Trust

...and trusting it

Reflections on Trusting Trust

Overview

- Two main points
  - How can we make sure trust is explicit?
  - How can we limit trust?

Two Kinds of Trust

- Transitive Trust
  - You trust the people that the people you trust trust
  - If A trusts B, and B trusts C, then A trusts C
  - Examples?

Two Kinds of Trust

- Transitive Trust
  - You trust the people that the people you trust trust
  - If A trusts B, and B trusts C, then A trusts C
  - Examples
    - If I give you a key to my house, you could give it to anybody you trust
    - Whenever I trust a company, I trust all of their employees, subcontractors, etc

Two Kinds of Trust

- Transitive Trust
  - Facebook's Data Policy

    We transfer information to vendors, service providers, and other partners who globally support our business, such as providing technical infrastructure services, analyzing how our Services are used, measuring the effectiveness of ads and services, providing customer service, facilitating payments, or conducting academic research and surveys. These partners must adhere to strict confidentiality obligations in a way that is consistent with this Data Policy and the agreements we enter into with them.

Two Kinds of Trust

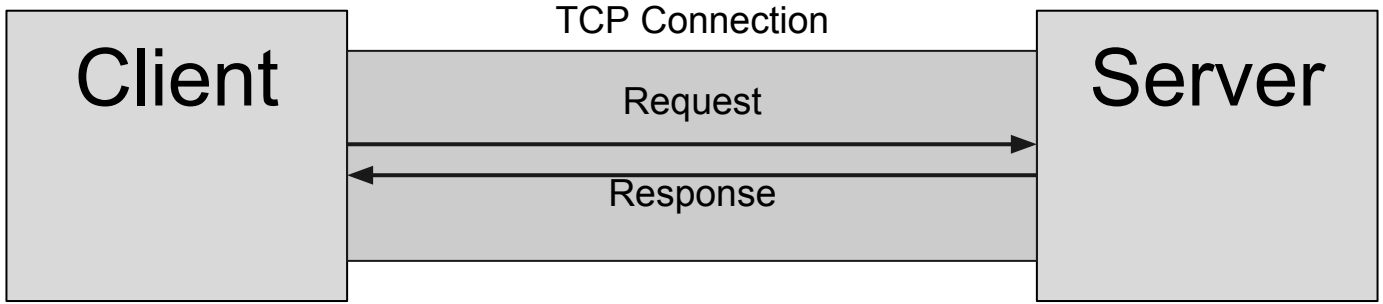
- Intransitive Trust
  - You trust only the people you trust
  - If A trusts B, and B trusts C, A does not necessarily trust C
  - Examples?

Two Kinds of Trust

- Intransitive Trust
  - You trust only the people you trust
  - If A trusts B, and B trusts C, A does not necessarily trust C
  - Examples
    - If I let you in my house, I don't have to let in the people you claim are trustworthy

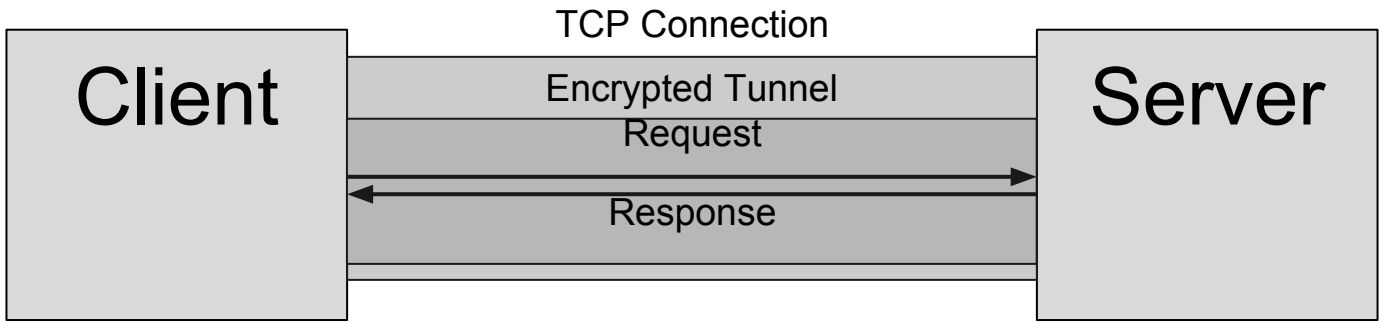
Certificate Authority System

- HTTP is a simple, unencrypted protocol
- Connect over TCP to server, request data



