Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage

Christina Garman

Matthew Green

Gabriel Kaptchuk

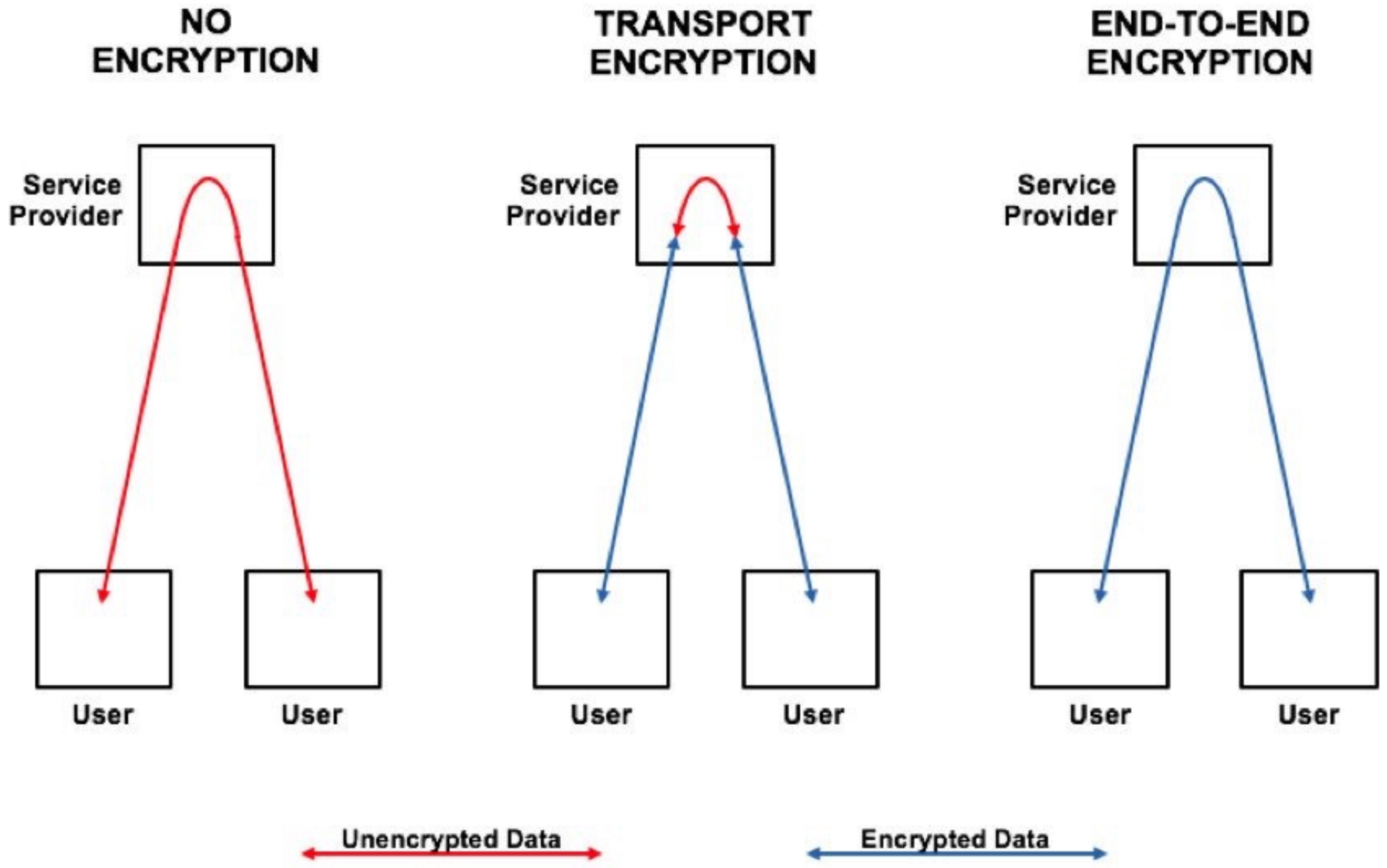
<u>Ian Miers</u>

Michael Rushanan

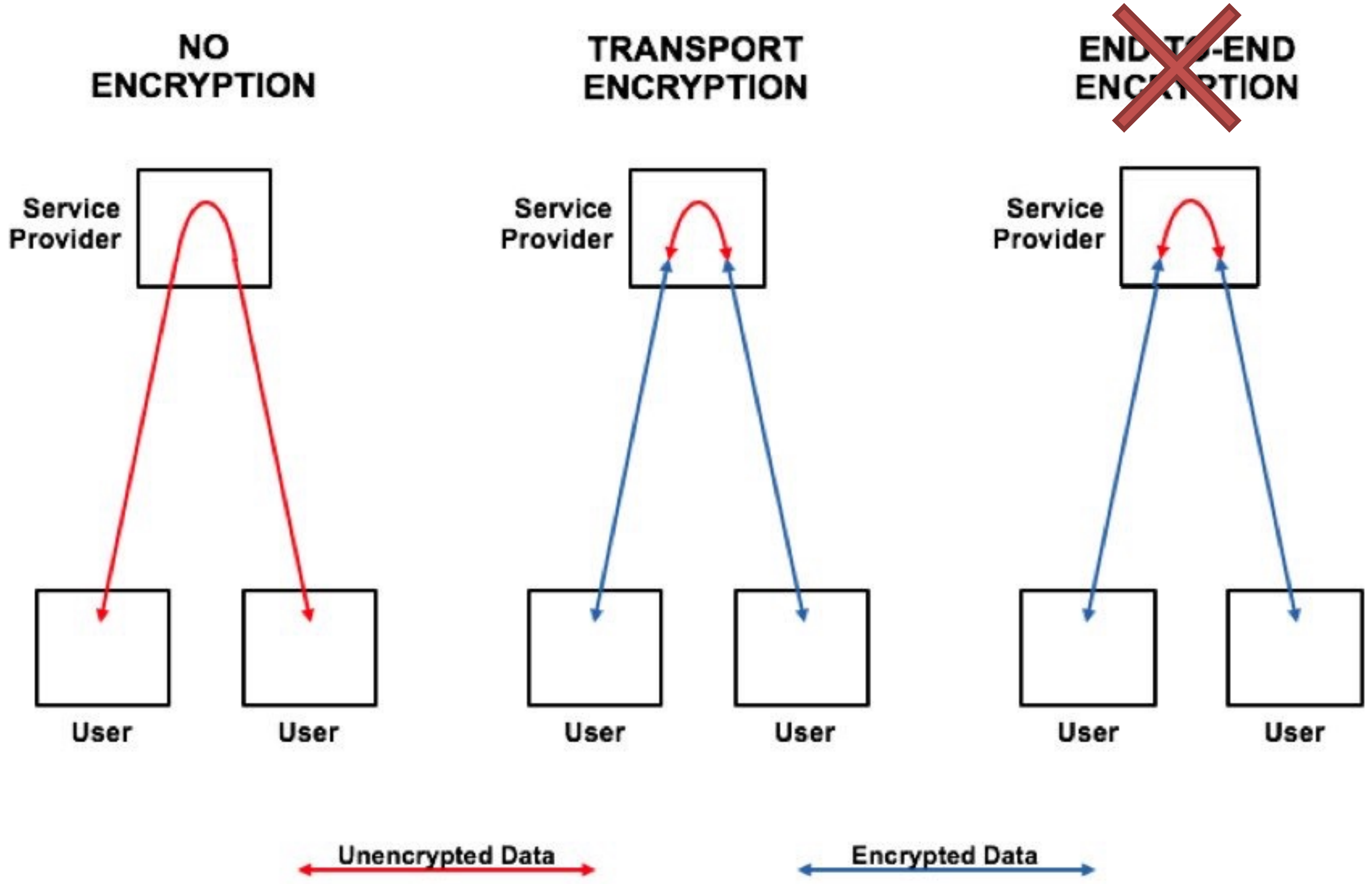
iMessage

- Created in 2011
- 1 billion deployed devices
- 200,000 messages per second peak
- First major deployment of end-to-end encrypted chat
- Used in other things:
  - Handoff
  - Other undisclosed products

