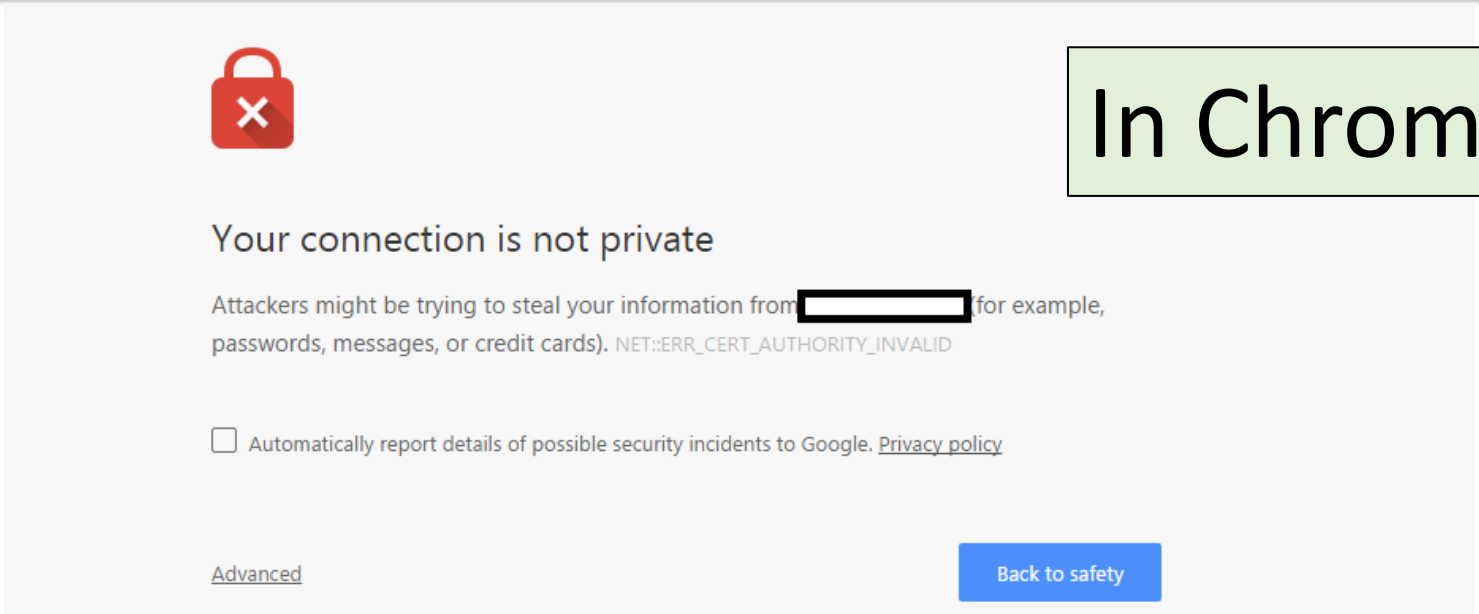
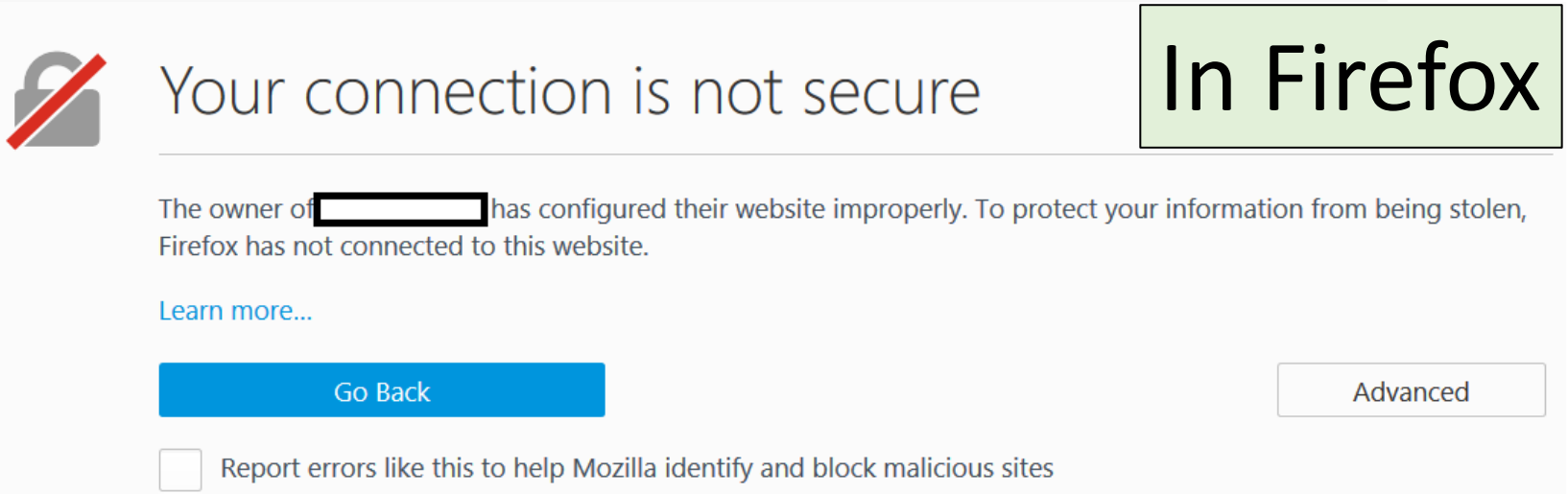
CS 4173/5173
