Security

17-313 Fall 2026

Foundations of Software Engineering

https://cmu-17313q.github.io

Administrivia

- **P5A slides due tonight.**
- **Tomorrow:** checkpoint presentations.

Learning Goals

- Describe how software and network communication can be attacked.
- Distinguish authentication, authorization, confidentiality, and integrity.
- Explain the roles of hashing, symmetric encryption, public key cryptography, digital signatures, certificates, and TLS.
- Identify engineering practices that reduce the likelihood and impact of security breaches.

Outline

- **Why Security Matters**
- Network Attacks & Concepts
- Cryptographic Tools
- Activity: Encrypt a Message
- Signatures, Certificates & TLS
- Security Beyond the Connection
- Wrap-Up

Security Is Everyone's Job



What Do We Mean by Security?

Security protects against unauthorized:

- Access: who can get in, and what they can read.
- Modification: what can be changed.
- Disruption: what can be broken or stopped.

Threats can target:

- Data at rest.
- Data in transit.

Today's Focus

- Execution environments (browsers, servers, mobile).
- People (social engineering).

Why Network Security Matters

- Networks are the weakest common denominator for nearly all applications.
- Networks are inherently **untrusted**: anyone between sender and receiver can potentially observe, inject, or modify traffic.
- Modern apps rely heavily on remote APIs, cloud services, Wi-Fi, and mobile networks.
- Most attacks begin with stealing or manipulating traffic.

Attacks Can Be Expensive

FT Financial Times

UK regulator hits Equifax with £11mn fine over cyber breach

The Financial Conduct Authority has fined credit reporting agency Equifax just over £11mn for failing to protect the data of nearly 14mn UK...



Oct

The Guardian

Uber fined $148m for failing to notify drivers they had been hacked

Failure to report 2016 data breach 'one of the most egregious cases we've ever seen', says Illinois attorney general.



CBS News

PlayStation Network breach has cost Sony $171 million

(CBS/AP) TOKYO - Sony has spent 14 billion yen, which translates to roughly $171 million, to cover the costs of the massive security breach...



May 24, 2

Reuters

UK car production fell by a quarter in September amid JLR cyberattack

Britain's car production fell a sharp 27% in September, industry data showed on Friday, as an unprecedented cyberattack halted production...

Oct 23, 2025

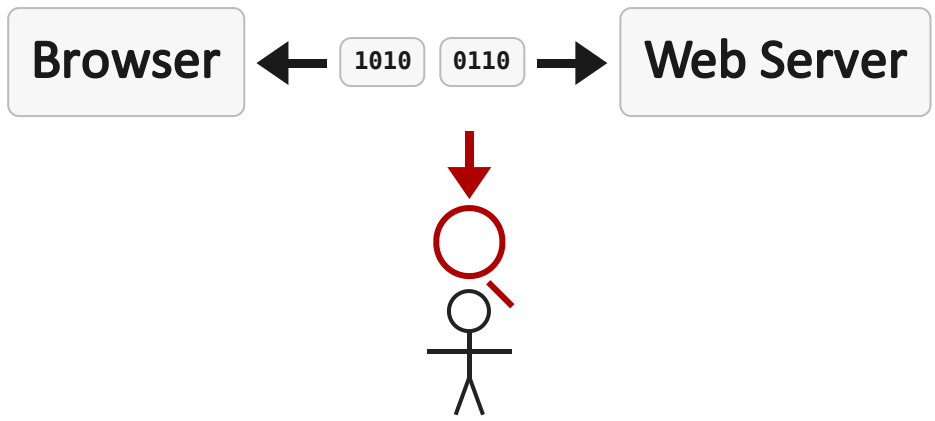


Outline

- Why Security Matters
- **Network Attacks & Concepts**
- Cryptographic Tools
- Activity: Encrypt a Message
- Signatures, Certificates & TLS
- Security Beyond the Connection
- Wrap-Up

Common Network Attacks: Sniffing

Eavesdropping: an attacker passively observes traffic between a browser and a web server.