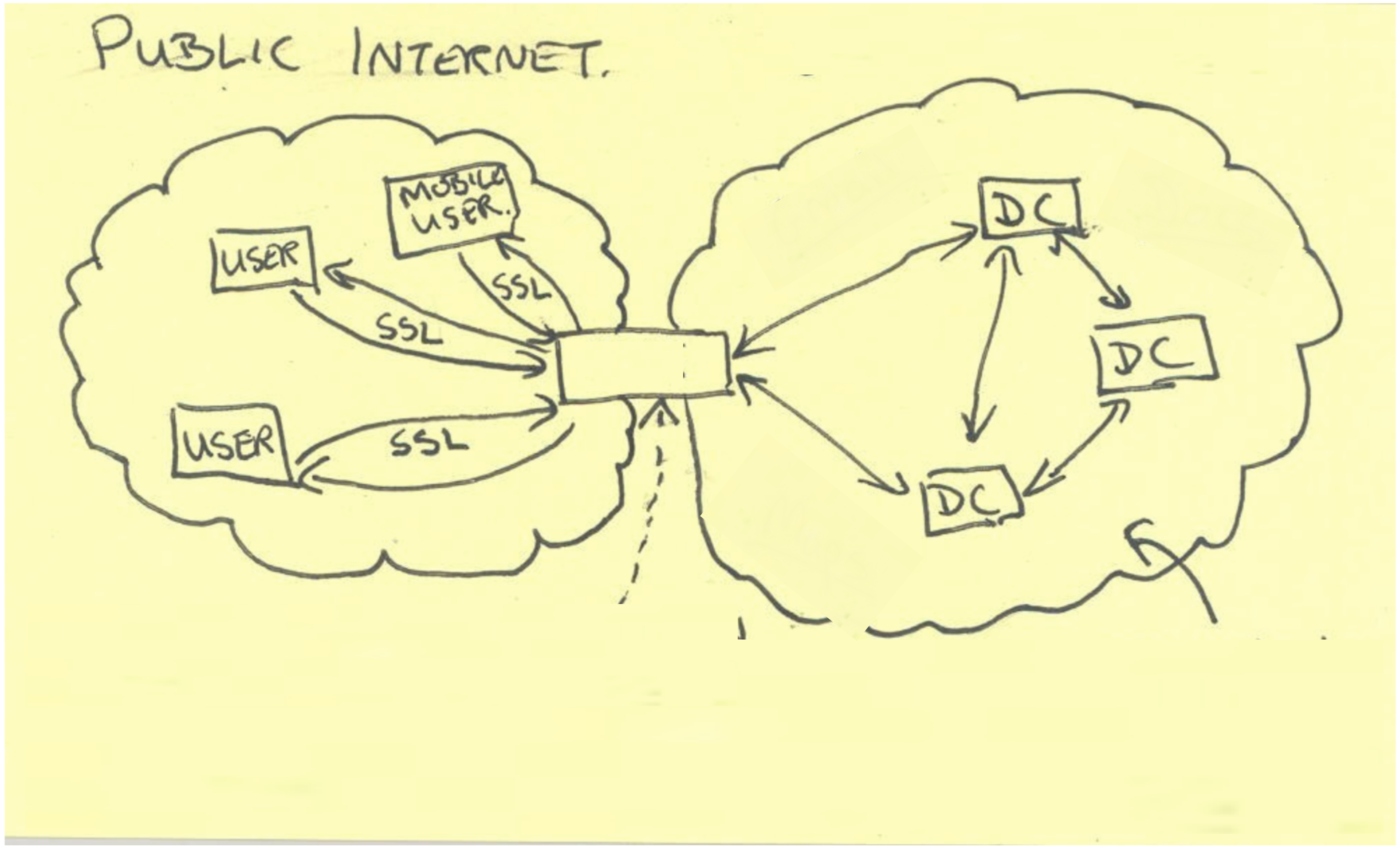
All encryption is not equal



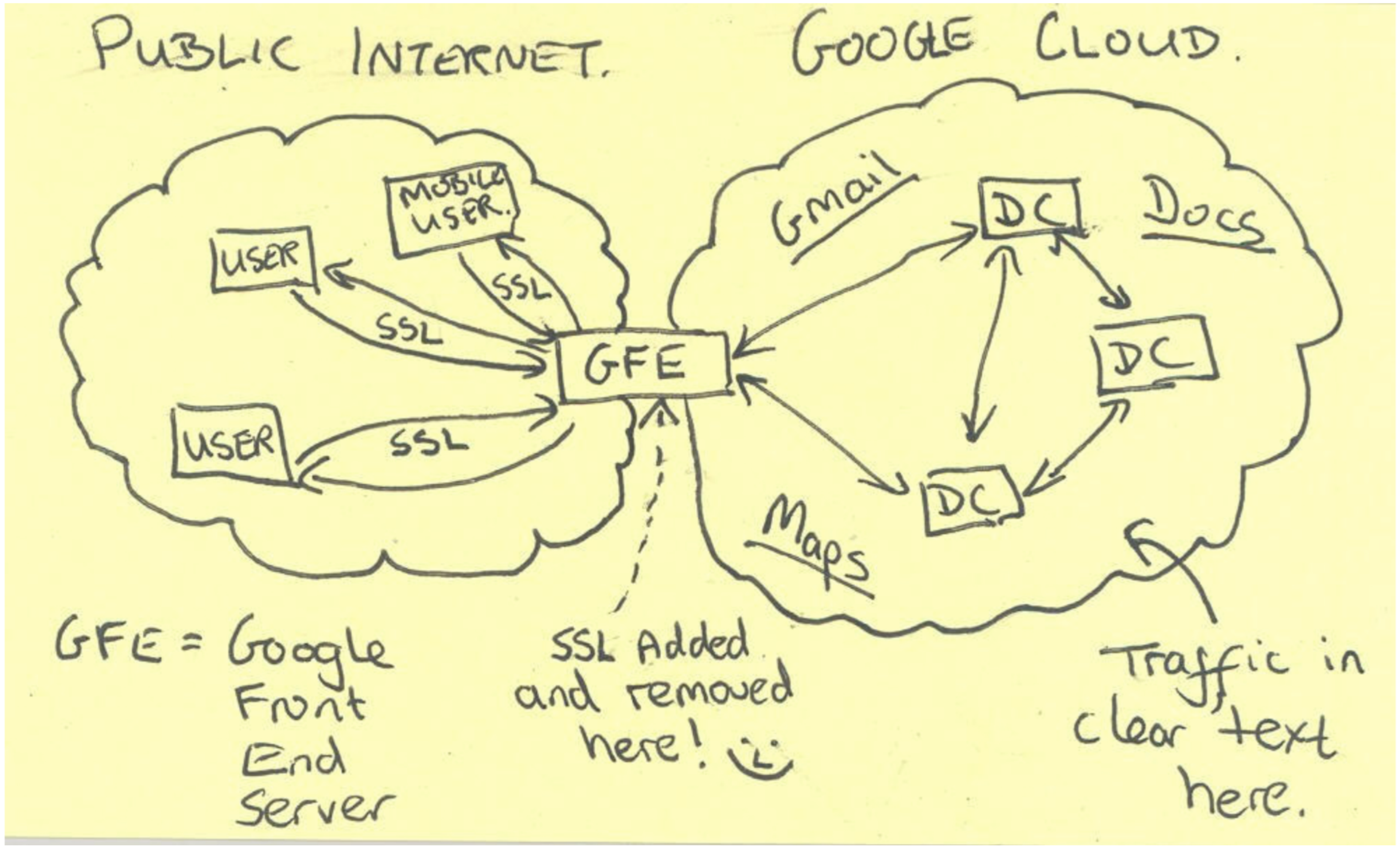
Reducing iMessage Security



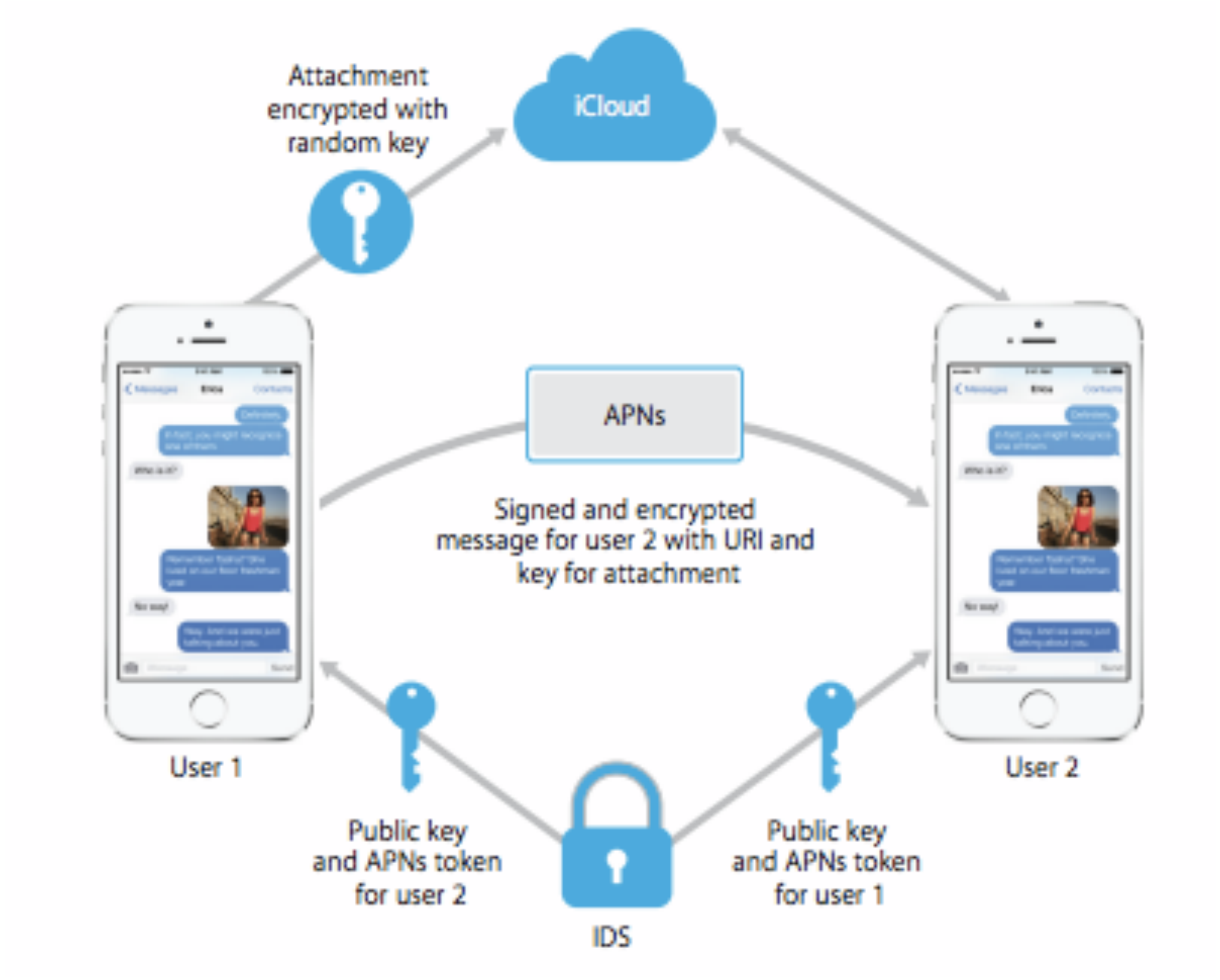
“Server” can be very complex and insecure …

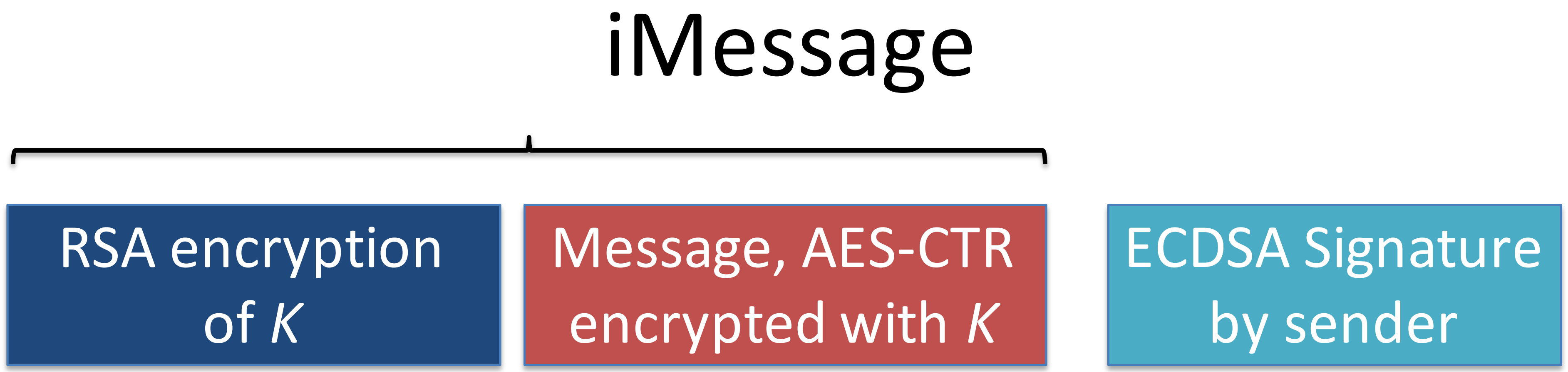


… And have skilled attackers

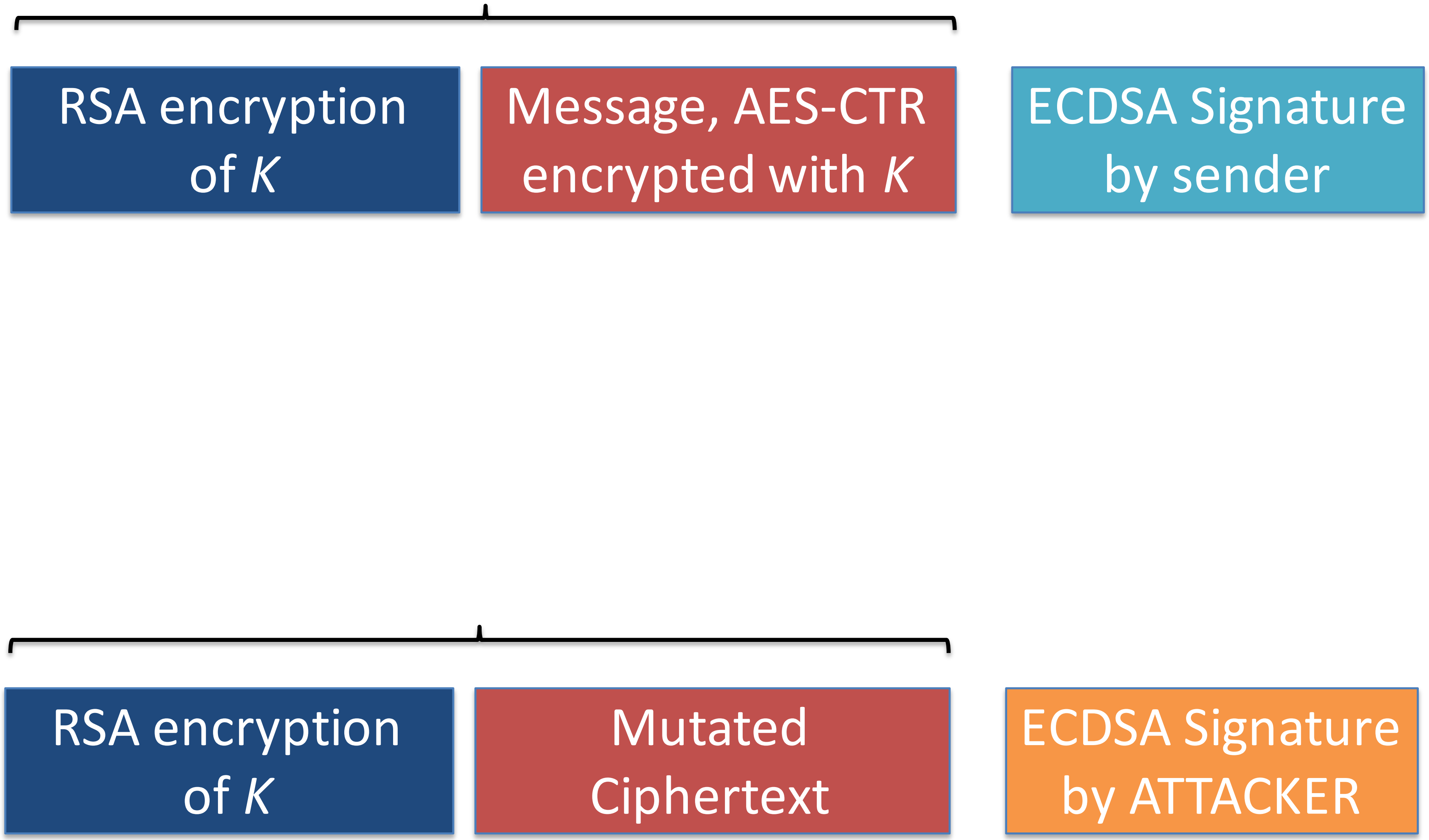

PUBLIC INTERNET.
GOOGLE CLOUD.
MOBILE USER.
USER
USER
SSL
SSL
SSL
GFE
Gmail
DC
Docs
DC
DC
Maps
GFE = Google Front End Server
SSL Added and removed here! :)
Traffic in clear text here.

iMessage

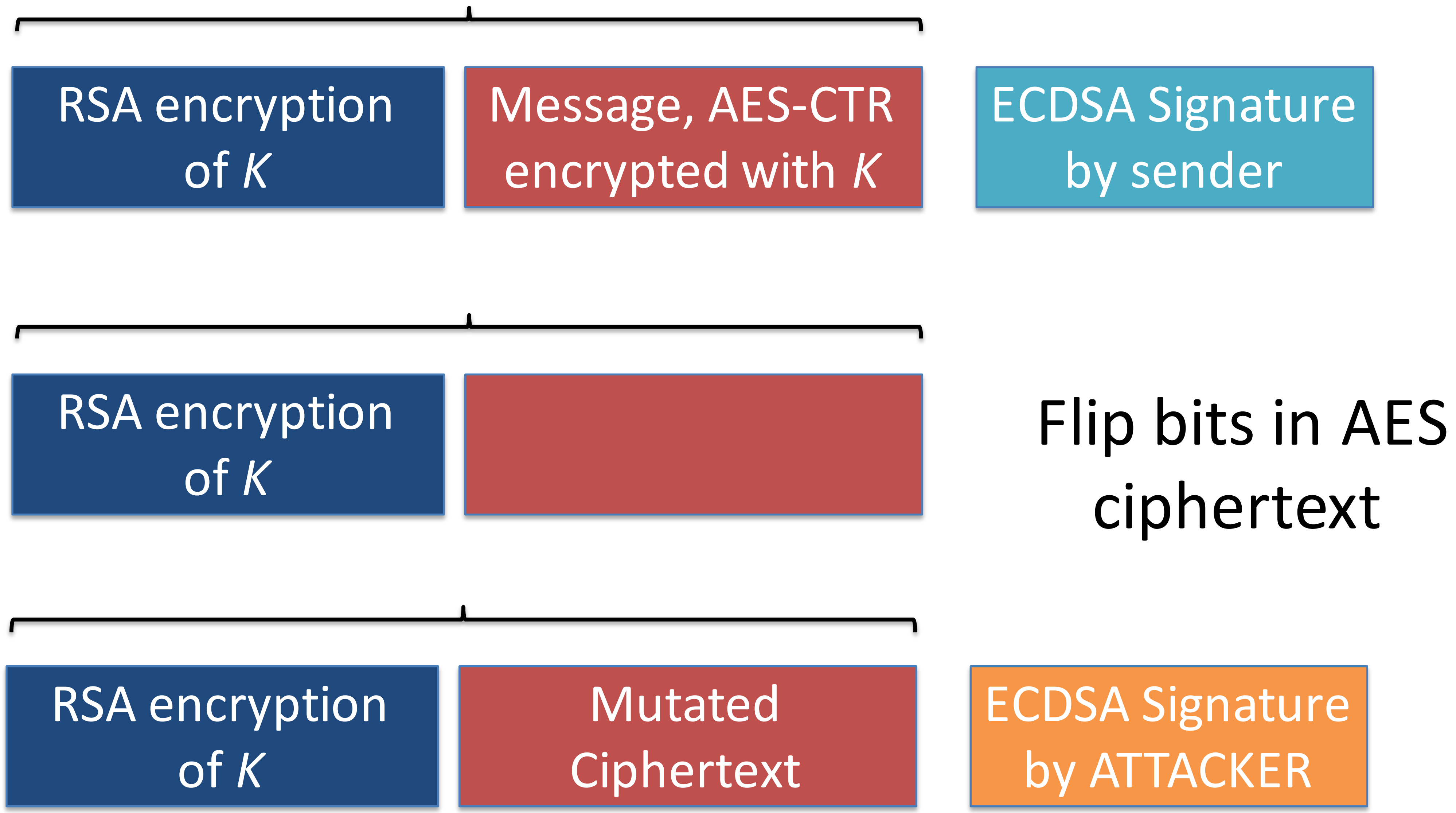

Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS


iMessage
RSA encryption of *K*
Message, AES-CTR encrypted with *K*
ECDSA Signature by sender

