

# **11. Network Security**

## **The tale of SSL/TLS and SET**

Computer Security Courses @ POLIMI  
*Prof. Carminati & Prof. Zanero*

# Issues of Transactions Security

- Problems of remoteness
  - Trust factor between parties
  - Use of sensitive data
  - Atomicity of transaction
- Internet protocol problems
  - Authentication
  - Confidentiality
- Transparency and critical mass problem

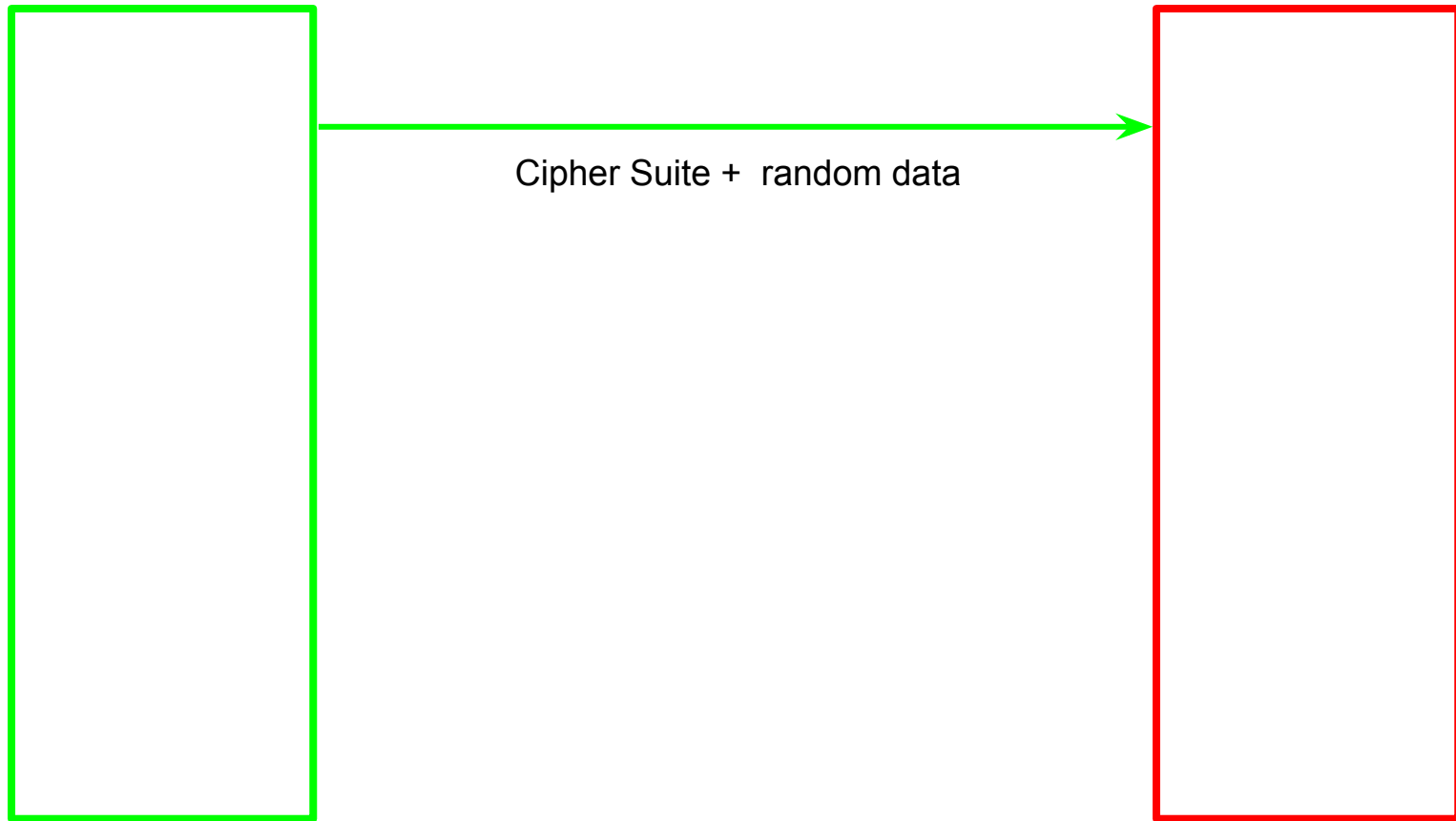
# A Tale of Two Protocols

- Two valiant protocols fought against the darkness
- HTTP over SSL (Secure Socket Layer), or HTTPS
  - Communication confidentiality and integrity
  - Mutual authentication
  - No guarantees on data usage
  - No strict authentication of client (in practice)
- SET (Secure Electronic Transaction)
  - Guarantees on data usage and transaction security enforcement
  - Missing critical mass support

# SSL and TLS

- Originally designed by Netscape for securing web communication
  - de facto standard also for other protocols
  - IETF standardized TLS, which comes after version SSL v3, and is now at version 1.3. All previous versions to TLS 1.2 are unsafe
- SSL/TLS enforces:
  - Confidentiality and integrity of the communications
  - Server authentication
  - Client authentication (optionally)
- Uses both symmetric and asymmetric cryptography for performance reasons

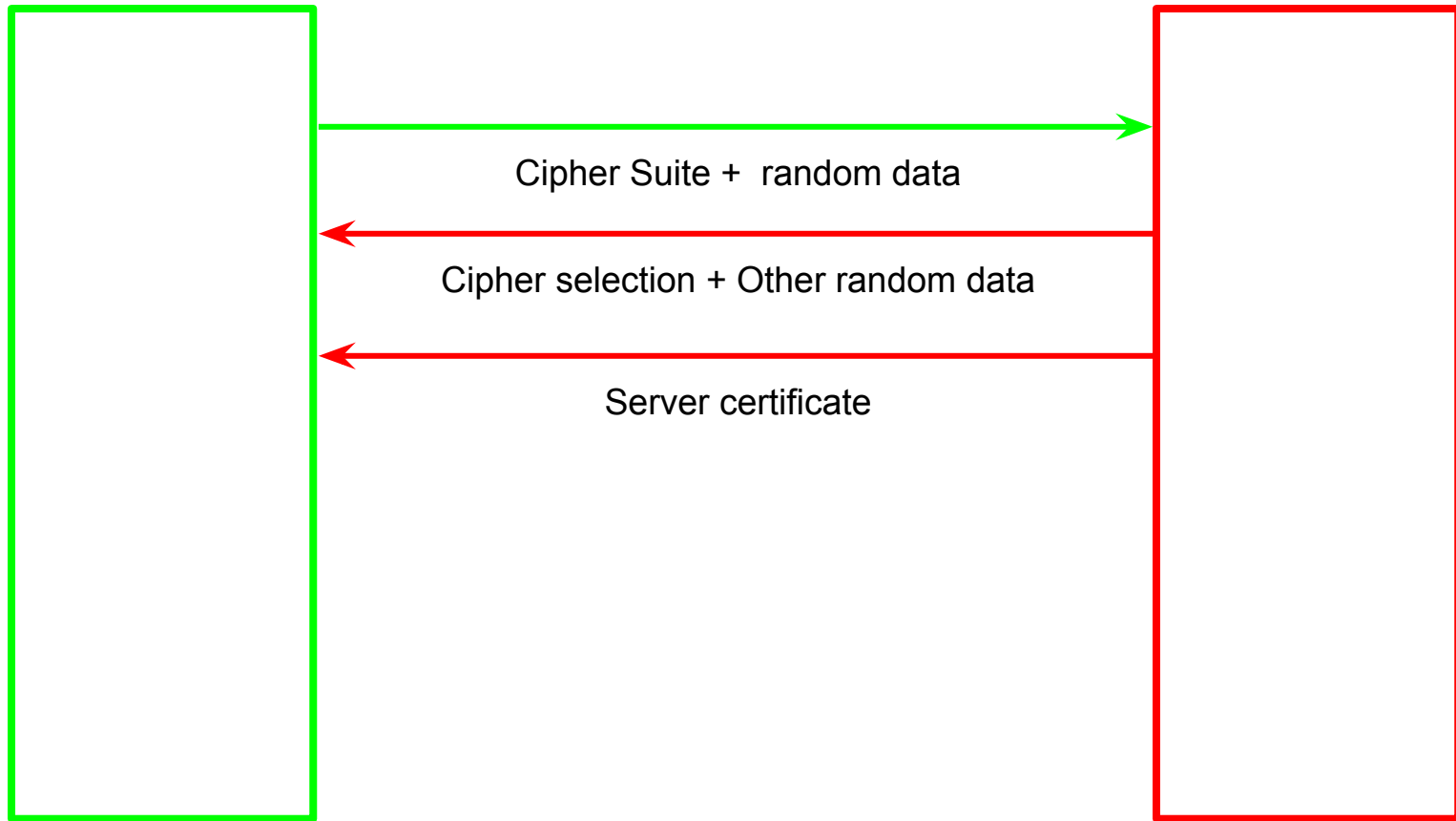
# SSL Handshake Phases



# Cipher Suite

- SSL designed to be flexible wrt to technical evolution
- Clients and servers may use different *suites* of algorithms for different functions
  - key exchange algorithm
  - bulk encryption algorithm
  - message authentication code algorithm
  - pseudorandom function
- During handshake, cipher suites are compared to agree on shared algorithms in order of preference
- The standard mandates implementation of a minimal cipher setting

# Server Authentication



# Verification of Server Certificate

- Is the certificate in the validity period?
- Is the root CA trusted?
- Is the certificate valid?
- Is it revoked?
- **Is the *name* of the server in the certificate the same that I requested?**

**Remember the implementation issues that we have learned a couple of months ago?**



## 11. Network Security

File Edit View



8

1

### Verification of Server Certificate

- Is the certificate in the validity period?
- Is the root CA trusted?
- Is the certificate valid?
- Is it revoked?
- Is the name of the server in the certificate the same that I requested?

