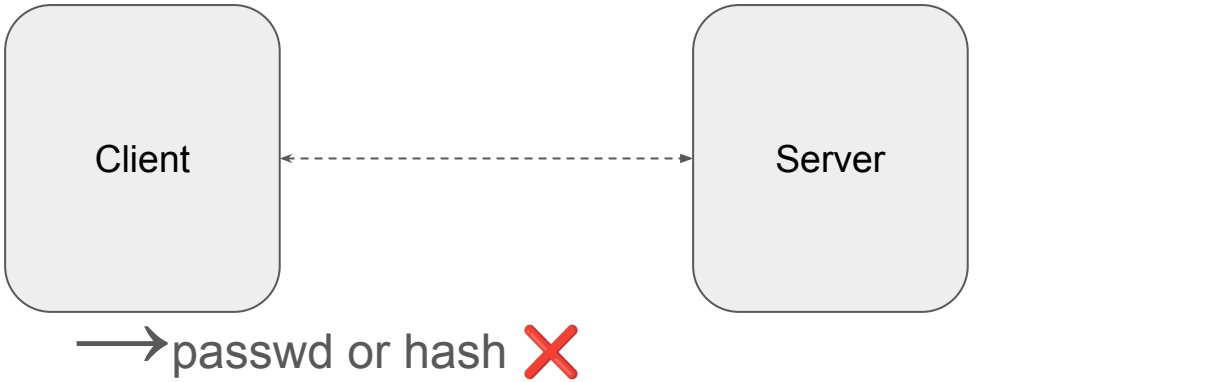
PARSEC

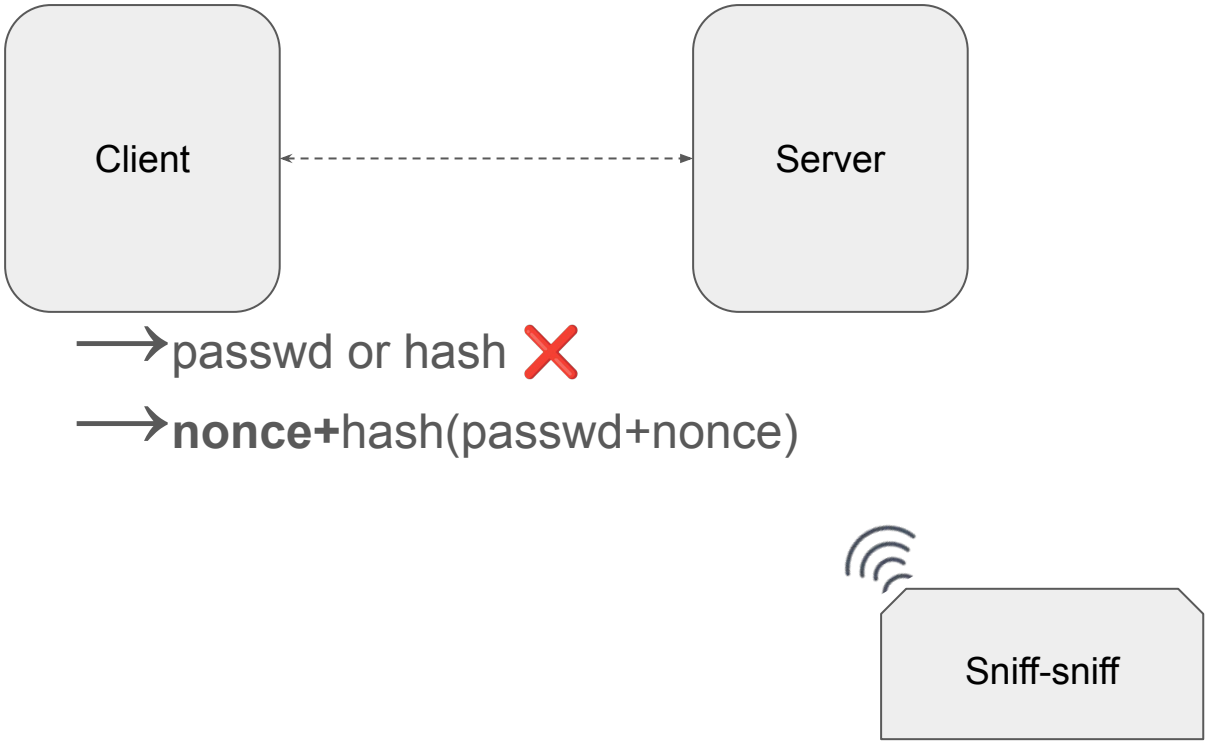
Password Authentication using Response Signed with Elliptic Curve

