpenSSL per eseguire il controllo della politica del certificato devono utilizzare X509_VERIFY_PARAM_set1_policies() o abilitare esplicitamente il controllo della politica chiamando X509_VERIFY_PARAM_set_flags() con l'argomento flag X509_V_FLAG_POLICY_CHECK. I controlli dei criteri di certificato sono disabilitati per impostazione predefinita in OpenSSL e non sono comunemente utilizzati dalle applicazioni.

Il controllo della politica di certificato durante la verifica di un certificato è una parte importante del processo di autenticazione e sicurezza in

- **Ottieni il certificato** : Ricevi il certificato da una fonte attendibile, ad esempio da un server Web tramite una connessione sicura o da una persona di fiducia. Un certificato X.509 standard contiene informazioni sul certificato stesso e sulla politica di certificato associata.
- **Estrai le informazioni del certificato** : Estrai le informazioni rilevanti dal certificato, come la politica di cert
- **Verifica la validità del certificato** : Prima di controllare la politica di certificato, verifica
- **Controlla la politica di certificato** : Verifica se il certificato soddisfa le politiche di certificato specifiche. Questo può comportare il confronto del campo OID nel certificato con le politiche di certificato richieste o accettate.
- **Gestisci le eccezioni** : In caso di discrepanze tra le politiche di certificato e il certificato stesso,
- **Registra e monitora** : Tieni traccia delle decisioni pre

CVE-2023-0464	È stata identificata una vulnerabilità di sicurezza in tutte le versioni supportate di OpenSSL relativa alla verifica delle catene di certificati X.509 che includono vincoli di policy. Gli aggressori potrebbero essere in grado di sfruttare questa vulnerabilità creando una catena di certificati dannosi che innesca un uso esponenziale delle risorse computazionali, portando a un attacco Denial of Service (DoS) sui sistemi interessati. L'elaborazione delle policy è disabilitata per impostazione predefinita ma può essere abilitata passando l'argomento "-policy"alle utilità della riga di comando o chiamando la funzione "X509_VERIFY_PARAM_set1_policies()".

programma che protegge dagli attacchi DOS un controllo dei danni e garantisce un riavvio del sistema

```
import time
import threading

class DoSProtectionSystem:
    def __init__(self):
        self.dos_counter = 0
        self.reboot_needed = False

    def monitor_traffic(self):
        while True:
            # Simulazione del controllo del traffico in ingresso
            traffic = self.measure_traffic()

            if traffic > 1000:
```

```
                # Potenziale attacco DoS rilevato
                self.dos_counter += 1
                if self.dos_counter > 5:
                    # Se ci sono stati troppi attacchi, richiedi un reboot del
sistema
                    self.reboot_needed = True
                    self.mitigate_dos_attack()

            # Simulazione del controllo ogni minuto
            time.sleep(60)

    def measure_traffic(self):
        # Simulazione di misurazione del traffico in ingresso
        return 1000  # Modifica il valore in base alle tue esigenze

    def mitigate_dos_attack(self):
        # Simulazione di misure di mitigazione
        print("Misure di mitigazione in corso...")
        # Esegui azioni di mitigazione reali qui

    def protect_system(self):
        # Avvia il monitoraggio del traffico in un thread separato
        monitoring_thread = threading.Thread(target=self.monitor_traffic)
        monitoring_thread.daemon = True
        monitoring_thread.start()

        while True:
            if self.reboot_needed:
                # Richiedi un reboot del sistema
                self.reboot_system()

            # Simulazione di altre operazioni di routine
            time.sleep(300)  # Esegui ogni 5 minuti

    def reboot_system(self):
        # Simulazione di un reboot del sistema
        print("Richiesta di reboot del sistema...")
        # Esegui le operazioni di reboot effettive qui
```

```
if __name__ == "__main__":
    dos_protection_system = DoSProtectionSystem()
    dos_protection_system.protect_system()
```

CVE-2023-0286	Esiste una vulnerabilità legata alla confusione dei tipi relativa all'elaborazione degli indirizzi X.400 all'interno di un GeneralName X.509. Gli indirizzi X.400 sono stati analizzati come ASN1_STRING ma la definizione della struttura pubblica per GENERAL_NAME specificava erroneamente il tipo del campo x400Address come ASN1_TYPE. Questo campo viene successivamente interpretato dalla funzione OpenSSL GENERAL_NAME_cmp come ASN1_TYPE anziché come ASN1_STRING. Quando il controllo CRL è abilitato (ovvero l'applicazione imposta il flag X509_V_FLAG_CRL_CHECK), questa vulnerabilità può consentire a un utente malintenzionato di passare puntatori arbitrari a una chiamata memcmp, consentendogli di leggere il contenuto della memoria o attuare un rifiuto di servizio. Nella maggior parte dei casi, l'attacco richiede che l'autore dell'attacco fornisca sia la catena di certificati che il CRL, nessuno dei quali deve necessariamente avere una firma valida. Se l'utente malintenzionato controlla solo uno di questi input, l'altro input deve già contenere un indirizzo X.400 come punto di distribuzione CRL, il che è raro. Pertanto, è molto probabile che questa vulnerabilità interessi solo le applicazioni che hanno implementato la propria funzionalità per il recupero dei CRL su una rete.