Identity Misbinding Attack

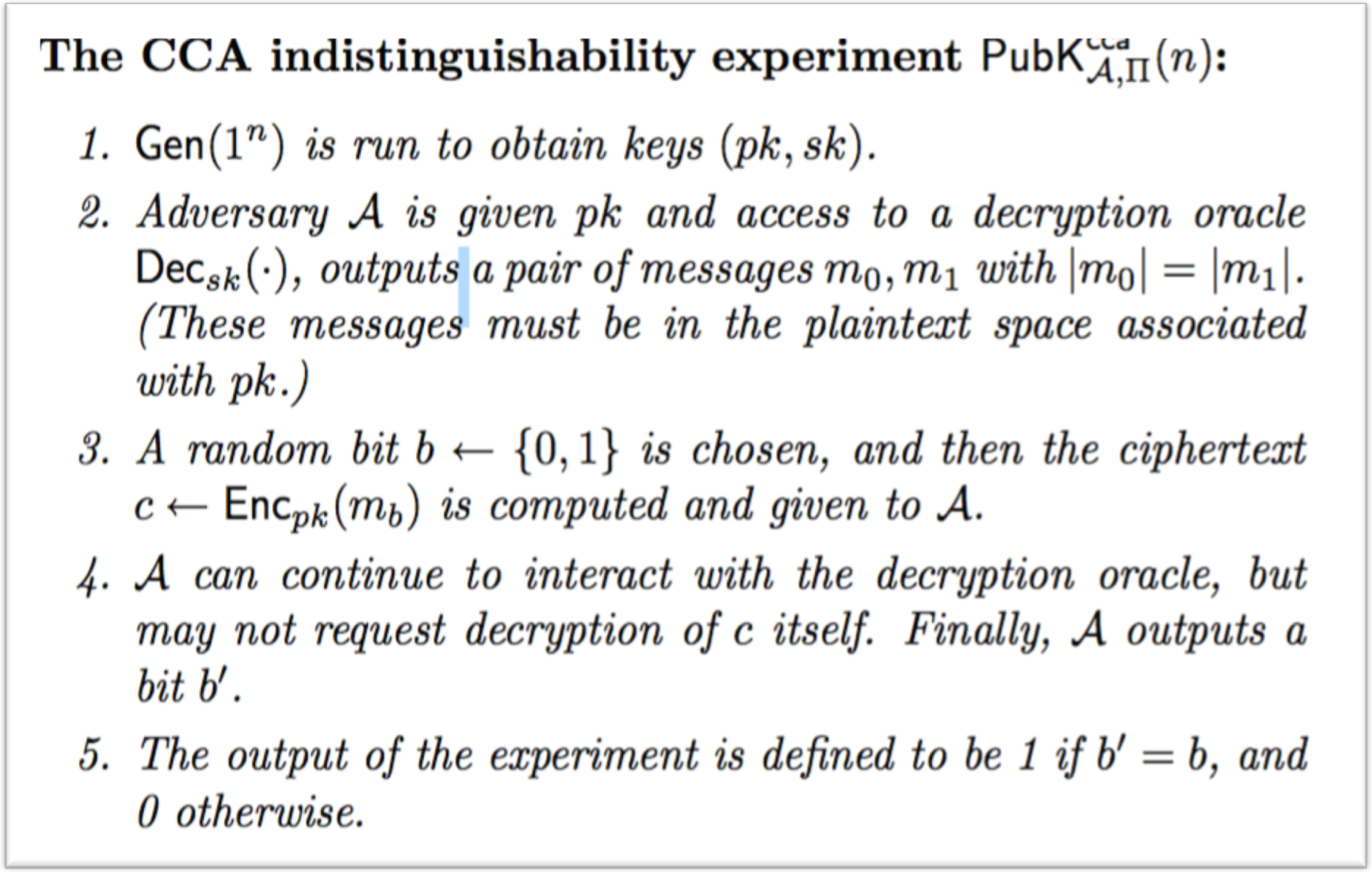

RSA encryption of *K*
Message, AES-CTR encrypted with *K*
ECDSA Signature by sender
RSA encryption of *K*
Mutated Ciphertext
ECDSA Signature by ATTACKER

Ciphertext malleability


RSA encryption of *K*
Message, AES-CTR encrypted with *K*
ECDSA Signature by sender
RSA encryption of *K*
Flip bits in AES ciphertext
RSA encryption of *K*
Mutated Ciphertext
ECDSA Signature by ATTACKER

Chosen Ciphertext Attack

- Attacker can query on any ciphertext but challenged one
- "Who would build such a system?"



The CCA indistinguishability experiment $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$**:**

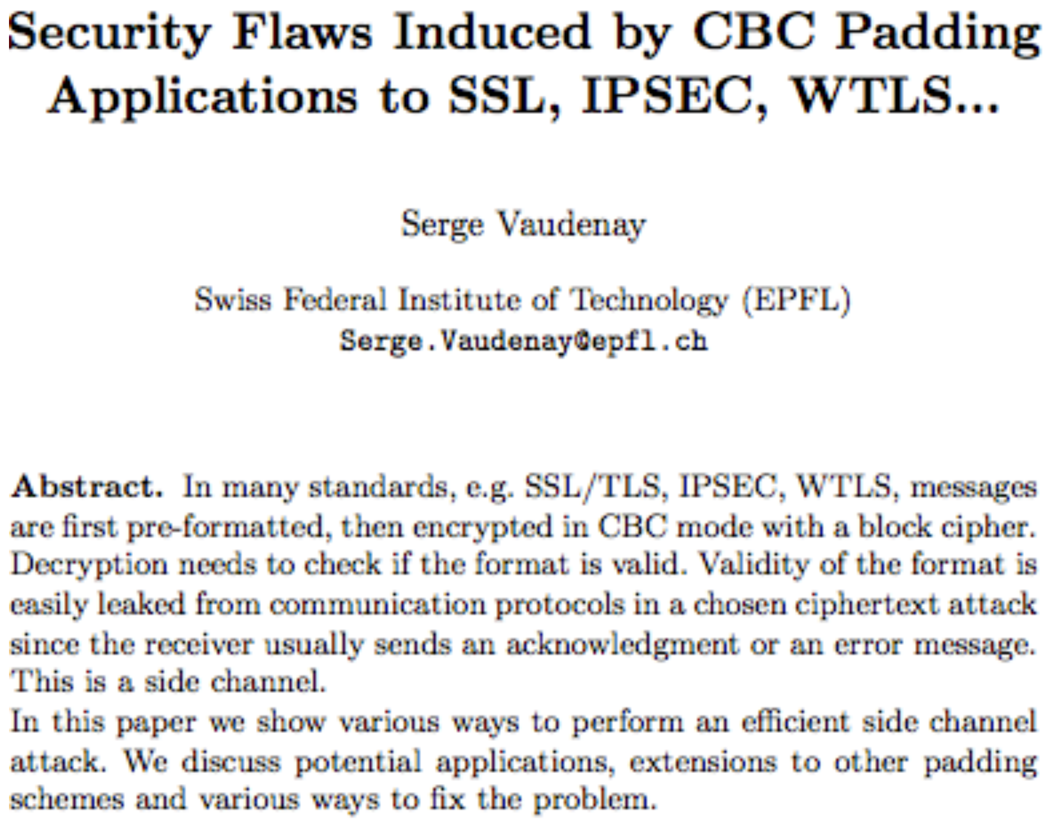
1. $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).
2. *Adversary* $\mathcal{A}$ *is given* pk *and access to a decryption oracle* $\mathsf{Dec}_{sk}(\cdot)$, *outputs a pair of messages* m_0, m_1 *with* $|m_0| = |m_1|$. *(These messages must be in the plaintext space associated with* pk.*)*
3. *A random bit* $b \leftarrow \{0,1\}$ *is chosen, and then the ciphertext* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ *is computed and given to* $\mathcal{A}$.
4. $\mathcal{A}$ *can continue to interact with the decryption oracle, but may not request decryption of* c *itself. Finally,* $\mathcal{A}$ *outputs a bit* b'.
5. *The output of the experiment is defined to be 1 if* $b' = b$, *and 0 otherwise.*

From Modern Cryptography Katz and Lindell

“Format Oracles”

- Suppose instead of decrypting the message, the server tells us if it is valid?
- E.g. Is the message the right length
- Or if it is encoded/serialized incorrectly

Does happen in the real world

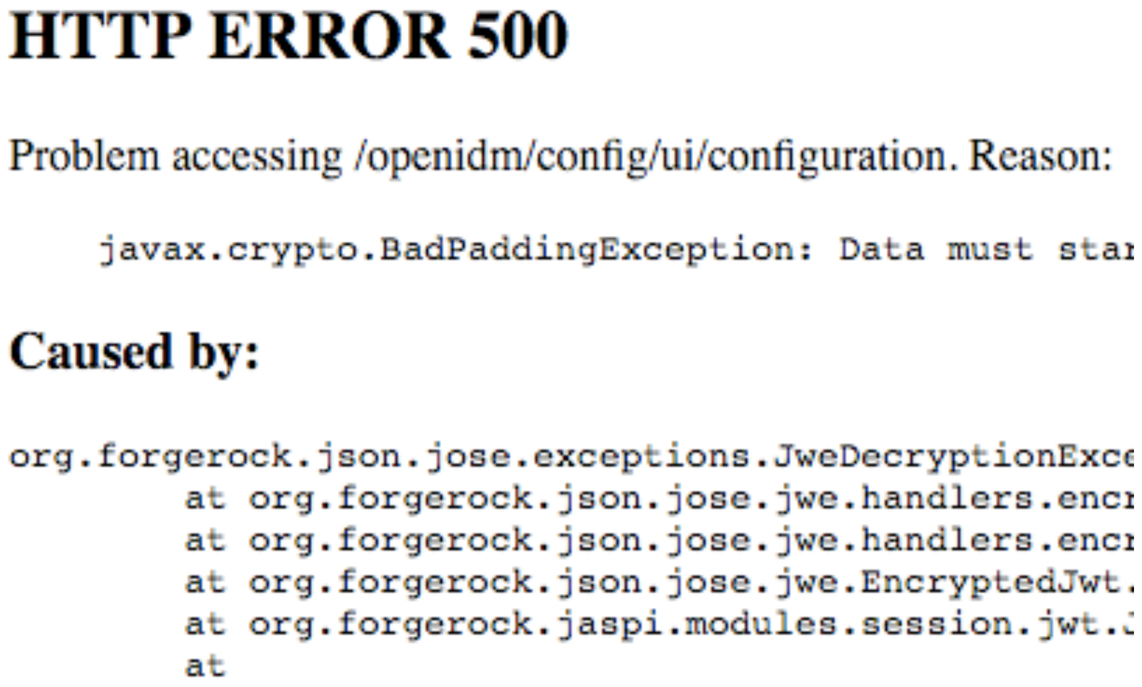


Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...

Serge Vaudenay

Swiss Federal Institute of Technology (EPFL)
Serge.Vaudenay@epfl.ch

Abstract. In many standards, e.g. SSL/TLS, IPSEC, WTLS, messages are first pre-formatted, then encrypted in CBC mode with a block cipher. Decryption needs to check if the format is valid. Validity of the format is easily leaked from communication protocols in a chosen ciphertext attack since the receiver usually sends an acknowledgment or an error message. This is a side channel.
In this paper we show various ways to perform an efficient side channel attack. We discuss potential applications, extensions to other padding schemes and various ways to fix the problem.



HTTP ERROR 500

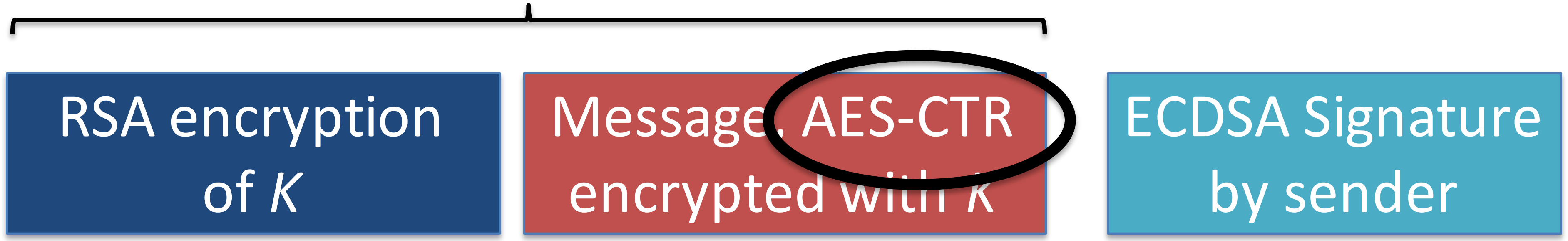
Problem accessing /openidm/config/ui/configuration. Reason:

```
    javax.crypto.BadPaddingException: Data must star
```

Caused by:

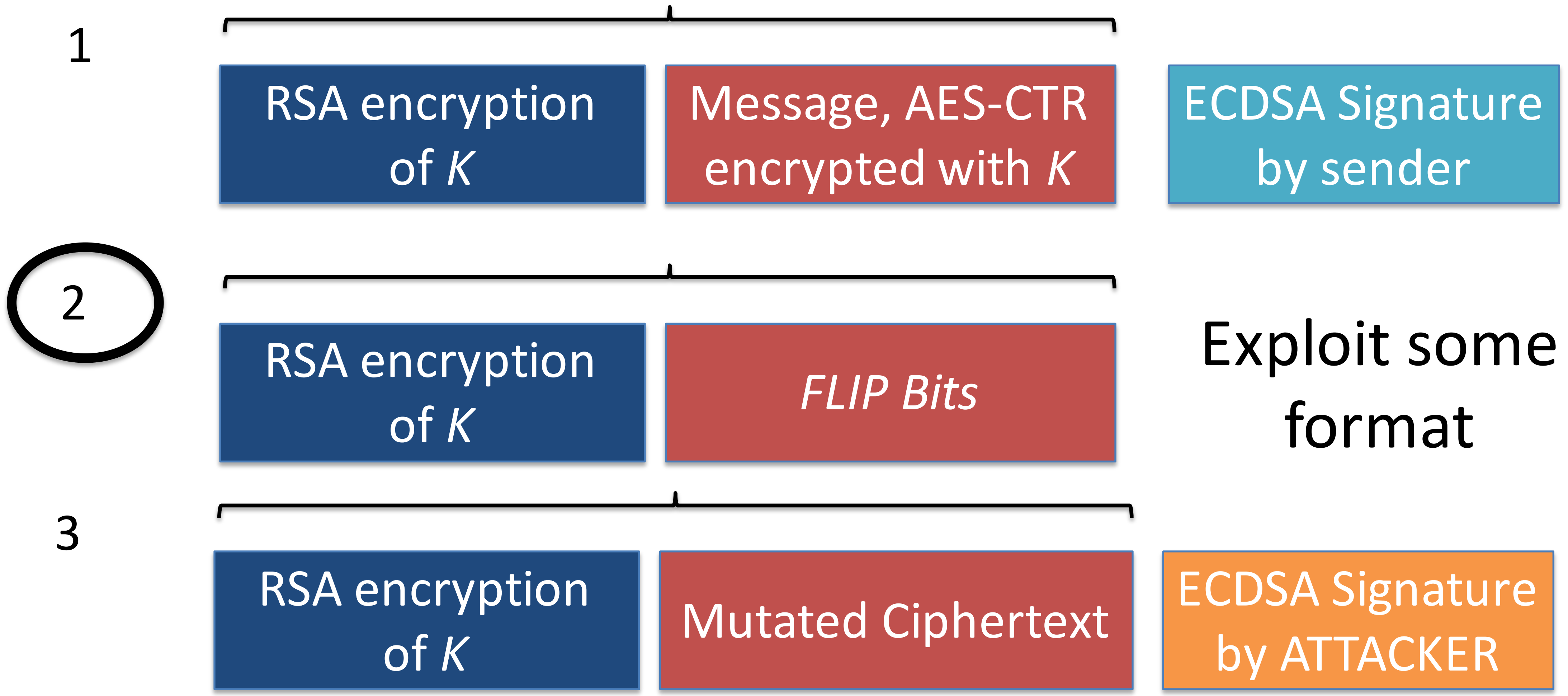
```
org.forgerock.json.jose.exceptions.JweDecryptionExce
        at org.forgerock.json.jose.jwe.handlers.encr
        at org.forgerock.json.jose.jwe.handlers.encr
        at org.forgerock.json.jose.jwe.EncryptedJwt.
        at org.forgerock.jaspi.modules.session.jwt.J
        at
```

iMessage: No padding, No XML, etc.