Attacker: reading the traffic

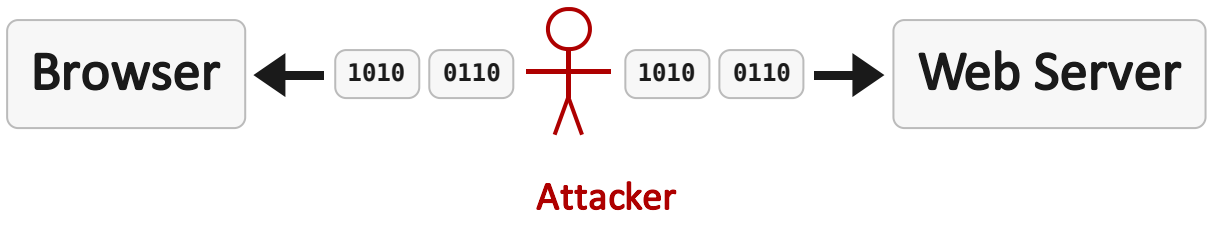
What harm can an attacker cause just by seeing your data in transit?

NOTES

Sniffing doesn't require controlling the network, only being positioned somewhere traffic happens to pass through. An attacker can capture packets wirelessly, by joining the same Wi-Fi network (especially an open or weakly secured one), or physically, by tapping into the same Ethernet segment, switch, or hub along the path. Tools like Wireshark or tcpdump then let them read any traffic that isn't itself encrypted. In concrete terms: they could read the contents of your emails, capture credit card numbers or other payment details submitted over an unencrypted connection, or harvest session cookies and passwords sent in the clear.

Common Network Attacks: Person-in-the-Middle

An attacker sits quietly between the browser and the server, altering traffic without either side noticing.



What kinds of damage can an attacker cause by altering traffic this way?

NOTES

A person-in-the-middle attacker gets into this position the same ways a sniffer does (a shared or rogue Wi-Fi network, a tap or compromised device on the path), but goes further: instead of just reading traffic, they actively intercept it and relay a modified version to each side, so browser and server each believe they're talking directly to each other. Because they control the data in both directions, they can change what either side receives: altering the amount or destination account in a financial transaction, injecting malicious script into an otherwise legitimate webpage, swapping a download link for a malware-laced one, or rewriting an API response to grant access it shouldn't.

Common Network Attacks: Spoofing

Pretending to be someone, or something, you're not.

- **IP spoofing:** pretending to be a client with a specific IP address.
- **Email spoofing:** pretending to be the sender of an email.
- **DNS spoofing:** fooling a DNS server into giving out the wrong IP address for a domain name.
- **Lookalike domains:** watch for typos or similar characters, e.g. a fake "mytimes.com," or "paypa1.com" instead of "paypal.com."

Three Useful Security Concepts

Not the complete set of security properties, but three that come up constantly:

- **Authentication:** verifies the identity of a user or system.
- **Authorization:** controls access to actions and resources.
- **Confidentiality:** protects data from being viewed by unauthorized parties.

A fourth property, **integrity** (protection against unauthorized modification), isn't one of today's three focus concepts, but it comes back later when we look at what TLS guarantees.

Terms Defined

Authentication

Knowing with whom you are communicating: the user knowing the server, the server knowing the user, or both.

Authorization

A user having privilege to perform a specific operation on the server.

Confidentiality

Communicating without others knowing what was said. By itself, this does not stop someone from changing the message in transit; that's integrity.

Poll: Which Concept Protects You?

Which of these concepts protects you against each attack?

Authentication **Authorization** **Confidentiality**

- Sniffing? **Confidentiality**
- Spoofing? **Authentication**
- Person-in-the-middle? **Authentication + Confidentiality**

Outline

- Why Security Matters
- Network Attacks & Concepts
- **Cryptographic Tools**
- Activity: Encrypt a Message
- Signatures, Certificates & TLS
- Security Beyond the Connection
- Wrap-Up

Concepts Every Software Engineer Needs

To protect authentication, authorization, and confidentiality, software engineers rely on a small set of cryptographic tools:

- One-way hashing
- Secret key (symmetric) encryption
- Public key (asymmetric) encryption
- Certificates

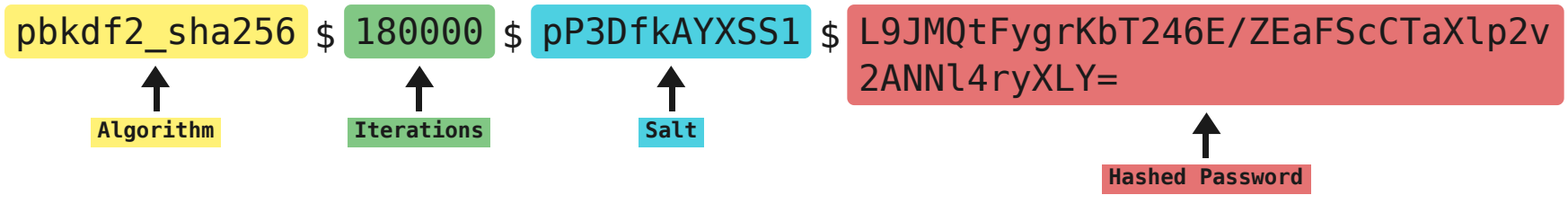
Hashing

Also called message digests, or one-way hashing.

