LUCRARE INDIVIDUALĂ

Sistemul de criptare ED25519

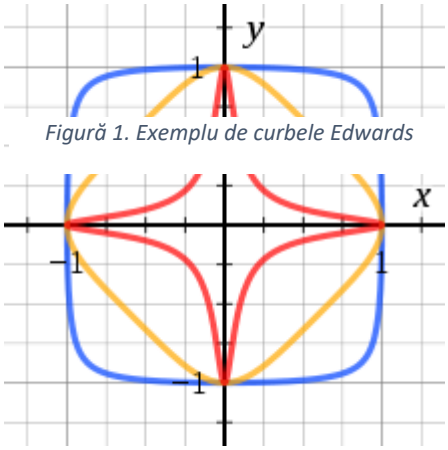
Zuza Marius

W-1942

1. Despre sistemul de criptare ED25519

Ed25519 este o schema de semnătură EdDSA, care foloseşte SHA-512 (SHA-2) şi Curve25519.

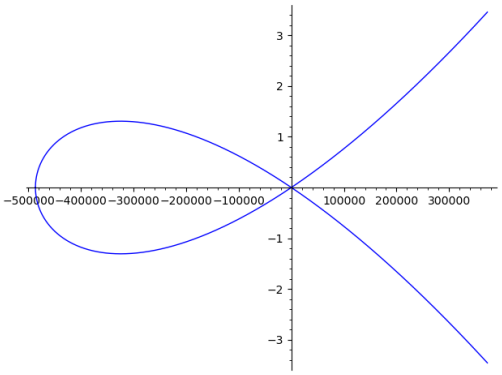
EdDSA este un algoritm de semnătură digitală care utilizează Curbe Edwards (vezi figura 1). În matematică, curbele Edwards sunt o familie de curbe eliptice studiate de Harold Edwards în 2007. Conceptul de curbe eliptice peste câmpuri finite este utilizat pe scară largă în criptografia cu curbe eliptice.



Figură 1. Exemplu de curbele Edwards

SHA-512 – o variaţie a funcţiei SHA-2, care utilizează valori cu dimensiunile de 512 biţi.

SHA-2 (Secure Hash Algorithm 2) este un set de funcţii hash criptografice concepute de Agenţia de Securitate Naţională a Statelor Unite (NSA) şi publicat pentru prima dată în 2001. Ele sunt construite folosind construcţia Merkle–Damgård, dintr-o funcţie de compresie unidirecţională în sine construită folosind structura Davies–Meyer dintr-un cifr bloc specializat.



Figură 2. Curve25519

Curve25519 este o curbă eliptică utilizată în criptografia cu curbă eliptică (ECC) care oferă 128 de biţi de securitate (dimensiunea cheii de 256 de biţi) şi concepută pentru a fi utilizată cu schema de acord de chei pentru curba eliptică Diffie–Hellman (ECDH). Este una dintre cele mai rapide curbe din ECC şi nu este acoperită de niciun brevet cunoscut.

Avantajele Ed25519:

- *Verificare rapidă cu o singură semnătură*
  Durează doar 273364 de cicluri pentru a verifica o semnătură pe liniile de procesoare Intel Nehalem/Westmere, implementate pe scară largă.

- *Semnarea foarte rapidă*
  Software-ul durează doar 87548 de cicluri pentru a semna un mesaj. Un Westmere quad-core de 2,4 GHz semnează 109000 de mesaje pe secundă.

- *Generare rapidă a cheilor*
  Generarea cheii este aproape la fel de rapidă ca semnarea.

- *Nivel ridicat de securitate*
  Acest sistem are o țintă de securitate de 2^128; spargerea acestuia are dificultăți similare cu spargerea NIST P-256, RSA cu chei de ~3000 de biți, cifruri bloc puternice de 128 de biți, etc.

- *Chei de sesiune sigure*
  Semnăturile sunt generate în mod determinist; generarea cheilor consumă o nouă aleatorie, dar noile semnături nu. Aceasta nu este doar o caracteristică de viteză, ci și o caracteristică de securitate, direct relevantă pentru recenta prăbușire a sistemului de securitate Sony PlayStation 3.

