Secure Sockets Layer

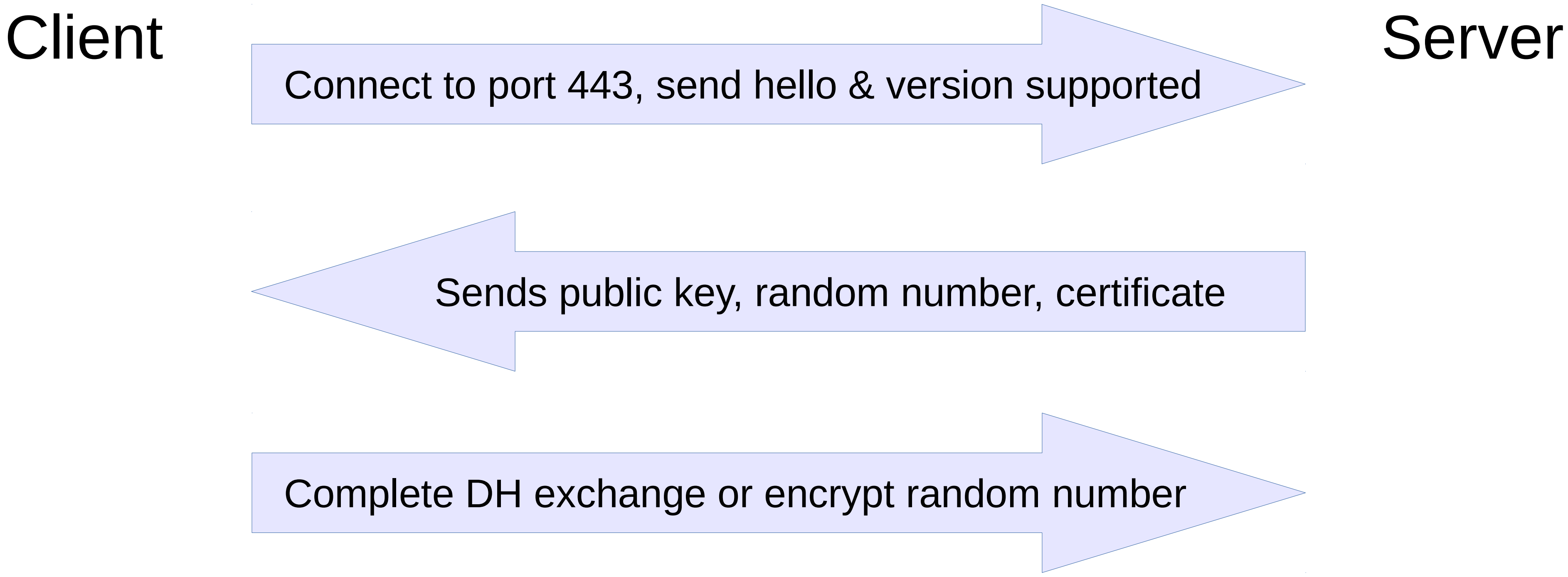
and

Transport Layer Security

Secure Sockets Layer

- Introduced by Netscape in 1995
- Initially a closed proprietary protocol
- SSL 3 protocol was published in 1996
- Standardized as Transport Layer Security (TLS) in 1999
- OpenSSL developed based on public spec
- Mozilla uses NSS (Network Security Services) originally developed by Netscape

Protocol



Client

Server

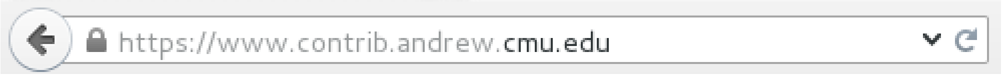
Connect to port 443, send hello & version supported

