

# Hardware Digital Signatures

Lionello Lunesu  
Enuma Technologies Limited

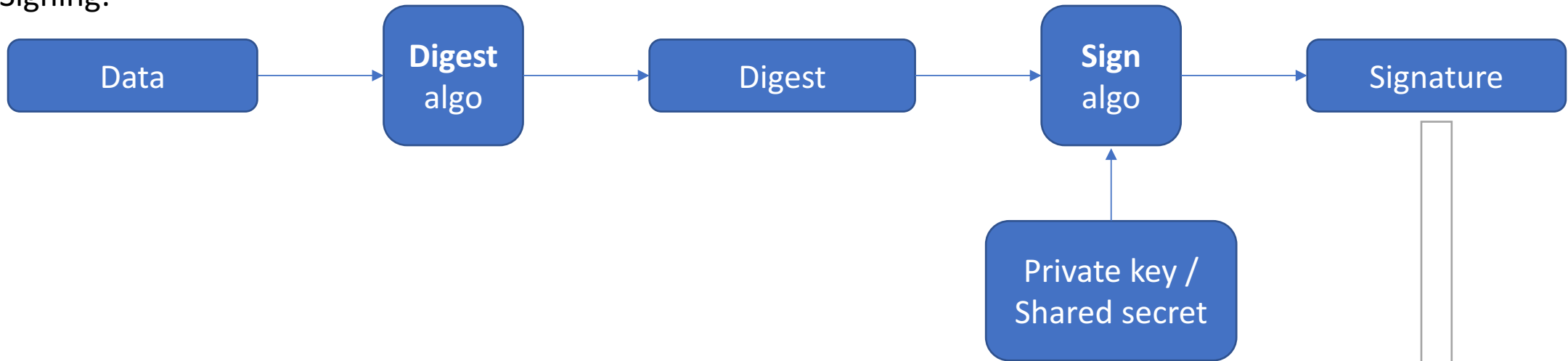
[lio@enuma.io](mailto:lio@enuma.io)

# Why?

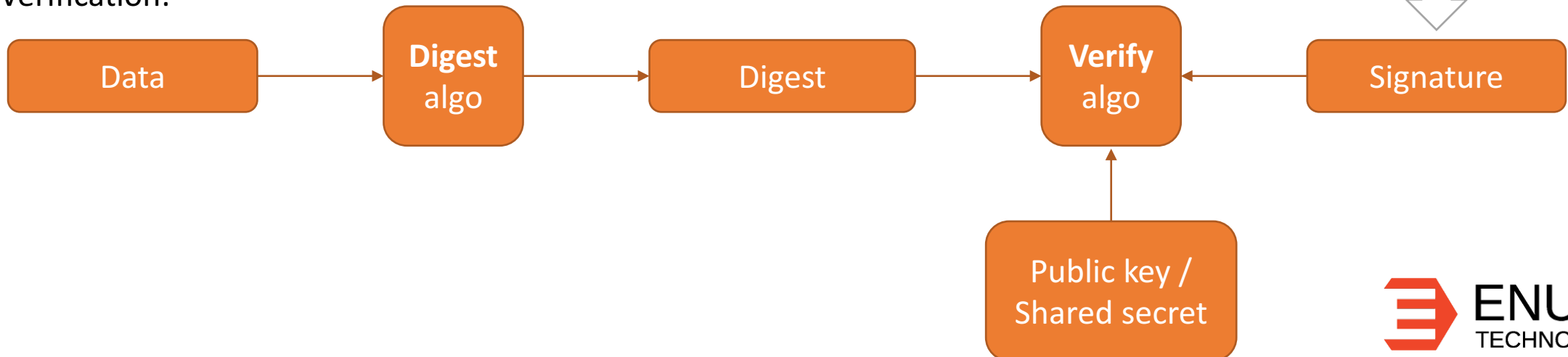
- SSL certificates (HTTPS)
- Bitcoin transactions
- Ethereum smart contracts
- Document signing (PDF)
- Debit/credit card (EMV)
- Message authentication (Signal/WhatsApp)

