



UNIVERSITA' DEGLI STUDI DI CAGLIARI
FACOLTÀ DI SCIENZE
Corso di Laurea in Informatica

Analisi della libreria crittografica PyCryptoDome

Docente di riferimento

Prof. Massimo Bartoletti

Candidato

Michele Melis (matr.65798)

ANNO ACCADEMICO 2022-2023



Indice

- Introduzione
- Algoritmi di hashing
- Schemi di cifratura a chiave privata
- Schemi di cifratura a chiave pubblica
- Schemi di firma digitale
- Protocolli di condivisione e funzioni di derivazione chiavi
- Conclusioni



Introduzione

L'elaborato pone il focus sull'analisi e l'utilizzo delle primitive crittografiche implementate dalla libreria **PyCryptoDome**.

PyCryptoDome è suddivisa in otto package e, per ogni package, viene implementato un tipo diverso di primitiva.

I package che costituiscono la libreria sono i seguenti:

- **Hash**: algoritmi di hashing e schemi di autenticazione;
- **Random**: funzioni per la generazione di numeri pseudo-casuali;
- **IO**: formati di memorizzazione chiavi;
- **Cipher**: schemi di cifratura a chiave privata;
- **PublicKey**: schemi di cifratura a chiave pubblica;
- **DigitalSignature**: schemi di firma digitale;
- **Protocol**: funzioni di derivazione chiavi e protocolli di condivisione;
- **Util**: funzioni accessorie per lo svolgimento di operazioni secondarie.



Algoritmi di hashing

Primitive implementate nel package: **SHA-2, SHA-3, BLAKE2, HMAC, CMAC e Poly1305**.

Alcuni casi d'uso degli algoritmi di hashing e degli schemi di autenticazione sono: garantire la sicurezza di una password, effettuare un controllo sull'integrità dei dati e generare un codice per l'autenticazione di un messaggio.

Test effettuati con gli algoritmi di hashing: confronto con il modulo hashlib sull'efficienza delle implementazioni delle primitive MD5, SHA256 e SHA512, utilizzando input di 100 MB e confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.

Test effettuato con lo schema di autenticazione HMAC: confronto con il modulo hmac sull'efficienza dell'implementazione della primitiva, utilizzando input di 100 MB e chiavi da 16 B, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.



Risultati dei test sulle primitive di hashing:

MD5			SHA256			SHA512		
	PyCryptoDome	hashlib		PyCryptoDome	hashlib		PyCryptoDome	hashlib
Media	0,152104955	0,117522817	Media	0,325290117	0,047640123	Media	0,224108052	0,108952003
Varianza	4,88481E-06	8,03577E-06	Varianza	5,90451E-06	2,32721E-07	Varianza	3,17389E-06	1,00749E-06
Osservazioni	100	100	Osservazioni	100	100	Osservazioni	100	100
Varianza complessiva	6,46029E-06		Varianza complessiva	3,06862E-06		Varianza complessiva	2,09069E-06	
Differenza ipotizzata per le medie	0		Differenza ipotizzata per le medie	0		Differenza ipotizzata per le medie	0	
gdl	198		gdl	198		gdl	198	
Stat t	96,20792742		Stat t	1120,756572		Stat t	563,1535117	
P(T<=t) una coda	1,7404E-168		P(T<=t) una coda	0		P(T<=t) una coda	0	
t critico una coda	2,345328349		t critico una coda	2,345328349		t critico una coda	2,345328349	
P(T<=t) due code	3,4808E-168		P(T<=t) due code	0		P(T<=t) due code	0	
t critico due code	2,600887278		t critico due code	2,600887278		t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore del modulo hashlib, con una differenza di circa 0,034s per la primitiva MD5, 0,277s per la primitiva SHA256 e 0,115s per la primitiva SHA512.



Risultati del test sulla primitiva HMAC:

HMAC	PyCryptoDome	hmac
Media	0,328573487	0,047934706
Varianza	4,8147E-05	3,96394E-06
Osservazioni	100	100
Varianza complessiva	2,60555E-05	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	388,7613229	
$P(T \leq t)$ una coda	1,0201E-287	
t critico una coda	2,345328349	
$P(T \leq t)$ due code	2,0401E-287	
t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore del modulo hmac, con una differenza di circa 0,281s.