- *Rezilienţa la coliziune*
  Ciocnirile cu funcţia hash nu distrug acest sistem. Acest lucru adaugă un strat de apărare împotriva posibilităţii de slăbiciune în funcţia hash selectată.

- *Semnături mici*
  Semnăturile se încadrează în 64 de octeți. Aceste semnături sunt de fapt versiuni comprimate ale semnăturilor mai lungi; timpii pentru compresie și decompresie sunt incluși în numărătoarea ciclului raportată mai sus.

- *Chei mici*
  Cheile publice consumă doar 32 de octeți. Timpii de compresie și decompresie sunt din nou incluși.

Slăbiciunea:

Punctul slab descoperit se referă la unele implementări care creează chei publice precalculate și care accelerează funcționarea acestora. Astfel, atunci când un utilizator semnează pentru o tranzacție, în mod normal am accesa cheia privată de

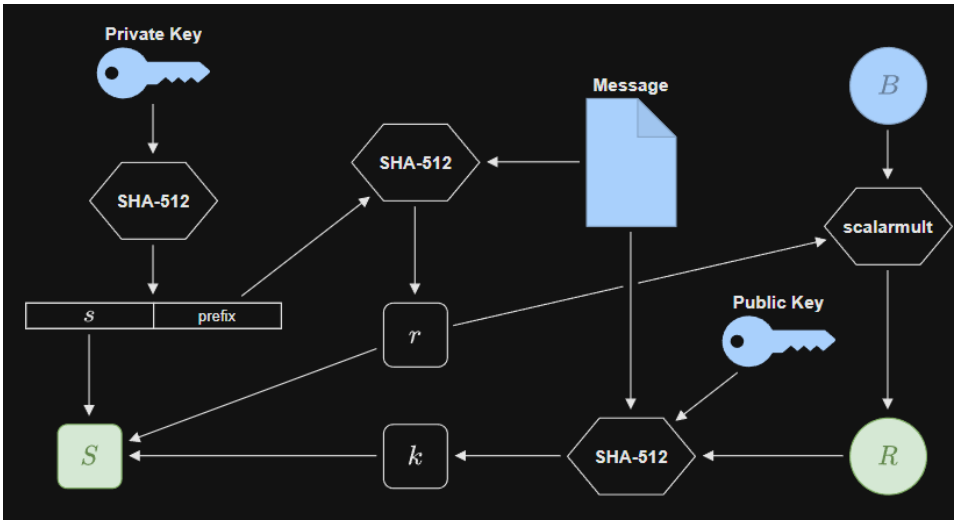
două ori: pentru a semna tranzacția; și să genereze cheia publică. Dar unele biblioteci nu verifică dacă cheia publică se referă de fapt la o cheie corespunzătoare și, prin urmare, nu verifică semnătura. Această abordare ar putea permite unui atacator să injecteze o serie de chei publice și mesaje în metoda semnăturii și, în cele din urmă, să descopere părți ale cheii private. Trebuie amintit că, chiar dacă un bit este descoperire, se reduce securitatea la jumătate, și doi biți cu un sfert și așa mai departe.

2. Descriere

Cheile Ed25519 încep viața ca un seed binar uniform aleatoriu de 32 de octeți (256 de biți). Seed-ul este apoi „hash-uit” folosind SHA512, care vă oferă 64 de octeți (512 biți), care este apoi împărțit într-o „jumătate din stânga” (primii 32 de octeți) și o „jumătate din dreapta”. Jumătatea stângă este masată într-un scalar privat curve25519 „a” prin setarea și ștergerea câțiva biți de ordin înalt/inferior. Cheia publică este generată prin înmulțirea acestui scalar secret cu „B” (generatorul), care dă un element de grup de 32 de octeți/256 de biți „A”.