Remember the implementation issues that we have learned a couple of months ago?

9

### Secret Transmission



10

11

### (Optional) Client Authentication



Equifax Secure Certificate Authority  
↳ GeoTrust Global CA  
↳ Google Internet Authority G2  
↳ \*.google.com



### \*.google.com

Issued by: Google Internet Authority G2

Expires: Tuesday 5 August 2014 02 h 00 min 00 s Central European Summer Time

✓ This certificate is valid

#### ▼ Details

Subject Name  
Country US  
State/Province California  
Locality Mountain View  
Organization Google Inc  
Common Name \*.google.com

Issuer Name  
Country US  
Organization Google Inc  
Common Name Google Internet Authority G2

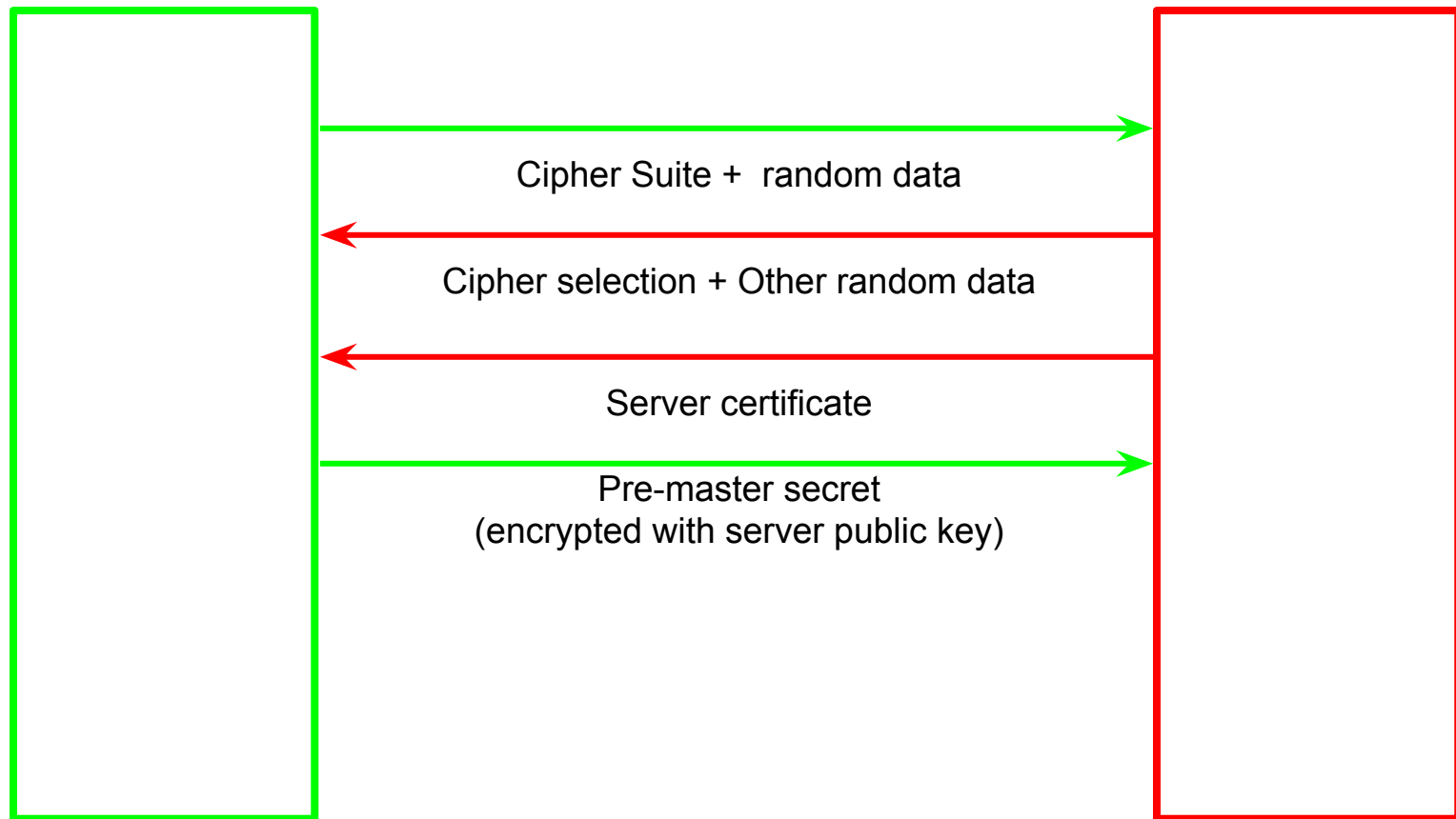
Serial Number 1405391832758935466  
Version 3

Signature Algorithm SHA-1 with RSA Encryption ( 1.2.840.113549.1.1.5 )  
Parameters none

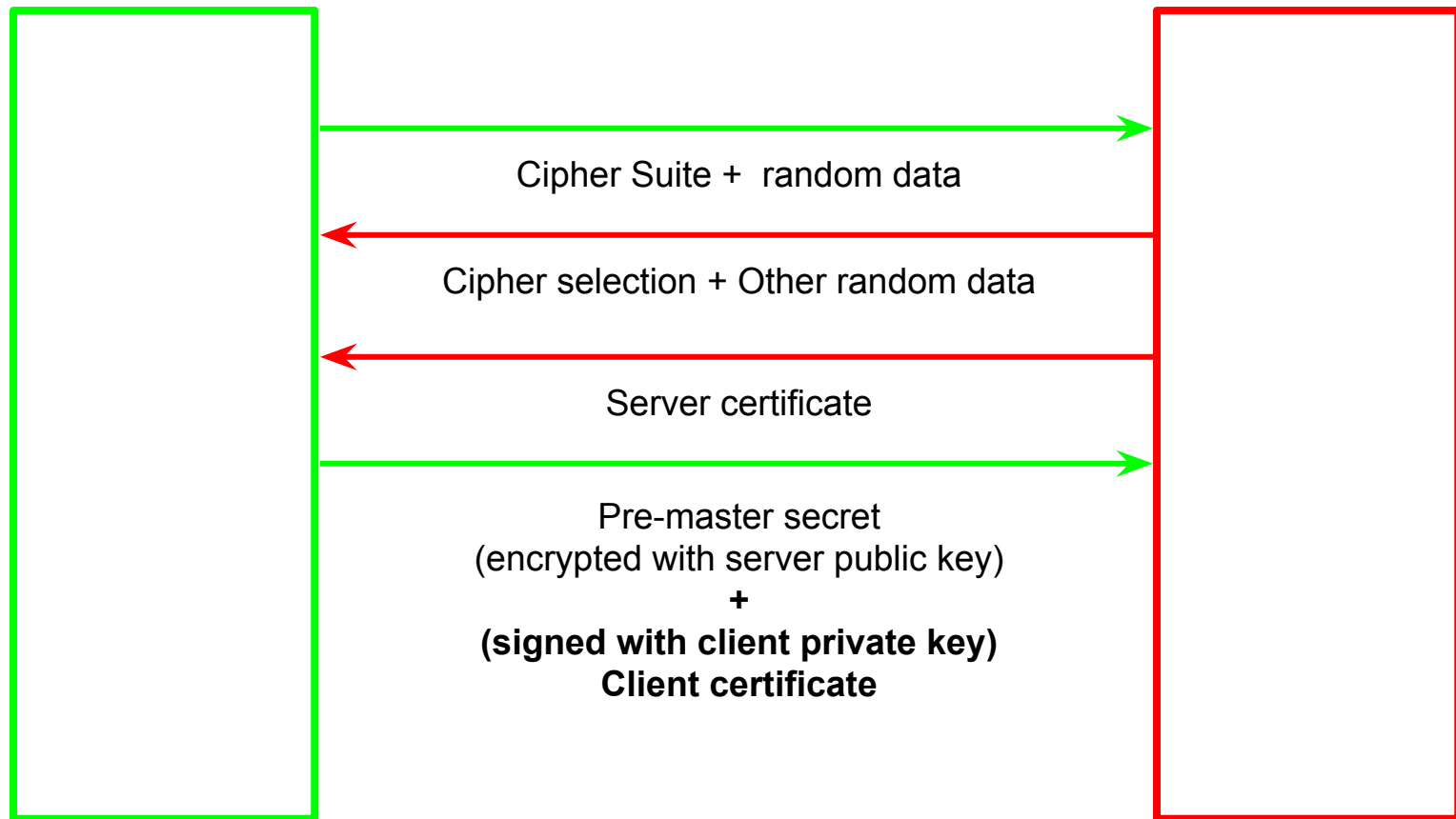
Not Valid Before Wednesday 7 May 2014 14 h 37 min 59 s Central European Summer Time

