



PARSEC

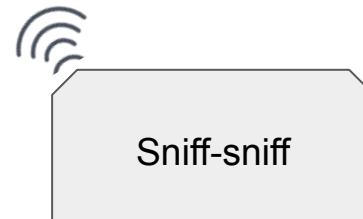
Password Authentication
using Response Signed
with Elliptic Curve

Nikita Maliavin

DIY: authentication plugin



→passwd or hash X

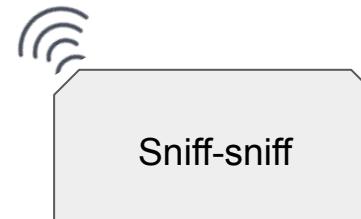


DIY: authentication plugin



→passwd or hash **X**

→nonce+hash(passwd+nonce)



DIY: authentication plugin



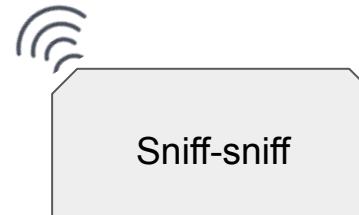
→ nonce+hash(passwd+nonce+scramble)

= old_password,

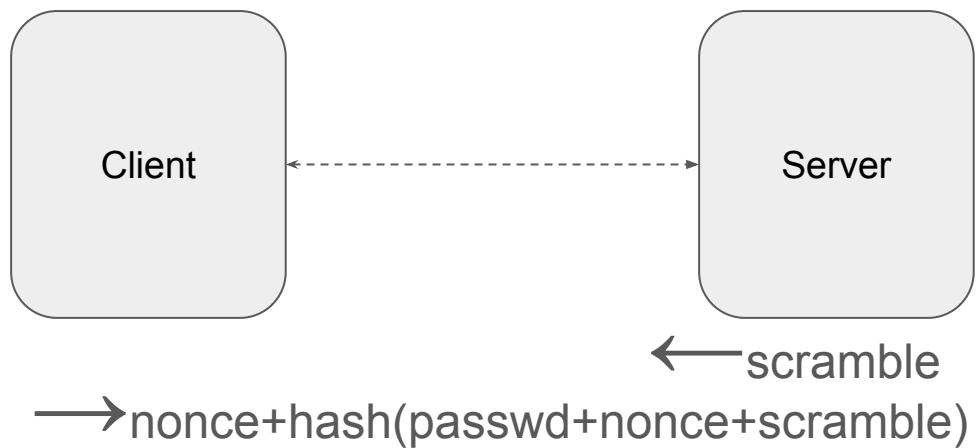
hash = ???

= native_password,

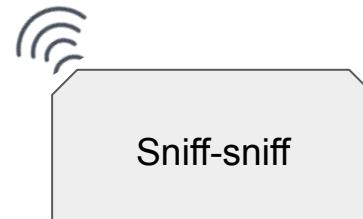
hash = sha1(sha1)



DIY: authentication plugin



⚠️ The password table can be stolen
🔧 Store:
salt+hash(passwd+salt)

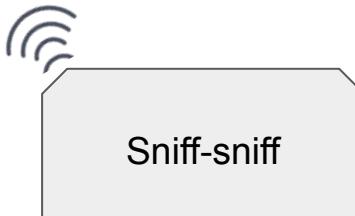


DIY: authentication plugin



The password table can be stolen
 Store:
salt+hash(passwd+salt)

$H = \text{hash}(\text{passwd} + \text{salt})$
→ nonce+hash(H+nonce+scramble)



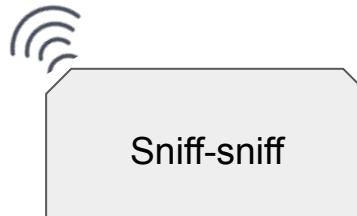
DIY: authentication plugin



The password table can be stolen
Store:
salt+hash(passwd+salt)
Regenerate **salt** if stolen



$H = \text{hash}(\text{passwd} + \text{salt})$
→ nonce + hash(H + nonce + scramble)



Asymmetric signatures

- We want to sign the message m and check the result.
- We'll generate the key pair: $K_{\text{priv}}, K_{\text{pub}}$

$\text{sgn} = \text{Sign}(m, K_{\text{priv}})$

$\text{signed?} = \text{Check}(m, K_{\text{pub}})$

- Can we build such pair of functions, Sign and Check?