Când se fac semnături, rezultă două valori: R și S (ambele de 32 de octeți, deci semnătura totală este de 64 de octeți). R depinde de jumătatea dreaptă a seed-ului extins și de mesaj. (În DSA tradițional, R este generat aleatoriu, iar securitatea cheii private depinde de calitatea acelei aleatorii, ceea ce duce la unele eșecuri importante). S este adevărata componentă a semnăturii și este o funcție a tuturor (inclusiv a pubkey-ului).

Procesul este descris în schema de mai jos:



Figură 3. Fluxul de date în procedura de semnare a Ed25519. Intrările sunt colorate în albastru, iar ieșirile sunt colorate în verde. Punctele curbei eliptice (și codificările lor) sunt desenate ca discuri, în timp ce scalarii sunt dreptunghiuri cu colțuri rotunjite.

3. Exemplu

Crearea cheii publice:

$pub=H(priv)[:32]\cdot B$

pub – cheia publică

priv – cheia privată

B - punctul de bază al curbei

H() - este funcţia de criptare (SHA-512)

Pentru a calcula semnătura, generăm:

$r=H\_int(H(priv)[32:]||M)(mod\ l)$

H_int() - o funcţie hash (SHA-512) şi convertită într-o valoare întreagă.

În acest caz, luăm cei 32 de octeţi superiori pentru calcul. Valoarea lui l este de ordinul curbei. Atunci M este valoarea octetului mesajului. Cu „ ||” ataşăm octeţii împreună. Apoi convertim acest lucru într-un punct de pe curbă cu:

$R=r\cdot B$

În continuare, calculăm:

$h=Hint(R||pub||M)(mod\ l)$

şi:

$s=H\_int(priv))[:32]$

$S=(r+h\cdot s)(modl)$

Semnătura este atunci (R,S). Bob îi trimite lui Alice M, R, S şi pub. Apoi Alice calculează:

$k=H(R||pub||M)(mod\ l)$

$P1=S\cdot B$

$P2=R+k\cdot pub$

iar dacă P1 este egal cu P2, semnătura se verifică. Acest lucru funcţionează deoarece:

$P1=S\cdot B=(r+h\cdot s(mod\ l))\cdot B=r\cdot B+h\cdot s\cdot B=R+h\cdot pub=P2$

4. Implementări

Codul în python cu implementarea Ed25519

```
import ed25519

privKey, pubKey = ed25519.create_keypair()
print("Private key (32 bytes):",
privKey.to_ascii(encoding='hex'))
print("Public key (32 bytes): ",
pubKey.to_ascii(encoding='hex'))

msg = b'Message for Ed25519 signing'
signature = privKey.sign(msg, encoding='hex')
print("Signature (64 bytes):", signature)

try:
  pubKey.verify(signature, msg, encoding='hex')
  print("The signature is valid.")
except:
  print("Invalid signature!")
```

Output

```
Private key (32 bytes):
b'1498b5467a63dffa2dc9d9e069caf075d16fc33fdd4c3b01bfadae6433767
d93'
Public key (32 bytes):
b'b7a3c12dc0c8c748ab07525b701122b88bd78f600c76342d27f25e5f92444
cde'
Signature (64 bytes):
b'6dd355667fae4eb43c6e0ab92e870edb2de0a88cae12dbd8591507f584fe4
912babff497f1b8edf9567d2483d54ddc6459bea7855281b7a246a609e3001a
4e08'
The signature is valid.
```

Utilizările ale Ed25519 includ OpenSSH, GnuPG și diverse alternative, precum și instrumentul signify de la OpenBSD.

- Apple Watch și iPhone folosesc cheile Ed25519 pentru autentificarea reciprocă IKEv2
- Botan
- Cardano
- Protocolul criptomonedei CryptoNote
- Dropbear SSH

- Implementarea I2Pd a EdDSA
- Kit de dezvoltare Java 15
- Libgcrypt
- Minisign și Minisign Diverse pentru macOS
- NaCl / libsodium
- OpenSSL 1.1.1
- Python
- Implementare de referință Supercop (limbaj C cu inline assembler)
- wolfSSL