OK

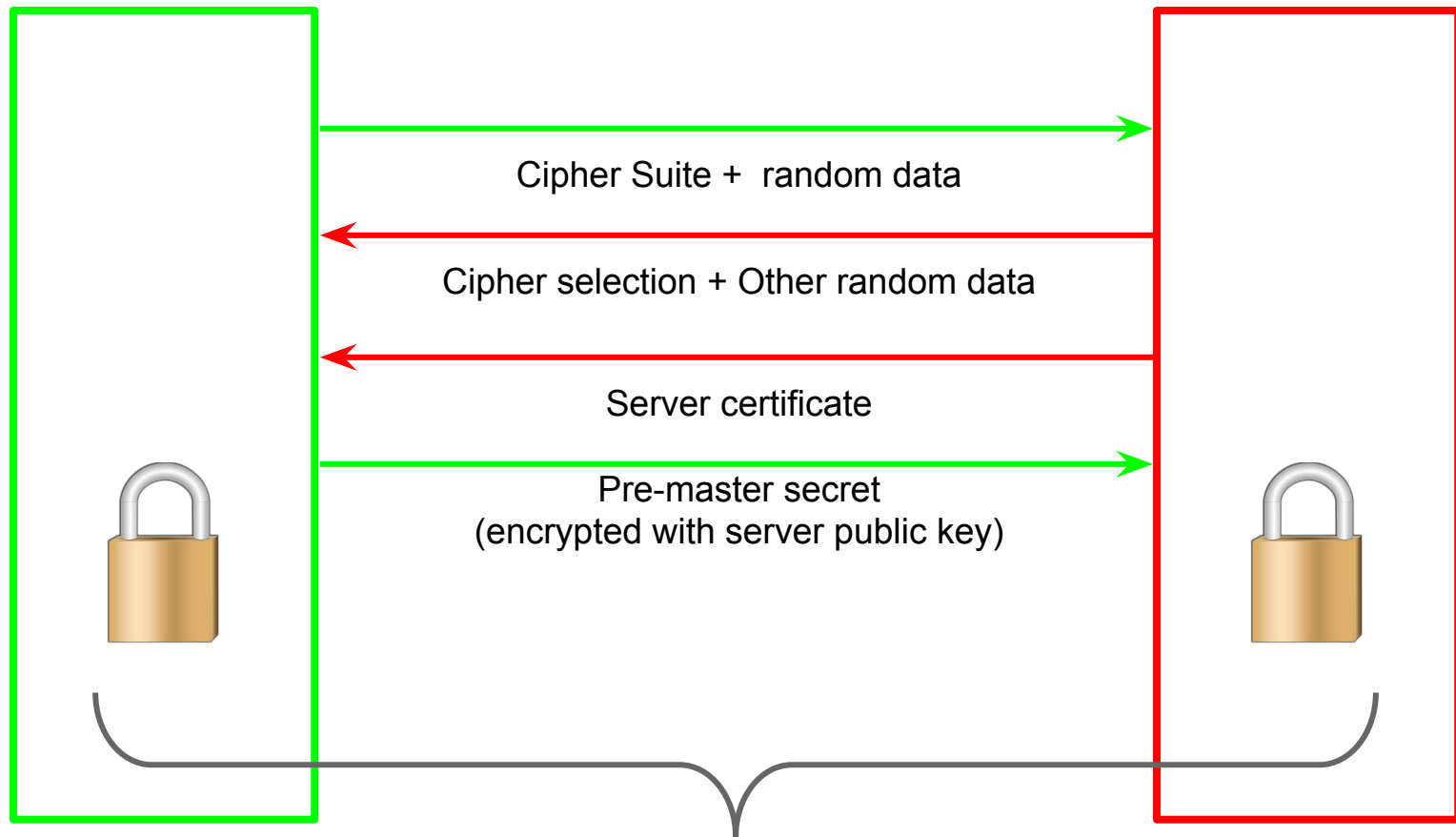
# Secret Transmission



# (Optional) Client Authentication

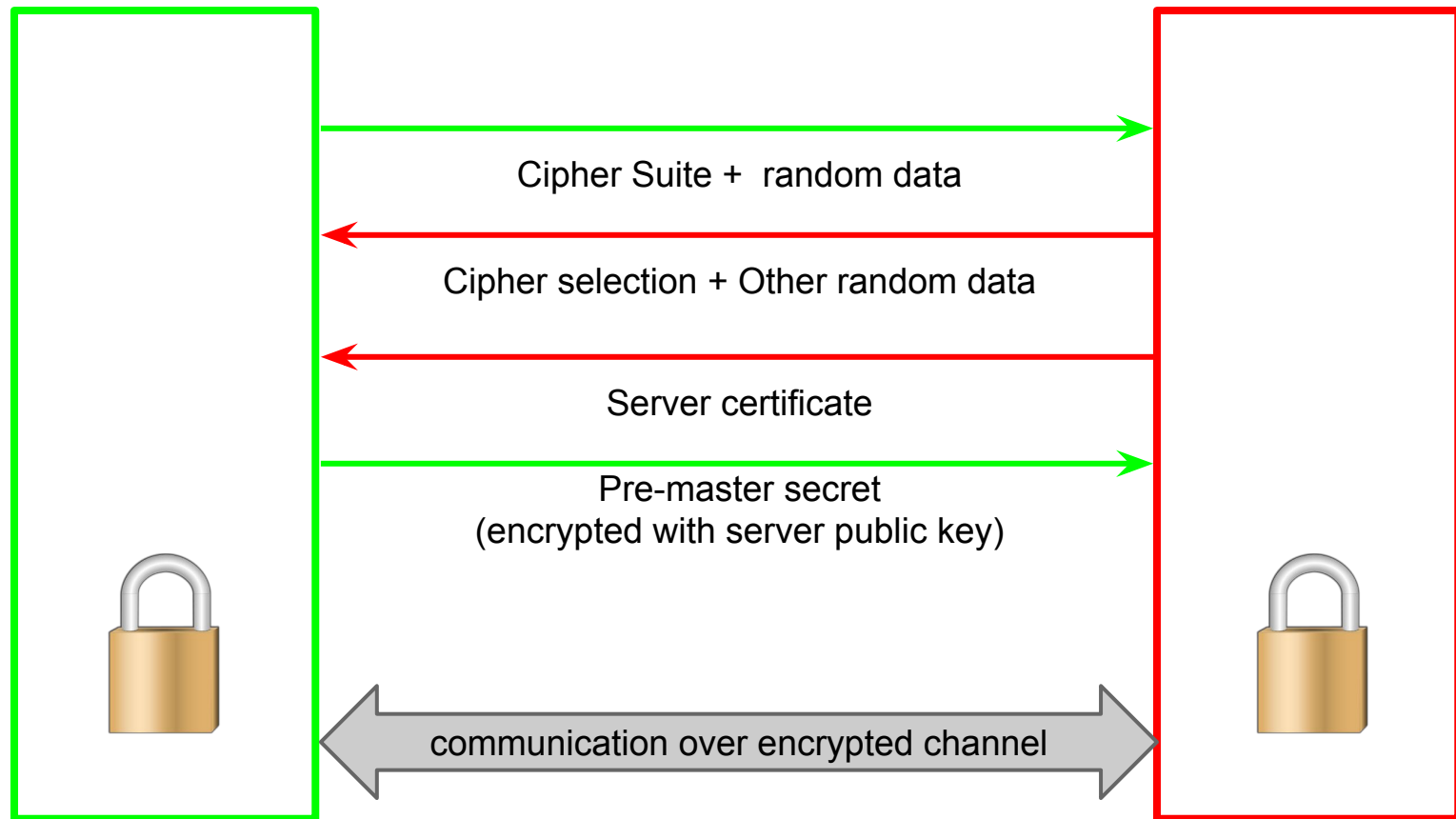


# Secret Computation



Compute shared secret from pre-master secret, client random data and server random data