- A hash function is a one-way encoding of data: the same input always gives the same output.
- Relatively easy to compute the hash; hard to compute its inverse (recover the input from the output).
- Different inputs can, in rare cases, produce the same output: a **collision**. Good hash functions make collisions extremely hard to find on purpose.

Hashing: Storing Passwords

We never store a plain password, only a salted, deliberately slow hash of it, so a stolen password file resists cracking.



NOTES

Storing passwords as plain hashes isn't enough: without a salt, two users with the same password get the same hash, and attackers can precompute hashes for common passwords (rainbow tables) once and reuse them against any leaked database. A per-password random salt defeats precomputation. "Deliberately slow" means using an algorithm with a tunable work factor (PBKDF2, bcrypt, scrypt, or Argon2) and a high iteration/cost count, so that even if the password file leaks, brute-forcing each password is expensive. A fast general-purpose hash like SHA-256 is the wrong tool here: being fast is exactly what makes brute-forcing cheap. The stored string packages everything the verifier needs to redo the computation: which algorithm, how many iterations, the salt, and the resulting hash.

Secret Key Cryptography

Also called symmetric, or private key, cryptography: like in the old spy movies.

- One shared key (`K`): 🔑 used to both encrypt and decrypt.
- Notation: `{data}K` means "data encrypted with key K."
- Both sides need the same key, kept secret from everyone else.



Public Key Cryptography

Also called asymmetric key cryptography.

- Each entity has a key pair: a **public key**, given out freely, and a **private key**, kept secret.
- Notation: for entity X, the keys are X_{pub} and X_{priv}.
- To send X a confidential message, encrypt it with X_{pub}; only X's X_{priv} can decrypt it. Notation: X_{pub}.

The two keys play different roles. This direction, public key encrypts, is for confidentiality. A different use, signing with the private key, comes up next.

Secret Key vs. Public Key Crypto

Secret Key Crypto



Public Key Crypto



NOTES

The main difference is how many keys each approach needs, and who has to already share one. Secret key crypto uses one shared key per pair of parties: both sides need the same key before they can talk, which means N people who all want to talk privately to each other need a key for every pair among them. Public key crypto gives each entity its own key pair, used for every conversation it has: X's public key can be handed out freely and reused by anyone who wants to send X something, so there's no need to pre-share a secret with each person individually. That's the key-distribution advantage; the tradeoff is that public key operations are computationally much more expensive than secret key ones, which is exactly why TLS later combines both.

Outline

- Why Security Matters
- Network Attacks & Concepts
- Cryptographic Tools
- **Activity: Encrypt a Message**
- Signatures, Certificates & TLS
- Security Beyond the Connection
- Wrap-Up

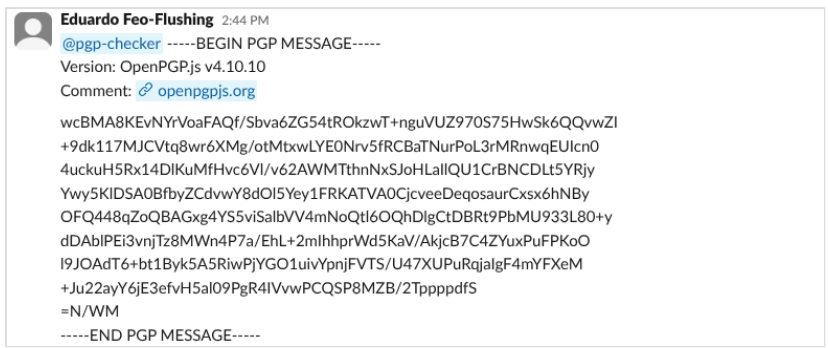
Activity: Encrypt a Message

Individually or in groups of 2-3: encrypt a short message using the instructor's public key, then post it to **#lecture-security** on Slack, tagging **@pgp-demo**.