COMPUTER SECURITY

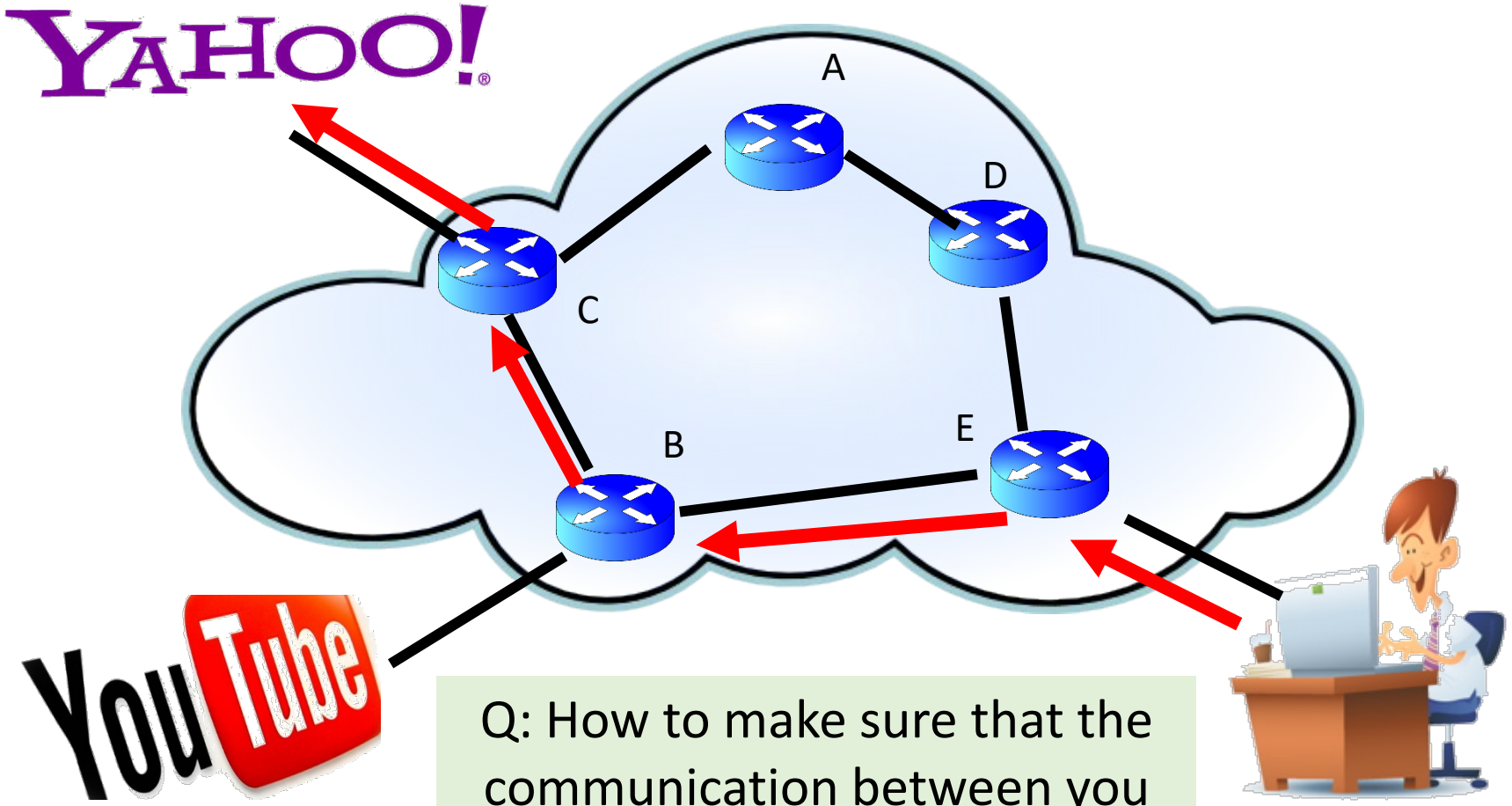
Digital Certificate and PKI

HAVE YOU EVER SEEN THIS?