# Encrypted Communication Phase

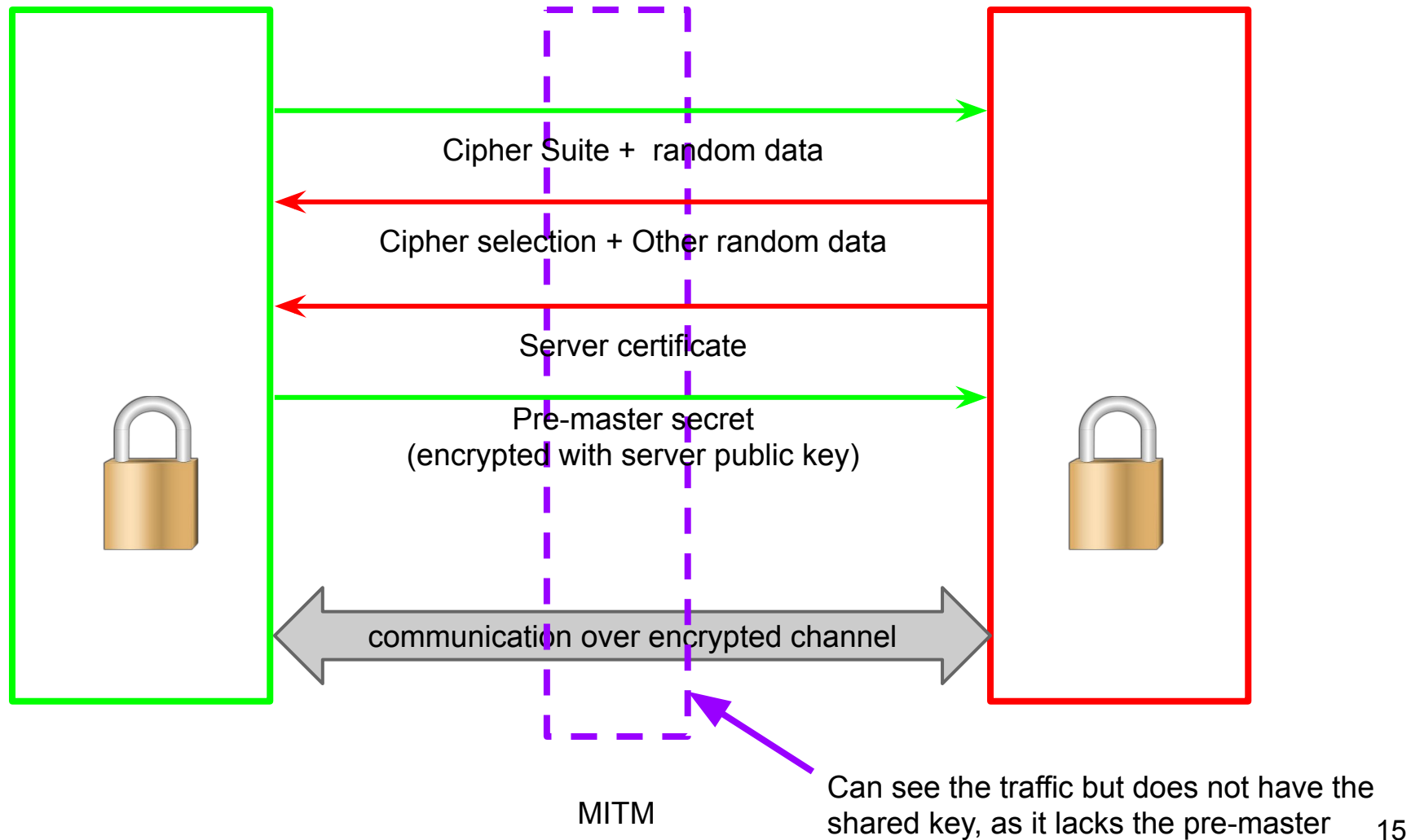


# Is SSL Resistant to MITM?

What could the MITM do?

- Let the original certificate through

# Cut out!



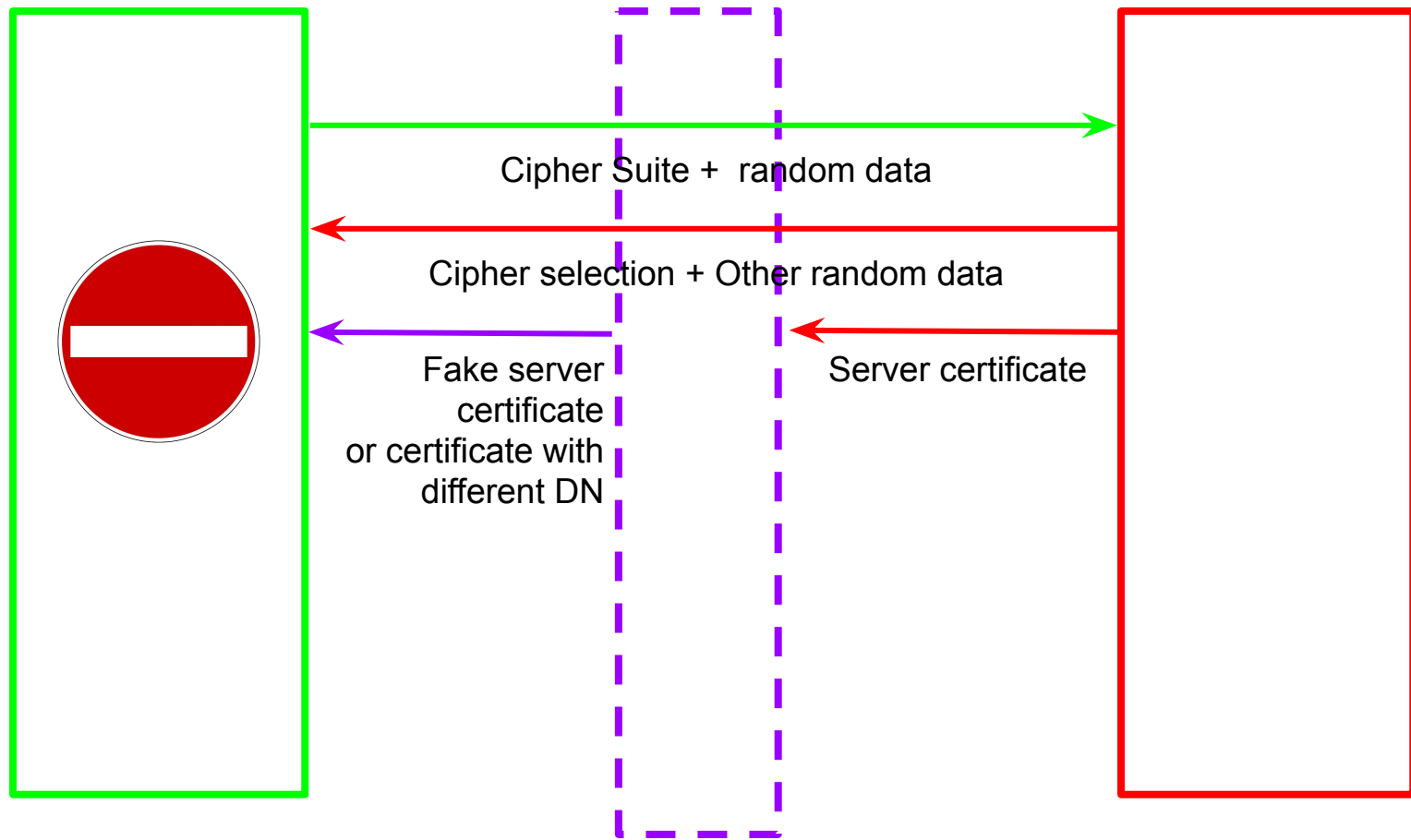
# Is SSL Resistant to MITM?

What could the MITM do?