Nikita Maliavin

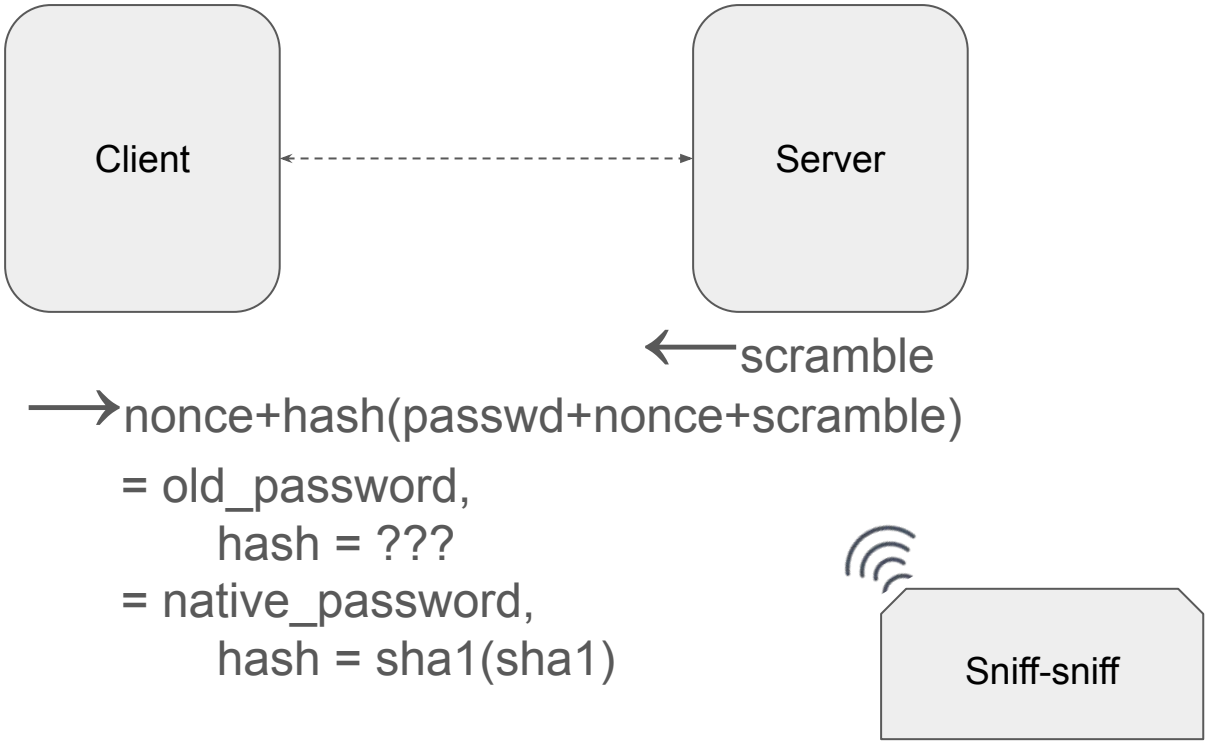
DIY: authentication plugin



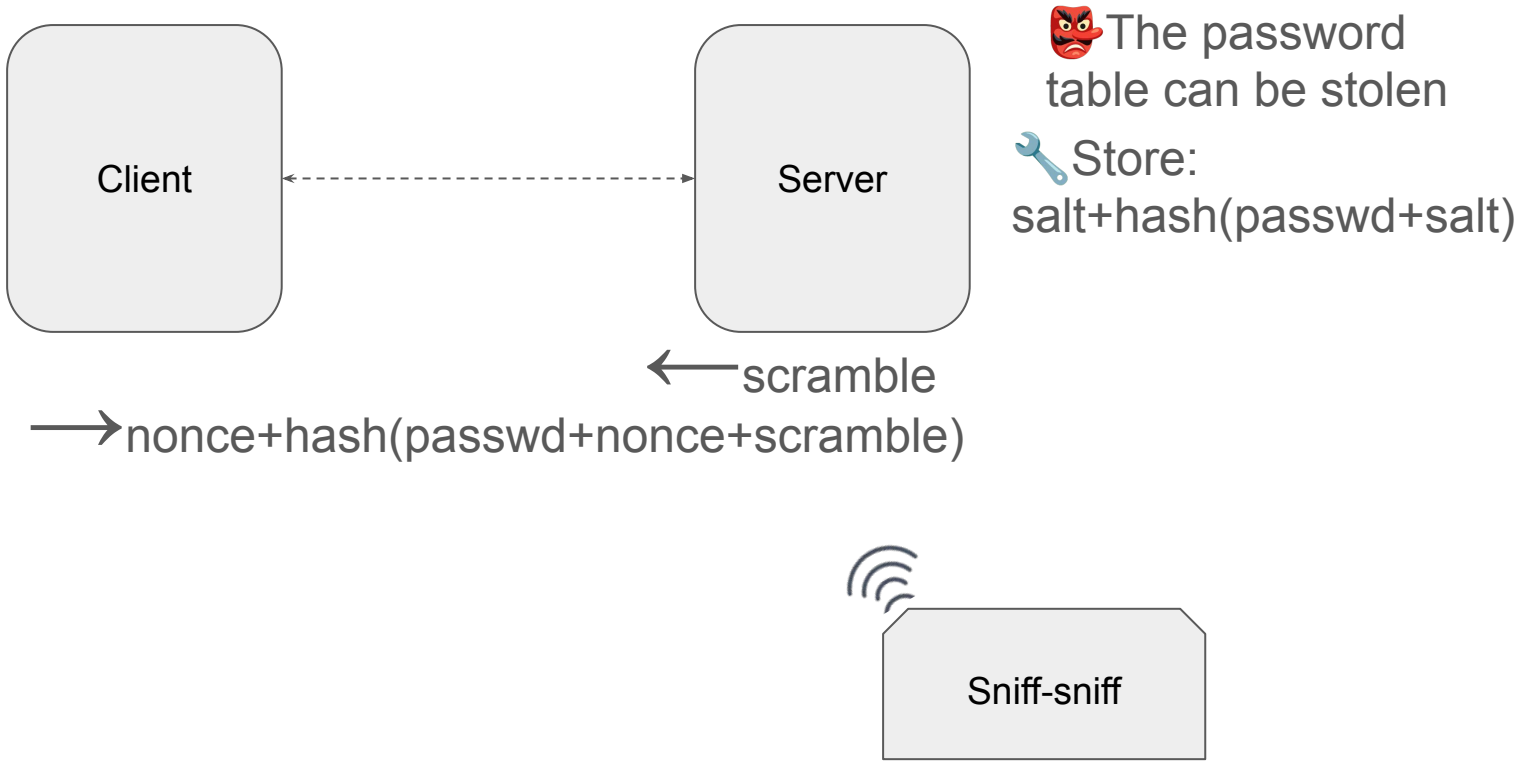
DIY: authentication plugin



