Bleichenbacher RSA Signature Forgery (2006)

oalieno

PKCS

PKCS

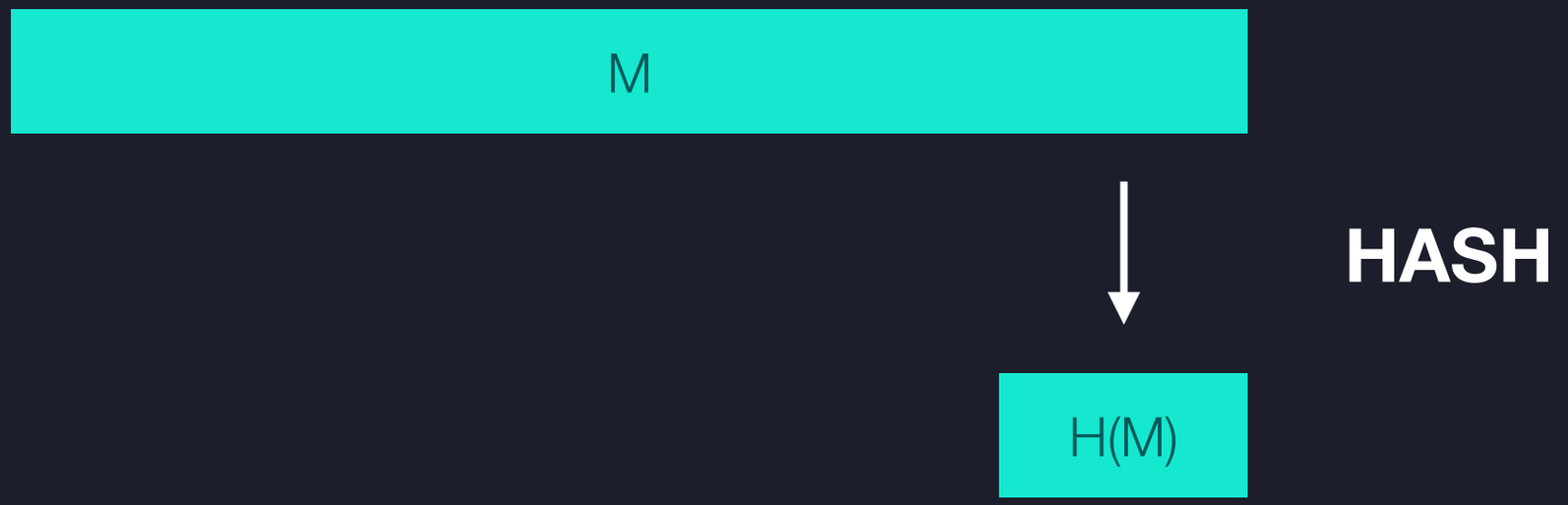
- PKCS (Public Key Cryptography Standards) 是公鑰密碼標準
- 制定了一系列從 PKCS#1 到 PKCS#15 的標準
- 其中 PKCS#1 是 RSA Cryptography Standard