# Digital Signatures

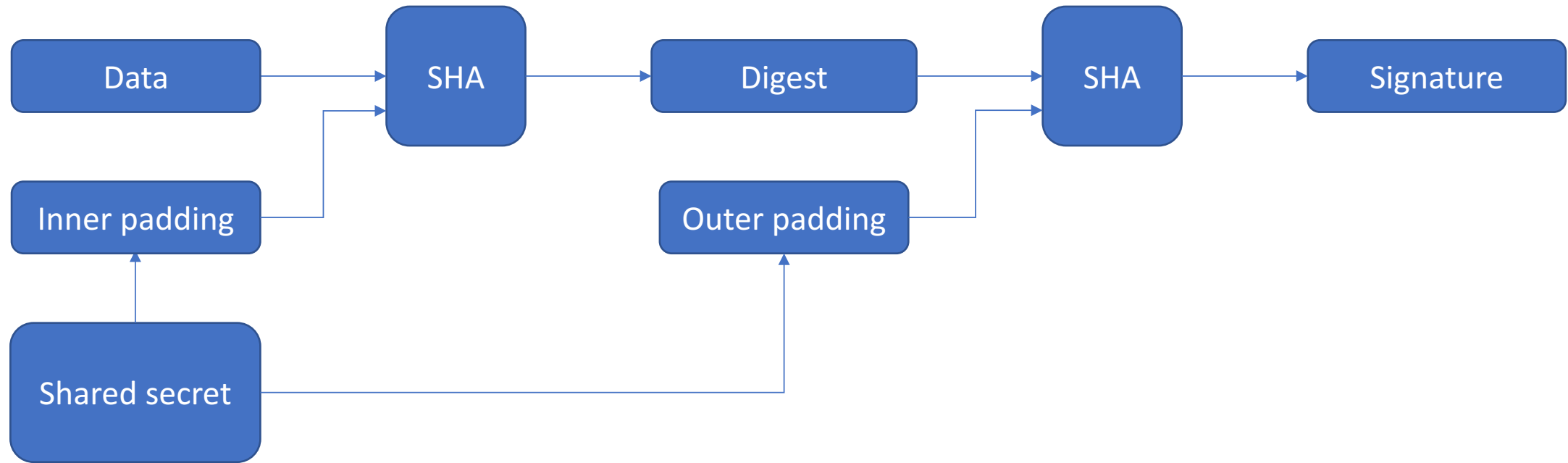
Signing:



Verification:

