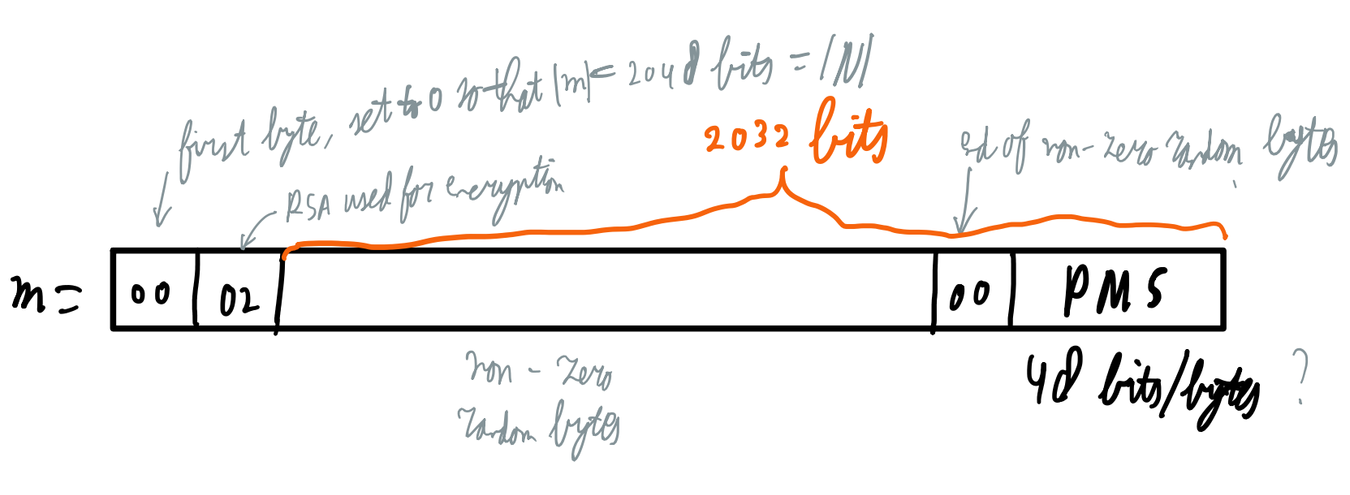
Lecture 6

dinsdag 29 november 2022 10:37

how does noise protocol fit on TLS ?

Bleichenbacher's padding oracle attack

→ RSA - PKCS #1 v1.5 → padding:



pre-master secret encrypted with RSA

homomorphic $c_1 c_2 = e_c(m_1) e_c(m_2) = e_c(m_1 m_2)$

client: sends $c \equiv m^e \pmod n$ to server

attacker: choose s, sends $s^e \cdot c \pmod n$ to server

with probability of $\approx \frac{1}{2^{16}}$, plaintext will start with 0002 and then has a zero-byte somewhere

server responds with alert if padding error

other if no padding error

then: $(s^e \cdot c)^d \equiv s \cdot m \pmod n$

$2 \cdot 2^{2032} < s \cdot m < 3 \cdot 2^{2032}$

min union of intervals, total (relative → compared with initial length) length: $\frac{2^{2032}}{2^{2048}} = \frac{1}{2^{16}}$

SSL strip: remove https from links

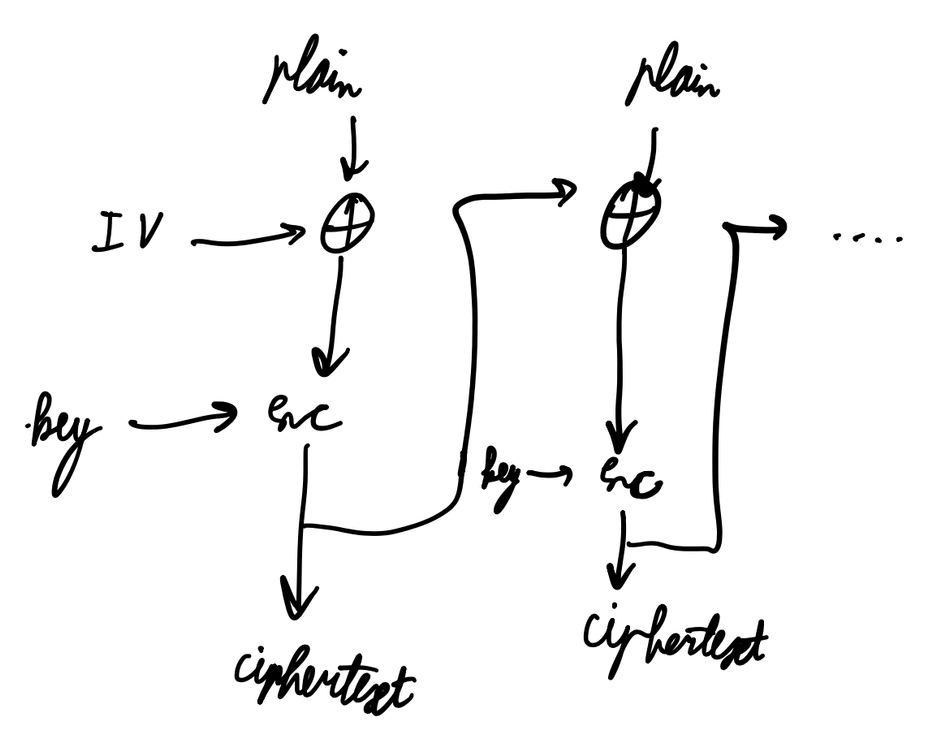
and change HTTP 302 redirects (from http to https) to http destinations