ASN.1

- ASN.1 是高階的抽象標準
- 具體的實作編碼規則有：BER, CER, DER, PER, XER

PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313



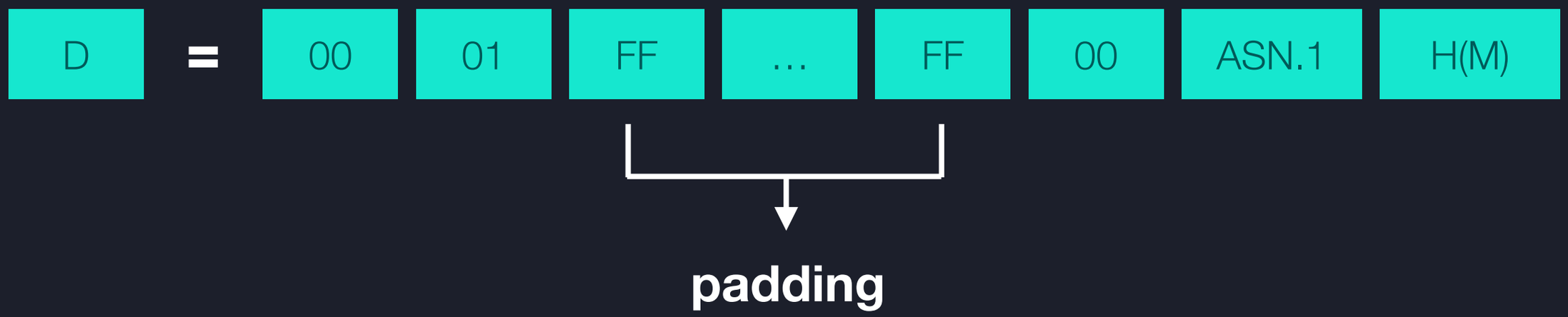
Sign

Step 1 : Message Digest

PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313

- **ASN.1 是編碼數據的格式，這裡紀錄了使用的 hash 演算法**



PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313

d

D % n = S

Sign

Step 3 : RSA encryption

PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313

$$S^e \% n = D$$

Verify

Step 1 : RSA decryption

PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313

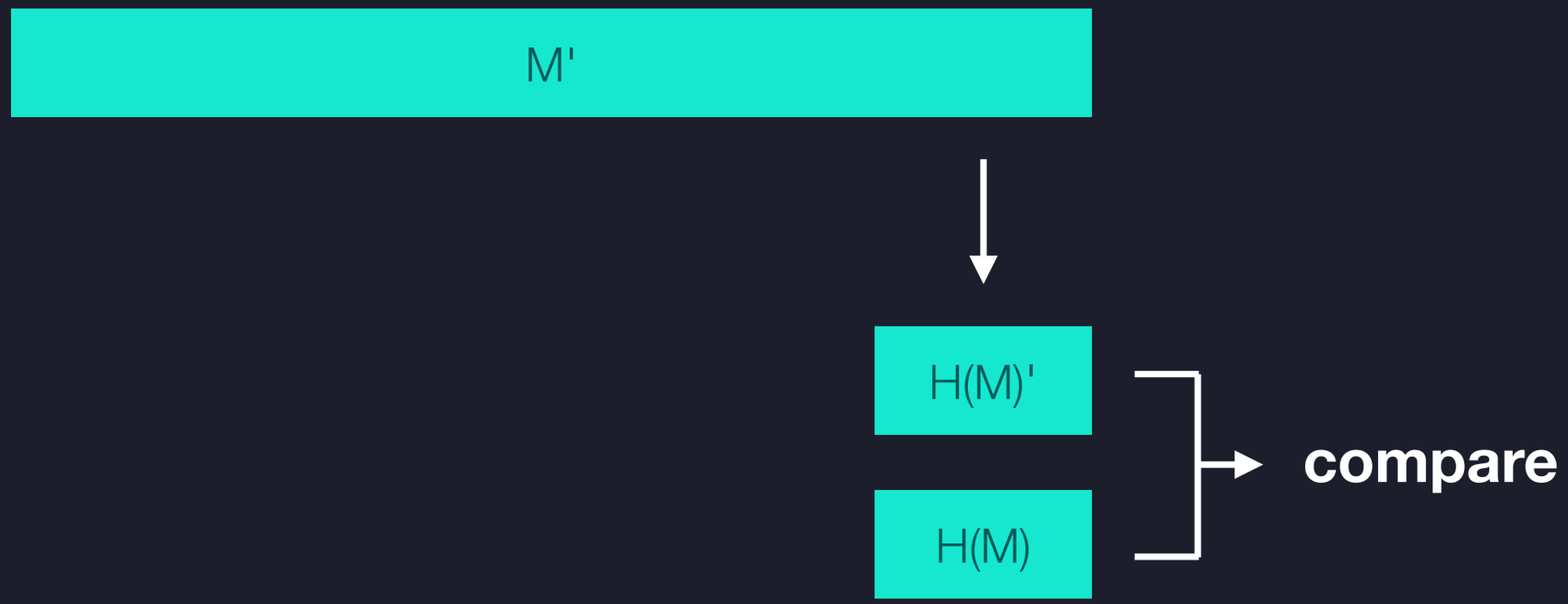
- 需要 parse 這個格式取出 H(M)
- 這個標準沒有說要怎麼 parse
- 如果 e 太小且沒有正確的 parse，就有機會偽造簽章