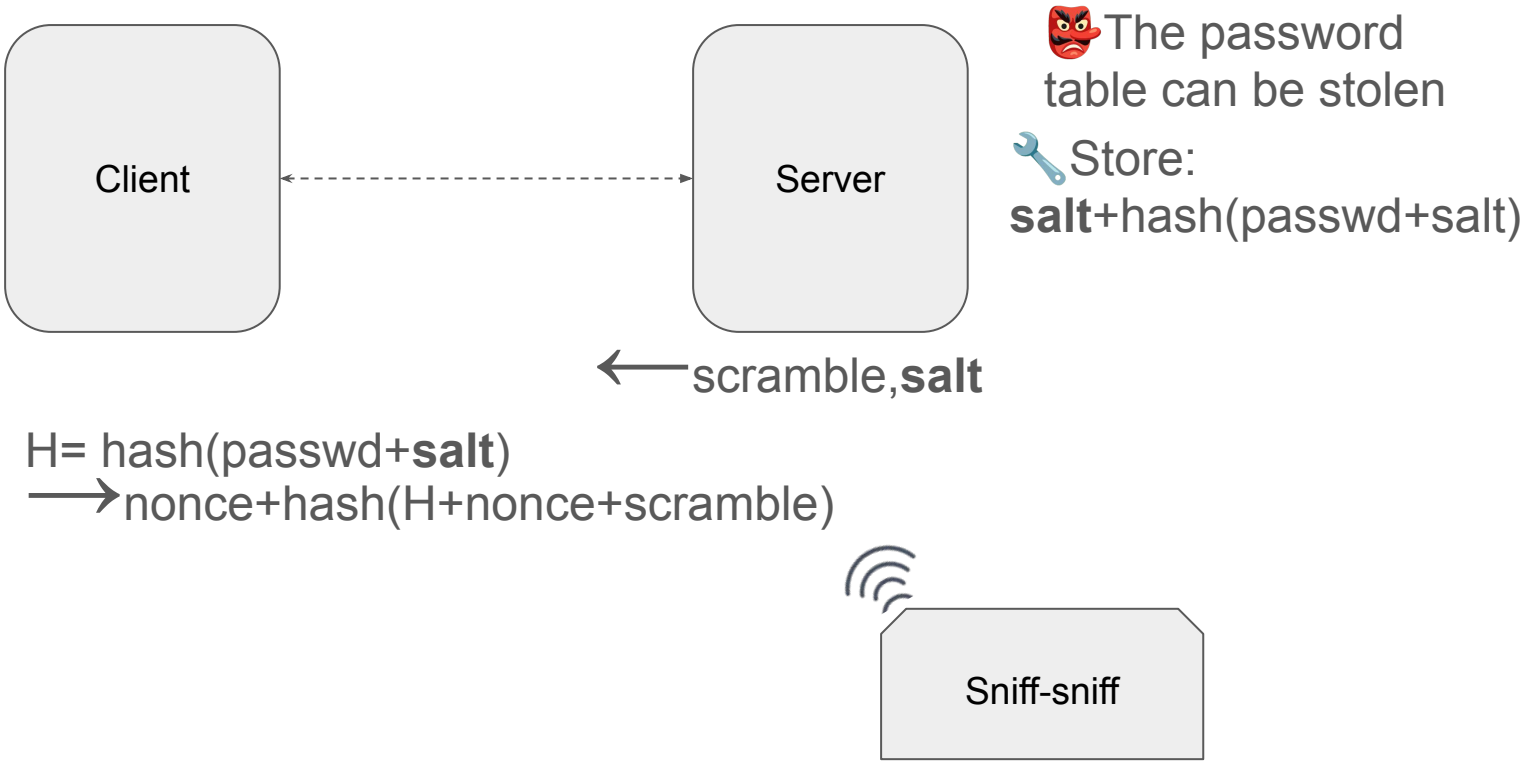
DIY: authentication plugin



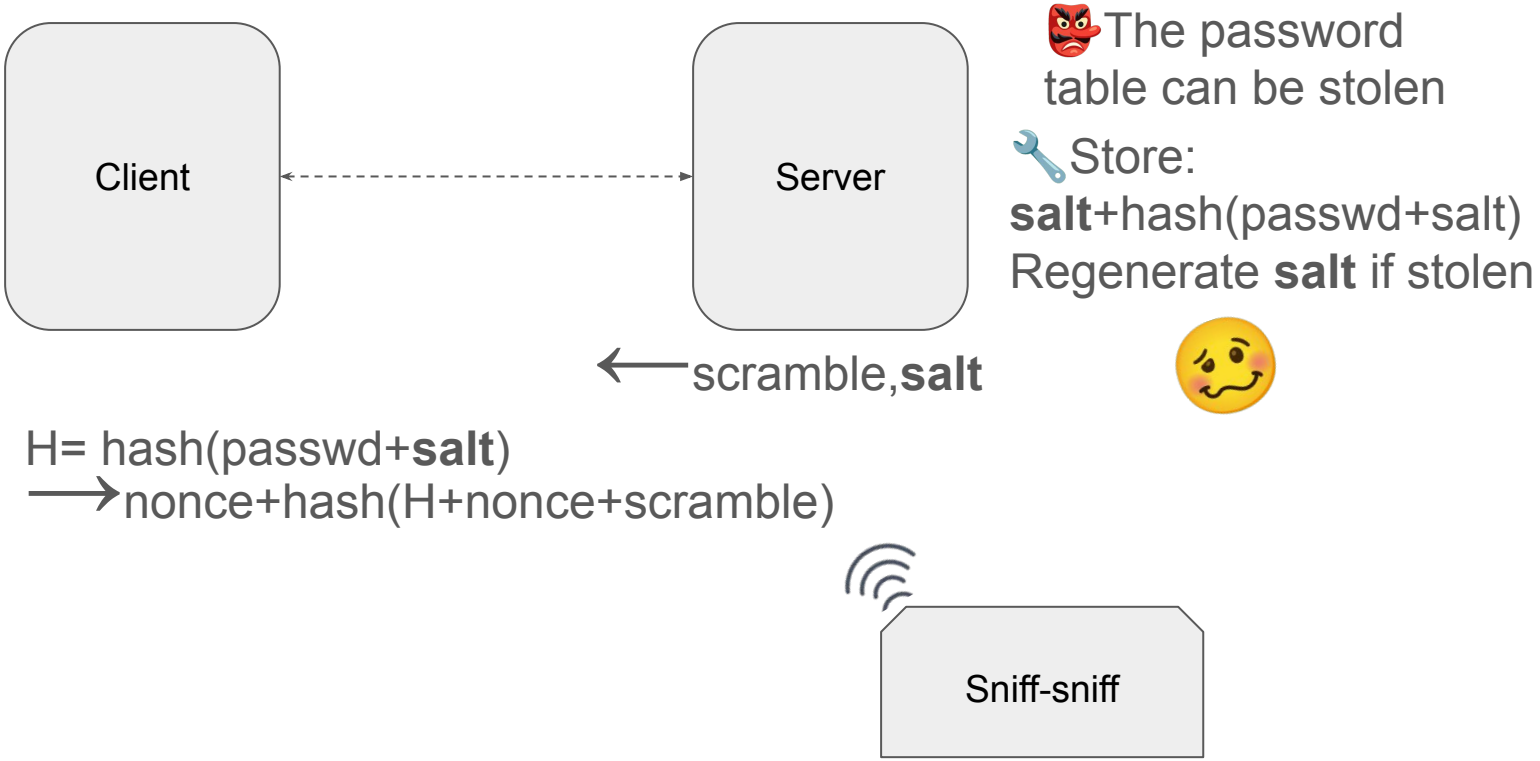
DIY: authentication plugin



DIY: authentication plugin



DIY: authentication plugin