Schemi di cifratura a chiave privata

Primitive implementate nel package: **Salsa20**, **ChaCha20**, **XChaCha20**, **AES**, **PKCS#1-OAEP**.

Alcuni casi d'uso degli schemi di cifratura a chiave privata sono: cifratura messaggio testuale, cifratura file e cifratura con codice di autenticazione.

Test effettuati:

- confronto con il modulo salsa20 sull'efficienza dell'implementazione della primitiva Salsa20 per la cifratura di un messaggio, utilizzando input di 100 MB e chiavi da 32 B, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.
- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva AES in modalità CBC per la cifratura di un messaggio, utilizzando input di 100 MB e chiavi da 16 B, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.
- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva ChaCha20-Poly1305 per la cifratura di un messaggio, utilizzando input di 100 MB e chiavi da 32 B, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.



Risultati dei test:

Salsa20	PyCryptoDome	salsa20	AES (CBC)	PyCryptoDome	Cryptography	ChaCha20-Poly1305	PyCryptoDome	Cryptography
Media	0,429594457	0,242384276	Media	0,163755975	0,138150549	Media	0,334090202	0,092152092
Varianza	7,64953E-05	4,14814E-05	Varianza	8,06165E-05	0,000139885	Varianza	1,70897E-06	5,52953E-06
Osservazioni	100	100	Osservazioni	100	100	Osservazioni	100	100
Varianza complessiva	5,89883E-05		Varianza complessiva	0,000110251		Varianza complessiva	3,61925E-06	
Differenza ipotizzata per le medie	0		Differenza ipotizzata per le medie	0		Differenza ipotizzata per le medie	0	
gdl	198		gdl	198		gdl	198	
Stat t	172,3579766		Stat t	17,24352319		Stat t	899,2491945	
P(T<=t) una coda	5,3001E-218		P(T<=t) una coda	1,36746E-41		P(T<=t) una coda	0	
t critico una coda	2,345328349		t critico una coda	2,345328349		t critico una coda	2,345328349	
P(T<=t) due code	1,06E-217		P(T<=t) due code	2,73493E-41		P(T<=t) due code	0	
t critico due code	2,600887278		t critico due code	2,600887278		t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore del modulo salsa20 e della libreria Cryptography, con una differenza di circa 0,188s per la primitiva Salsa20, 0,026s per la primitiva AES in modalità CBC e 0,242s per la primitiva ChaCha20-Poly1305.



Schemi di cifratura a chiave pubblica

La funzione del package è unicamente quella di generare la coppia di chiavi pubblica/privata. Il compito di cifrare e decifrare viene demandato al package Cipher.

Primitive implementate nel package: **RSA, DSA, ECC e ElGamal.**

Test effettuati:

- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva RSA per la generazione di una chiave pubblica di 256 B e confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.
- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva ECC per la generazione di una chiave pubblica di 256 b (NIST P-256) e confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.



Risultati dei test:

RSA	PyCryptoDome	Cryptography
Media	0,817756367	0,048573773
Varianza	0,318989591	0,000763744
Osservazioni	100	100
Varianza complessiva	0,159876667	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	13,60259936	
P(T<=t) una coda	1,72766E-30	
t critico una coda	2,345328349	
P(T<=t) due code	3,45532E-30	
t critico due code	2,600887278	

ECC	PyCryptoDome	Cryptography
Media	0,000318367	0,000872552
Varianza	2,17852E-07	6,11818E-05
Osservazioni	100	100
Varianza complessiva	3,06998E-05	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	-0,707248158	
P(T<=t) una coda	0,240121923	
t critico una coda	2,345328349	
P(T<=t) due code	0,480243845	
t critico due code	2,600887278	

Osservazioni:

Il risultato del test sulla primitiva RSA suggerisce una differenza significativa tra le medie dei tempi di esecuzione, a favore della libreria Cryptography, con una differenza di circa 0,769s; il risultato del test sulla primitiva ECC non rivela invece una differenza significativa tra le medie dei tempi di esecuzione, mostrando una differenza di circa 0,0006s.