La riservatezza legata alla confusione dei tipi, spesso indicata come attacco di "type confusion", può verificarsi quando i dati vengono interpretati erroneamente come un tipo diverso da quello previsto. Nel contesto X.509, questa riservatezza può essere

Per risolvere una vulnerabilità di questo tipo all'interno di un campo GeneralName in un certificato X.509, sono necessarie misure di sicurezza e correzioni nel software o nelle librerie che elaborano questi certificati. Ecco alcuni passi che possono essere intrapresi per mitigare questa vulnerabilità:

- **Aggiornamento delle librerie** : Assicurarsi di utilizzare le versioni più recenti delle librerie crittografiche e dei software che gestiscono i certificati X.509. Gli sviluppatori spesso rilasciano correzioni per problemi di sicurezza noti, incluso il tipo confusione.
- **Validazione dei certificati** : implementare una rigorosa validazione dei certificati durante la comunicazione
- **Impostazione di politiche di sicurezza** : Definire e seguire politiche di sicurezza ben definite per l'uso dei certificati
- **Controllo dei campi GeneralName** : Prestare particolare attenzione alla gestione dei campi GeneralName nei certificati X.509. Assicurarsi che vengano interpretati correttamente e che non ci siano ambiguità
- **Audit delle applicazioni** : effettuare audit e test di sicurezza regolari sulle applicazioni che elaborano certificati X.509. Ciò può aiutare a identificare un'eventuale nota pericolosa o sosp
- **Partecipazione alla comunità di sicurezza** : Tenersi aggiornati sulle vulnerabilità note e le migliori pratiche di sicurezza, partecipando
- **Risposta all'incidente** : avere un piano di risposta agli incidenti in caso di violazione della sicurezza o tentativo di sfruttare la vulnerabilità. Questo piano dovrebbe definire le azioni da

intraprend

La risoluzione della debolezza legata alla confusione dei tipi richiede un approccio completo alla sicurezza, compreso il monitoraggio costante e l'aggiornamento delle difese contro le minacce. Inoltre, è importante collaborare con la comunità di sicurezza e con altri operatori di sistemi per mantenere un ambiente sicuro e resiliente.

CVE-2023-0215	La funzione API pubblica BIO_new_NDEF è una funzione di supporto utilizzata per lo streaming di dati ASN.1 tramite un BIO. Viene utilizzato principalmente internamente a OpenSSL per supportare le funzionalità di streaming SMIME, CMS e PKCS7, ma può anche essere chiamato direttamente dalle applicazioni dell'utente finale. La funzione riceve un BIO dal chiamante, antepone un nuovo BIO del filtro BIO_f_asn1 sulla parte anteriore per formare una catena BIO, quindi restituisce la nuova testa della catena BIO al chiamante. In determinate condizioni, ad esempio se la chiave pubblica di un destinatario CMS non è valida, il nuovo filtro BIO viene liberato e la funzione restituisce un risultato NULL che indica un errore. Tuttavia, in questo caso, la catena BIO non viene ripulita adeguatamente e il BIO passato dal chiamante conserva ancora puntatori interni al BIO filtro precedentemente liberato. Se il chiamante continua a chiamare BIO_pop() sul BIO, si verificherà un use-after-free. Ciò molto probabilmente provocherà un incidente. Questo scenario si verifica direttamente nella funzione interna B64_write_ASN1() che potrebbe causare la chiamata di BIO_new_NDEF() e successivamente chiamerà BIO_pop() sul BIO. Questa funzione interna è a sua volta chiamata dalle funzioni API pubbliche PEM_write_bio_ASN1_stream, PEM_write_bio_CMS_stream, PEM_write_bio_PKCS7_stream, SMIME_write_ASN1, SMIME_write_CMS e SMIME_write_PKCS7. Altre funzioni API pubbliche che potrebbero essere interessate da questo includono i2d_ASN1_bio_stream, BIO_new_CMS, BIO_new_PKCS7, i2d_CMS_bio_stream e i2d_PKCS7_bio_stream. Le applicazioni a riga di comando OpenSSL cms e smime sono interessate in modo simile.