Verify

Step 2 : Data Decoding

PKCS#1 1.5 Signature

https://tools.ietf.org/html/rfc2313



Verify

Step 3 : Message digesting and comparison

Bleichenbacher RSA Signature Forgery (2006)

Bleichenbacher RSA Signature Forgery (2006)

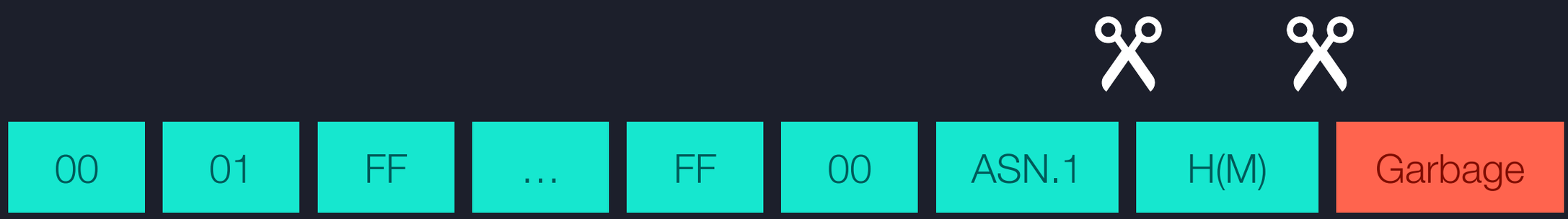
https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE

- 又稱作 BB06
- 針對 PKCS#1 1.5 (RFC 2313)
- RSA 簽章偽造

Bleichenbacher RSA Signature Forgery (2006)

https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE

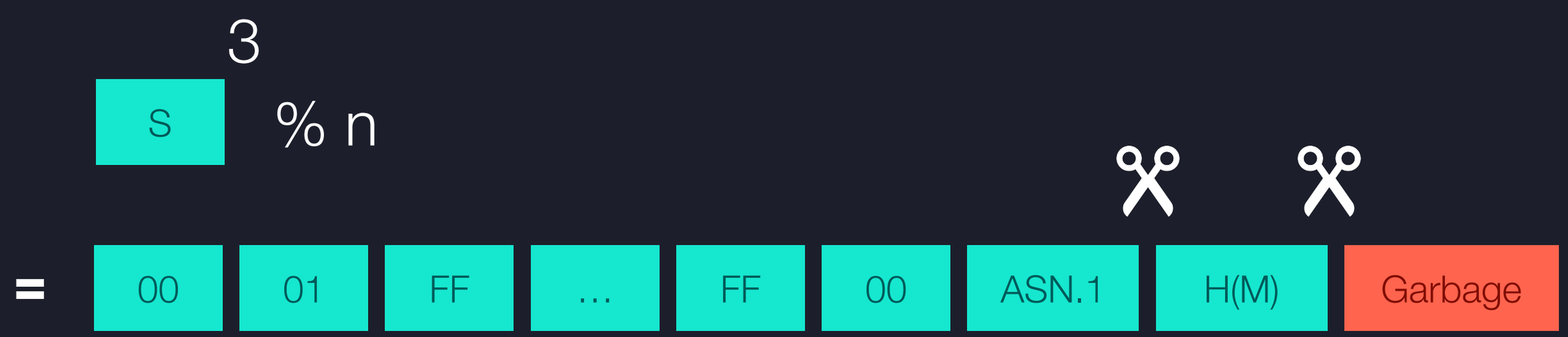
- 實作缺陷：可以有多餘的字元在後面
- parse 的時候直接取出後面固定長度的 H(M)
- 沒有檢查後面還有沒有東西



Bleichenbacher RSA Signature Forgery (2006)