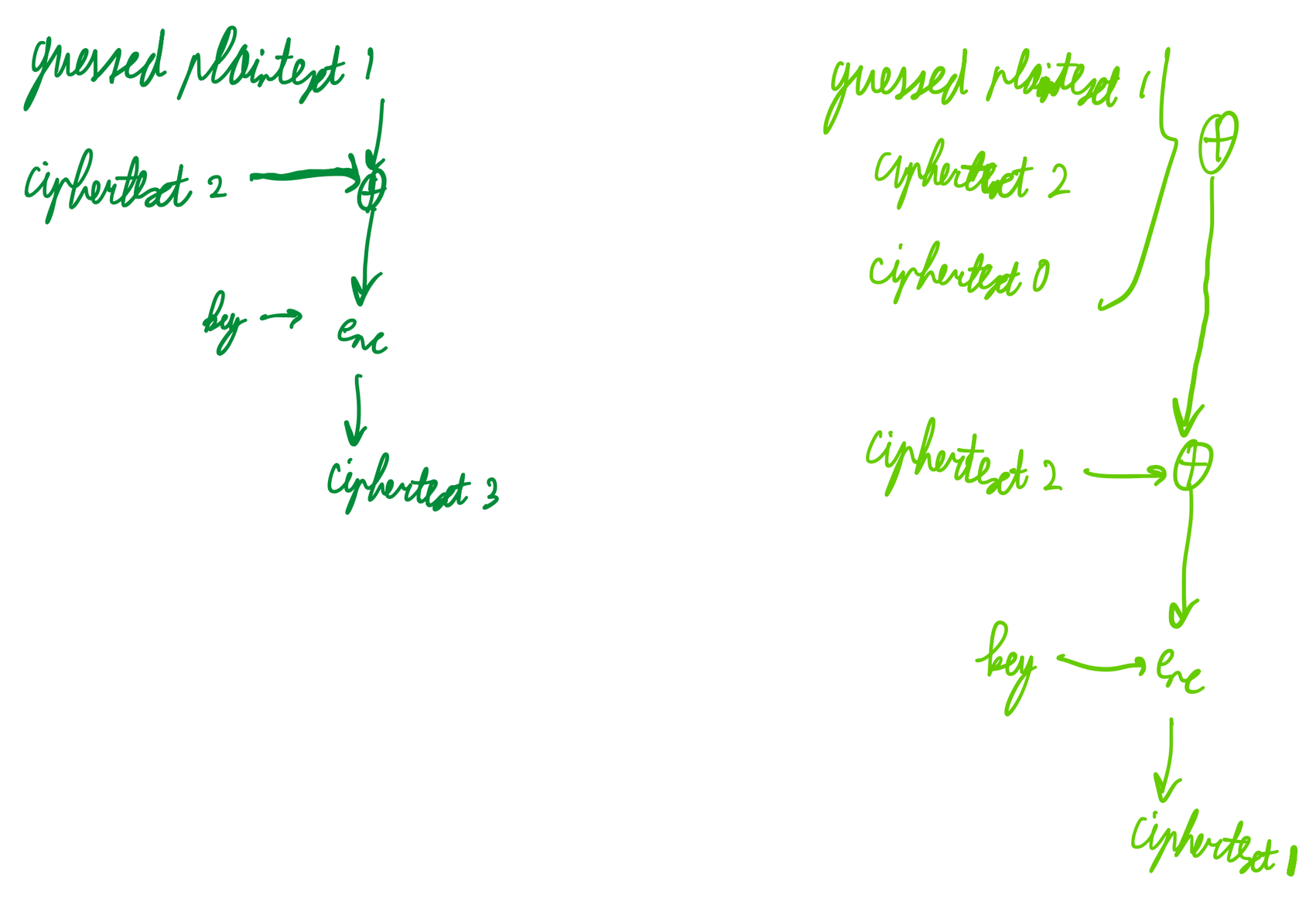
countermeasures: HSTS

BEAST attack

cipher block chaining mode

CBC mode





browser exploit against SSL/TLS

'force' client to send deliberate data, steal e.g. cookies byte-by-byte

CRIME: compressed ratio info-leak made easy

exploit compression; second reference to same data will be compressed as back-reference

↓

good guess for secret value ⇒ smaller compression

no compression in TLS 1.3

↓

BREACH → apply compression trick to HTTP content (still compressed) instead of TLS content (not compressed anymore)

padding oracle: pad data with (length of padding) before using AES-CBC to get correct blocksize

at server: decrypt,
check padding, → else: decryption failed
check MAC → else: bad_record_mac

attacker learns whether last bytes of a plaintext were a valid pad

can find plaintext in 16 · 256 chosen ciphertext queries

works well for IMAP (password sent often)

solution: do not give (detailed) error messages

and watch out for timing attacks

Lucky13 attack: exploit timing differences

best solution: use authenticated encryption

POODLE: downgrade TLS version using e.g. man-in-the-middle

do not implement insecure cipher suites

RC4: many biases/problems → do not use!

perhaps NSA can decrypt this?

FREAK: in past: only weak keys e.g. RSA-512 were used due to export restrictions

but may still be in use

do not use/implement weak keys/ciphers

LOGJAM: similar (to FREAK), but with DH

standard primes are often used...

Heartbleed: OpenSSL memory leak in messages for finding out whether server is still alive

Sweet 32: find colliding ciphertext with known plaintext, then XOR out the CBC mode ciphertext difference

$C_i = C_j$

$P_i = P_j \oplus C_{j-1} \oplus C_{i-1}$

remove 3DES

DROWN: Bleichenbacher after downgrade

do not distinguish crypto failure cases

do not use vulnerable crypto

do not support broken crypto/protocols for compliance

do not re-use keys in different protocols