RSA encryption of *K*
Message, AES-CTR encrypted with *K*
ECDSA Signature by sender

Over the lip of the Volcano

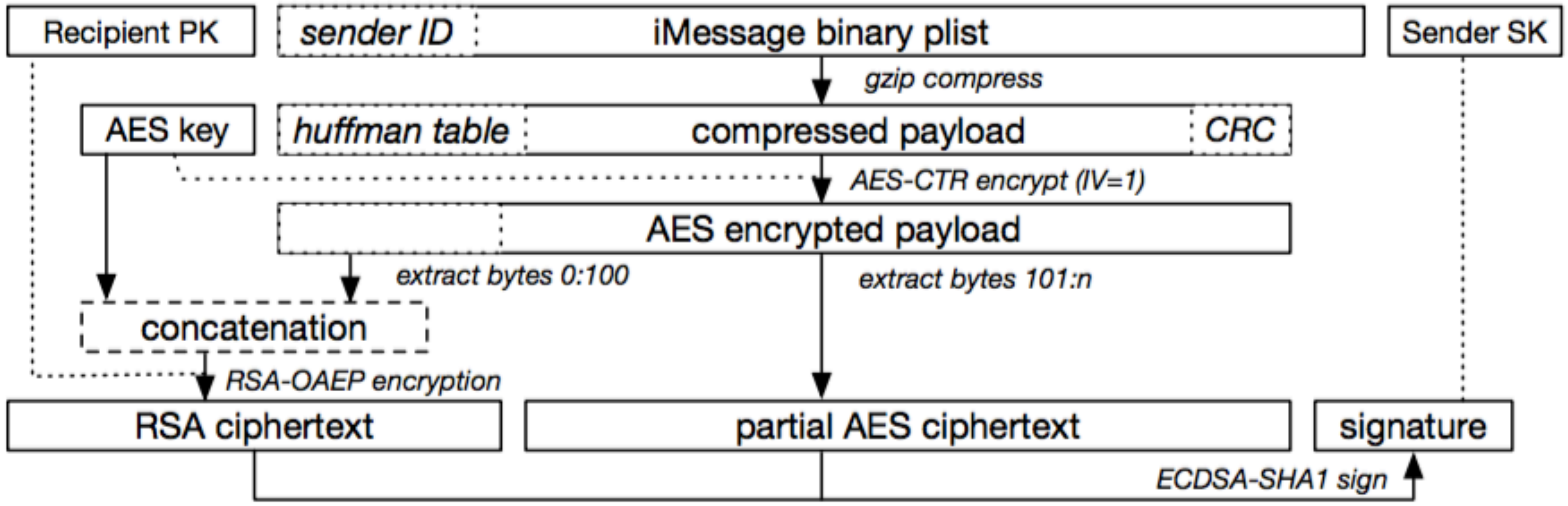
Finding a new format oracle



4 Check if mutated ciphertext decrypts and validates

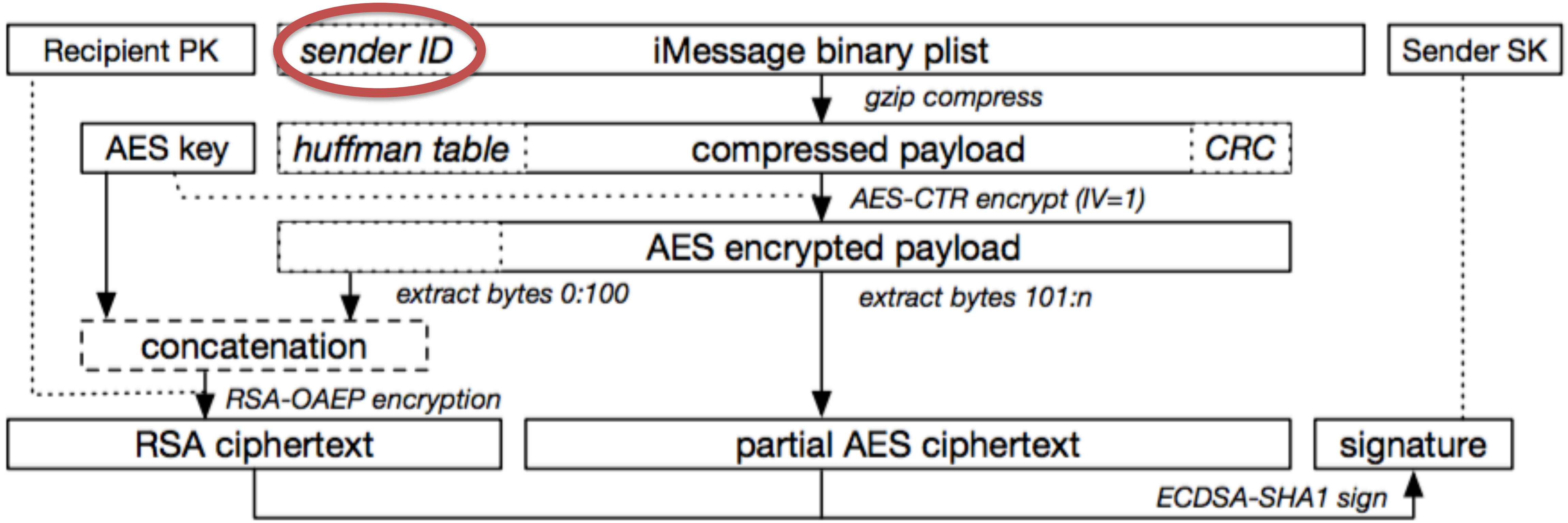
iMessage Format: What's in the box?

- Builds on a partial RE by Quarks Lab
- Ciphertext is a GZIP compressed binary plist
- Part of the message is put in the RSA ciphertext to save space



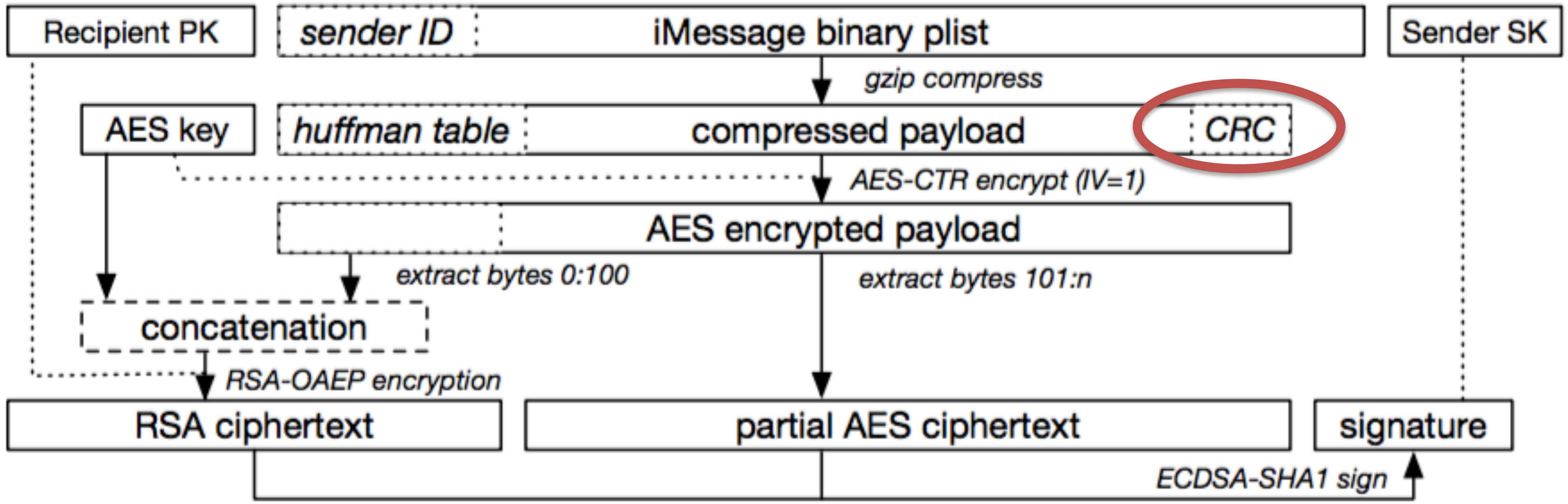
Countermeasure

- The sender ID is stored in the ciphertext
- Client rejects if internal sender ID does not match external ID
- Luckily, the ciphertext is malleable!!



GZIP: another catch

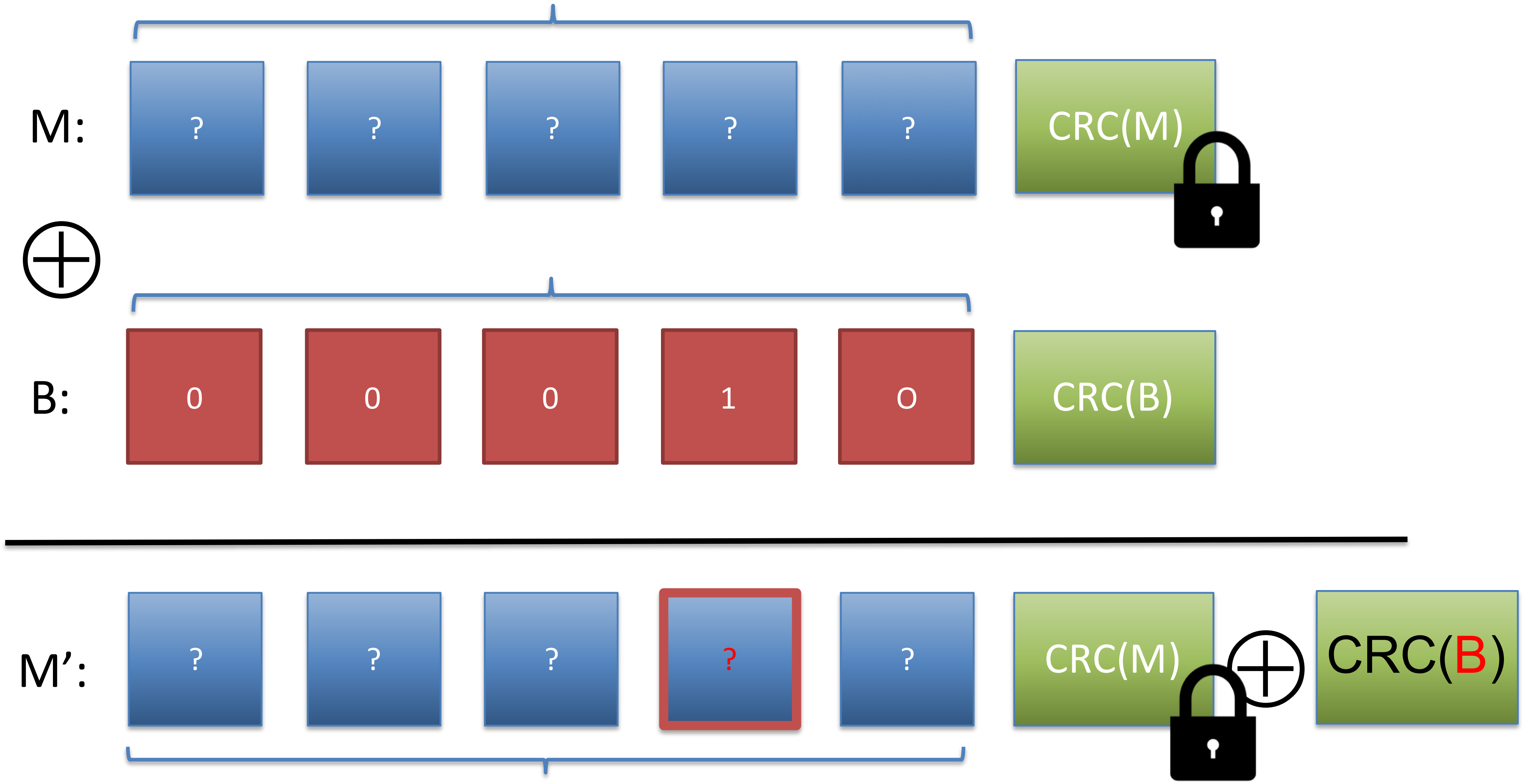
- HEADER + compressed message + CRC32
- CRC checksum of decompressed message
- Decompression fails if the checksum is wrong!