Asymmetric signatures

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

sgn = Sign(m, K_{priv})
signed? = Check(m, K_{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

  Sign(m, K_{priv}) := m + **E**(sha512(m), K_{priv})

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

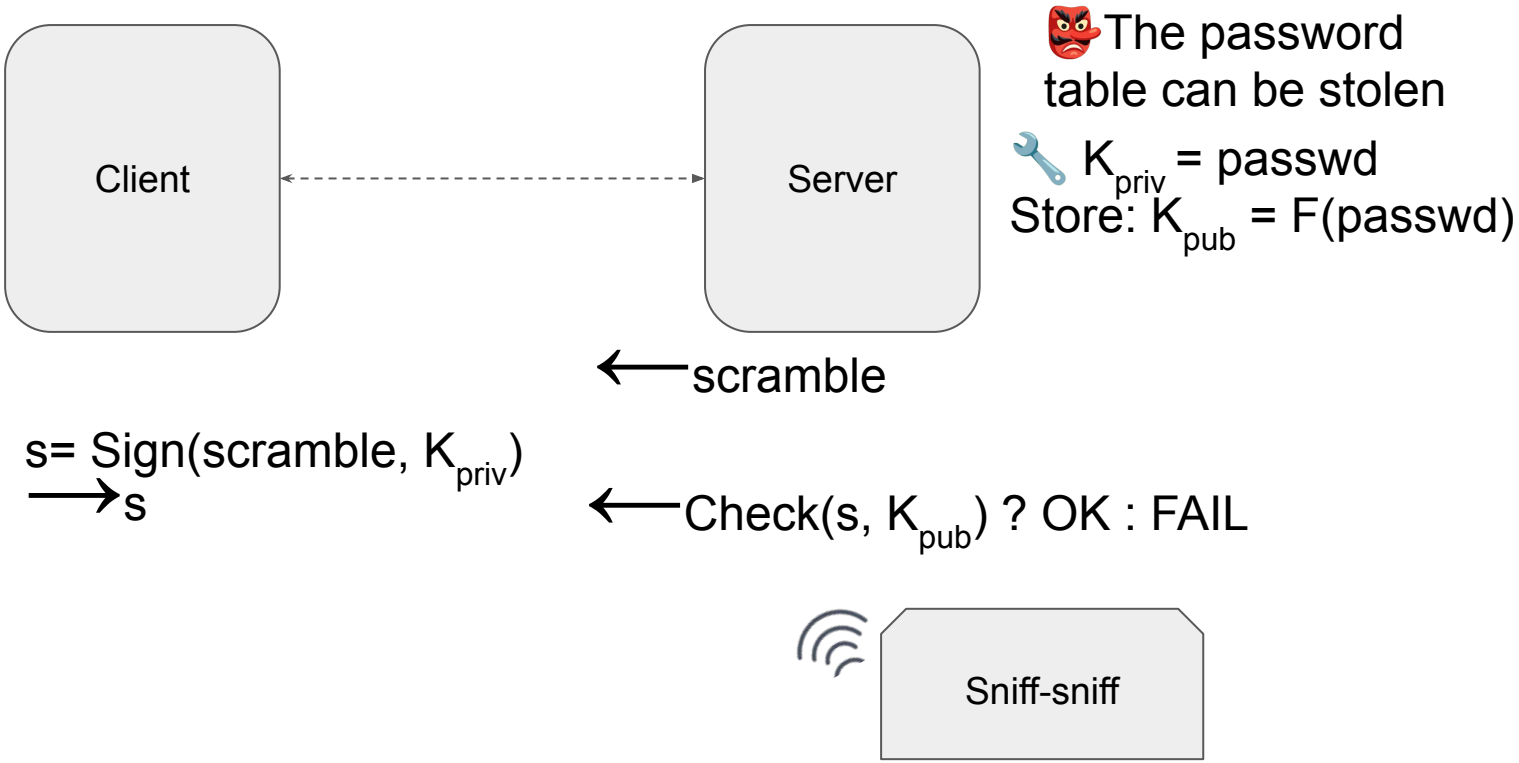
    m := s[:-64]