Sends public key, random number, certificate

Complete DH exchange or encrypt random number

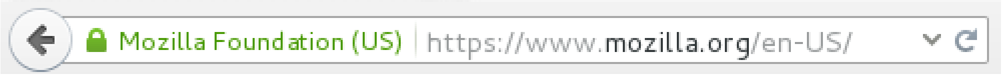
Certificate Authorities

Domain validation



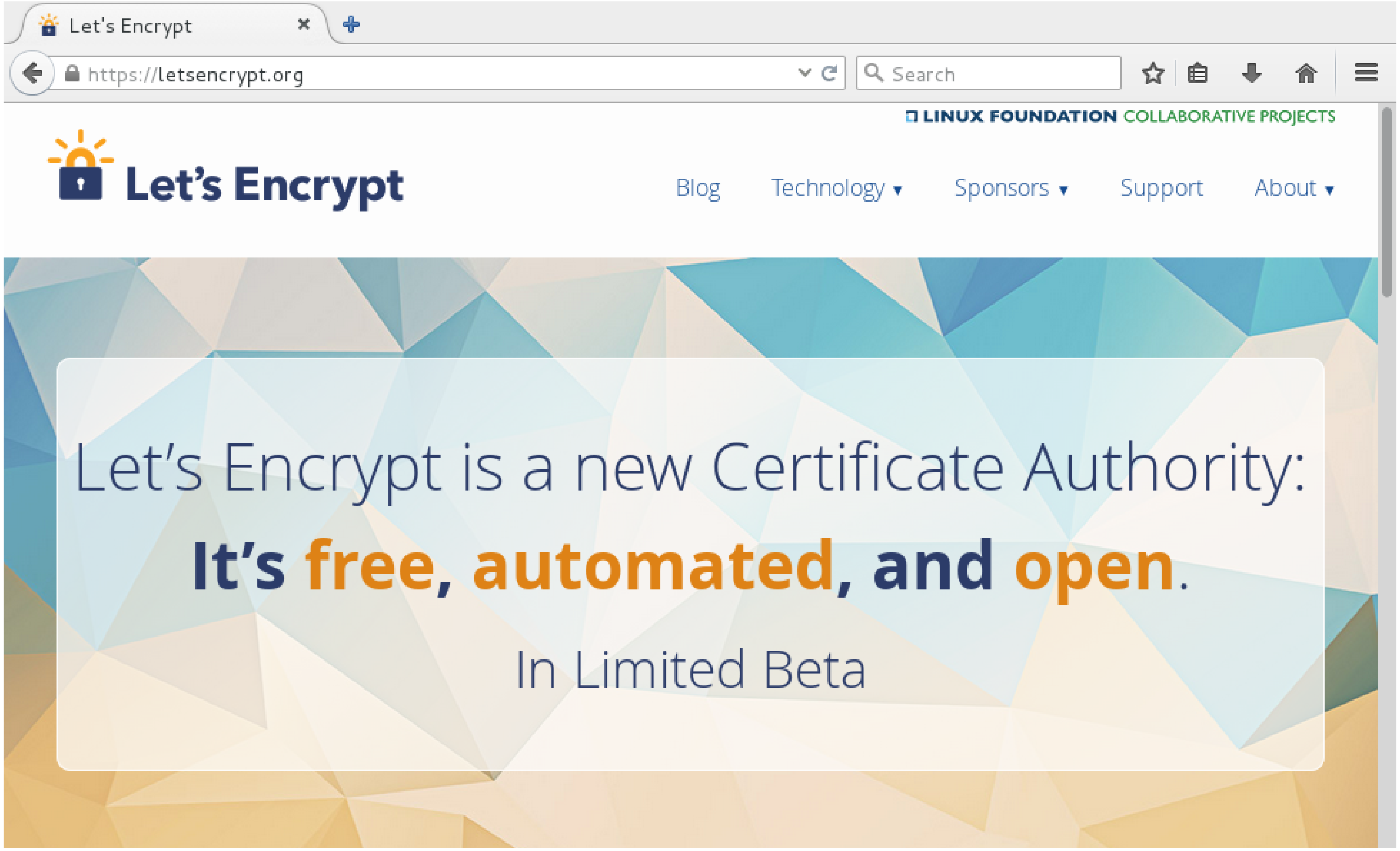
https://www.contrib.andrew.cmu.edu

Extended validation



Mozilla Foundation (US) https://www.mozilla.org/en-US/

Cost & complexity of certificates



Let's Encrypt is a new Certificate Authority:

It's free, automated, and open.