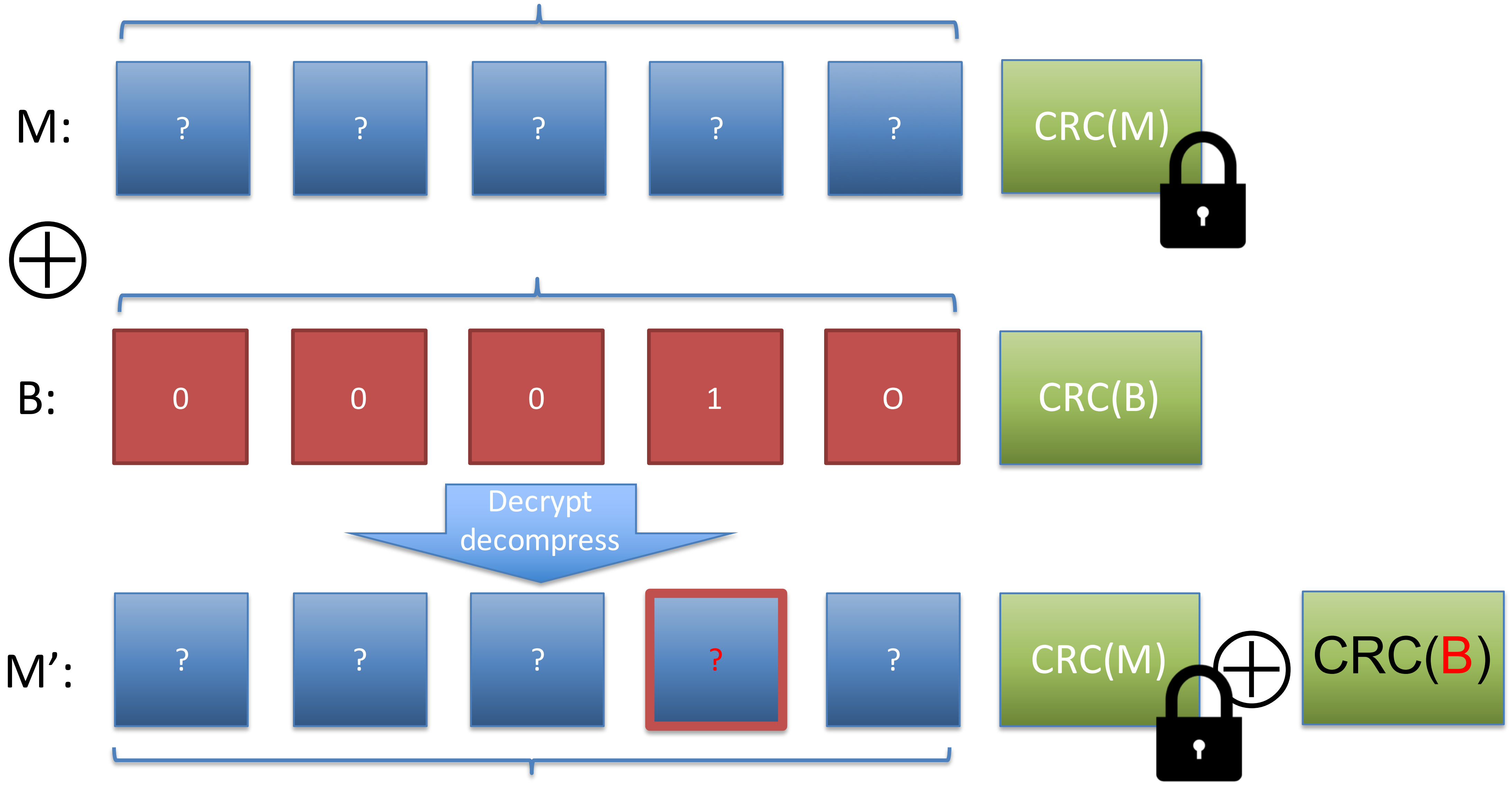
Fun with CRC32

- Interesting mathematical fact: $\mathrm{CRC}(a) \oplus \mathrm{CRC}(b) = \mathrm{CRC}(a \oplus b)$
- {slightly different for non-zero IVs}