    h := s[-64:]

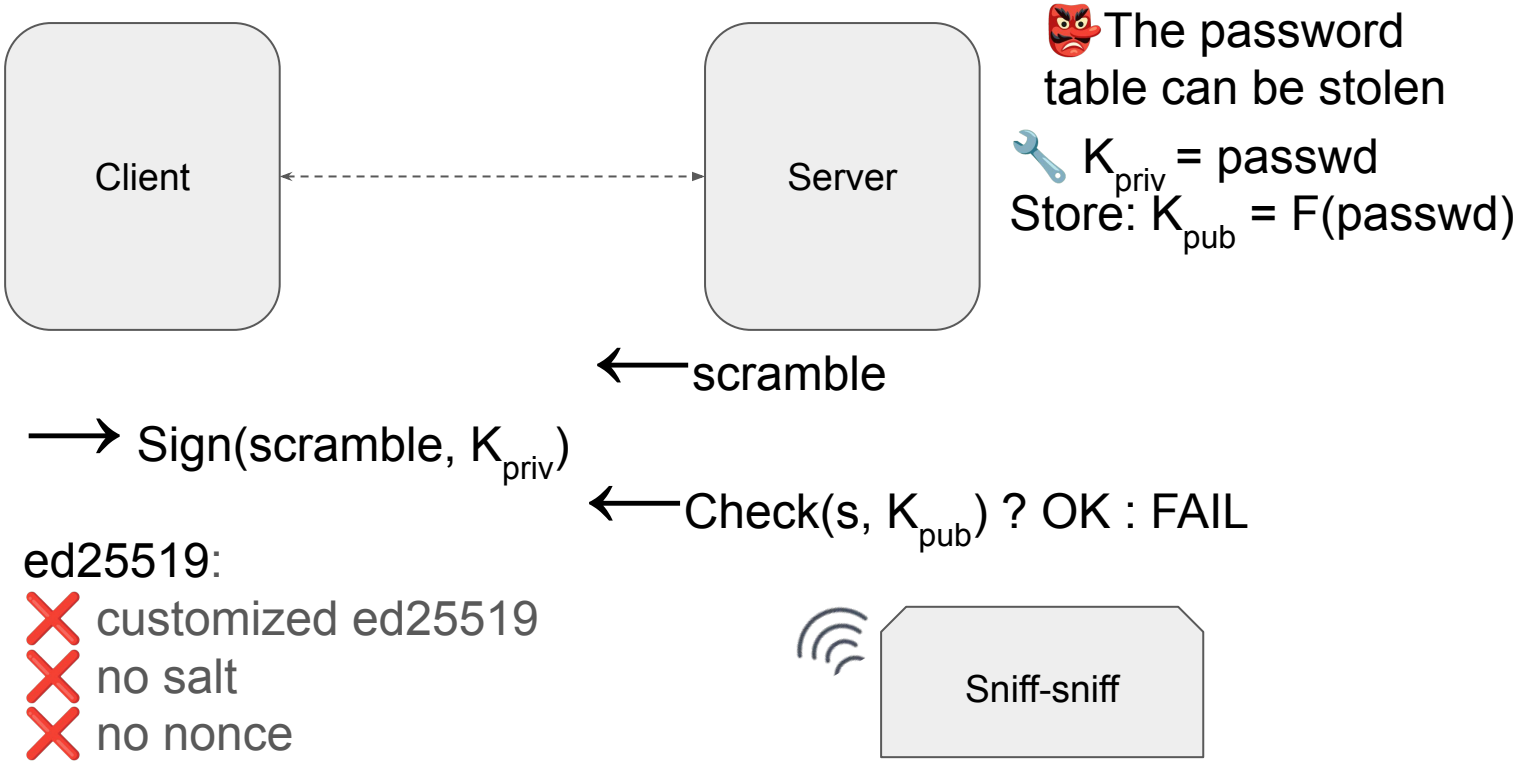
    return := sha512(m) == **D**(h, K_{pub})