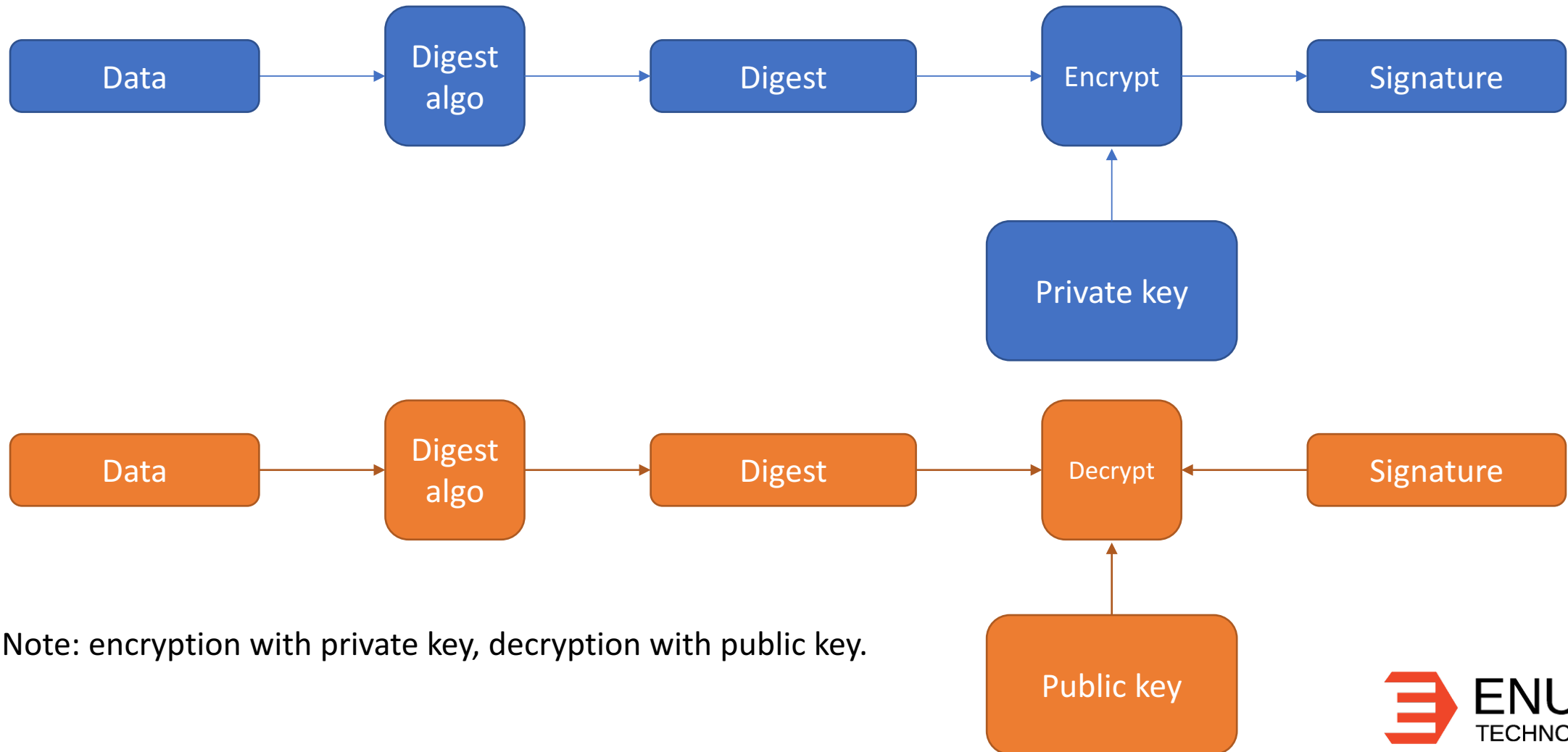


# Digital Signatures : HMAC

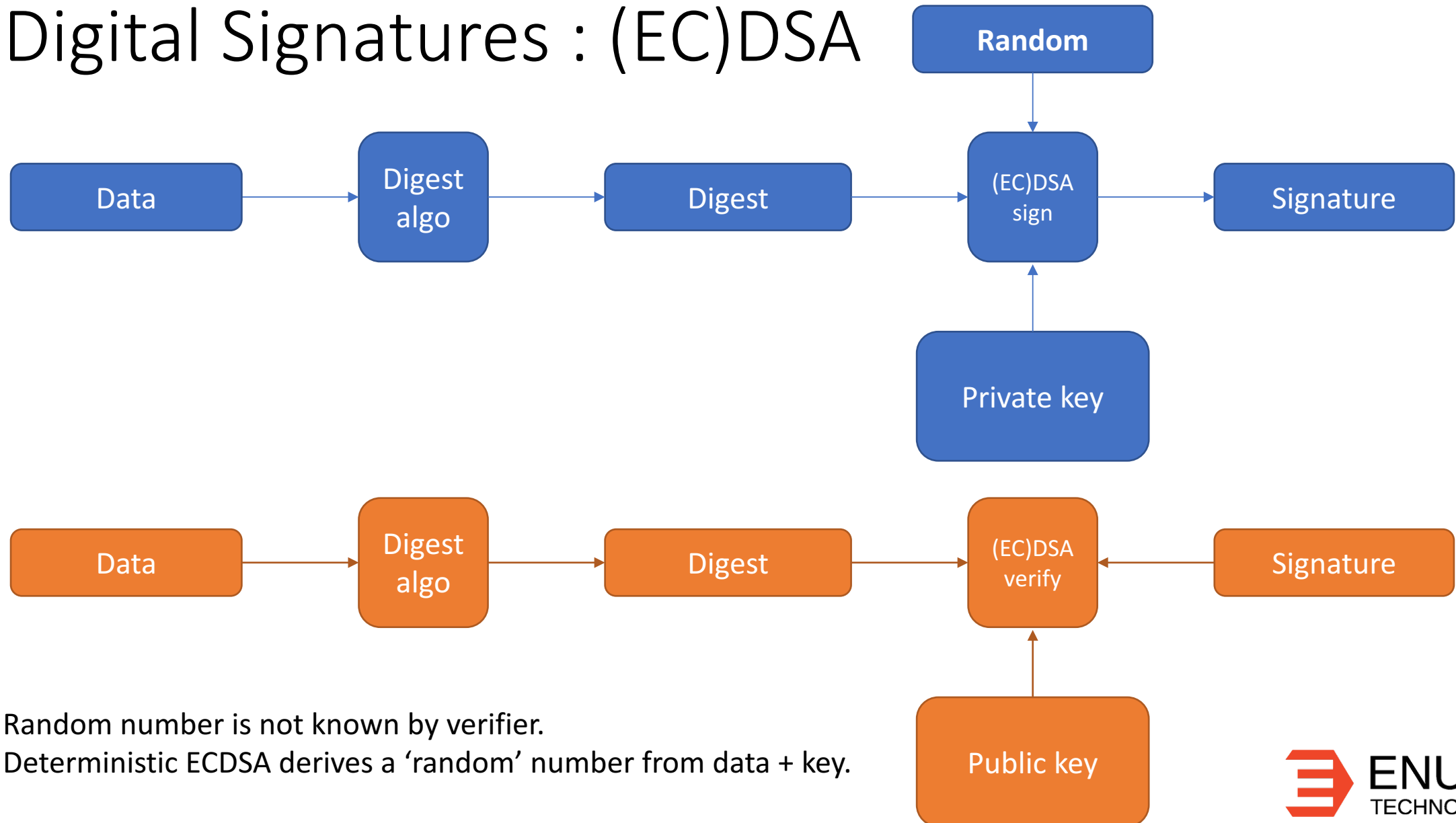


Both the signer and verifier follow the same process and compare the final signature.

# Digital Signatures : plain RSA



# Digital Signatures : (EC)DSA



Random number is not known by verifier.

Deterministic ECDSA derives a 'random' number from data + key.

# Digital Signature inside Bitcoin transaction

```
000: 0100000001ac18fa31f68e5597d4d1580ec1bdea10be30a39d10dddfd41360b3
020: f7bbc40fac000000006a47304402206d3e58f553c0605c3a663d6baa7258cd53
040: 6f3c50f4aac96c361695f82ab2017d022003ad05075cff6be0185d4dcc7581a6
060: 0634de35a33585f009ab2f0d1beb9ccbd801210374b22e7dd641b4d24c483023
080: a275fc808c813fe89f8cb4a9c97ef0f3431afb37fffffffff0202000000000000
0A0: 001976a914a24d41cca0b9baba81ce4f43747d97e24846ca6088acee3dcd1d00
0C0: 0000001976a91444635889ad4ba11e76c14d347867c48dba4069b388ac000000
0E0: 00
```

A 256-bit ECDSA signature consists of two 256-bit DER-encoded integers.

# Attacks on Software Digital Signatures

- Bad random number generation
  - PS3
  - Android: bitcoin wallet
- Bad deterministic key generation
  - BIP032 vulnerability
- Side-channel monitoring of key generation
  - Trezor v1
- Key compromise
  - Cross-VM snooping
  - Row hammer



# PS3 attack in a nutshell

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

© xkcd 221

# Crypto Hardware

- iOS Secure Enclave
- ARM Trusted Execution Environment (used by Android M and up)
- iNTEL SGX
- Atmel ECC crypto element
- Infineon Security controller
- NXP Secure authentication microcontroller
- FPGA-based
- SafeNet Luna SA (Amazon CloudHSM)
