Running headers & footers

This document tests whether a PDF-to-text pipeline correctly strips repeated boilerplate. Every page below carries the same header line and a footer with a page number. Neither is part of the actual content, and neither should show up mid-sentence in extracted text.

Encryption in transit is handled by TLS, which runs a handshake before any application data moves: the client and server agree on a protocol version and cipher suite, the server proves its identity with a certificate signed by a trusted authority, and both sides derive a shared session key from that exchange.

From that point on, every byte between client and server is encrypted with the session key, which is why a network observer can see that a connection exists and roughly how much data moved, but not the request path, headers, or body being exchanged.

Certificate validation is what stops this from being trivially spoofable. A browser checks that the certificate's domain matches the one being requested, that it has not expired, and that it chains up to a root certificate authority the browser already trusts. Fail any of those checks and the connection is refused before data flows.

None of this protects against every threat: TLS secures the connection, not the endpoints. A compromised server still sees plaintext data after decryption, and a user tricked into visiting a look-alike domain with its own valid certificate is still handing data to the wrong server, just over an encrypted channel.

This is also why the padlock icon in a browser only ever meant "this connection is encrypted," never "this site is trustworthy." That distinction matters more as certificate issuance has become free and automatic for nearly any registered domain.

Certificate transparency logs add a further check on top of the browser-side validation above. Every publicly trusted certificate is required to be published to append-only, publicly auditable logs, so a certificate authority that mis-issues a certificate for a domain it should not have, by mistake or by compromise, cannot do so quietly.

HSTS (HTTP Strict Transport Security) closes a different gap: without it, a browser that has only ever been told "this site uses HTTPS" by a previous visit has no way to know that before the very first request on a new device, leaving a window where a network attacker could intercept a plain-HTTP request and strip the upgrade. An HSTS response header tells the browser to refuse plain HTTP to that domain for a set period, no matter how the address was typed or linked.

Mixed content is what happens when an HTTPS page loads a subresource, such as a script, a stylesheet, or an image, over plain HTTP. Browsers block "active" mixed content like scripts outright, because a network attacker who can tamper with that script can control the whole page even though

the top-level document itself was fetched securely.

Cipher suite negotiation during the TLS handshake determines exactly which encryption and key-exchange algorithms protect a given connection, and it is renegotiated for every new connection. A server can drop support for an older, weaker suite the moment it decides the security risk outweighs compatibility with older clients, without any change to the certificate itself.