- ~~Let the original certificate through~~
  - Needs to actually have the key on that cert!
- Send a fake certificate (i.e., signed by a non-trusted CA)
- Send a good certificate with a fake name



# Rejected!



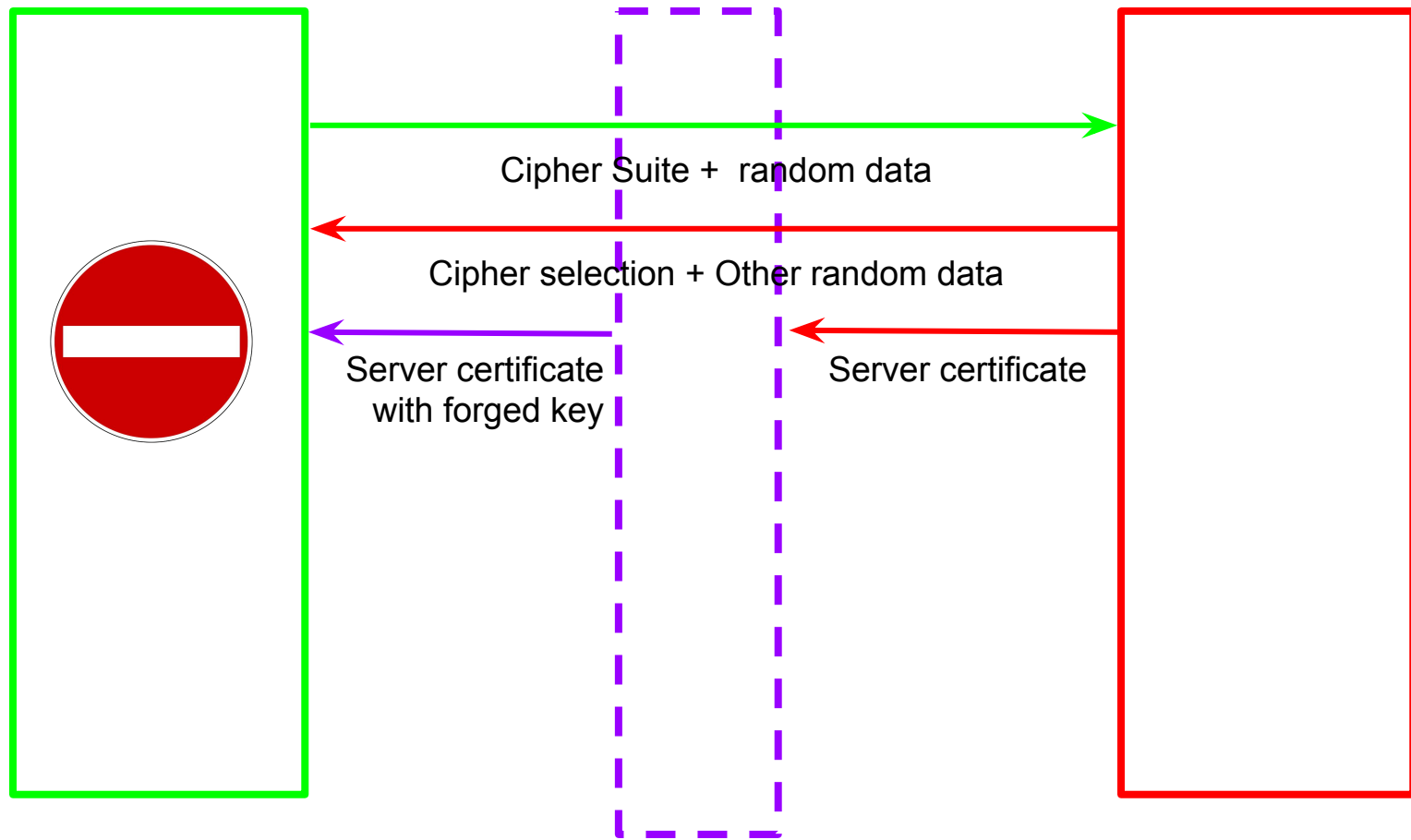
MITM

# Is SSL resistant to MITM?

What could the MITM do?

- ~~● Let the original certificate through~~
- ~~● Send a fake certificate (i.e. signed by a non-trusted CA)~~
- ~~● Send a good certificate with a fake name~~
- Send a good certificate but substitute the public key (making it invalid)

# Rejected!



MITM

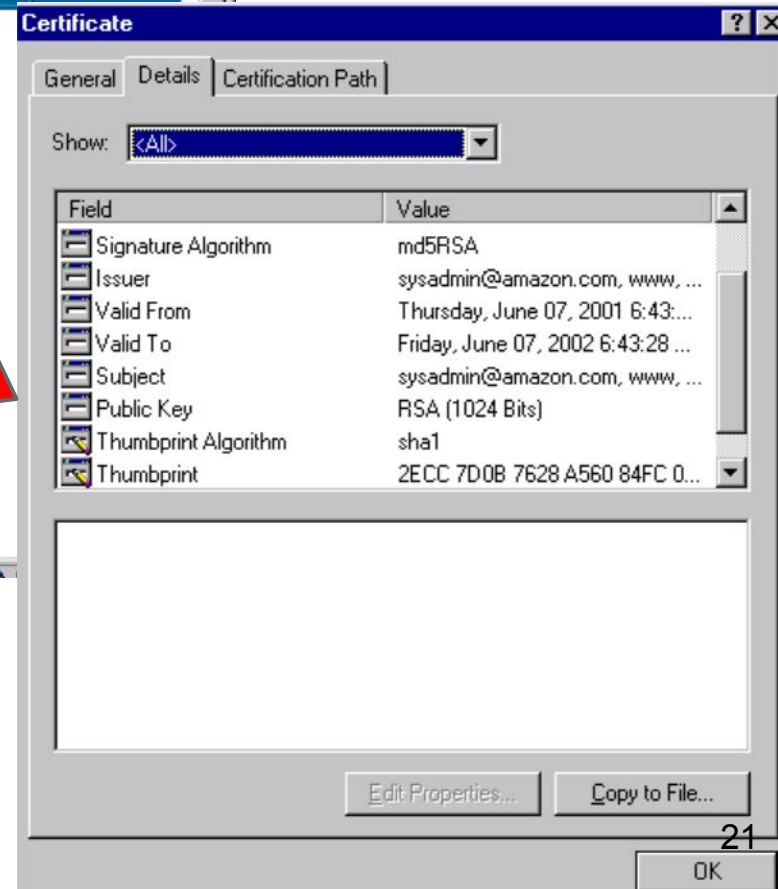
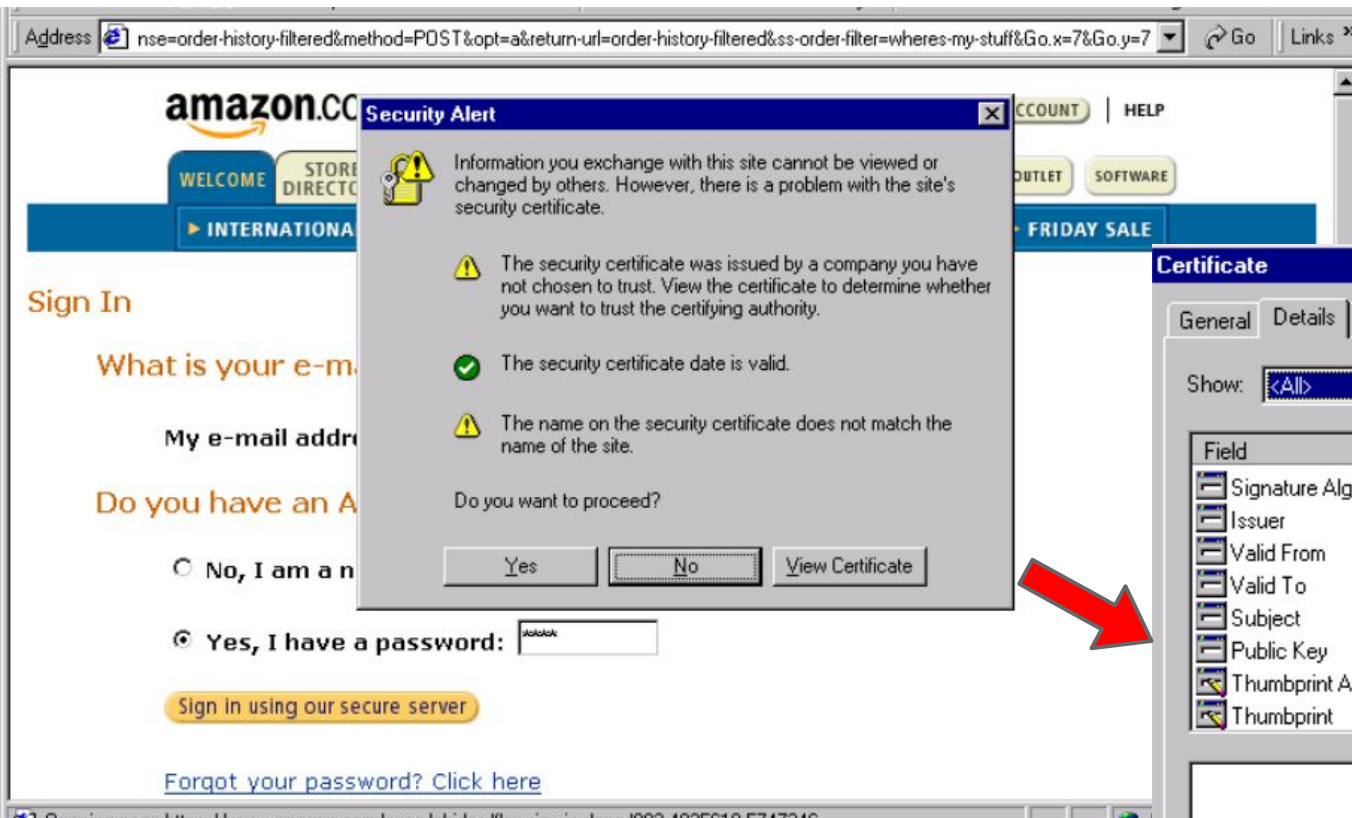
# Is SSL resistant to MITM?