  end

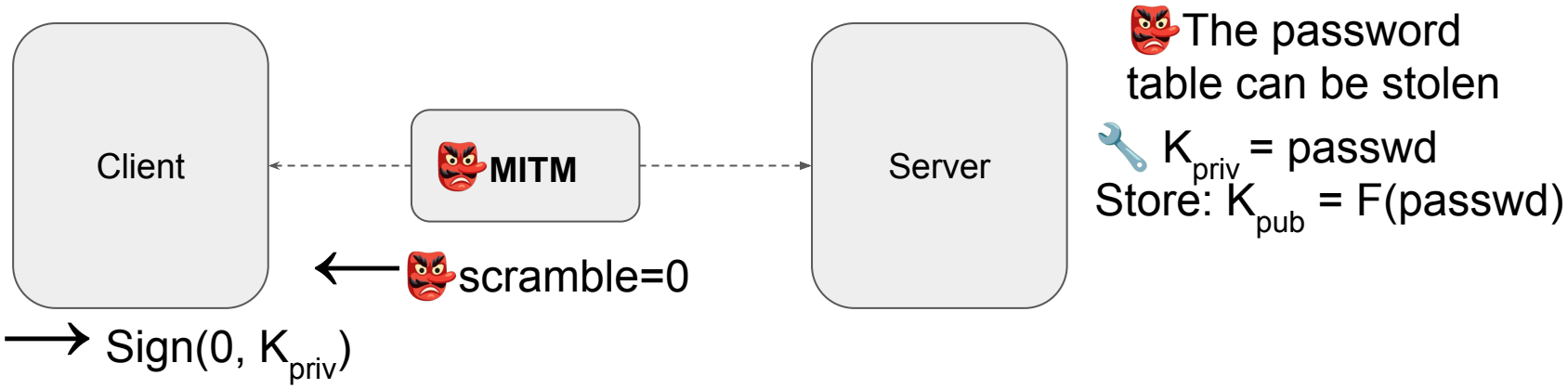
DIY: authentication plugin



DIY: authentication plugin

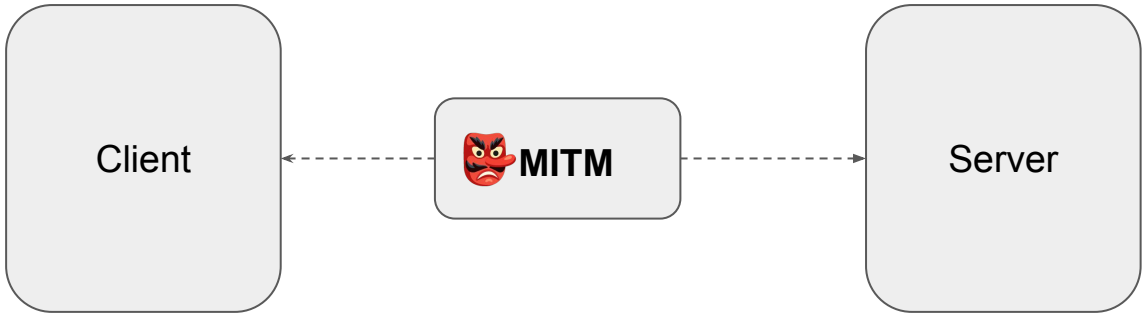


DIY: authentication plugin



Precalculates Sign(0, *) for every *

DIY: authentication plugin



The password table can be stolen

🔧 K_{priv} = passwd

Store: K_{pub} = F(passwd)