- **Line 1 (required):** first word `Hello` (any case), then your andrew id(s).
- **Line 2 onward (optional):** your best dad joke, posted publicly (keep it appropriate); the bot quotes it back.
- Use any encryption tool; we recommend onlinepgp.com.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: OpenPGP.js v.1.20130420
Comment: http://openpgpjs.org
...
-----END PGP PUBLIC KEY BLOCK-----
```

Example:



7 minutes

Debrief: Some Considerations

- Who can read the messages posted to the channel? Do we have confidentiality?
- How could the instructor respond to a message privately?
- Does successfully encrypting a message confirm who sent it?

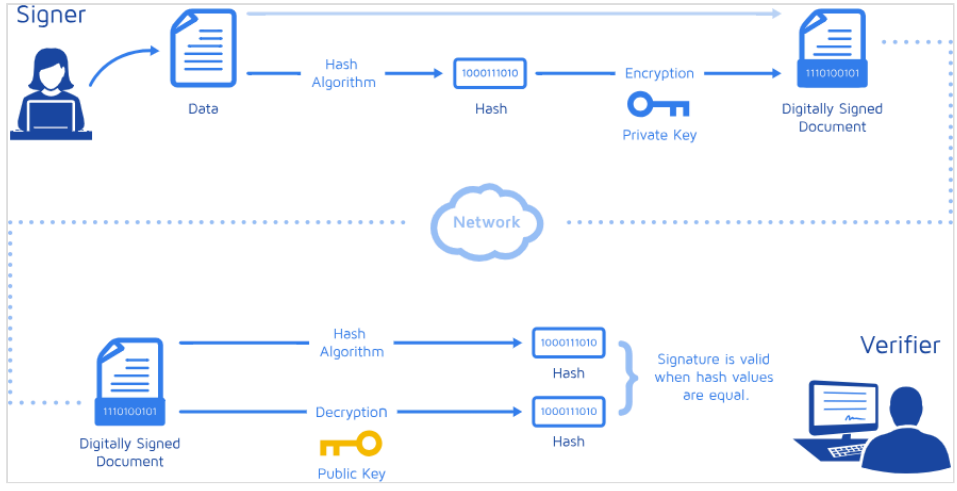
NOTES

Only the holder of the matching private key can read a message encrypted with the instructor's public key, so the channel gives confidentiality (in one direction). But the public key is, by design, public: anyone can use it to encrypt and send a message. Successfully encrypting a message proves nothing about who sent it, so this activity does not establish the sender's identity.

There's a deeper version of this problem, too: how do you know the public key you used really belongs to the instructor? If you got it from a Slack message, a webpage, or anywhere else that could be tampered with, an attacker could have substituted their own public key in its place. If so, every "confidential" message you encrypted was readable by the attacker, not the instructor, and you'd have no way to tell just from the activity. Verifying a signature (coming up next) proves that whoever signed a message holds a specific private key; it does not, by itself, prove that key pair belongs to the instructor. That binding between a public key and an identity is a separate trust problem, solved later by certificates.

Public Key Cryptography: Digital Signatures

A digital signature on a message gives a recipient confidence that the message really came from a sender they already know, and was not changed in transit.



We'll walk through exactly how this works next, before the activity.

Diagram: docusign.com

Digital Signatures: What We Need

How do I know that a sender is really who they say they are? We need a way for the sender to produce something that:

- Only the real sender could have created.
- Anyone else can verify.
- Works even over an untrusted network.

Digital Signatures: Signing and Verifying



The sender signs the message: they hash it, then sign the hash using their **private** key.

Can the receiver confirm that only the private-key holder could have created this signature?

The receiver hashes the data themselves, verifies the signature using the sender's **public** key, and checks: does it match `hash(data)`?

Match: sender verified. No match: reject.

NOTES

Signing and verifying are two separate operations, not the same operation run in opposite directions: the sender signs using their private key, and the receiver verifies using the sender's public key. A match confirms that whoever created the signature holds the private key paired with that public key. It does not, by itself, confirm that this key pair actually belongs to the person the receiver thinks they're talking to. That binding between a public key and an identity is a separate problem, which is exactly what certificates exist to solve.

Activity Part 2: Verify a Signature

This time, the instructor posted **two** messages, each with a different joke. Only one is really signed by the instructor.

- Find both messages posted on Slack.
- Go to onlinepgp.com.
- Check each one against the instructor's **public key**. Which one actually verifies?

Example format (not a real signature):

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Why did the JavaScript bug feel better? You console.log'd
-----BEGIN PGP SIGNATURE-----

iQFMBpEBCgA2FiETZQj8HLw5WyPQIy6kOOxG9RFewjoFAJsC1n0YHGVmZW
QGFuZMJldy5jbXUuZWR1AAoJEHjsRPURXsIbLfYIAK8DoEQHEJK5Iddhrt
Xq9dPLGIBDUpH2g3PvNhR8GTz11hcBi2PSSIdN4gWg+V6eTu1YWibe9zoI
ygXz0L9Ik6mmS5e2pfSQkJEodH/T/NWV4B/Pru21sxtK7cNgpvYU9gqir4
WxIWmHqbK9fl4yxyUYOE0aofcGRoBJGxDe8ct35tITjjAFKCi8w5Jcofgm
W6T/IRVptQk6sLml/0ujgOTFQ1ItNyR0wNJStDbNb+A1vP2j8scnVRDIz1
vo2XinzRVetL0vCXI8Bq15ZYINjvGmmZAv+Yd0leVReHyvWZRh9Pdgjtzf
-----END PGP SIGNATURE-----
```