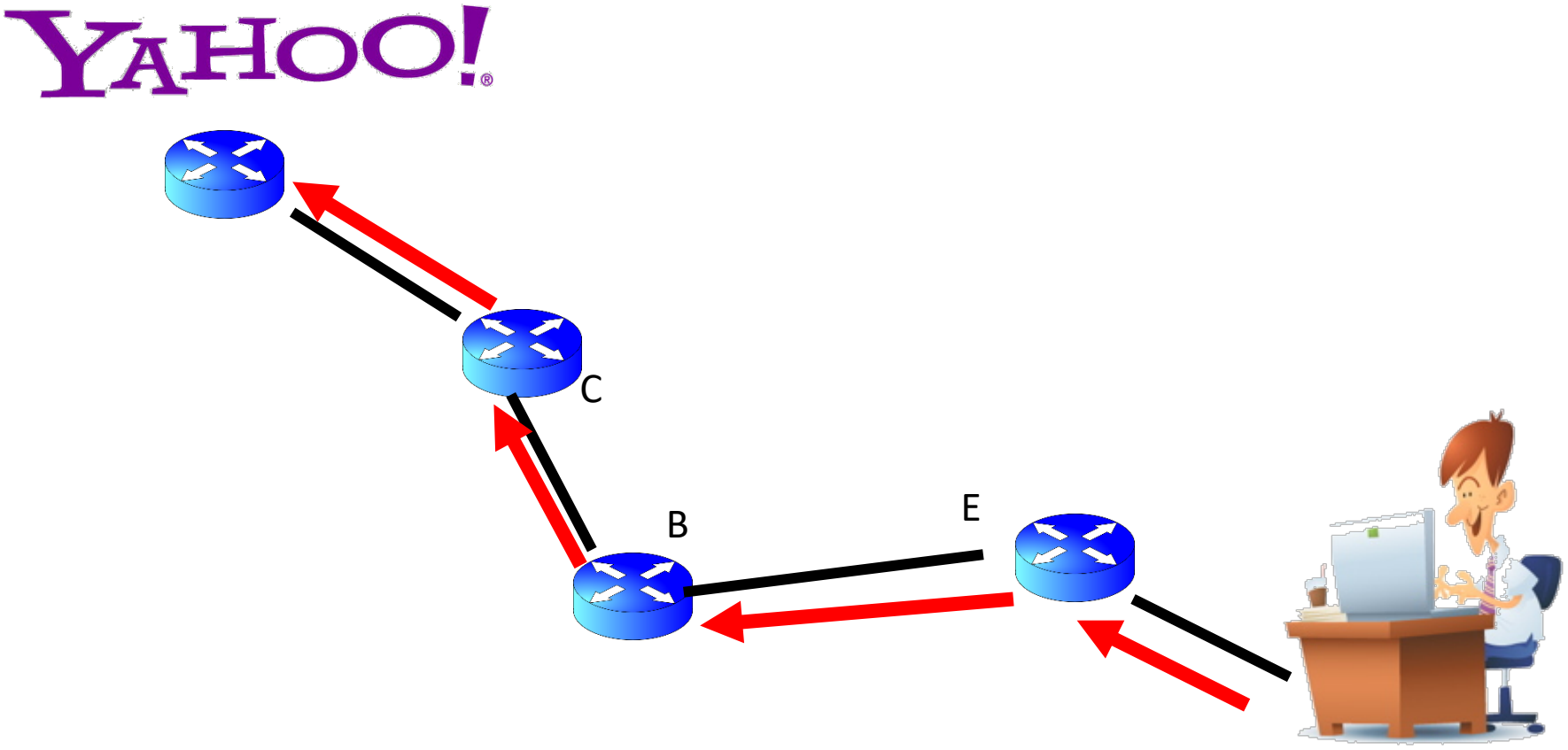


HOW TODAY'S NETWORK WORKS



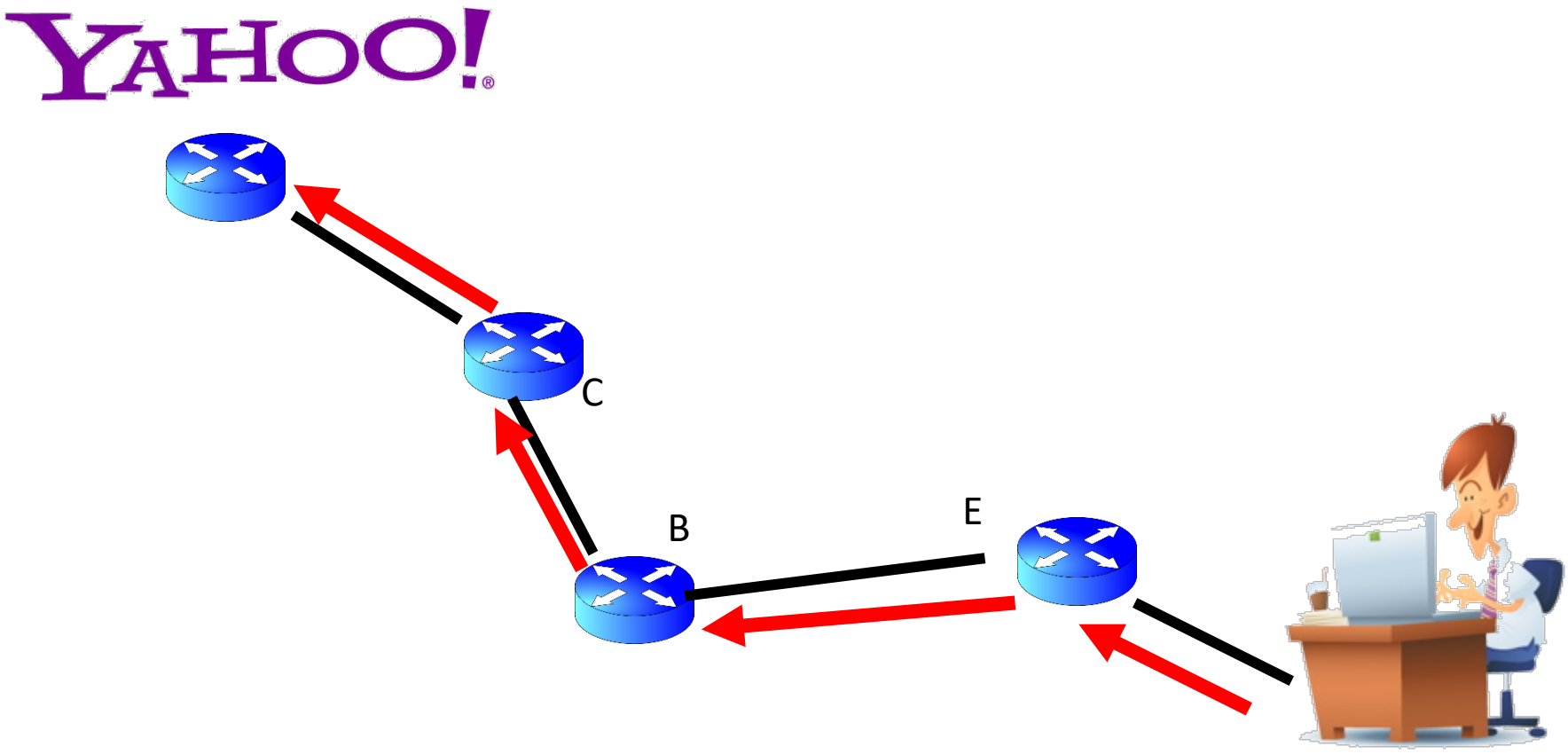
Q: How to make sure that the communication between you and Yahoo is secure!