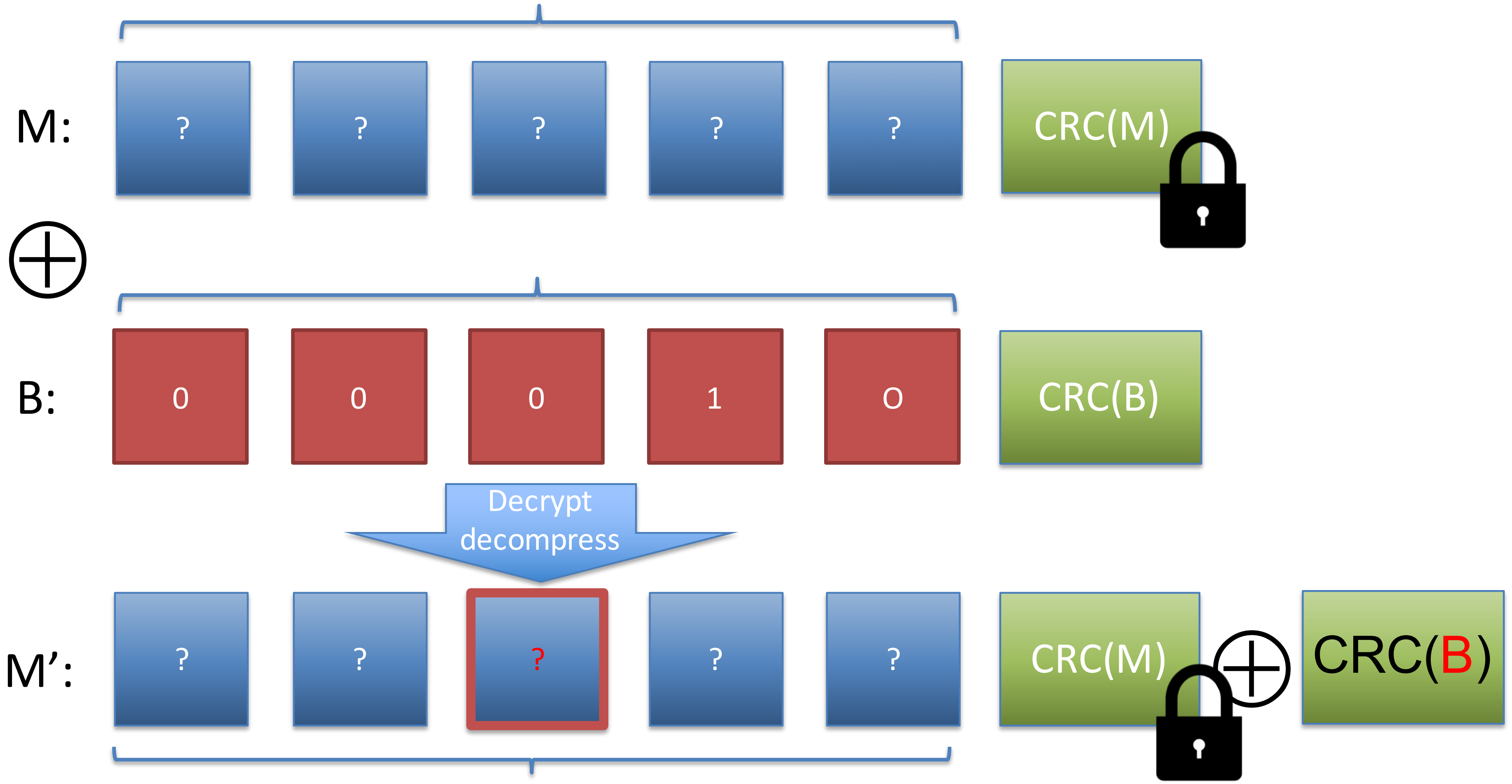
Correcting CRCs for bit flips



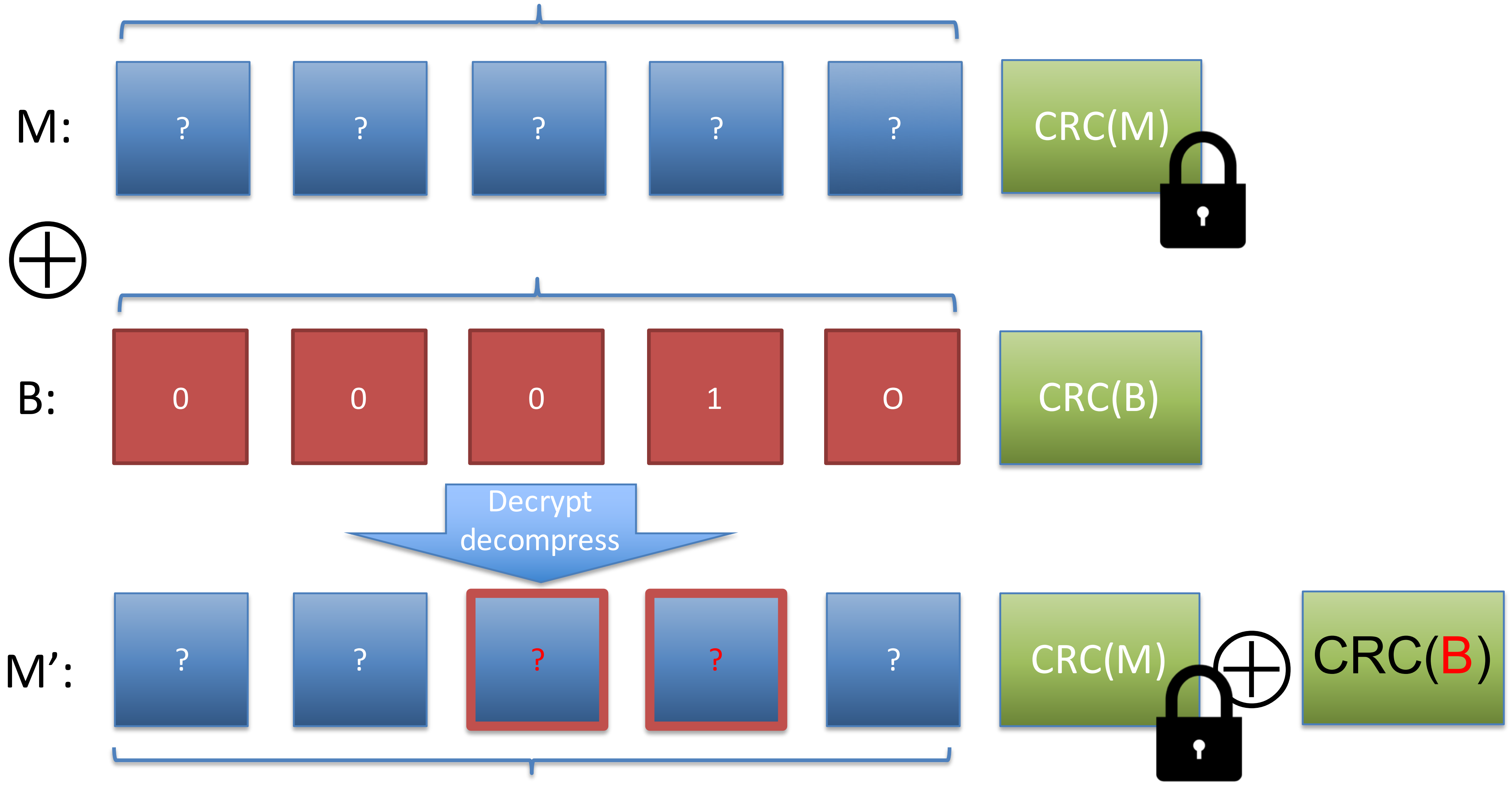
There’s a catch



There's a catch



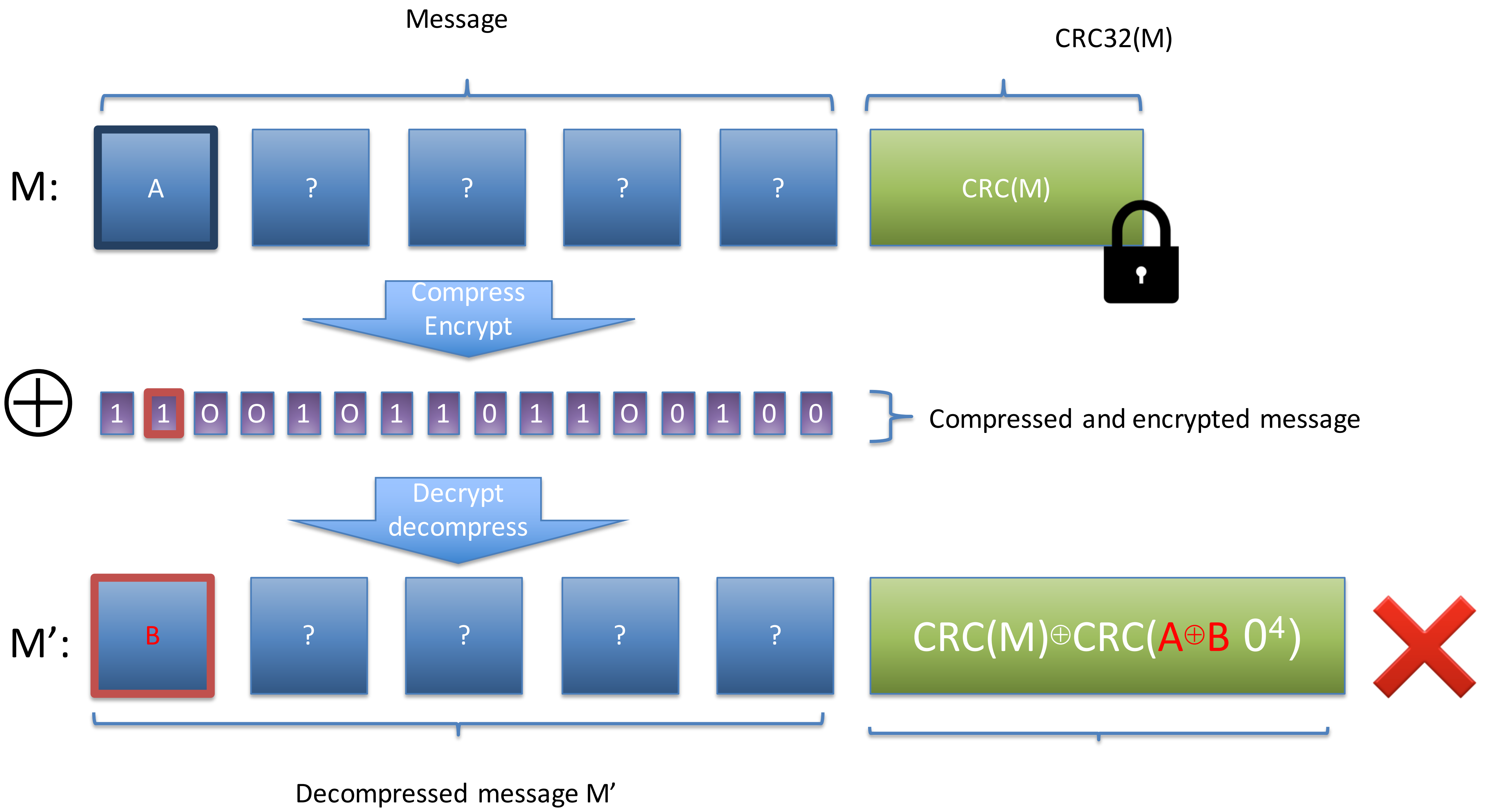
There's a catch

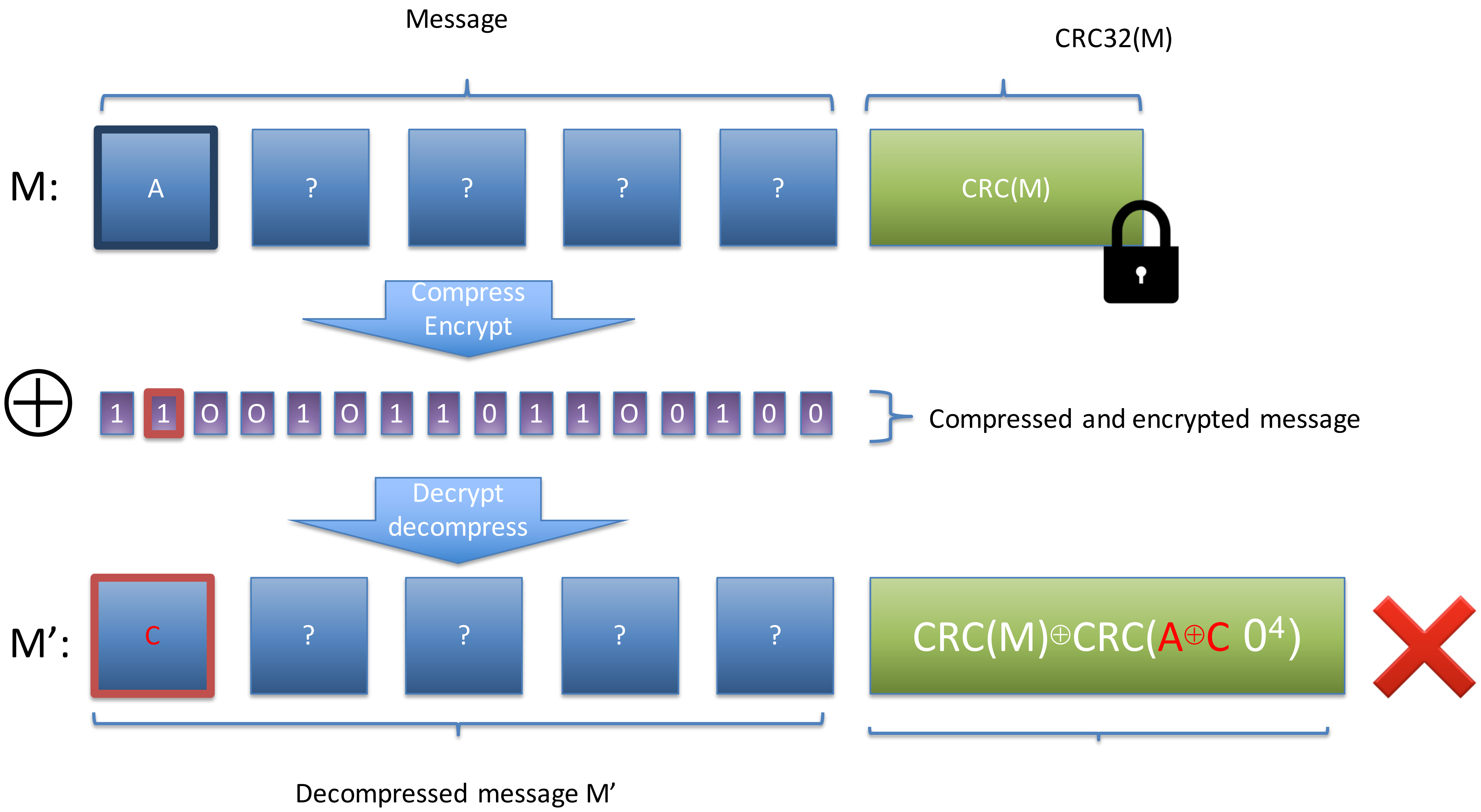


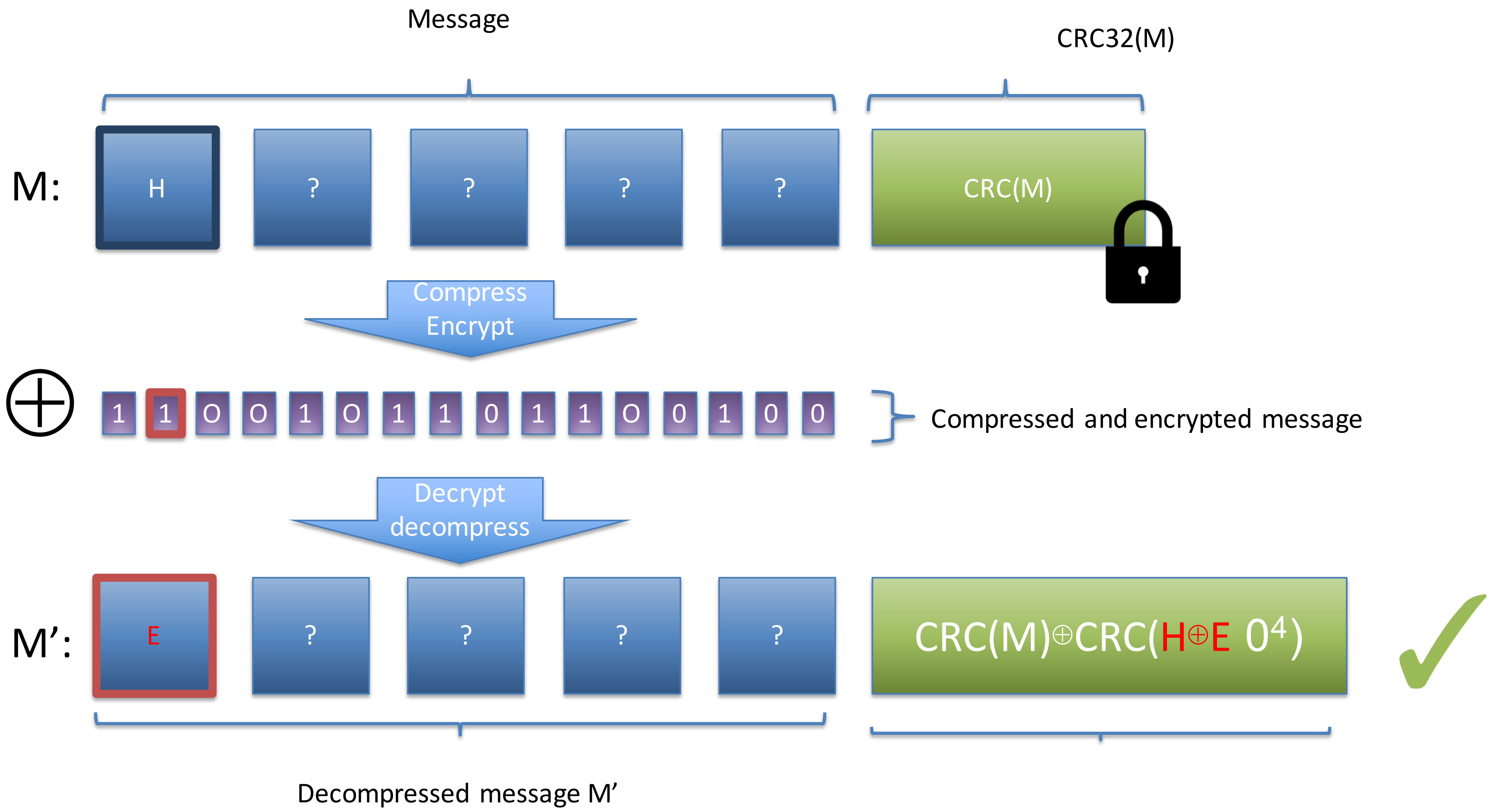
DEFLATE

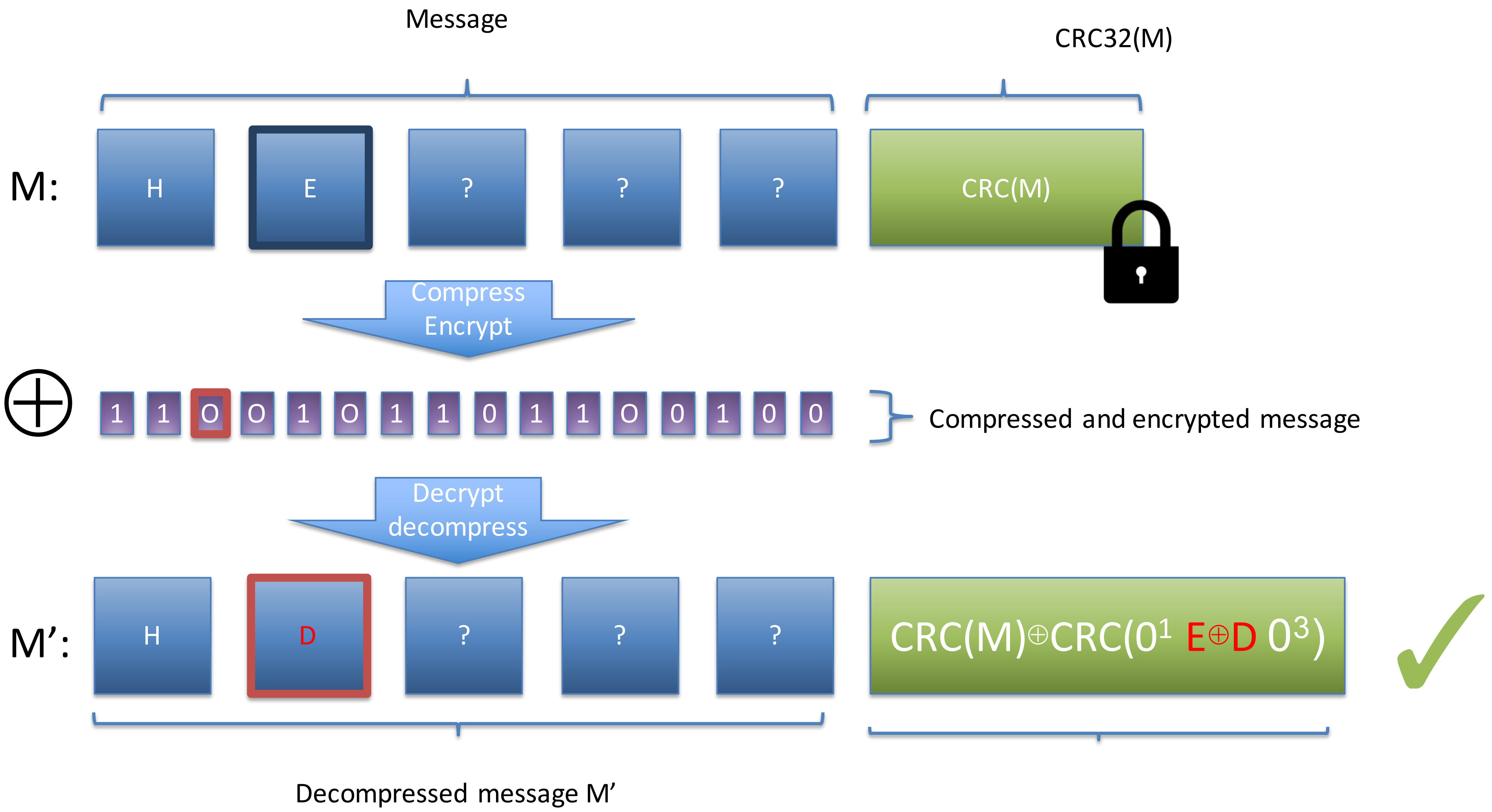
- GZIP uses DEFLATE for compression
- DEFLATE is
  - Lempel – Ziv encoding for repeated strings
  - Huffman coding of the resulting stream
- Flipping a bit in a Huffman symbol MAY NOT flip the same bit in the decoded character

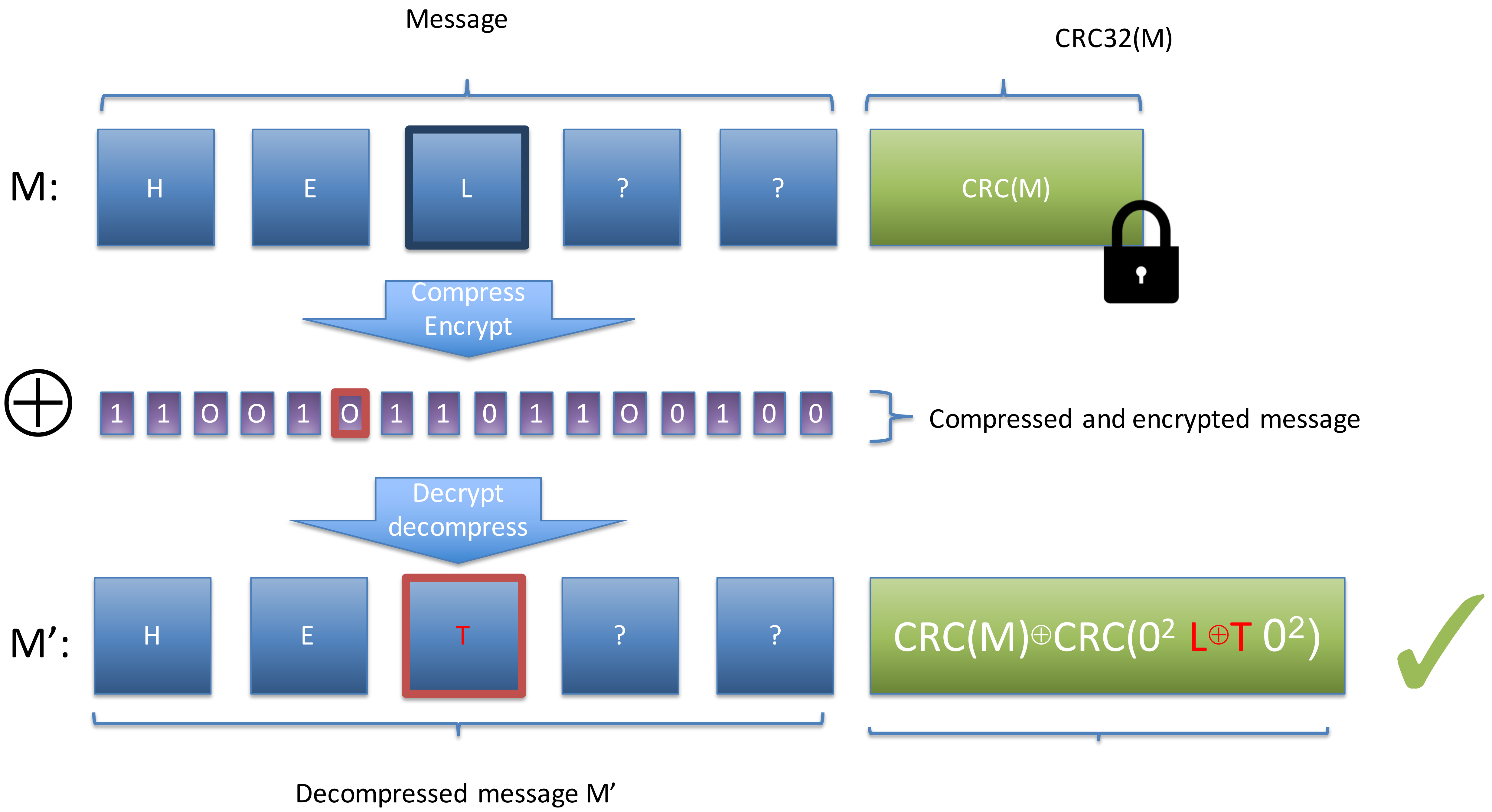
	Huffman Symbol	ASCII
E	11	01000101
H	10	01001000
I	101	01001001