ASN.1 (Abstract Syntax Notation One) è un formato di codifica utilizzato per rappresentare dati strutturati in modo interoperabile tra sistemi diversi. ASN.1 viene spesso utilizzato in crit

BIO (I/O bufferizzato) è una str

Se stess`d2i_`(decodifica da formato DER in una struttura dati) e `i2d_`(codifica da una struttura dati in formato DER). Queste funzioni

```
BIO *bio = BIO_new(BIO_s_mem()); // Crea un oggetto BIO in memoria
BIO_write(bio, dati_ASN1, lunghezza_dei_dati); // Scrivi i dati ASN.1 nell'oggetto BIO

// Decodifica i dati ASN.1 dall'oggetto BIO in una struttura dati
MY_ASN1_STRUCTURE *asn1_data = d2i_MY_ASN1_STRUCTURE(NULL, (const unsigned char **)&bio->ptr, BIO_number_written(bio));
```

// Ora "asn1_data" contiene i dati decodificati

// Rilascia l'oggetto BIO
BIO_free(bio);

Ricorda che il codice specifico può variare in base al tipo di dati ASN.1 che stai trattando e alle tue esigenze specifiche. Assicurati di consultare la documentazione di OpenSSL o della libreria crittografica che stai utilizzando per dettagli specifici e per garantire che la gestione dei dati ASN.1 sia sicura e conforme alle tue esigenze.

CVE-2022-4450	La funzione PEM_read_bio_ex() legge un file PEM da una BIO e analizza e decodifica il "nome"(ad esempio "CERTIFICATO"), eventuali dati di intestazione e dati di carico utile. Se la funzione ha esito positivo, gli argomenti "name_out", "header"e "data"vengono popolati con puntatori ai buffer contenenti i relativi dati decodificati. Il chiamante è responsabile della liberazione di tali buffer. È possibile costruire un file PEM che risulti in 0 byte di dati di payload. In questo caso PEM_read_bio_ex() restituirà un codice di errore ma popolerà l'argomento dell'intestazione con un puntatore a un buffer che è già stato liberato. Se il chiamante libera anche questo buffer, si verificherà un doppio rilascio. Ciò molto probabilmente porterà a un incidente. Questo potrebbe essere sfruttato da un utente malintenzionato che ha la capacità di fornire file PEM dannosi da analizzare per ottenere un attacco di negazione del servizio. Le funzioni PEM_read_bio() e PEM_read() sono semplici wrapper attorno a PEM_read_bio_ex() e quindi anche queste funzioni sono direttamente interessate. Queste funzioni vengono anche chiamate indirettamente da una serie di altre funzioni OpenSSL tra cui PEM_X509_INFO_read_bio_ex() e SSL_CTX_use_serverinfo_file(), anch'esse vulnerabili. Alcuni usi interni OpenSSL di queste funzioni non sono vulnerabili perché il chiamante non libera l'argomento dell'intestazione se PEM_read_bio_ex() restituisce un codice di errore. Queste posizioni includono le funzioni PEM_read_bio_TYPE() nonché i decodificatori introdotti in OpenSSL 3.0. Anche l'applicazione della riga di comando OpenSSL asn1parse è interessata da questo problema

Un file PEM (Privacy-Enhanced Mail) di solito è costituito da un'intestazione e da un corpo dati, entrambi codificati in base64. L'intestazione specifica il tipo di dati contenuti e altre informazioni relative al file. Mentre è possibile creare un file PEM con un corpo dati vuoto, l'intestazione non sarà mai vuota, in quanto deve specificare il tipo

Un file PEM ha generalmente un formato simile a questo:

-----BEGIN TIPO_DATI-----

Dati_codificati_in_base64

-----END TIPO_DATI-----

Dove "TIPO_DATI" è un'etichetta che identifica il tipo di dati, e "Dati_codificati_in_base64" è la rappresentazione codificata in base64 dei dati effettivi.

Ecco un esempio di un file PEM con un corpo dati vuoto:

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

In questo caso, l'intestazione specifica che

In breve, puoi avere un file PEM con un corpo dati vuoto, ma l'intestazione deve essere presente e specificare il tipo di dati. Un file PEM contenente solo dati di intestazione e nessun payload non avrebbe molto significato pratico, ma è certamente possibile crearlo seguendo il formato specifico di PEM.