What could the MITM do?

- ~~● Let the original certificate through~~
- ~~● Send a fake certificate (i.e. signed by a non-trusted CA)~~
- ~~● Send a good certificate with a fake name~~
- ~~● Send a good certificate but substitute the public key (making it invalid)~~

**Nothing: SSL is by design resistant to MITM!**

# Social Engineering = Fail

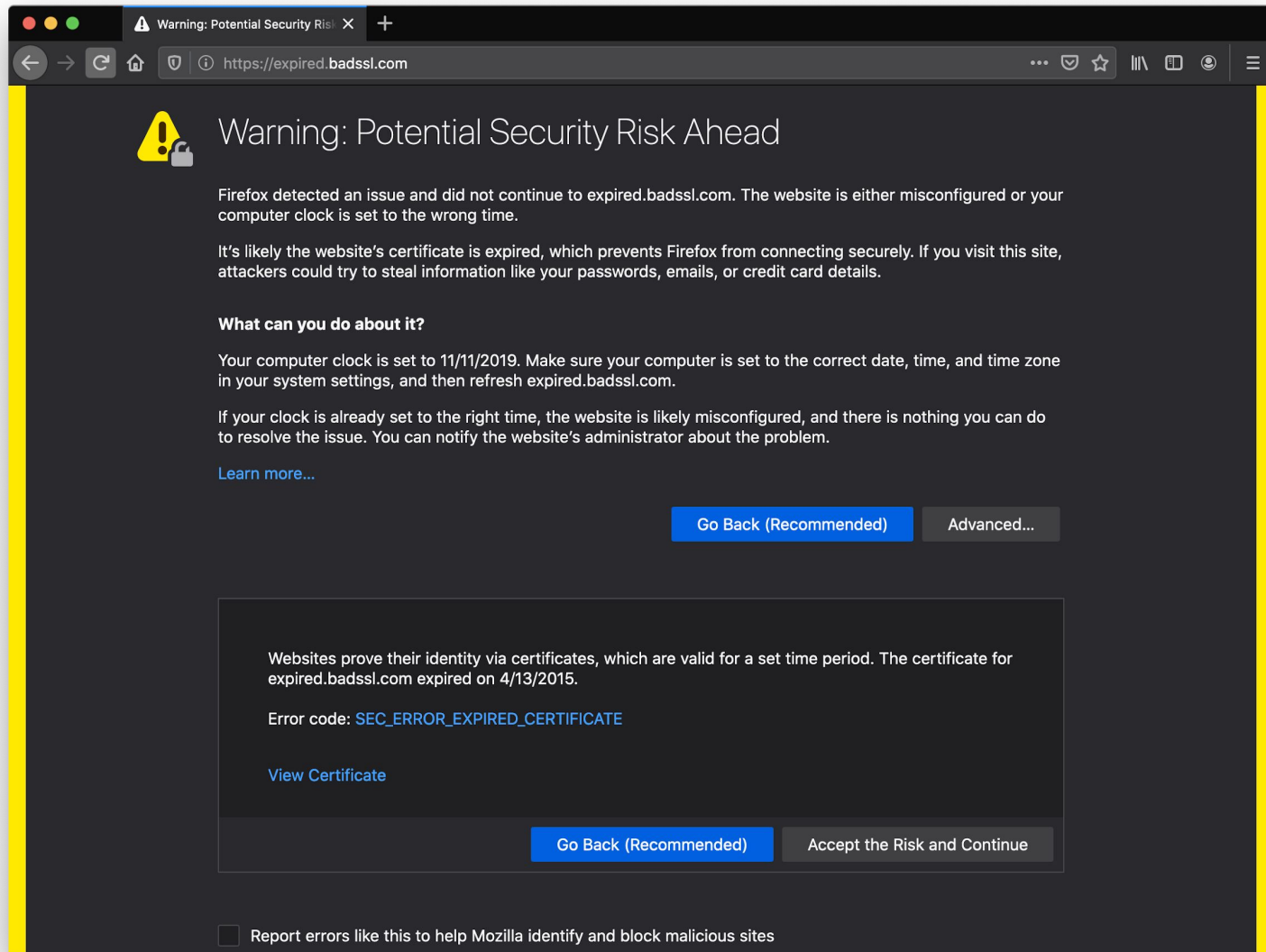


If the user clicks “yes” the assumptions of SSL are violated and attack is successful.

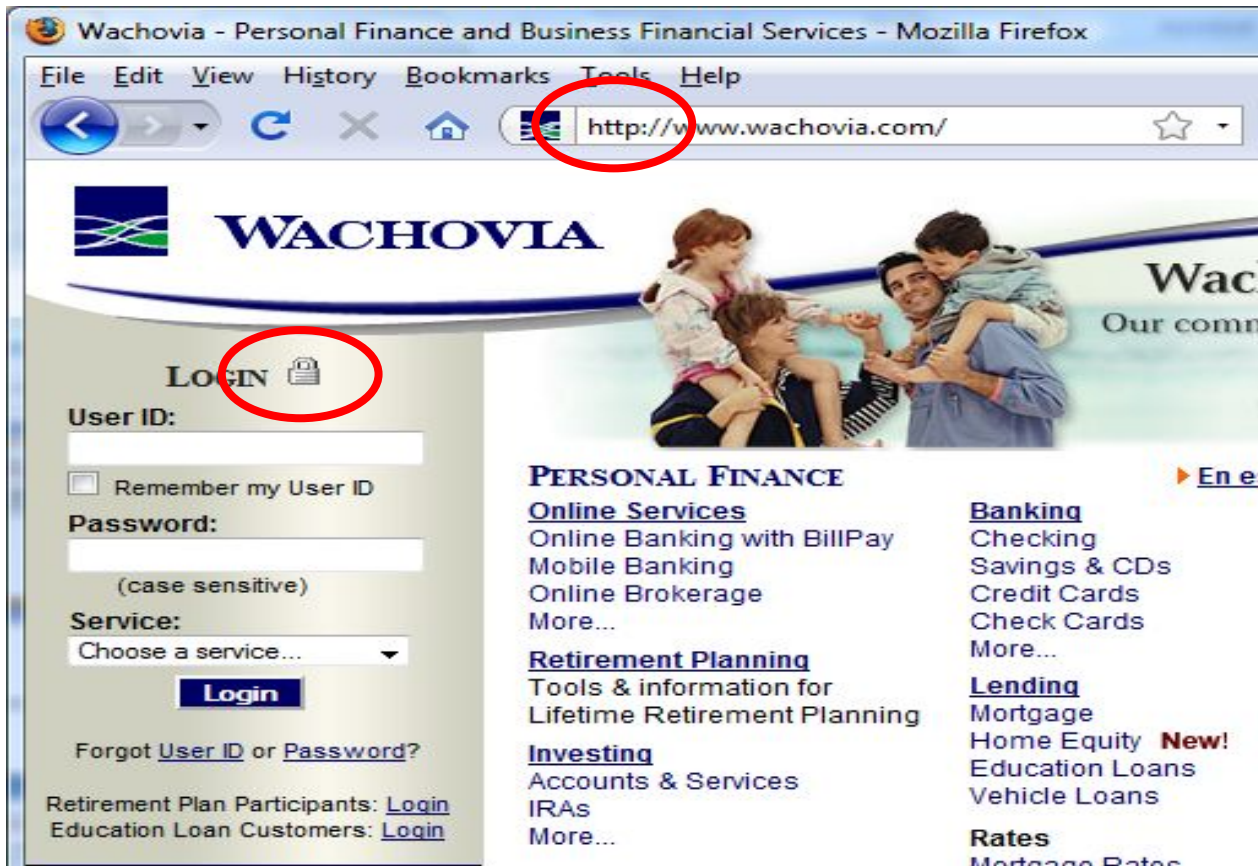
If he clicks no, he cannot buy his book.

Guess what’s going to happen...