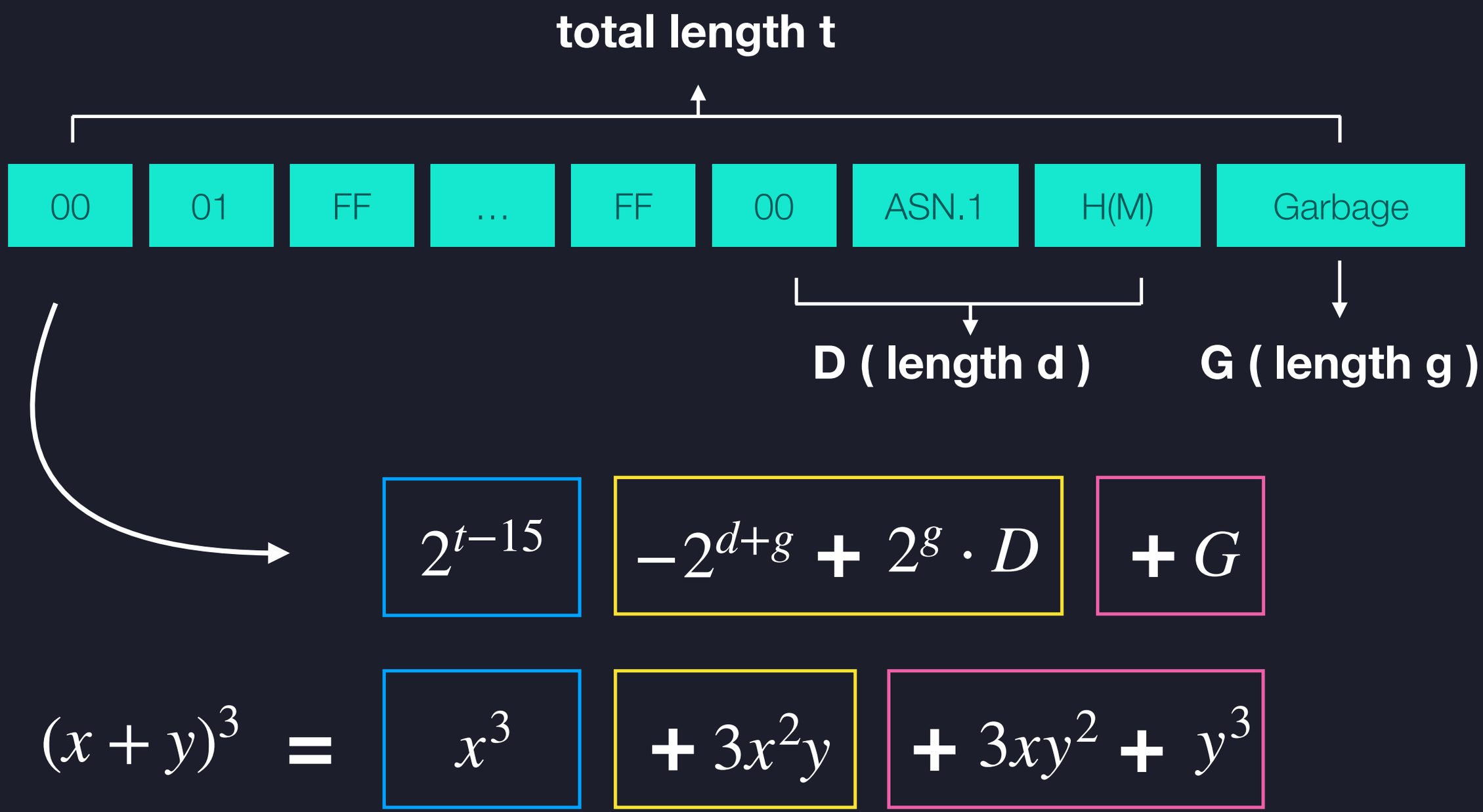
https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE

- 在 e = 3 的情況下可以 forge signature
- 嘗試構造 ED 讓 ED 的三次方不超過 n 且滿足以下格式



Bleichenbacher RSA Signature Forgery (2006)

https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE



Bleichenbacher RSA Signature Forgery (2006)

https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE

$$x = 2^{\frac{t-15}{3}}$$

$$y = \frac{(D - 2^d) \cdot 2^g}{3 \cdot 2^{\frac{2(t-15)}{3}}}$$

Bleichenbacher RSA Signature Forgery (2006)

https://mailarchive.ietf.org/arch/msg/openpgp/5rnE9ZRN1AokBVj3VqblGlP63QE

- 假設
  - Key 長度為 3072 bit
  - Garbage 長度為 2072 bit
  - 使用 SHA-1 的話，D 的長度是 288 bit
- 最後 ED = x + y 就是我們構造出的合法簽章

$$x = 2^{1019}$$

$$y = \frac{(D - 2^{288}) \cdot 2^{34}}{3}$$

RSA Signature Forgery in python-rsa (2016)