Asymmetric signatures at home

- Let **E** and **D** be the asymmetric encryption and decryption functions

$$\text{Sign}(m, K_{\text{priv}}) := m + \mathbf{E}(\text{sha512}(m), K_{\text{priv}})$$

Check(s, K_{pub}) := begin

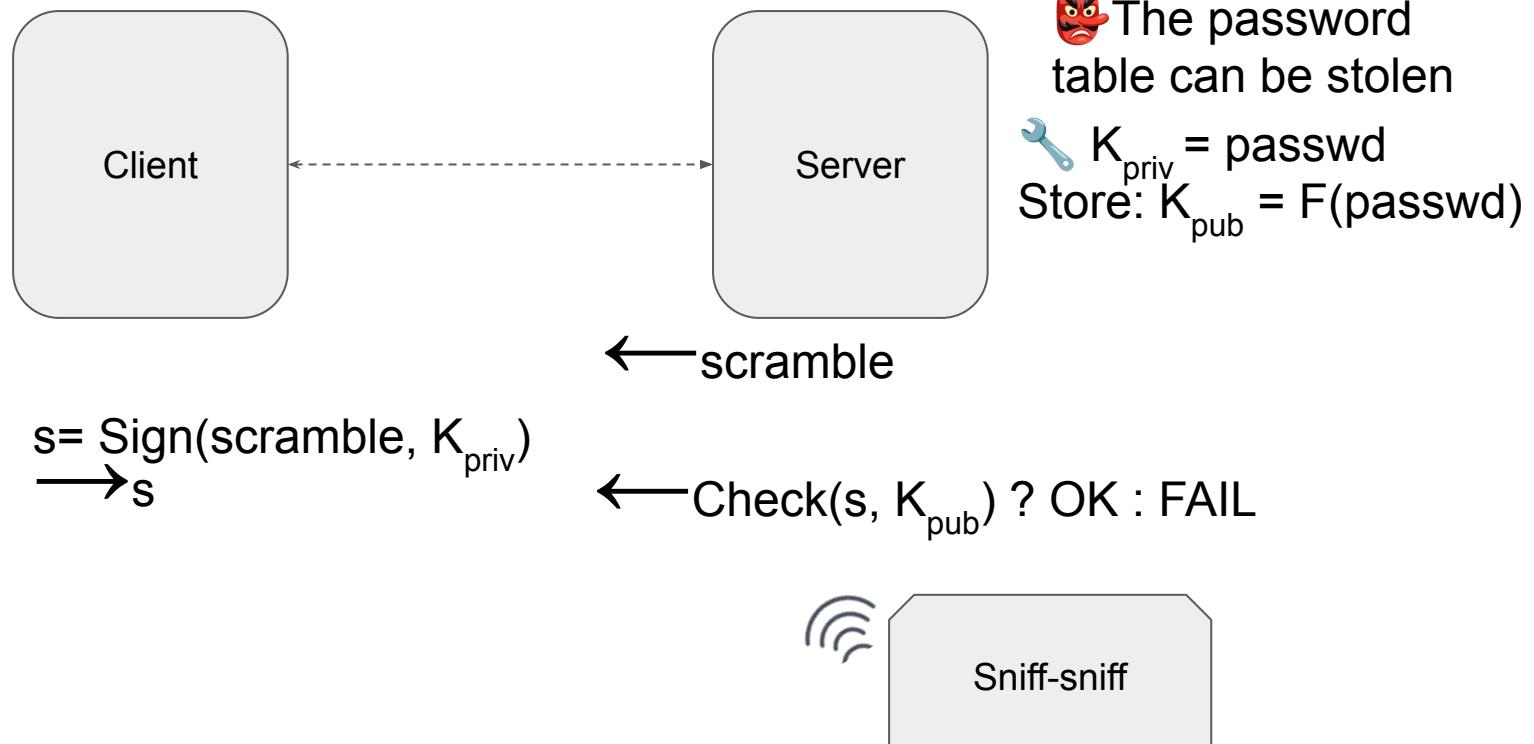
 m := s[:-64]

 h := s[-64:]

return := sha512(m) == D(h, K_{pub})

end

DIY: authentication plugin



DIY: authentication plugin



The password table can be stolen
 $K_{priv} = \text{passwd}$
Store: $K_{pub} = F(\text{passwd})$