Signing with a private key is a different operation from encrypting with it: here we are checking authorship, not reading a secret. The signature's hash is SHA-512.

Outline

- Why Security Matters
- Network Attacks & Concepts
- Cryptographic Tools
- Activity: Encrypt a Message
- **Signatures, Certificates & TLS**
- Security Beyond the Connection
- Wrap-Up

Comparing Public and Secret Key Crypto

- Both are very secure, unless the keys are compromised.
- Public key cryptography is computationally expensive; secret key cryptography is relatively fast.
- But it’s hard to distribute a secret key between parties who have never met.

This is why they're often combined: use public key cryptography to establish a shared secret key, then use that fast secret key for the rest of the conversation.

How Do We Distribute Public Keys?

If a public key alone doesn't prove who it belongs to, how does a browser know which key really belongs to a given server?

Certificate Authorities

- A Certificate Authority (CA) confirms an entity's (usually a server's) public key, after checking out the requester.
- The CA issues a **certificate** containing that information, signed with the CA's own private key.
- The CA's public key is well-known: pre-installed in your browser or OS, or added to your configuration.

Certificate Chain of Trust



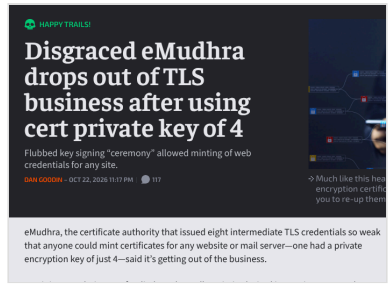
What happens if a private key anywhere in this chain is compromised?

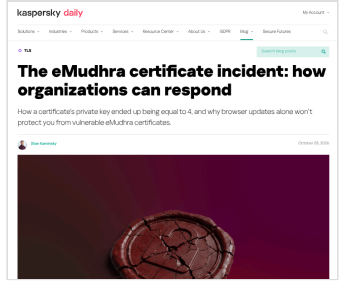
NOTES

Trace one path through this diagram: a trusted root certificate authority signs an intermediate CA's certificate, and that intermediate signs the actual website's certificate. A browser trusts the website's certificate only because it can follow this chain back to a root it already trusts. If a compromised intermediate's private key were used to sign a certificate for any website, a browser would trust that fraudulent certificate too, since it validly chains back to the same trusted root.

eMudhra Intermediate CA Compromise (October 2026)

- A faulty key-generation process left the private keys of eight intermediate CAs recoverable: one had a private key of just **4**.
- Six were TLS intermediates. Their keys could have been used to issue fraudulent website certificates, impersonating sites to clients that still trusted the affected chain.
- eMudhra revoked the intermediates. Mozilla blocked them in Firefox and removed trust in the root that signed them.





If the browser trusts the root CA, why is a compromised intermediate CA dangerous?

NOTES

"A private key of 4" means the actual numeric value of the private key was the integer 4, where a cryptographically secure private key should be an enormous, unpredictable random number. A flawed key-generation script produced private keys that were small and effectively guessable instead of properly random, making them recoverable by an attacker. The case is a reminder that cryptographic algorithms being sound doesn't help if the engineering controls around key generation are broken.