CVE-2016-1494

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

- 實作缺陷： padding bytes 可以是任意字元

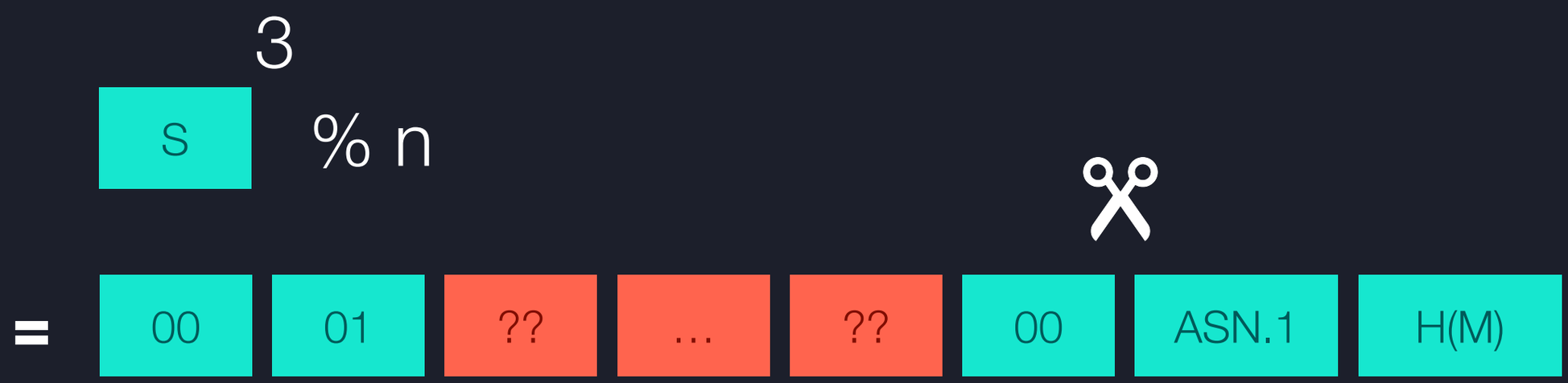
```
...
# If we can't find the signature  marker, verification failed.
if clearsig[0:2] != b('\x00\x01'):
    raise VerificationError('Verification failed')

# Find the 00 separator between the padding and the payload
try:
    sep_idx = clearsig.index(b('\x00'), 2)
except ValueError:
    raise VerificationError('Verification failed')

# Get the hash and the hash method
(method_name, signature_hash) = _find_method_hash(clearsig[sep_idx+1:])
message_hash = _hash(message, method_name)
...
```

直接取第二個 0x00 沒有檢查中間的 padding bytes

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

- 在 e = 3 的情況下可以 forge signature
- 嘗試構造 ED 讓 ED 的三次方不超過 n 且滿足以下格式
  - ED^3 的後綴是 ASN.1 + H(M)
  - ED^3 的前綴是 \x00\x01



RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

S	0	0	0	1
S^3	0	0	0	1
目標	1	1	0	1

match

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

S	0	0	0	1
S^3	0	0	0	1
目標	1	1	0	1

match

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

S	0	0	0	1
S^3	0	0	0	1
目標	1	1	0	1

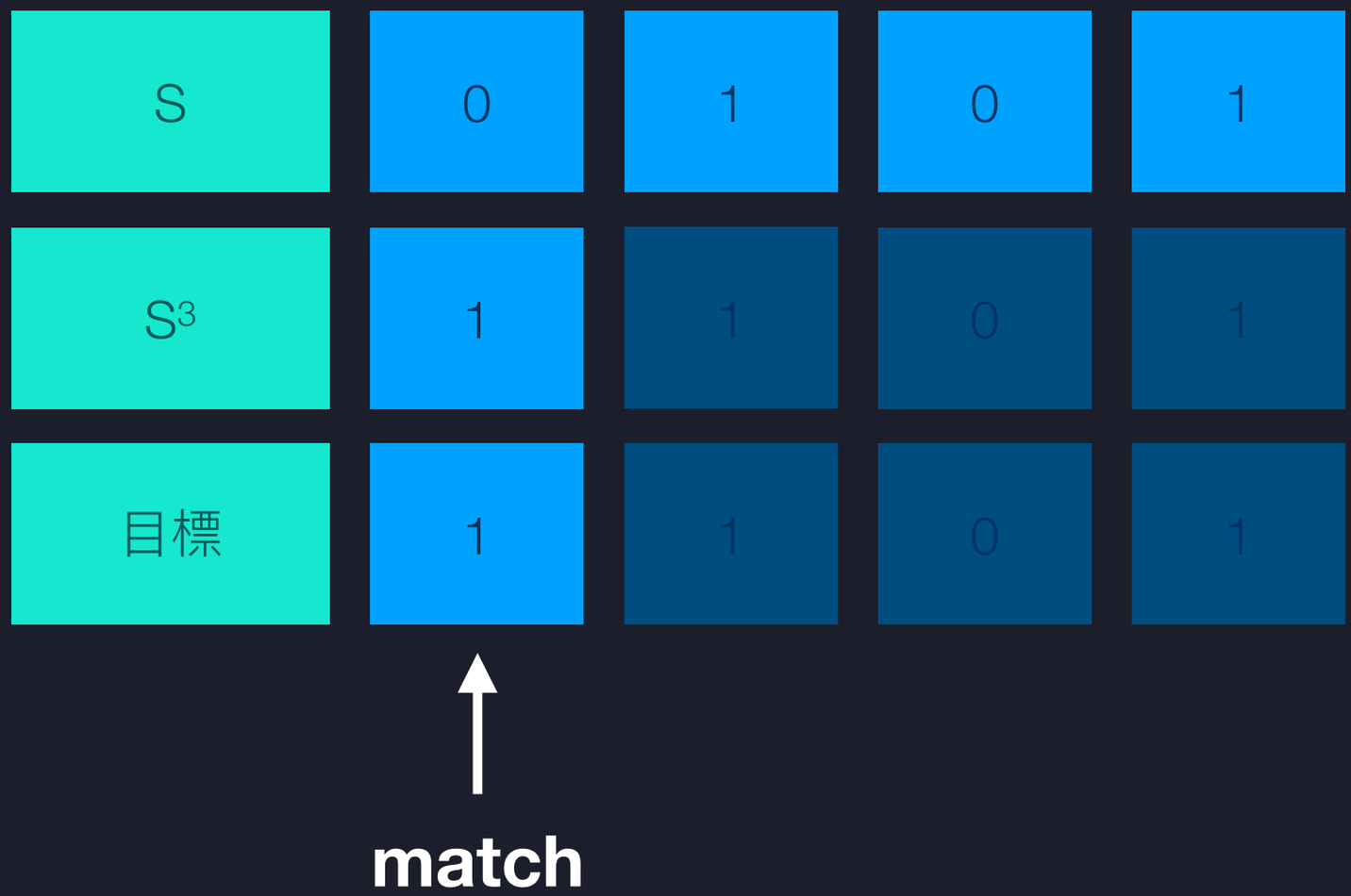
mismatch

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/



RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

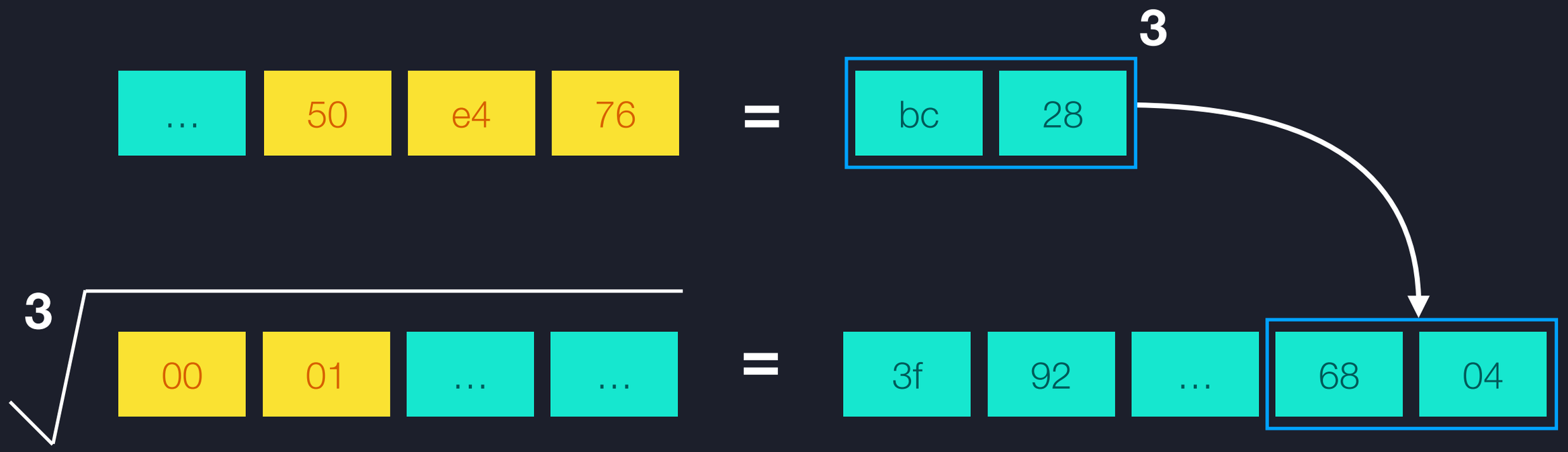
S	0	1	0	1
S^3	1	1	0	1
目標	1	1	0	1

$0101^3 = 11111101$

RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

- 要讓 ED^3 的前綴是 \x00\x01 只要把 \x00\x01... 開三次方
- 最後再把開完三次方的值的後綴換成前面算出來的後綴
- 就可以成功自己構造合法簽章了



RSA Signature Forgery in python-rsa (2016)

https://blog.filippo.io/bleichenbacher-06-signature-forgery-in-python-rsa/

A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works (2019)