In Limited Beta

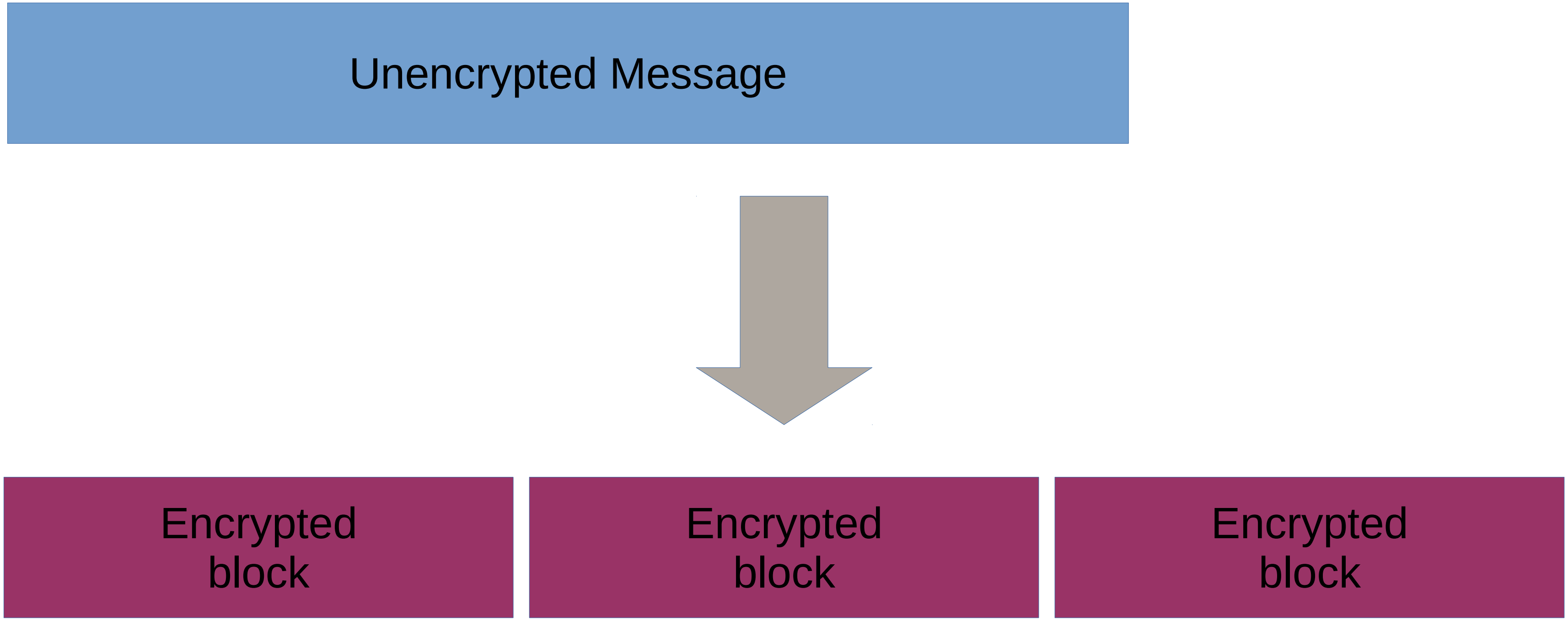
What if the certificate authority is not trustworthy?

- Certificate authorities have issued fraudulent certificates
- Superfish spyware installed its own CA key
- PrivDog allowed any certificate without any validation

Downgrade Attacks

- Old versions of SSL had security weaknesses
- Initial negotiation of supported versions is not encrypted or authenticated
- Active attacker can block initial negotiation and cause fallback to older, insecure version of SSL

Padding Oracle On Downgraded Legacy Encryption (POODLE)



Unencrypted Message

Encrypted block

Encrypted block

Encrypted block

The square-and-multiply algorithm

Calculate: n^2 … n^4 … n^8 … n^{16} … n^{32} ...

Example: $n^{42} = n^2 \times n^8 \times n^{32}$

Problem: The square and multiply operations differ in CPU time, power usage, and memory access pattern

- Timing attacks
- Radio frequency emission
- Cache analysis

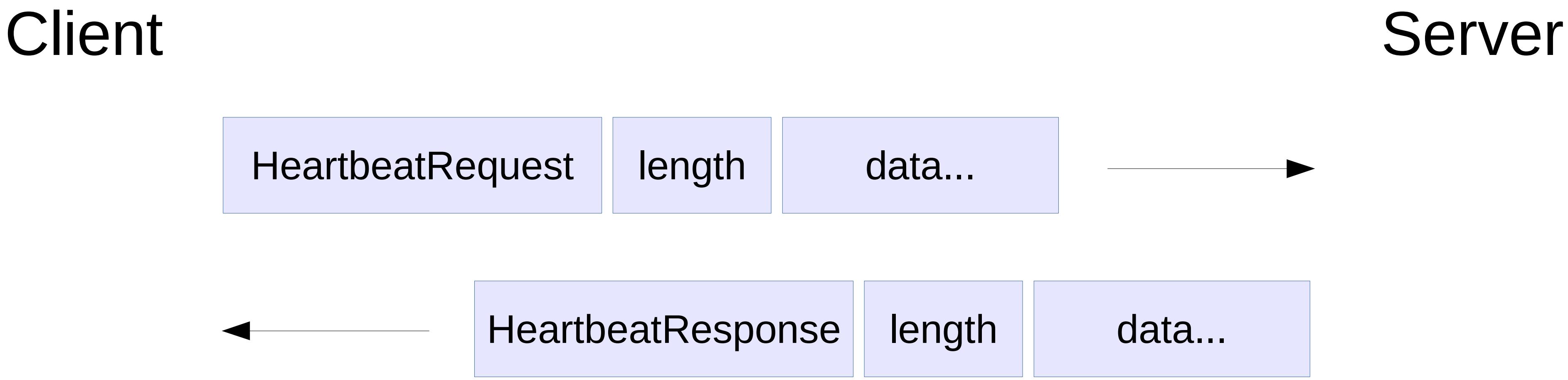
Data Compression Exploits

Even if the encryption is secure, the size of messages may leak information

Compression Ratio Info-leak Made Easy (“CRIME")