SOLUTION 0



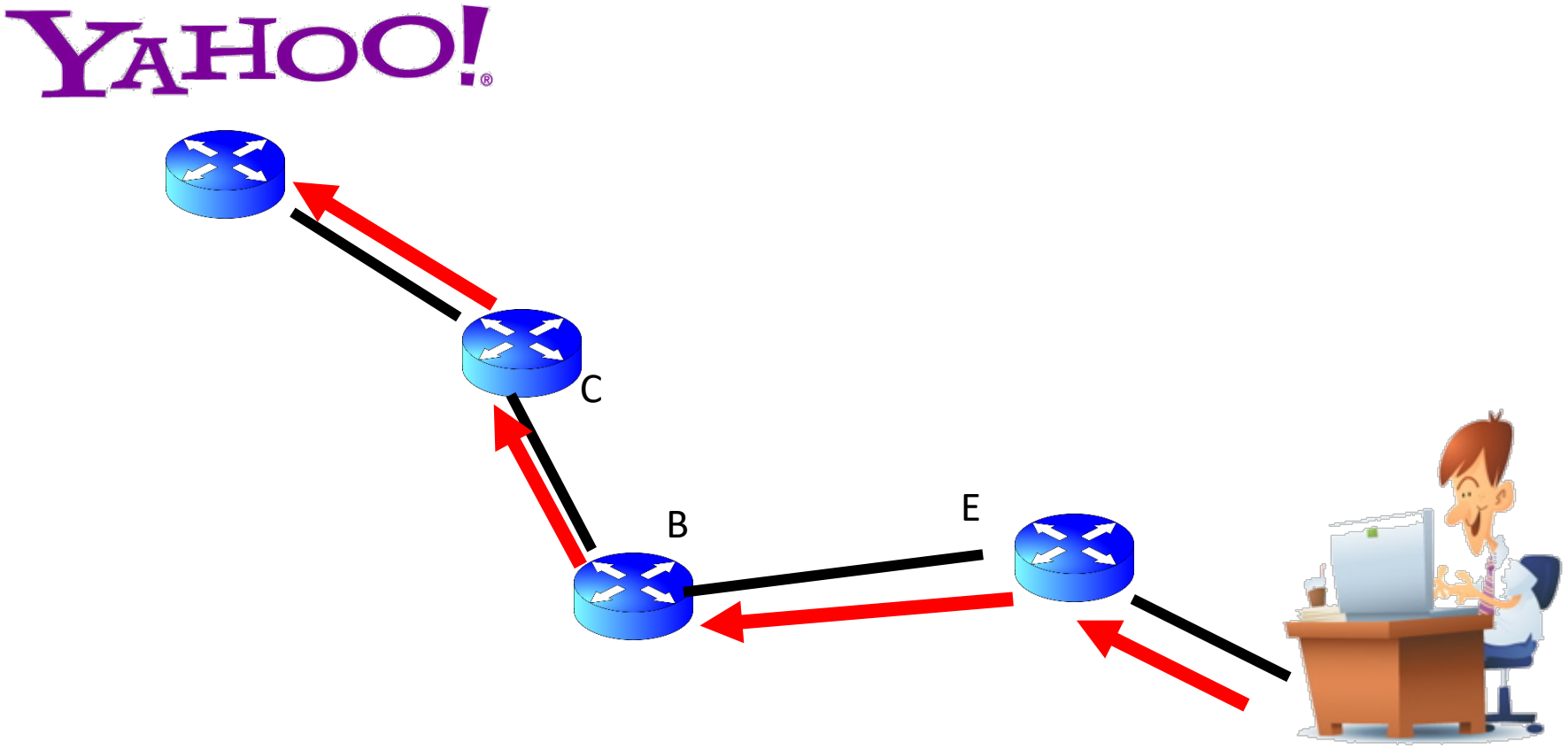
1. User and Yahoo use the same shared key (e.g. AES 128)
   Security and Efficiency?