A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works (2019)

https://i.blackhat.com/USA-19/Wednesday/us-19-Chau-A-Decade-After-Bleichenbacher-06-RSA-Signature-Forgery-Still-Works.pdf

- 整個格式固定是 n 這麼長
- 用 Symbolic Execution 去找到可以任意亂塞的部分有多長

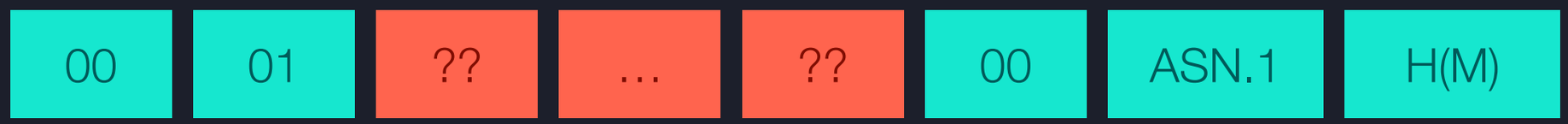
Name - Version	***Overly lenient***	*Practical exploit under small e*
axTLS - 2.1.3	YES	YES
BearSSL - 0.4	No	-
BoringSSL – 3112	No	-
Dropbear SSH – 2017.75	No	-
GnuTLS – 3.5.12	No	-
LibreSSL – 2.5.4	No	-
libtomcrypt – 1.16	YES	YES
MatrixSSL – 3.9.1 (Certificate)	YES	No
MatrixSSL – 3.9.1 (CRL)	YES	No
mbedTLS – 2.4.2	YES	No
OpenSSH – 7.7	No	-
OpenSSL – 1.0.2l	No	-
Openswan – 2.6.50 *	YES	YES
PuTTY – 0.7	No	-
strongSwan – 5.6.3 *	YES	YES
wolfSSL – 3.11.0	No	-

A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works (2019)

https://i.blackhat.com/USA-19/Wednesday/us-19-Chau-A-Decade-After-Bleichenbacher-06-RSA-Signature-Forgery-Still-Works.pdf