CVE-2022-4304	Nell'implementazione OpenSSL RSA Decryption esiste un canale laterale basato sui tempi che potrebbe essere sufficiente per recuperare un testo in chiaro attraverso una rete in un attacco in stile Bleichenbacher. Per ottenere una decrittazione efficace, un utente malintenzionato dovrebbe essere in grado di inviare un numero molto elevato di messaggi di prova da decrittografare. La vulnerabilità colpisce tutte le modalità di riempimento RSA: PKCS#1 v1.5, RSA-OEAP e RSASVE. Ad esempio, in una connessione TLS, RSA viene comunemente utilizzato da un client per inviare un segreto pre-master crittografato al server. Un utente malintenzionato che avesse osservato una connessione autentica tra un client e un server potrebbe sfruttare questa falla per inviare messaggi di prova al server e registrare il tempo impiegato per elaborarli. Dopo un numero sufficientemente elevato di messaggi, l'aggressore potrebbe recuperare il segreto pre-master utilizzato per la connessione originale e quindi essere in grado di decrittografare i dati dell'applicazione inviati su quella connessione

Un attacco di tipo Bleichenbacher è un attacco laterale basato

Per proteggere un sistema dall'attacco di Bleichenb

- **Padding sicuro** : Assicurati di utilizzare una forma di pad crittografico sicuro con le operazioni RSA, come OAEP (Optimal Asymmetric Encryption Padding) o PKCS#1 v1.5 pad. Un riempimento sicuro aiuta a nascondere le informazioni temporali che possono essere sfruttate dagli attaccanti.
- **Costanti di tempo** : Assicurati che le operazioni di crittografia e decrittazione hanno costi temporali costanti, indipendentemente dai dati in input. Utilizza funzioni crittografiche ottimizzate per evitare il rilevamento di variazioni nei tempi di esecuzione.
- **Limita i tentativi di errore** : Limita il numero di tentativi di errore in una sessione di crittografia. Ad esempio, in caso di errore, ritarda la risposta o blocca la sessione per un periodo fisso. Ciò renderebbe difficile per un attaccante eseguire numerosi giochi
- **Monitoraggio e registrazione** : Monitora e registra le attività sospette oi tentativi di attacco a canale laterale. Il rilevamento precoce può aiutare a mitigare gli attacchi
- **Patch e aggiornamenti** : Mantieni aggiornato il software e le librerie crittografiche per beneficiare di correzioni e miglioramenti relativi alla sicurezza.

- **Validazione dei certificati** : Assicurati che i certificati
- **Riduci l'esposizione dei dati di tempo** :

Queste misure di sicurezza possono contribuire a mitigare il rischio di attacchi di tipo Bleichenbacher e altri attacchi a canale laterale. Tuttavia, la protezione da tali attacchi è ricca

Rigenerare

CVE-2022-2097	**5.0**La modalità AES OCB per piattaforme x86 a 32 bit che utilizzano l'implementazione ottimizzata dell'assembly AES-NI non crittograferà la totalità dei dati in alcune circostanze. Ciò potrebbe rivelare sedici byte di dati preesistenti nella memoria che non sono stati scritti. Nel caso speciale della crittografia "sul posto", verrebbero rivelati sedici byte del testo in chiaro. Poiché OpenSSL non supporta le suite di crittografia basate su OCB per TLS e DTLS, entrambe non sono interessate. Risolto il problema in OpenSSL 3.0.5 (interessato da 3.0.0-3.0.4). Risolto il problema in OpenSSL 1.1.1q (interessato da 1.1.1-1.1.1p).

La modalità OCB (Offset Codebook Mode) è una modalità di crittografia che offre autenticazione e cifratura allo stesso tempo. Tuttavia, il tuo messaggio sembra indicare che c'è un problema con l'implementazione di questa modalità AES-OCB su piattaforme x86 a 32 bit che utilizzano istruzioni ottimizzate AES-NI.

L'implementazione crittografica di OCB richiede attenzione ai dettagli per garantire che sia sicura e funzionante correttamente. Le implementazioni crittografiche possono essere complesse, e pic

Per risolvere un problema di questo tipo, è fondamentale che gli sviluppatori correggano l'implementazione crittografica in modo che i crittografi l'intero messaggio in tutte le circostanze. Inoltre, dovrebbe essere condotta un'attenta revisione e analisi delle implementazioni crittografiche per identificare e correggere eventuali debolezze o potenziali debolezze.

La soluzione a questo problema coinvolge la correzione

Se sei un utente di un software che utilizza questa implementazione crittografica, ti consiglio di cercare aggiornamenti o patch forniti dagli sviluppatori per risolvere questa ricerca. La correzione dovrebbe essere inclusa nelle versioni più recenti del software. Assicurati di mantenere il tuo software crittografico aggiornato per garantire la sicurezza del sistema.

In generale, le vulnerabilità della sicurezza nelle implementazioni crittografiche sono preoccupanti e richiedono interventi tempestivi per proteggere i dati sensibili. Inoltre, è consigliabile seguire le migliori pratiche di sicurezza crittografica e assicurarsi che tutte le implementazioni crittografiche siano sottoposte a una rigorosa revisione per identificare e risolvere potenziali problemi di sicurezza.