- Thales nShield (Azure Key Vault)

# Hardware keys on iOS

```
var dict: [String: AnyObject] = [  
    String(kSecAttrKeyType) : kSecAttrKeyTypeEC,  
    String(kSecAttrKeySizeInBits) : 256 as AnyObject  
]  
  
#if !((arch(i386) || arch(x86_64)) && os(iOS) && !NO_SE)  
    dict[String(kSecAttrTokenID)] = kSecAttrTokenIDSecureEnclave  
#endif  
  
let result = SecKeyGeneratePair(dict as CFDictionary, &publicKey, &privateKey)
```

# Hardware keys on Android

```
KeyPairGenerator keyPairGenerator =  
    KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC,  
                                "AndroidKeyStore");  
  
KeyGenParameterSpec.Builder builder =  
    new KeyGenParameterSpec.Builder("some key alias",  
                                    KeyProperties.PURPOSE_SIGN);  
  
keyPairGenerator.initialize(  
    builder  
        .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))  
        .setDigests(KeyProperties.DIGEST_SHA256, KeyProperties.DIGEST_NONE)  
        .build());  
  
KeyPair keyPair = keyPairGenerator.generateKeyPair();
```

# Hardware key in Azure

```
npm install azure -g
```

```
azure keyvault create my-vault --resource-group free-hk --location eastasia --sku premium
```

```
azure keyvault key create --vault-name my-vault --key-name MyKey --destination HSM
```

# Limitations of Crypto Hardware

- Key size
  - often  $\leq 256$  bits EC,  $\leq 2048$  bits RSA
- Algorithms
  - EC, SHA, ECDHE, RSA?
- Curve domain parameters
  - Secp256r1
  - usually no Secp256k1

# Thank you

- Related info:

<http://blog.enuma.io/update/2016/11/01/a-tale-of-two-curves-hardware-signing-for-ethereum.html>

lio@enuma.io