Openswan 2.6.50 | CVE-2018-15836

- 實作缺陷 : padding bytes 可以是任意字元

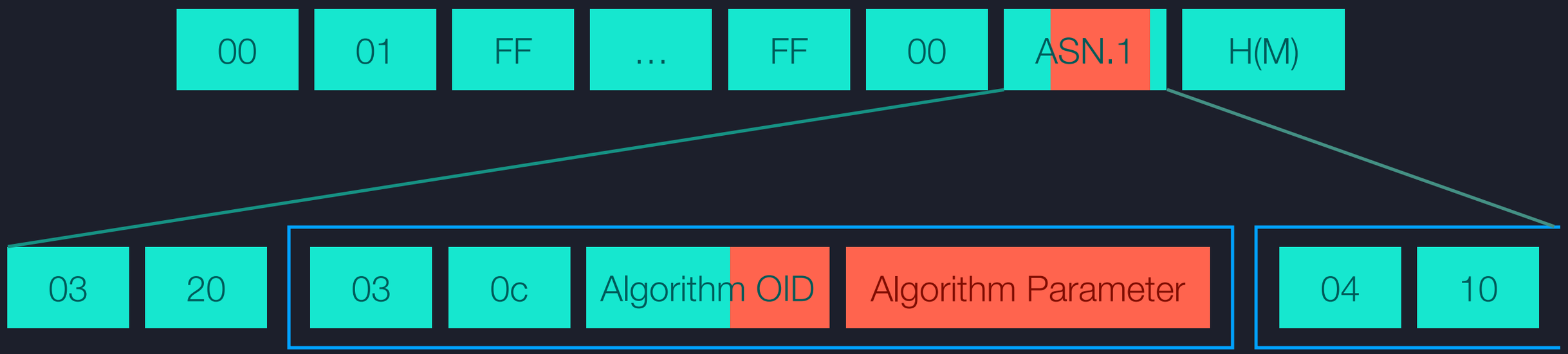


A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works (2019)

https://i.blackhat.com/USA-19/Wednesday/us-19-Chau-A-Decade-After-Bleichenbacher-06-RSA-Signature-Forgery-Still-Works.pdf

strongSwan 5.6.3 | CVE-2018-16152

- 實作缺陷：
  - Algorithm Parameter 可以是任意字元
  - Algorithm OID 後面可以有多餘的字元

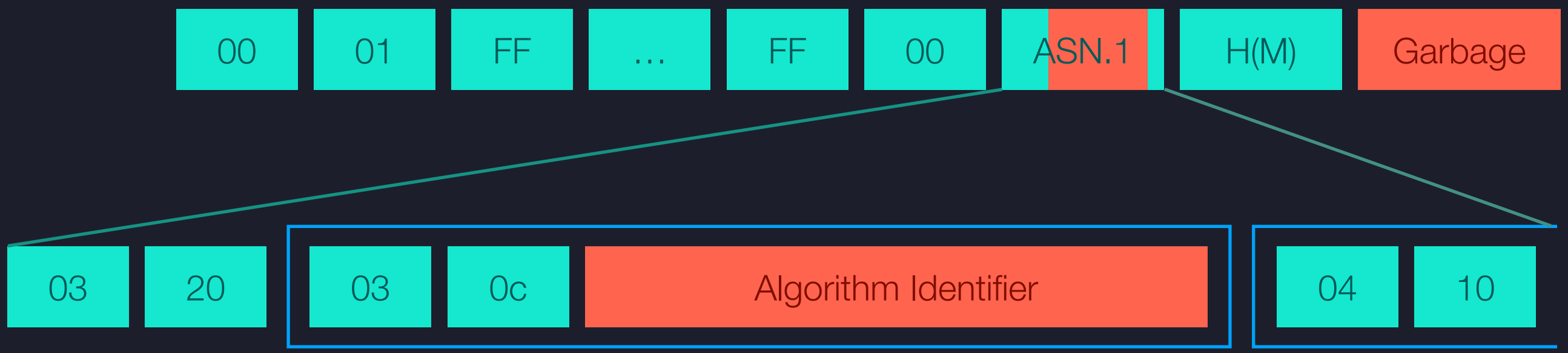


A Decade After Bleichenbacher '06, RSA Signature Forgery Still Works (2019)

https://i.blackhat.com/USA-19/Wednesday/us-19-Chau-A-Decade-After-Bleichenbacher-06-RSA-Signature-Forgery-Still-Works.pdf

axTLS 2.1.3 | CVE-2018-16150

- 實作缺陷：
  - 可以有多餘的字元在後面
  - Algorithm Identifier 可以是任意字元



Defense against RSA Signature Forgery

How to defense?

- 用其他的簽章演算法，比如說 ECDSA
- 用更大的 e，比如 65537
- ~~parsing based~~ → comparison based