TLS: Combining Two Ciphers

- Public key cryptography has great key-distribution properties, but is computationally expensive.
- Secret key cryptography is fast, but the two sides need to agree on a key first.
- TLS uses public key cryptography for an **authenticated key establishment** step, then switches to fast symmetric encryption for the rest of the connection.

NOTES

"Authenticated" is the key word: the key establishment step doesn't just agree on a shared secret, it also validates the server's certificate against the certificate chain we just covered. That one step is what grounds all three guarantees on the next slide: the certificate validation gives server authentication, and the symmetric key it establishes is what protects confidentiality and integrity for the rest of the connection.

What Does TLS Give You?

TLS can protect any TCP/IP connection. Once a connection is validated, you get:

- **Confidentiality:** traffic on the connection is encrypted.
- **Server authentication:** contingent on actually validating the server's certificate.
- **Integrity:** tampering with traffic in transit is detectable.

User authentication is separate: typically your own userids and passwords, layered on top of TLS.

NOTES

Each of these traces back to the previous slide's authenticated key establishment: validating the server's certificate is what gives server authentication, and the symmetric key that validation establishes is what the connection's cipher uses to provide both confidentiality and integrity for the rest of the session. None of this authenticates the user: TLS authenticates the server to the client (and, less commonly, the client to the server), not a logged-in person. That's still the application's job, typically through userids and passwords layered on top of TLS.

Outline

- Why Security Matters
- Network Attacks & Concepts
- Cryptographic Tools
- Activity: Encrypt a Message
- Signatures, Certificates & TLS
- **Security Beyond the Connection**
- Wrap-Up

User Authentication With Passwords

TLS authenticates the server to you. How does the server authenticate you?

Most often, still with a password, hashed and salted as we saw earlier, sometimes alongside a second factor.

Other Risks: Implementation and Deployment

- Injection flaws and XML External Entities (XXE) attacks.
- Cross-Site Scripting (XSS): lets an attacker execute scripts in a victim's browser.
- Deployment misconfigurations: default accounts, unused features left enabled, overly broad cloud permissions.
- Using dependencies with known vulnerabilities (CVEs).
- Insufficient logging and monitoring.

CVE: A Real Example



Equifax's breach traced back to a known, published vulnerability that had a patch available for months before the breach.

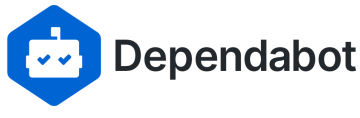
NOTES

The vulnerability behind the Equifax breach was not a secret flaw someone discovered in Equifax's own code: it was a publicly disclosed CVE in a dependency (Apache Struts), with a vendor-supplied patch already available. The gap was entirely in the "using dependencies with known vulnerabilities" category from the previous slide: Equifax had months to apply that patch and didn't. A published vulnerability only protects you once someone actually does the work of finding it in your dependencies and applying the fix.

Tracking and Patching Vulnerabilities

A CVE (Common Vulnerabilities and Exposures) is a public record of a known vulnerability. Tools can track CVEs in your dependencies and help you patch them.





Logging and Monitoring

- Early detection of security incidents.
- Forensic analysis and compliance.
- Proactive threat hunting.
- Audit trails for accountability.



Grafana Prometheus

Outline

- Why Security Matters
- Network Attacks & Concepts
- Cryptographic Tools
- Activity: Encrypt a Message
- Signatures, Certificates & TLS
- Security Beyond the Connection
- **Wrap-Up**

Wrap-Up: What You Need to Know

- **Attacks exploit untrusted networks:** sniffing, spoofing, and person-in-the-middle attacks are why authentication, confidentiality, and integrity matter.
- **Hashing is one-way, not reversible:** passwords need a salted, deliberately slow hash, never a fast general-purpose one.
- **Two kinds of key cryptography, different tradeoffs:** secret key is fast but needs a pre-shared key; public key solves distribution but is slower, and each entity reuses one key pair.
- **A signature isn't encryption:** it proves a message came from whoever holds a specific private key, checked with the matching public key, not who that key belongs to.
- **Certificates bind a public key to an identity:** a compromised certificate authority breaks that trust chain, as eMudhra showed.
- **TLS combines both ciphers:** authenticated key establishment gives confidentiality, integrity, and server authentication, not user authentication.
- **Security doesn't stop at the protocol:** implementation bugs, deployment misconfigurations, unpatched dependencies, and weak logging are still on you.