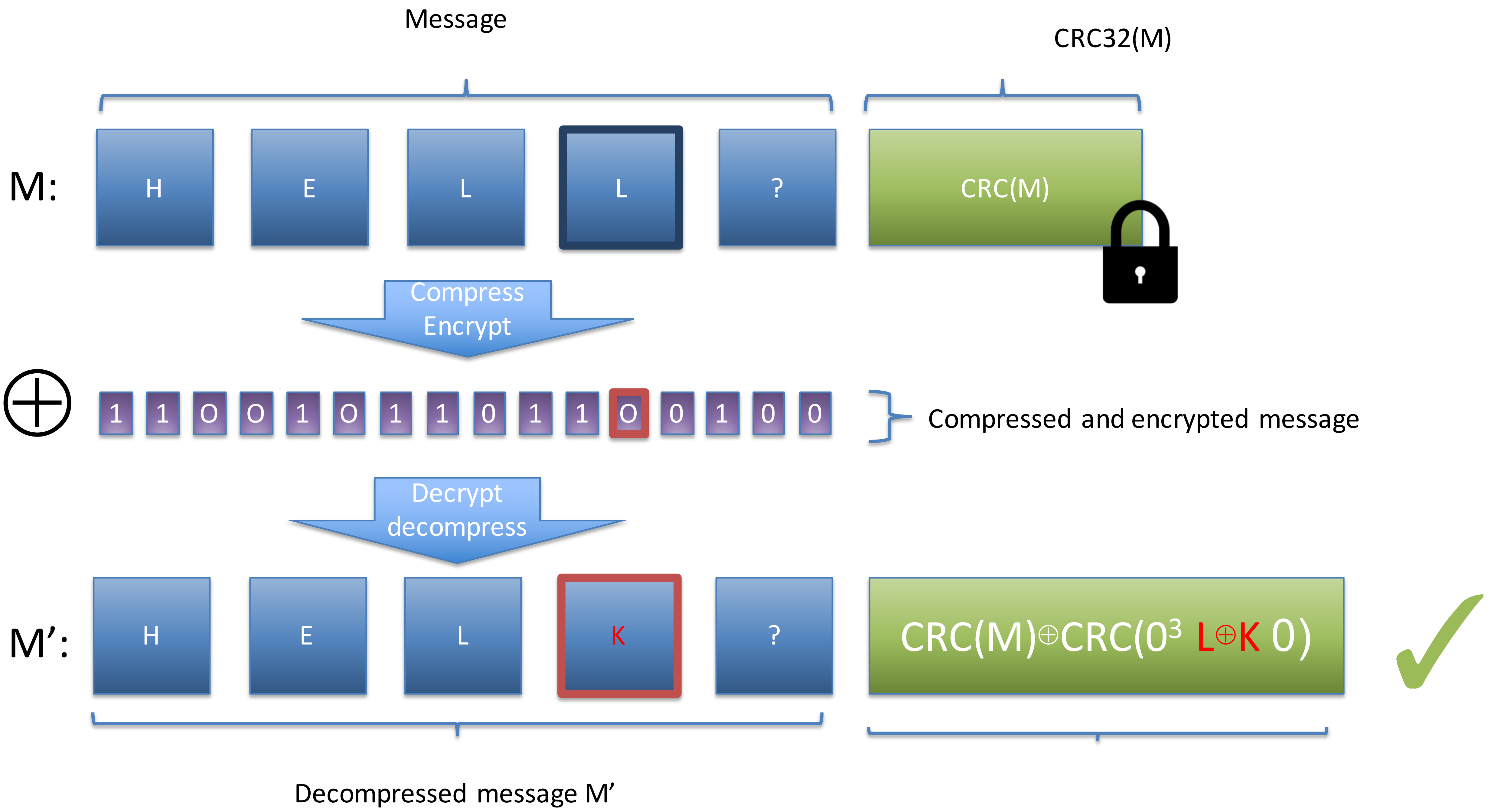

Message
CRC32(M)
M:
A
?
?
?
?
CRC(M)
Compress
Encrypt
⊕
1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
B
?
?
?
?
CRC(M)⊕CRC(A⊕B 0⁴)
Decompressed message M'

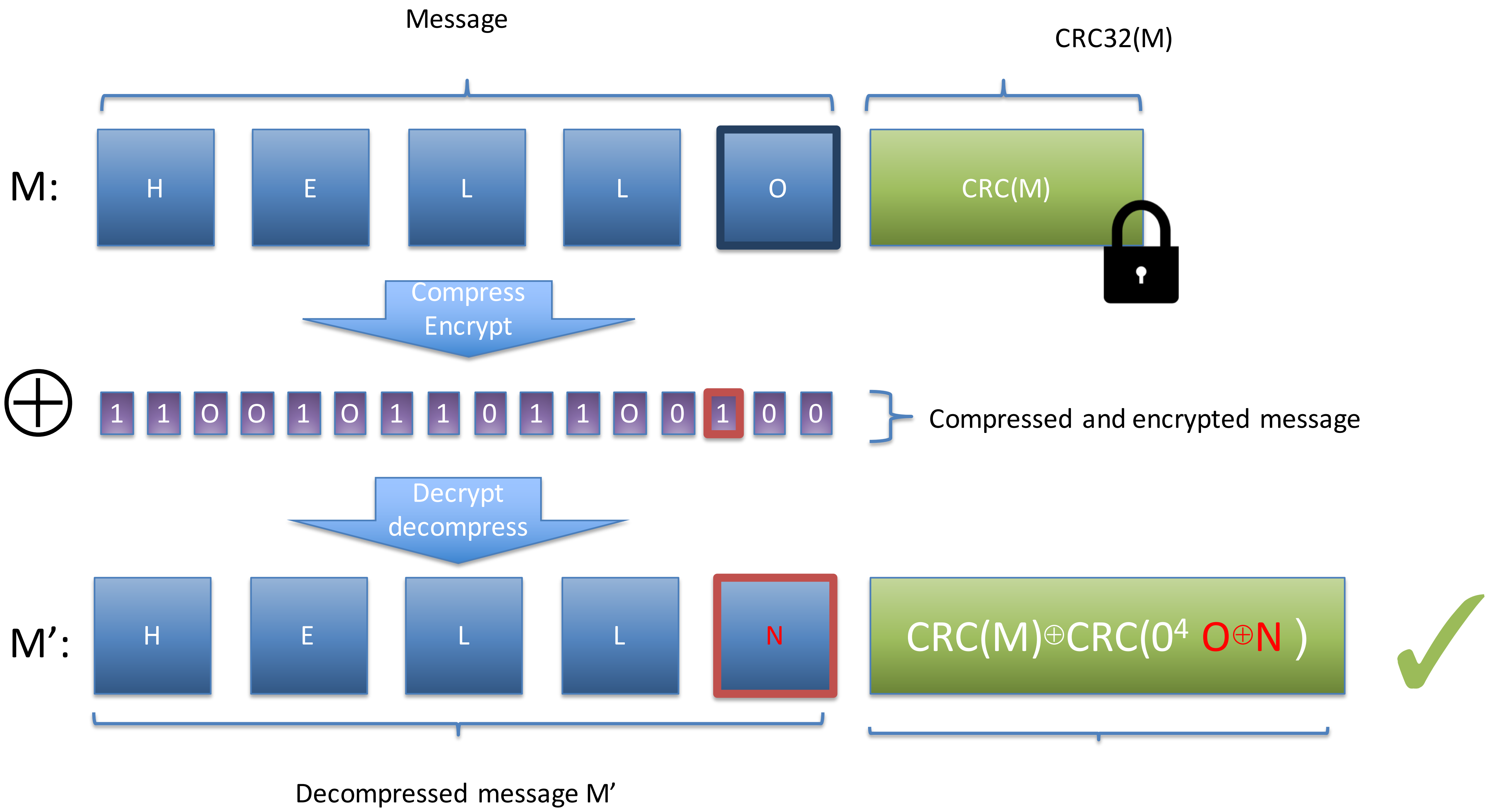

Message
CRC32(M)
M:
A
?
?
?
?
CRC(M)
Compress
Encrypt
⊕
1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
C
?
?
?
?
CRC(M)⊕CRC(A⊕C 0⁴)
Decompressed message M'


Message
CRC32(M)
M:
H
?
?
?
?
CRC(M)
Compress
Encrypt
1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
E
?
?
?
?
CRC(M)⊕CRC(H⊕E 0⁴)
Decompressed message M'

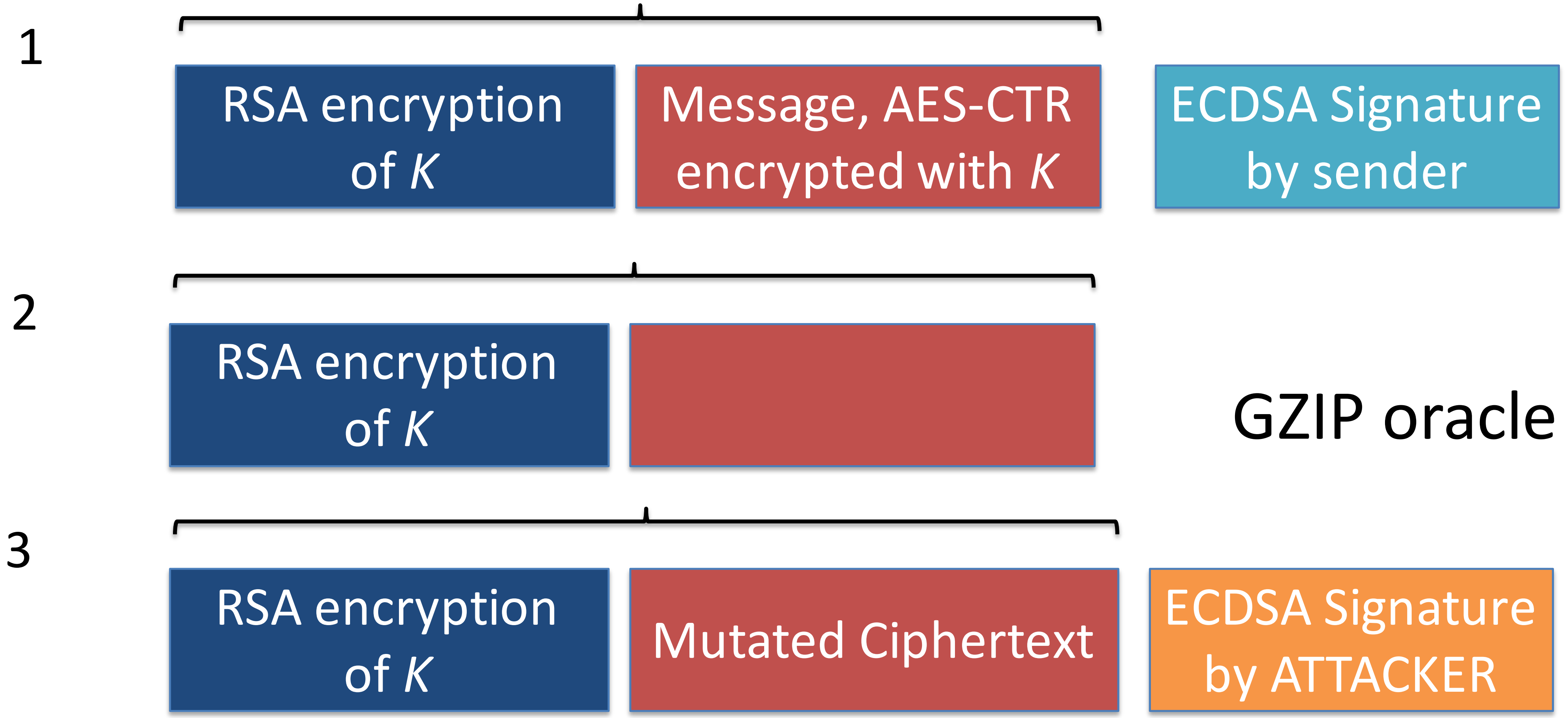

Message
CRC32(M)
M:
H
E
?
?
?
CRC(M)
Compress
Encrypt
⊕
1 1 O O 1 O 1 1 0 1 1 O 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
H
D
?
?
?
$CRC(M) \oplus CRC(0^1\ E \oplus D\ 0^3)$
✓
Decompressed message M'


Message
CRC32(M)
M:
H
E
L
?
?
CRC(M)
Compress
Encrypt
⊕
1 1 O O 1 O 1 1 0 1 1 O 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
H
E
T
?
?
CRC(M)⊕CRC(0² L⊕T 0²)
✓
Decompressed message M'


Message
CRC32(M)
M:
H
E
L
L
?
CRC(M)
Compress
Encrypt
⊕
1 1 O O 1 O 1 1 0 1 1 O 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
H
E
L
K
?
CRC(M)⊕CRC(0³ L⊕K 0)
✓
Decompressed message M'


Message
CRC32(M)
M:
H
E
L
L
O
CRC(M)
Compress
Encrypt
⊕
1 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0
Compressed and encrypted message
Decrypt
decompress
M':
H
E
L
L
N
CRC(M)⊕CRC(0⁴ O⊕N)
Decompressed message M'

Compression oracle?