# Security UI: Evolution



# Security UI Pitfalls & SSL strip



View source:  
<form method="post"

Source: <https://cs155.stanford.edu/>

action="**https://onlineservices.wachovia.com/...**"

# Security UI: Evolution

## Treatment of HTTP pages

Current (Chrome 67)

ⓘ example.com

July 2018 (Chrome 68)

ⓘ Not secure | example.com

## Treatment of HTTPS pages

Current (Chrome 67)

🔒 Secure | example.com

Sep. 2018 (Chrome 69)

🔒 example.com

Eventually

example.com



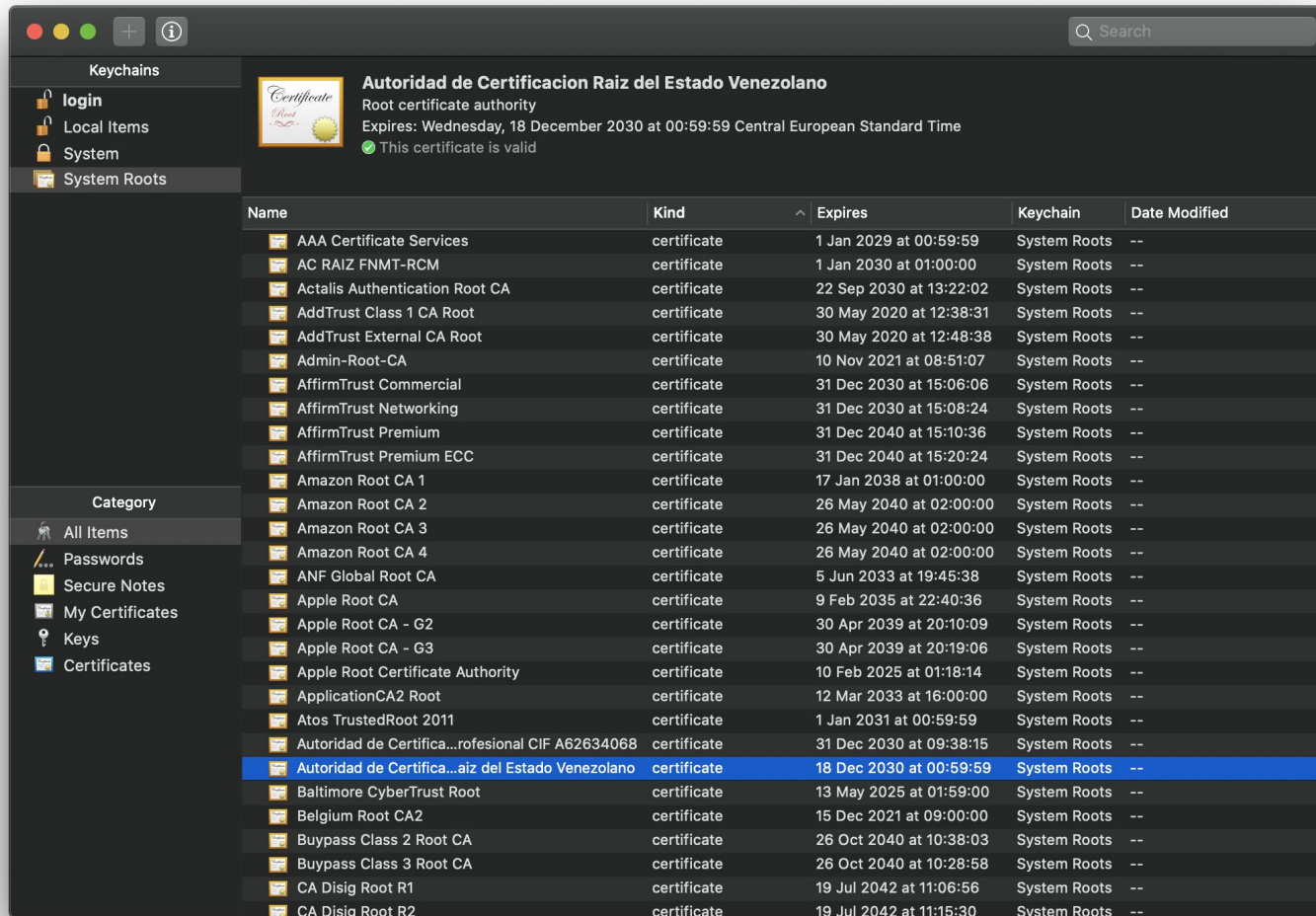
# SSL: Pros and Cons

- Protects transmissions
  - Confidentiality
  - Integrity
- Ensures authentication
  - Of server
  - Of client (optionally)
- No protection before or after transmission
  - On server
  - On client (e.g., trojan)
  - By abuser (e.g., non-honest merchant)
- Relies on PKI
- Not foolproof

# TLS cons: reliance on PKI

- Security guarantees of TLS depend on the security and trust of the root and intermediate CAs
- CAs can generate certs for any domain
  - They are responsible for domain\org validation
- In order to be included in browsers and OS root programs, CAs must abide to a set of requirements (CA/Browser Forum baseline requirements)
  - Dropping a non-compliant CA is a very difficult decision as it breaks websites

# Reminder: CAs your OS trusts



Corollary: you need to trust the list of CA pre-installed in your computer.  
It is not always so: see the 2015 Lenovo Superfish incident...

# Pitfalls: A few CA Incidents

- **2011:** Diginotar as well as some Comodo resellers are compromised → rogue certs (at least 500 for Diginotar)
  - Diginotar is distrusted on all major platforms
- **2009 - 2015:** Symantec → test certificates for existing domains (e.g., Google)
  - Caught through CT logs
  - Outcome: CA gradually distrusted (2016-2018)
- **2012:** Trustwave issues a MITM certificate for a data loss prevention appliance
- **2012:** TurkTrust mistakenly gives two CA certificates to users
- **2016:** WoSign / StartSSL → various issues, including cert misissuance due to vulnerability in the domain verification.
  - Outcome: CA distrusted

# Overcoming TLS/PKI limitations

## HSTS (HTTP Strict Transport Security)

- HTTP header to tell the browser to always connect to that domain using HTTPS
- Browsers (e.g., Chrome) implement a HSTS preload list: some websites are simply never loaded over plain HTTP
- Defends against SSL stripping

# Overcoming TLS/PKI limitations

## HPKP (HTTP Public Key Pinning)

- HTTP header to tell the browser to “pin” a specific certificate or a specific CA
- Browser will refuse certificates from different CAs for that origin
- Defends against trust cert mis-issuance
- **Deprecated!** Can you think about why?

# Overcoming TLS/PKI limitations

## Certificate Transparency

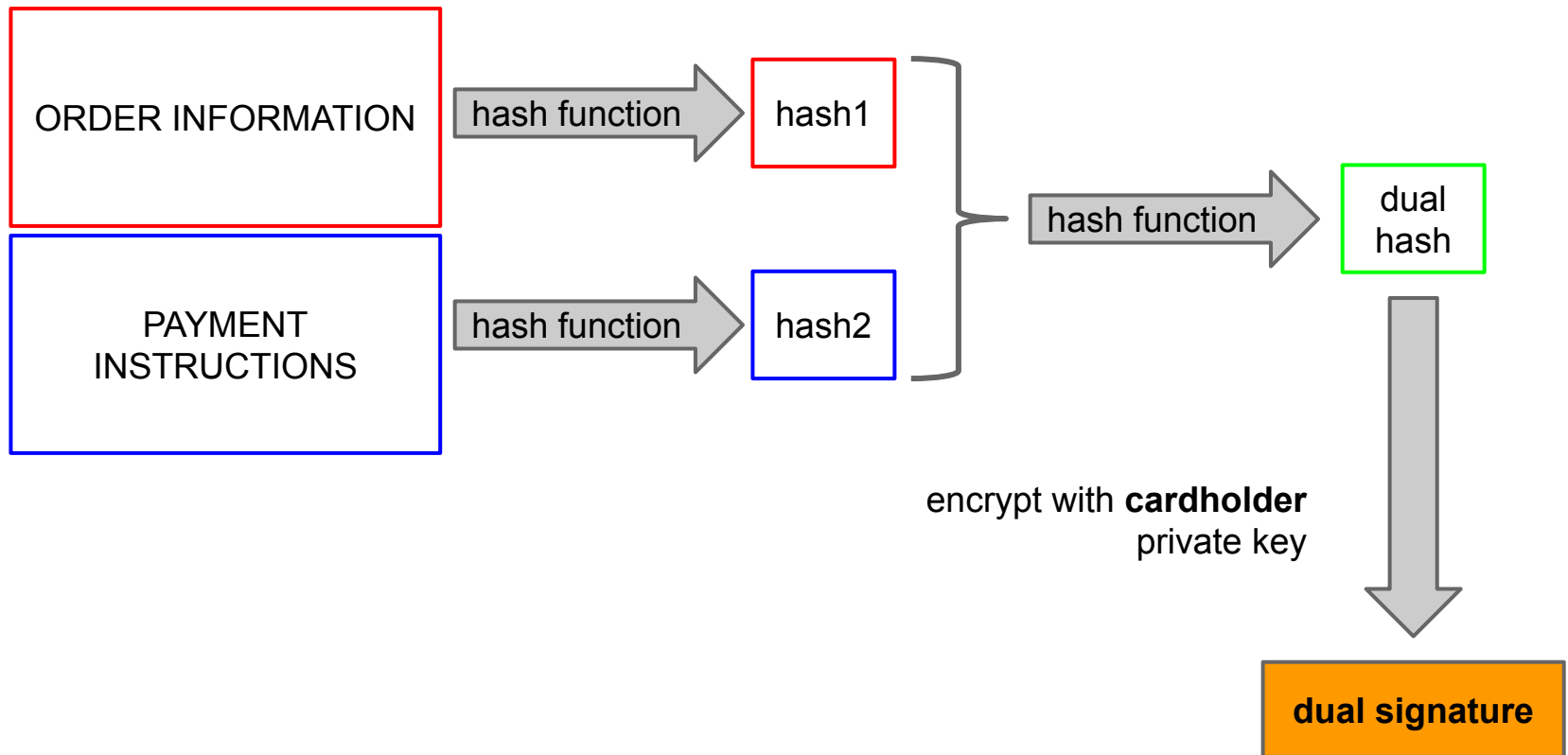
- CA submit the metadata of every issued certificate to a (independent, replicated) log
- Can be enforced by browsers
  - browsers refuse certs not logged in CA logs
- Defends against certificate misissuance: site owner can check \ be notified of certificates issued for the properties they manage
- You can look at the CT logs: <https://crt.sh>