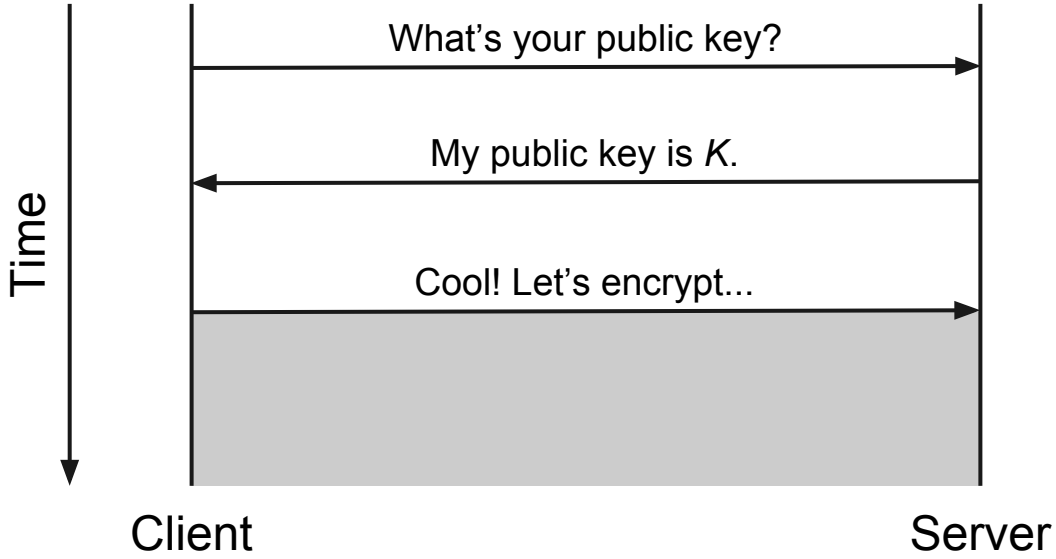
Certificate Authority System

- HTTPS is the secure version of HTTP
- Create encrypted tunnel, perform HTTP



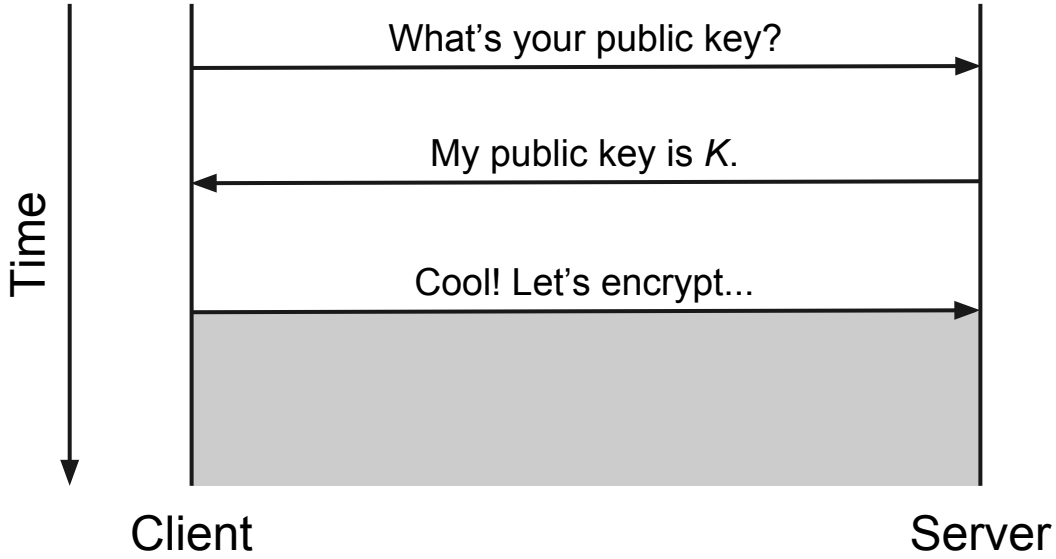
Certificate Authority System