← scramble

→ $\text{Sign}(\text{scramble}, K_{priv})$

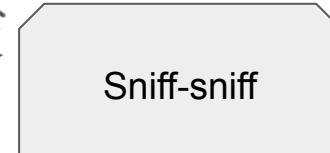
← $\text{Check}(s, K_{pub}) ? \text{OK} : \text{FAIL}$

ed25519:

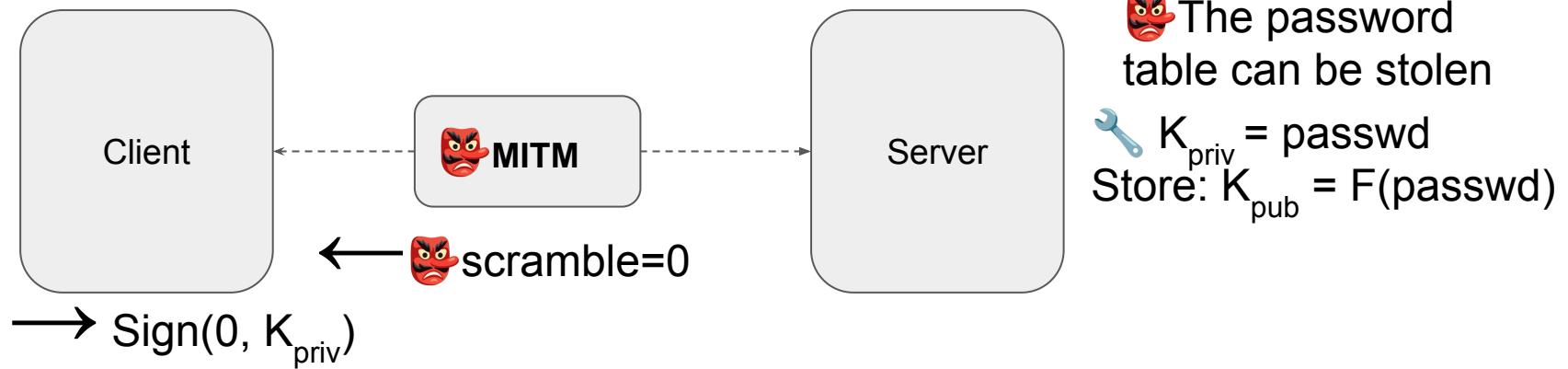
customized ed25519

no salt

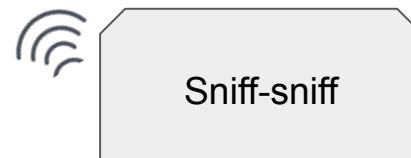
no nonce



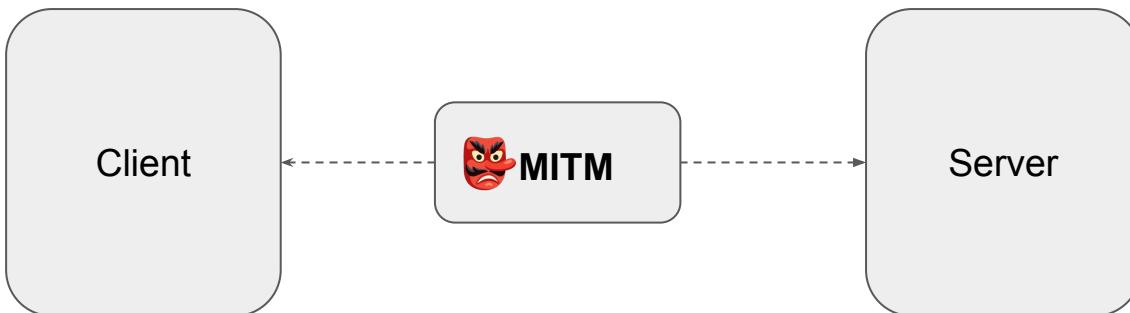
DIY: authentication plugin



❗ Precalculates $\text{Sign}(0, *)$ for every *



DIY: authentication plugin



← scramble,salt

→ nonce+Sign(s+nonce+scramble, passwd)