- Trick client into accessing various URLs (eg embedded images, iframes, javascript)
- Make guesses at session cookie in URL
- If a substring matches, data compression will shorten the message!
- Mitigate by not compressing headers

TLS Heartbeat Extension (RFC 6520)



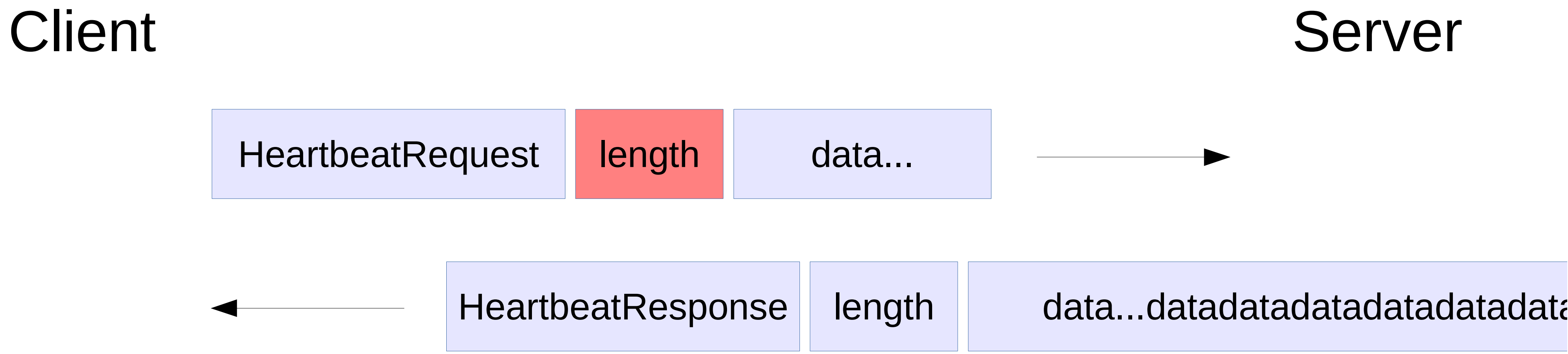
Client

Server

HeartbeatRequest | length | data...

HeartbeatResponse | length | data...

OpenSSL had a bug...



Client

Server

HeartbeatRequest

length

data...

HeartbeatResponse

length

data...datadatadatadatadatadata

Heartbleed

- OpenSSL used the requested length rather than the actual length in its response
- Returned whatever happened to be in that area of memory
- Allowed reading up to ~64KB per request
- Potentially leaked private keys

Impact of exposed keys

- Can spoof the server
- If Diffie-Hellman was not used, exposes all prior traffic
- Need to get new certificate for new key
- Need to revoke old certificate

Certificate Revocation

- Even if the certificate is free, revocation often is not (Startcom wanted $25)
- How to communicate revoked status to browsers?

Online Certificate Status Protocol (OCSP)

- Browser checks status of certificate with CA
- Typically unencrypted http
- Generally ignored on timeout
- Responses often lack nonce or expiration

Attacking Diffie-Hellman

The security of Diffie-Hellman key exchange is based on the difficulty of finding discrete logarithms in a finite field.

Logarithms have the property that the sum of logarithms is equal to the logarithm of the product, eg:

$$\log(2) + \log(3) = \log(6)$$

To break Diffie-Hellman, find logarithms of many small primes, then combine them to find the logs of larger numbers.

Avoiding the precomputation attack

- Use a modulus large enough to make this precomputation infeasible (2048+ bits)

  ...but encryption is $O(n^3)$ for n-bit modulus

- Don't use the same modulus as everyone else

  ...but generating strong primes takes time

- Use elliptic curve Diffie-Hellman

Conclusion

- Disable legacy weak crypto
- Don't allow downgrade to SSL 3
- Don't use RC4, DES or other weak algorithms
- Use ephemeral DH with ≥2048-bit modulus, or elliptic curve
- If using finite-field DH, generate a custom strong prime (see weakdh.org)
- Don't mix secure and insecure content