- Simplified version of the protocol



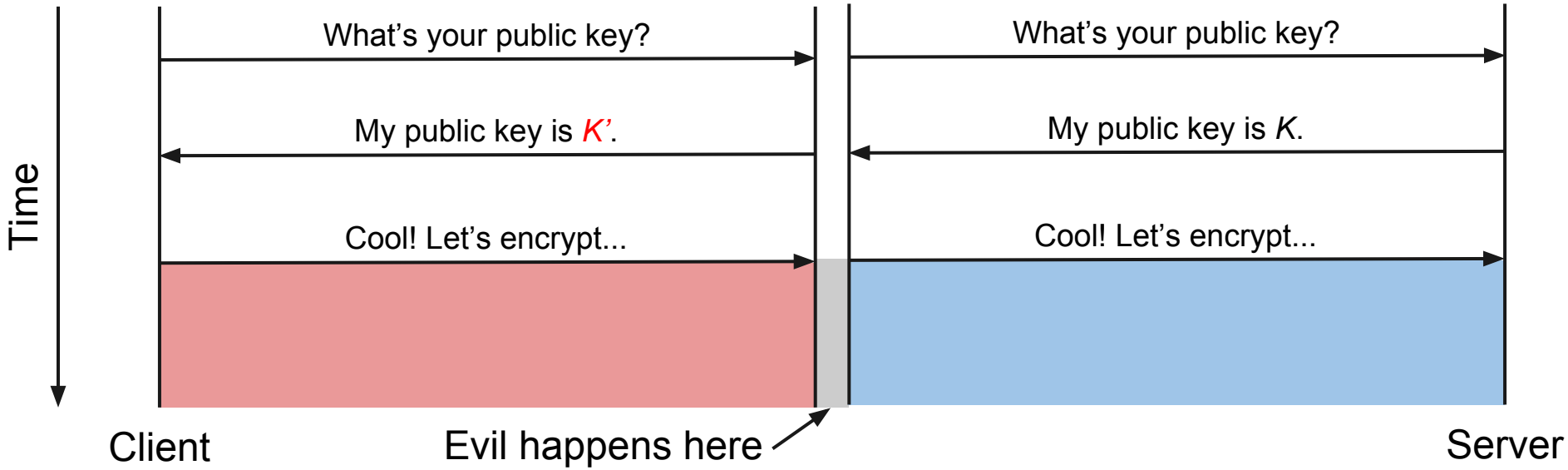
Certificate Authority System

- What's wrong with this?



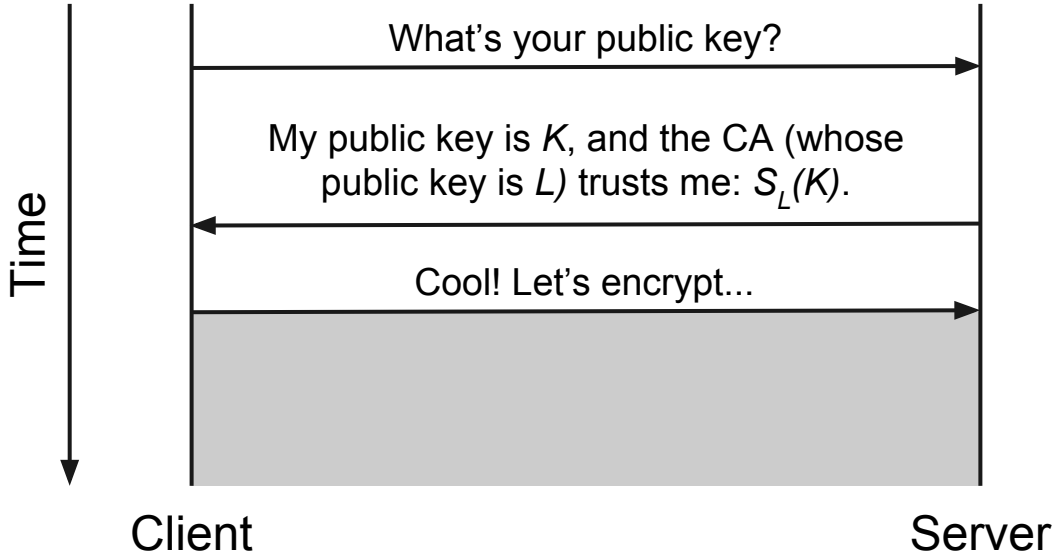
Certificate Authority System

- Man-in-the-Middle!