👹 The password table can be stolen

🔧 $K_{priv} = \text{passwd}$
Store: $K_{pub} = F(\text{passwd})$

👹 Precalculates $\text{Sign}(0, *)$ for every *



PARSEC:

✓ Uses OpenSSL, GnuTLS/nettle for ed25519,
randoms, hashes

✓ Uses PBKDF2/SHA512 for salt with > 1k
iterations

😓 can't use bcrypt/ncrypt from windows

Conforms NIST recommendations:

- 32-byte scramble
- 32-byte nonce
- 18-byte salt
- 64-byte sig, key is 32-byte

PARSEC:

ext-salt = (hash-alg, iterations, salt)

hash-alg = PBKDF2

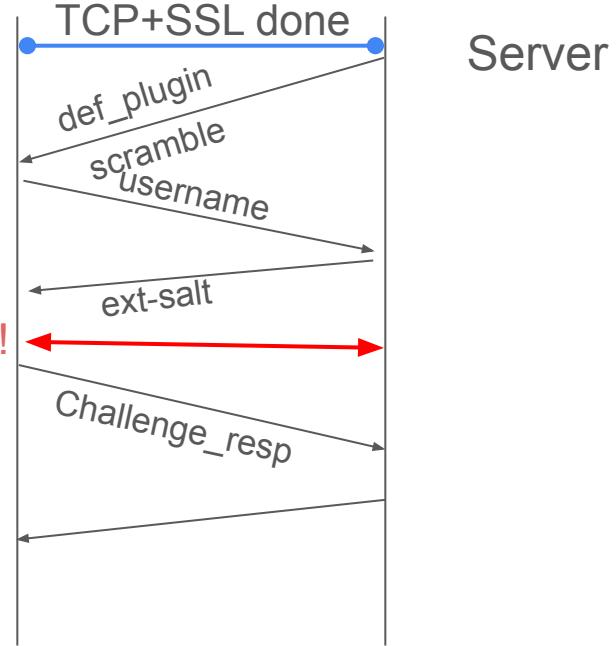
Challenge_resp = nonce +
ed25519_sign(nonce+scramble, K_{priv})

K_{priv} = PBKDF2(passwd)

Client

Server

+1 roundtrip:
change plugin!



PARSEC/Soon:

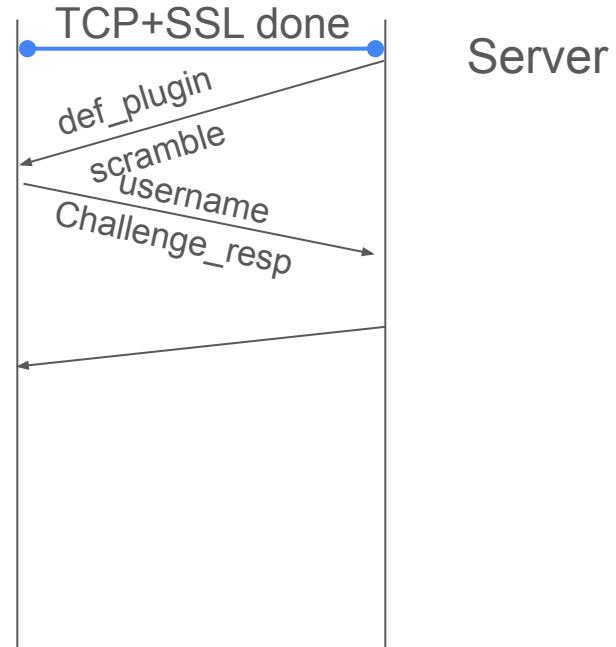
- Configurable as default (-1 roundtrip)
- Cache ext-salt or save in ini (-1 roundtrip)

Some day:

- Migration from native/old plugins
- HKDF (why not?)

Client

Server



Nikita Maliavin

Senior Software
Engineer
@MariaDB