Schemi di firma digitale

Primitive implementate nel package: **PKCS#1 v1.5**, **PKCS#1 PSS**, **DSA**, **EdDSA** e **ECDSA**.

Alcuni casi d'uso degli schemi di firma digitale sono: firma di un documento file e sistema di autorizzazione utente.

Test effettuato con lo schema di firma digitale DSS: confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva, seguendo lo standard FIPS-186-3, con chiave ECC NIST P-256 e algoritmo di hashing SHA256, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.



Risultati del test sulla primitiva DSS:

DSS	PyCryptoDome	Cryptography
Media	0,000340095	4,9994E-05
Varianza	2,26793E-07	4,79683E-08
Osservazioni	100	100
Varianza complessiva	1,37381E-07	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	5,534401839	
$P(T \leq t)$ una coda	4,91139E-08	
t critico una coda	2,345328349	
$P(T \leq t)$ due code	9,82277E-08	
t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore della libreria Cryptography, con una differenza di circa 0,0003s.



Protocolli di condivisione e funzioni di derivazione chiavi

Primitive implementate nel package: **Shamir's secret sharing**, **Diffie-Hellman** e le funzioni di derivazione chiave: **PBKDF1**, **PBKDF2**, **scrypt**, **bcrypt**, **HKDF** e **SP 800-180**.

Alcuni casi d'uso delle funzioni di derivazione chiave e dei protocolli di condivisione sono: garantire la sicurezza a dati sensibili, permettere l'accesso ad un messaggio condiviso tra più utenti e condividere una chiave di sessione tra mittente e destinatario.

Test effettuati:

- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva scrypt per la generazione di una chiave di 32 B, utilizzando input di 100 MB, un sale di 16 B e la tupla ($n = 2^{14}$, $r = 8$, $p = 1$), confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.
- confronto con la libreria Cryptography sull'efficienza dell'implementazione della primitiva DH per la generazione di una chiave condivisa, utilizzando come schema di generazione chiavi pubbliche ECC NIST P-521 e come funzione di derivazione la funzione identità, confrontando i tempi medi di esecuzione di 100 campioni tramite test t con significatività 1%.



Risultati del test sulla primitiva script:

script	PyCryptoDome	Cryptography
Media	21,72173472	0,245917394
Varianza	0,037038838	0,000170206
Osservazioni	100	100
Varianza complessiva	0,018604522	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	1113,333938	
$P(T \leq t)$ una coda	0	
t critico una coda	2,345328349	
$P(T \leq t)$ due code	0	
t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore della libreria Cryptography, con una differenza di circa 21,5s.



Risultati del test sulla primitiva DH:

DH	PyCryptoDome	Cryptography
Media	0,002129598	0,001920042
Varianza	1,1447E-07	7,42369E-08
Osservazioni	100	100
Varianza complessiva	9,43534E-08	
Differenza ipotizzata per le medie	0	
gdl	198	
Stat t	4,82398156	
P(T<=t) una coda	1,40154E-06	
t critico una coda	2,345328349	
P(T<=t) due code	2,80309E-06	
t critico due code	2,600887278	

Osservazioni:

I risultati dei test suggeriscono una differenza significativa tra le medie dei tempi di esecuzione, a favore della libreria Cryptography, con una differenza di circa 0,0002s.



Conclusioni

In conclusione, possono essere elencati i pregi e i difetti trovati durante l'analisi della libreria.

Pregi:

- Primitive: la libreria implementa un gran numero di primitive, rendendola utile per lo svolgimento diversificato di un gran numero di operazioni crittografiche (e non);
- Sintassi: la libreria offre una sintassi semplice, favorendone la sua implementazione e la successiva manutenzione del codice;
- QoL: la libreria viene mantenuta e aggiornata con regolarità, aggiungendo primitive e funzioni al passo con gli standard correnti;
- Retrocompatibilità: la libreria può essere utilizzata con standard deprecati ed è compatibile con i sistemi che utilizzando la libreria (deprecata) **PyCrypto**;

Difetti:

- Efficienza: la libreria ha dei tempi di esecuzione maggiori rispetto alla concorrenza.