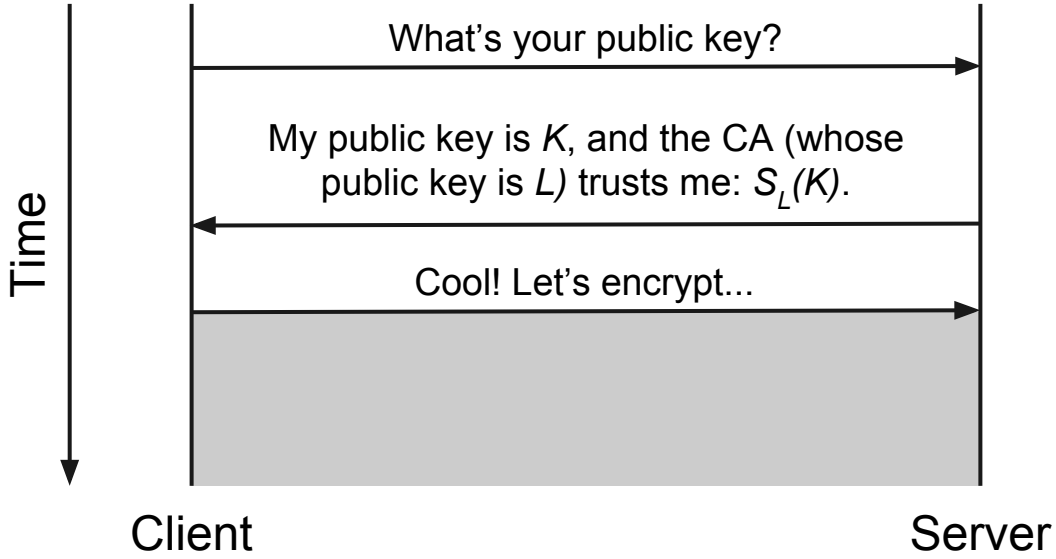
Certificate Authority System

- Second attempt: trusted “certificate authority” verifies



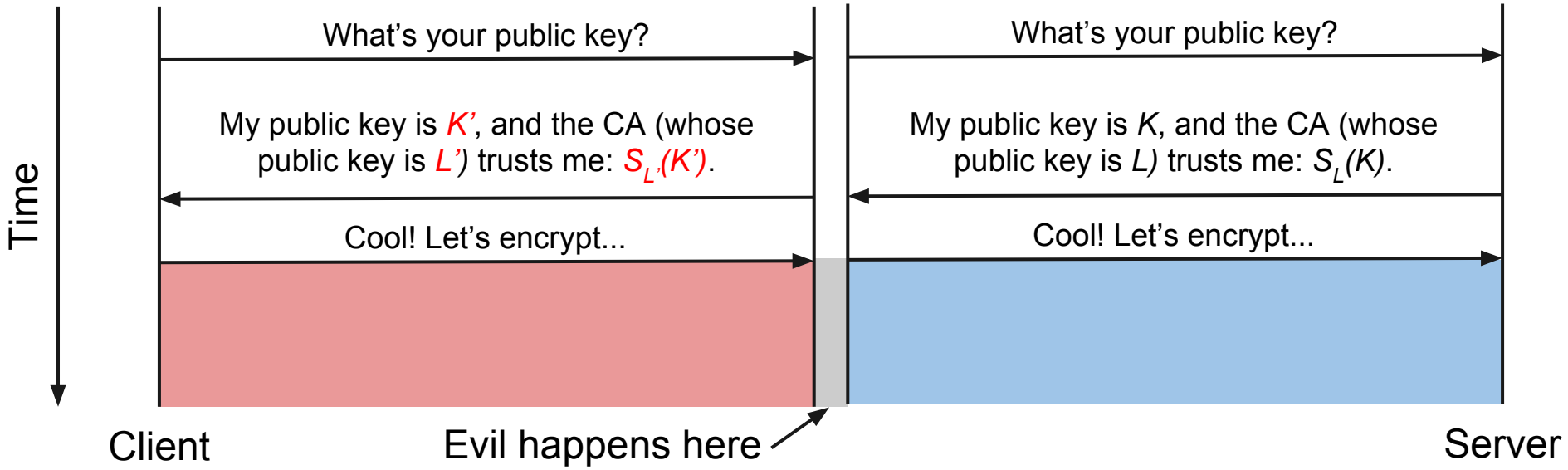
Certificate Authority System

- What could go wrong?



Certificate Authority System

- Man-in-the-Middle!



Certificate Authority System

- Clearly this will go on forever…
- Is there a way to solve this problem?