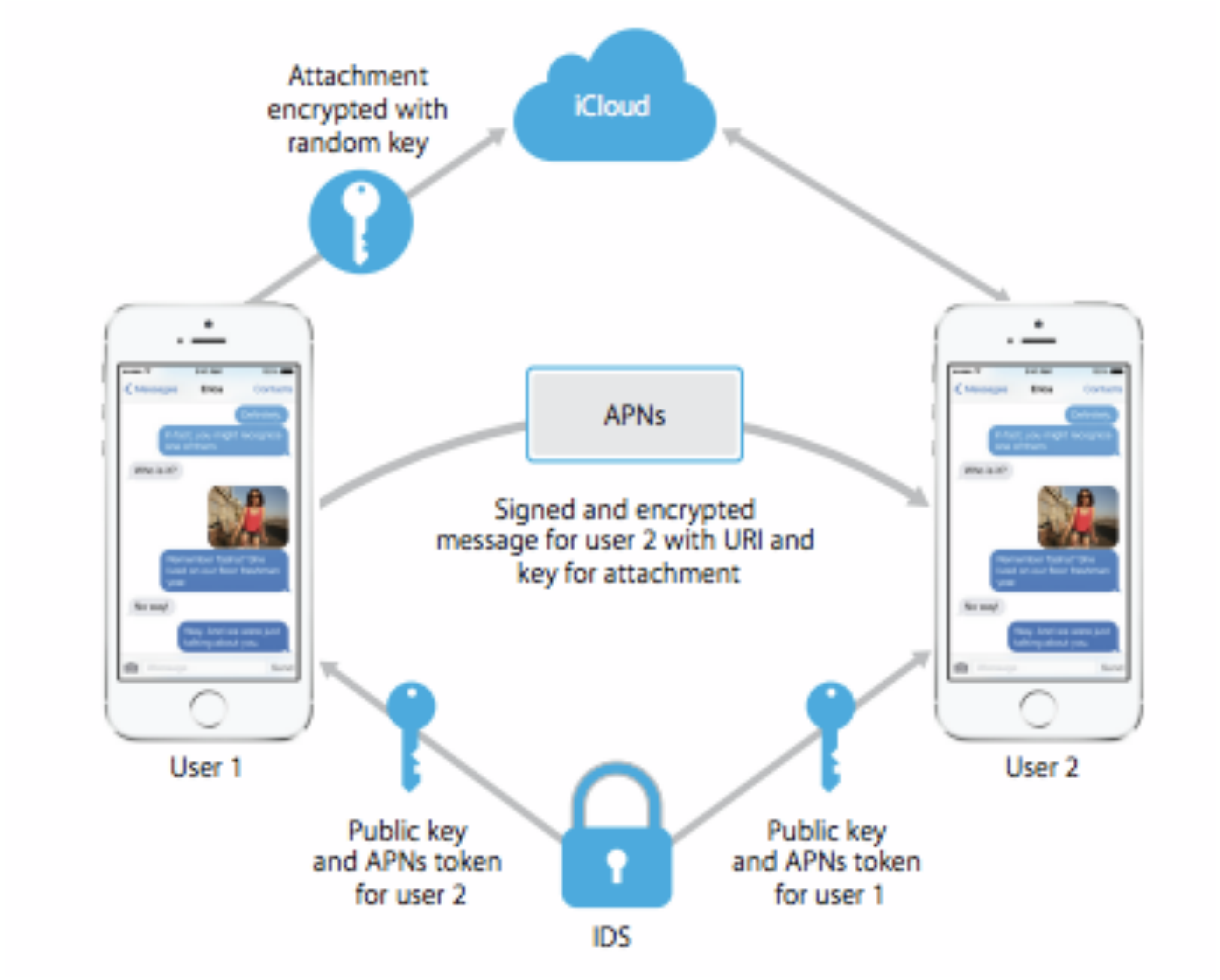


4 Check if mutated ciphertext decrypts and validates

Observing the oracle

- We need to see if when a message is received, it decompresses successfully
- iMessage does not report errors to the sender
- Read receipts require someone to see the message

iMessage



Attachment encrypted with random key
iCloud
APNs
Signed and encrypted message for user 2 with URI and key for attachment
User 1
User 2
Public key and APNs token for user 2
Public key and APNs token for user 1
IDS

Attachment messages

- Can see if message decompresses
- Requests block download response to hide message
- Can mutate message to point download request to attacker controlled server (e.g. i8loud.com)

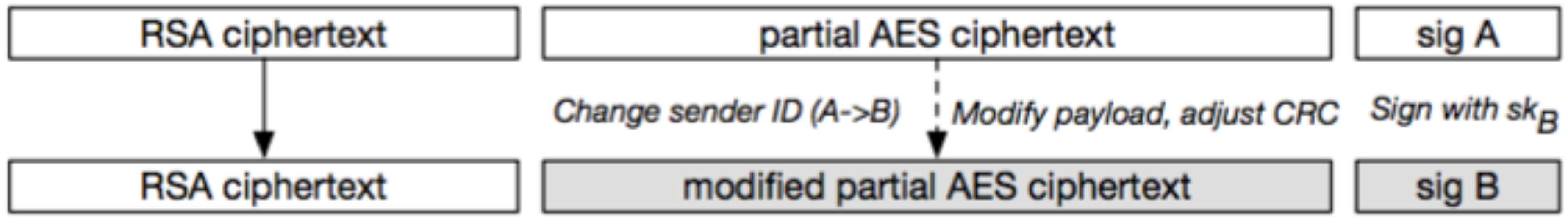
Attachment message payload

- {'gv': '8', 'pv': 0, 'p': ['mailto:alice.jhuisi@gmail.com', 'mailto:jhuisiscratch@gmail.com'], 'gid': 'A9CD06B6-6198-4289-A2C1-678B4E43ED77', 't': u'\ufffc', 'v': '1', 'x': '<html><body><FILE name="04duck.png" width="480" height="673" datasize="489847" mime-type="image/png" uti-type="public.png" mmcs-owner="MAB49B97D4B303E44942B4D05829B4F68012E577BBF0242A03E714F4B3F9D69CD.C01USN00" mmcs-url="https://p10-content.icloud.com/MAB49B97D4B303E44942B4D05829B4F68012E577BBF0242A03E714F4B3F9D69CD.C01USN00" mmcs-signature-hex="01AB6ED842CC96A19C19D1CF3FECA0CB37CE17B07D" file-size="489847" decryption-key="00F49B0E7388F578592FBB1618052675079DE82F0ABDE4BD5C4B2F5AF1426061DC"/> [OPTIONAL MESSAGE] </body></html>'}

Attack gets harder

- Attachment messages use a dynamic Huffman table which we don’t know
- We must recover the table
  - We basically have to edit known plaintext in the message
  - Variable length symbols, so we don’t know which decompressed byte we are affecting
  - Detect symbol edges (with high probability) with double bit flips

Complete Attack

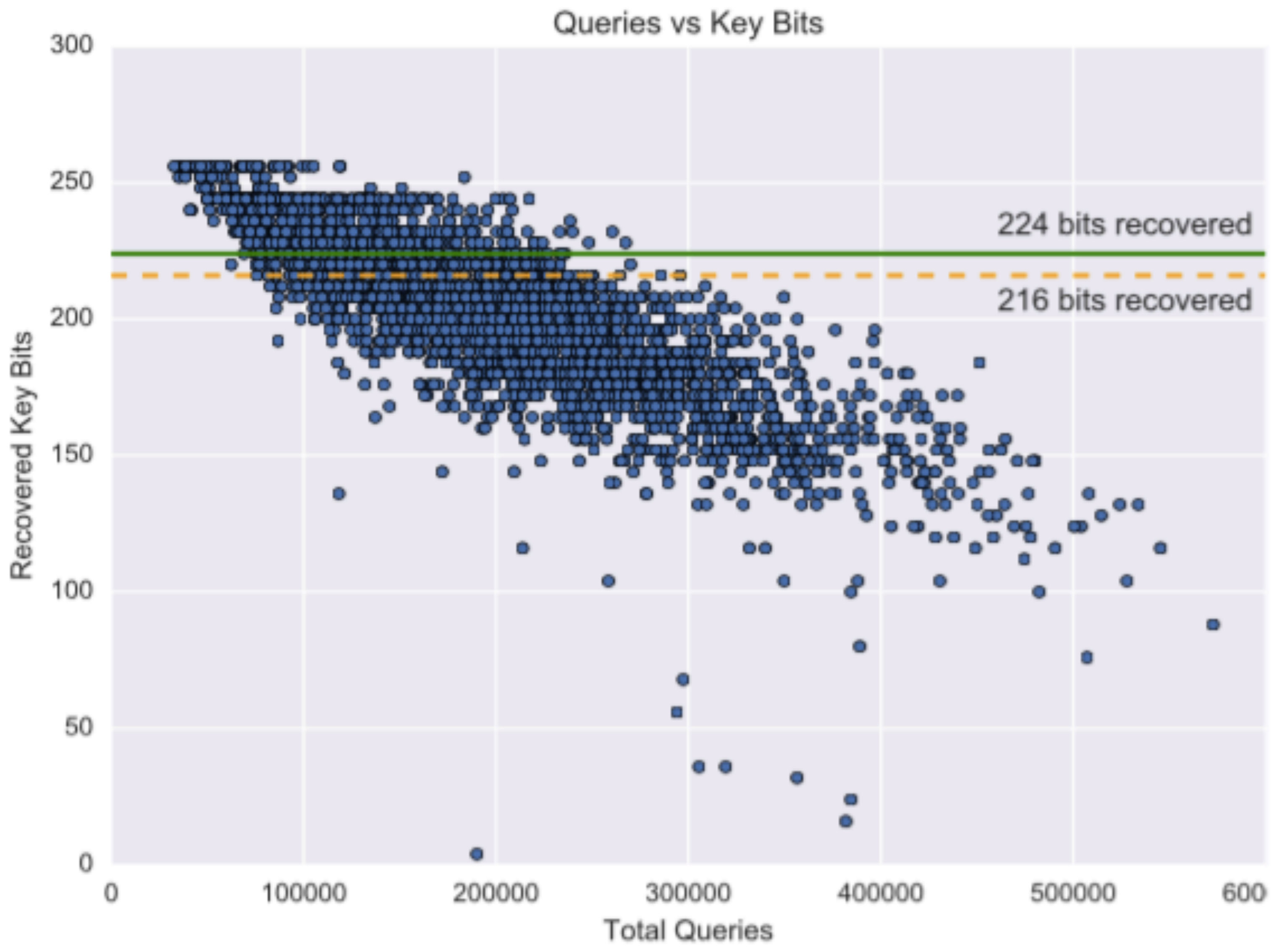


- Get message
- Change sender ID
- Use CRCs guess and check for chosen ciphertext attack to:
  - Recover Huffman table
  - Read attachment key
- Decrypt attachment with recovered key

Real attack

- Requires 2^18 oracle queries
- Long tail on message processing times with an upper bound of 1 second, average of 390 ms
- Takes 73 hours to execute attack (reducible to 35 hours via backtracking)
- Recovered 232 of 256 bits in the encryption key for the attachment

Simulating larger numbers of attacks



- Recovers all but 40 bits of the key for 34% of messages
  (brute force < 24 hour)

- Recovers all but 24 bits of key for 23% of messages
  (brute force < 1 hours)

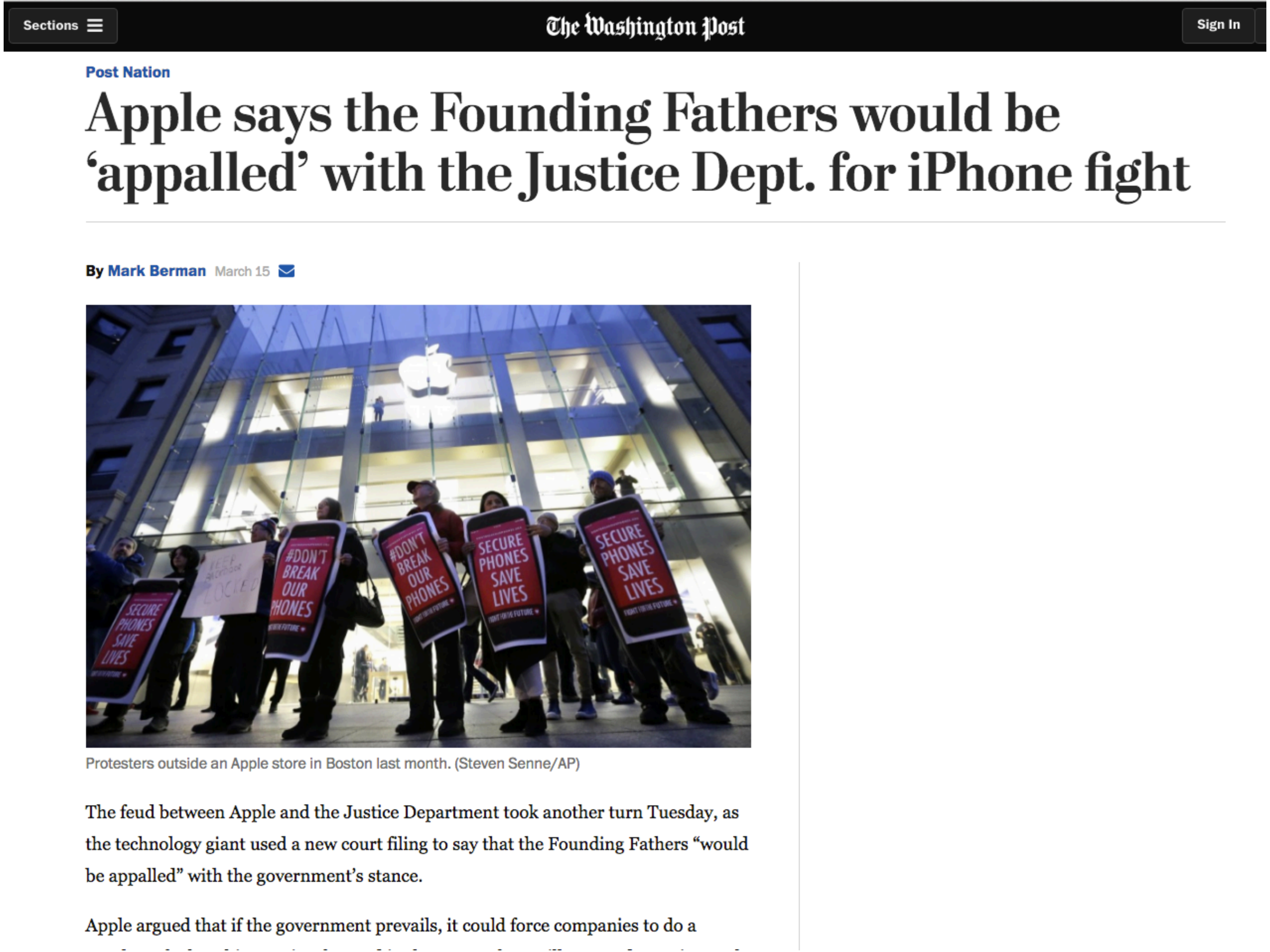
Ideal world solutions:

- Use Axolotl/Signal
- Just use authenticated encryption
  - AES-GCM/OCB
  - Include an HMAC
- Breaks backward compatibility
- Hard to do with 1 billion deployed devices

Real World Mitigations

Without breaking existing devices

- Recommended backward-compatible mitigations
  - Prevent the identity misbinding attack by moving sender ID to non-malleable RSA-OAEP ciphertext
  - Prevent chosen ciphertext attack by blacklisting RSA-OAEP ciphertexts that fail to decrypt
- RSA blacklisting deployed in IOS 9.3+ and OSX 10.11.4+
- Took Apple 4 months and 30 engineers to deploy
- Released on March 21, 2016



Sections ☰ The Washington Post Sign In

Post Nation

Apple says the Founding Fathers would be 'appalled' with the Justice Dept. for iPhone fight

By Mark Berman March 15

Protesters outside an Apple store in Boston last month. (Steven Senne/AP)

The feud between Apple and the Justice Department took another turn Tuesday, as the technology giant used a new court filing to say that the Founding Fathers "would be appalled" with the government's stance.

Apple argued that if the government prevails, it could force companies to do a

This shaky edifice could crumble at any moment

Questions?