Precalculates Sign(0, *) for every *

← scramble,**salt**

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

👆 PARSEC 🦾

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

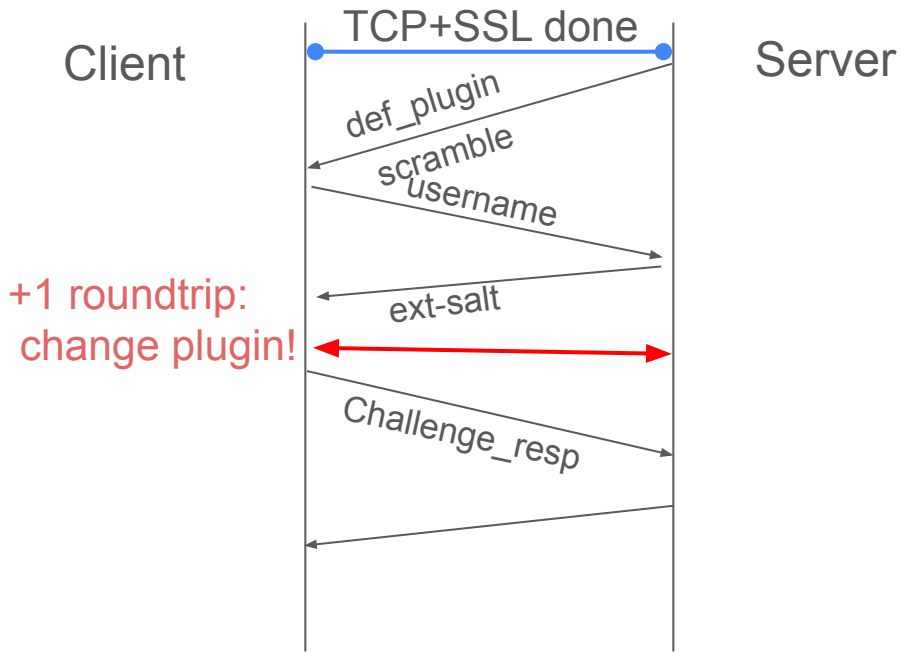
PARSEC:

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

hash-alg = PBKDF2

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

K_{priv} = PBKDF2(passwd)

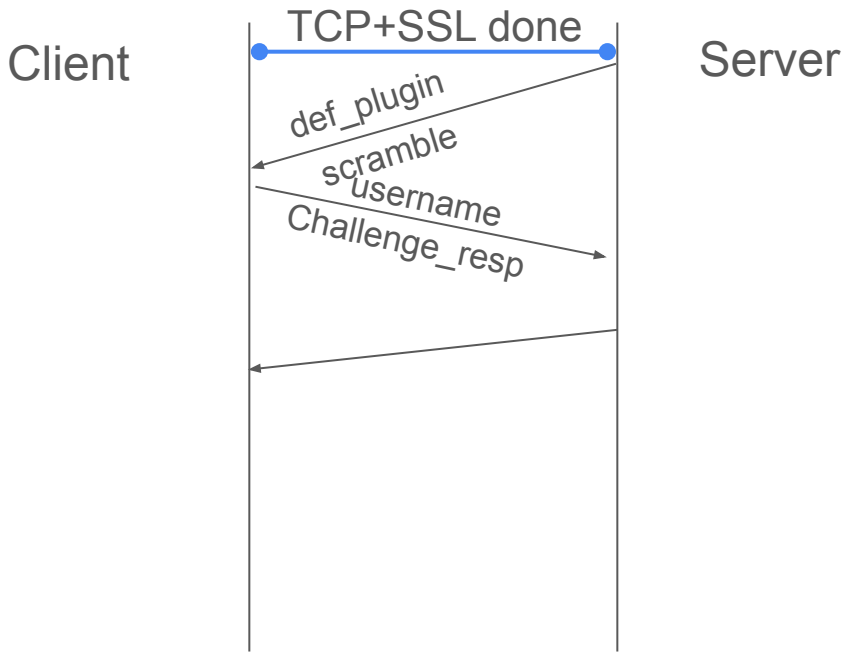


PARSEC/Soon:

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

Some day:

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



Nikita Maliavin

Senior Software Engineer
@MariaDB

MariaDB