Certificate Authority System

- The answer: Root CAs
- Know the root CAs' certificates ahead of time
- Root CA -> CA A -> CA B -> … -> Website
- This is a form of *explicit transitive trust*
  - It's *explicit* because you actively decide to allow transitivity on a case-by-case basis
  - Each trust relationship is cryptographically signed

Certificate Authority System

- The answer: Root CAs
- Know the root CAs' certificates ahead of time
- Root CA -> CA A -> CA B -> … -> Website
- This is a form of *explicit transitive trust*
  - It's *explicit* because you actively decide to allow transitivity on a case-by-case basis
  - Each trust relationship is cryptographically signed
  - What could go wrong?

Certificate Authority System

- What could go wrong?
- It's *hard to gauge trustworthiness*
- 2011: Dutch CA DigiNotar compromised
  - Attackers stole private keys
  - Forged fake certificates (including for *.google.com)
- Was DigiNotar trustworthy?
- Were the CAs that trusted them trustworthy?

Certificate Authority System

- About a week ago…
- Lenovo got flak for shipping a fake root certificate from the advertising company Superfish.
- Advertisers could inject advertising into pages, the computer would raise no alarms
- Basically, a Man-in-the-Middle attack

Certificate Authority System

- The NSA performs *interdiction*
  - Intercepts packages in shipping
  - Modifies them
  - Sends them on their way
- Could easily install fake root certificates
- In practice, it's often hardware bugs or malware

Certificate Authority System

- 10-Second Plug
- EFF, Mozilla, etc creating a CA in mid-2015
- Let's Encrypt
- Makes setting up HTTPS very easy, free
- If you run a website, you should do it!
- Ok, back to the lecture...

Signed Apps

- Download app from iOS, Android app stores
- App is cryptographically signed by developer
- Developer's public key signed by Apple, Google

Signed Apps

- Trust is inherently transitive and implicit
  - Signature only says “developer certifies this app”
  - Doesn’t say *why* they certify the app
  - Doesn’t say how they made the app
  - Doesn’t say what other parties were involved
  - You can’t reason explicitly about these trust relationships

Signed Apps

- What could go wrong?

Signed Code

- Developer releases, signs *source code*
- We compile it ourselves
  - Don’t have to trust their compiler
  - Don’t have to trust their filesystem
  - Don’t have to trust their network
  - Etc
- Makes trust *less transitive*

Signed Code

- But you all read *Reflections on Trusting Trust*
- What could go wrong?

Signed Code

- But you all read *Reflections on Trusting Trust*
- What could go wrong?
  - How do you trust your compiler?

Reproducible Builds

- One technique is called *reproducible builds*
- Relies on a deterministic compiler
  - Same source code always yields same executable
- Developer signs source code and a *hash of the executable* that should be produced
- Builds are reproducible, so hash should match

Reproducible Builds

- Much more trustworthy!
- To defeat, would have to break both the consumer's compiler *and* the developer's compiler
- Probably have to break *all* consumers' compilers in order to avoid detection

Reproducible Builds

- What could go wrong?

Reproducible Builds

- Demo!
  - I've written a program in the language *foo*
  - You've obtained a *foo* compiler
  - I sign: "If you compile my program containing only the word `foo`, the resulting executable should have the SHA-1 hash `df3aafe8f06eecbc0796b9ebc13c3028a54e608f`."
  - Run `sha1sum <file>` on Linux or Mac

Reproducible Builds

- What went wrong?

Reproducible Builds

- What went wrong?
  - Maybe you don’t have to trust the compiler…
  - But you have to trust the apps used to verify!
    - The hashing program
    - The signature verification program
    - The hard drive (could alter these or the executable after it’s been verified)
    - The kernel (could lie to you about nearly everything)
    - Etc

Reproducible Builds

- Last week, Kaspersky Labs released a report
- The *Equation Group*
  - (They love cryptography)
- They wrote a virus which infected the firmware on hard drives
- Can’t trust anything which relies on your hard drive… which is basically everything

Bunkers, Anyone?

- OK, so who's scared?

Bunkers, Anyone?

- OK, so who's scared?
- Don't be.

Recap

- Two kinds of trust: *transitive* and *intransitive*
  - Transitive: You trust the people the people you trust trust
  - Intransitive: You don't
- If you have to have transitive trust, make sure it's *explicit* and *verified*
- If you can help it, try to force the trust to be *intransitive*

Fin!

- Let’s Encrypt: letsencrypt.org/
- Demo source: github.com/synful/reproducible-builds-demo