# Introducing SET

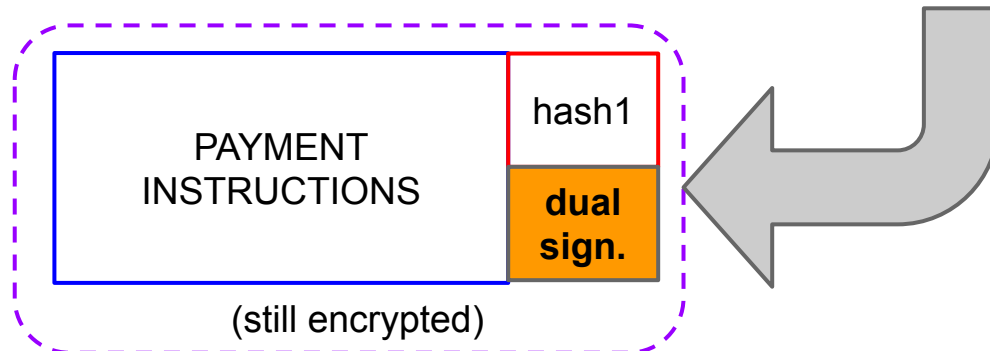
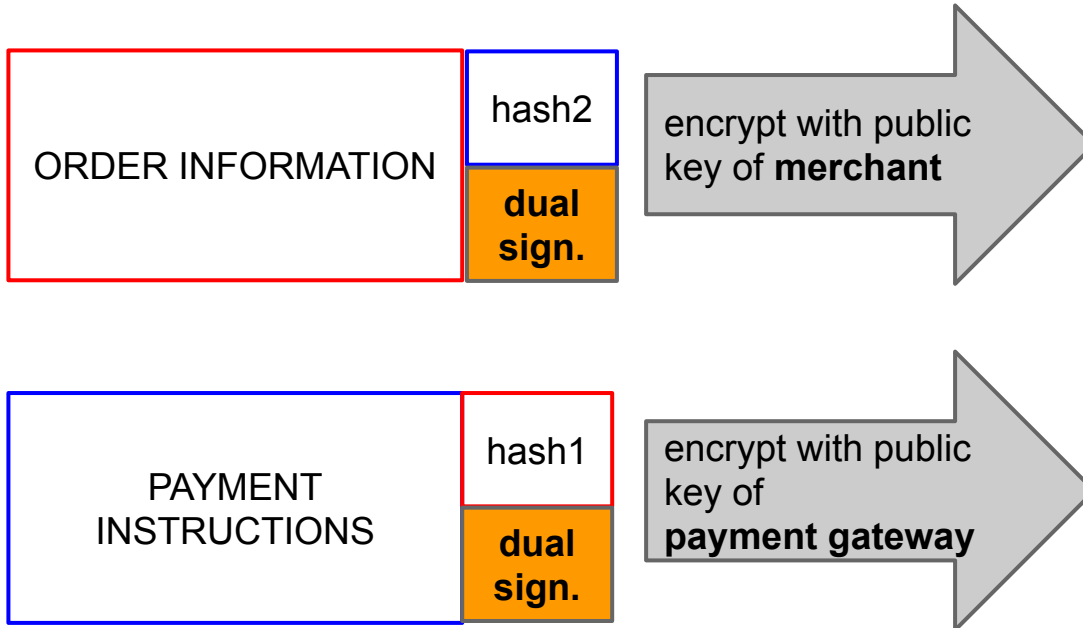
- Joint effort VISA+MasterCard consortium
- Protect *transactions*, not connections
- Approach
  - **Cardholder** sends the **merchant** only the order of goods
  - Send the payment data to the **payment gateway**
  - Empower gateway to verify correspondence
- Uses the concept of a dual signature
  - A signature that joins together the two pieces of a message, directed to two distinct recipients



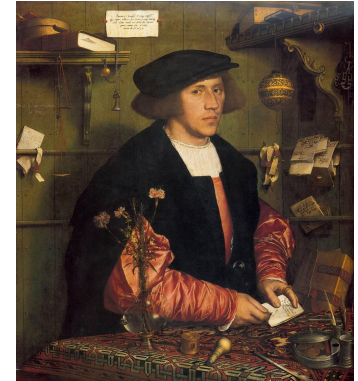
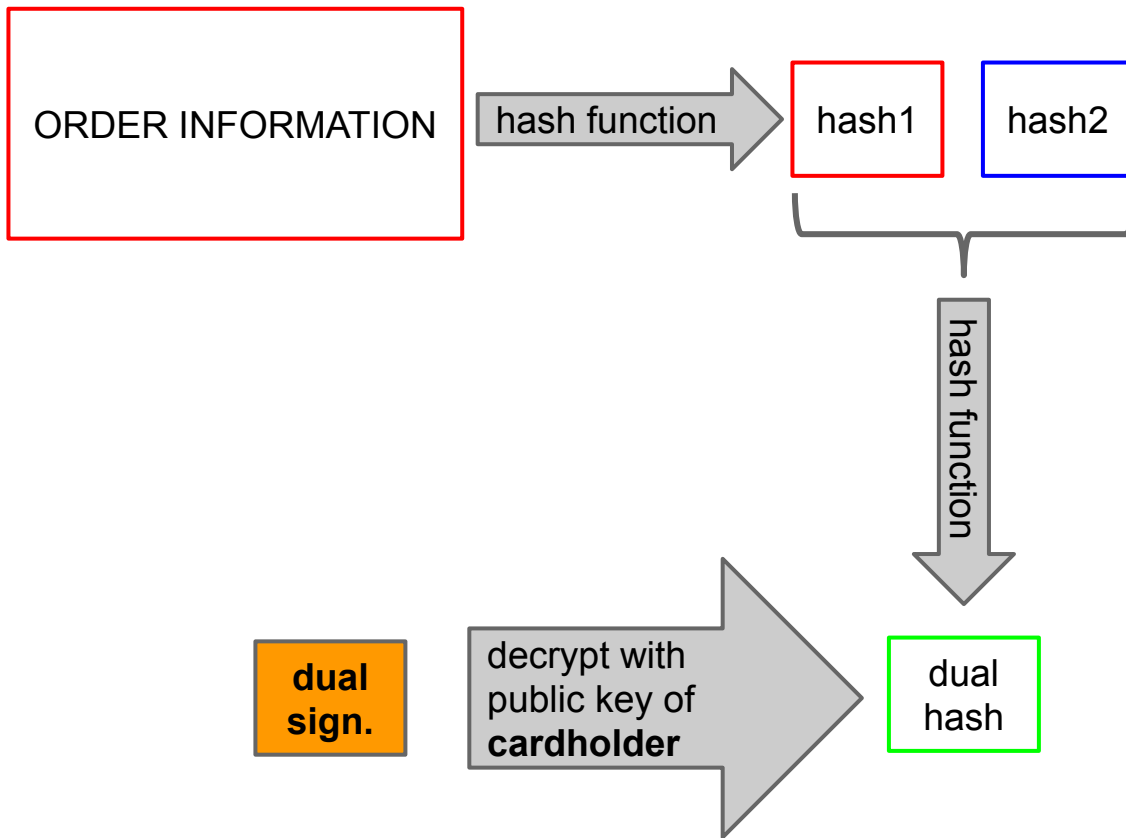
# Dual Signature Generation



# Data Transmission



# Verification (merchant side)



(the payment gateway side verification is the perfect dual)

# Why did SET Fail?

- SET requires a **digital certificate** from:
  - Merchant: OK, reasonable and feasible
  - Payment Gateway: OK, reasonable and feasible
  - **Cardholder: KO, does not scale!**
- Therefore, a pre-registration of the cardholder is needed! (won't buy that book)
- Non-transparent = less critical mass = failure
- Nowadays a simple redirect with a token to the website of the bank is commonly used
  - **Exercise:** think how this is implemented securely ;-)