SOLUTION 1



1. User and Yahoo use Diffie-Hellman to negotiate a key?
   - **Security and Efficiency?**

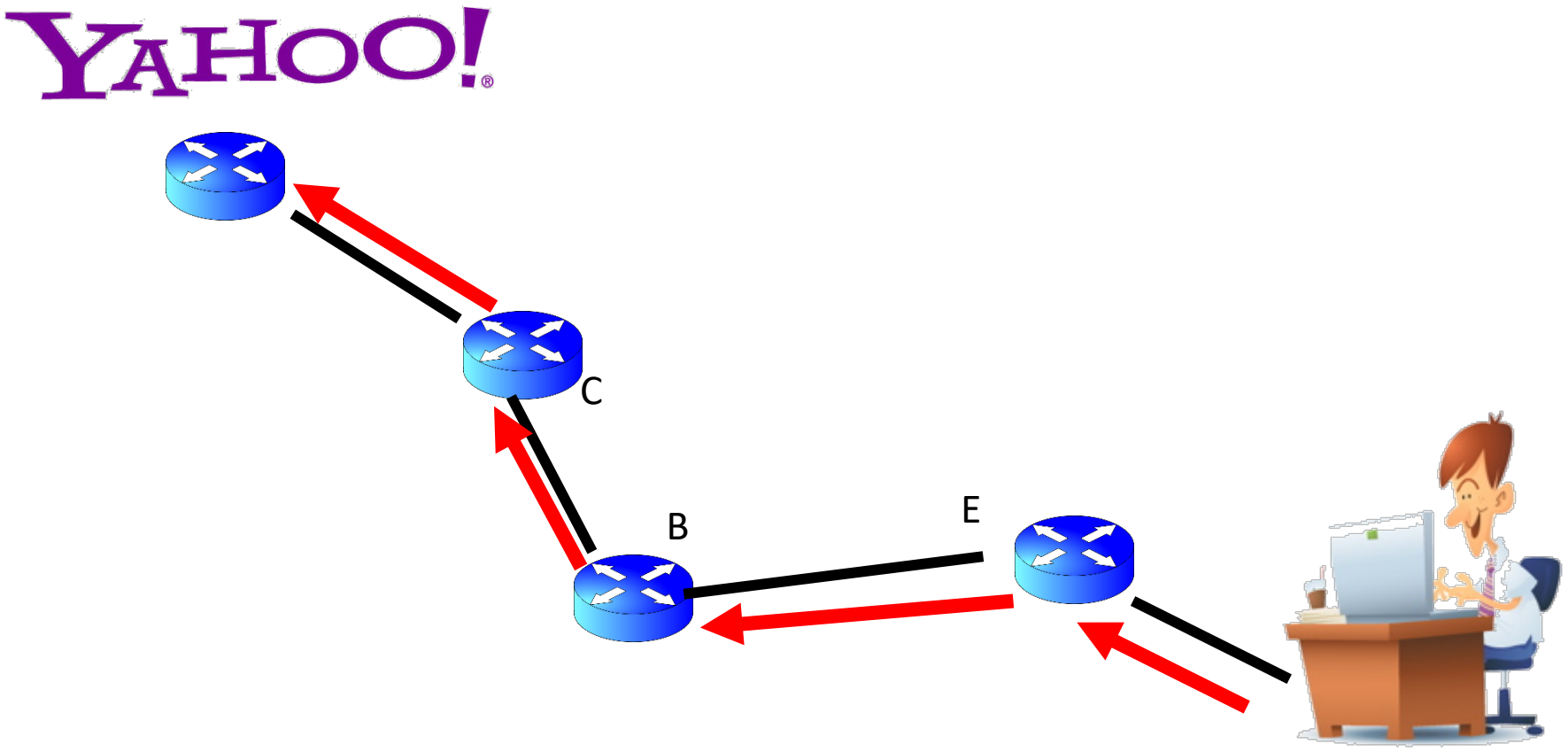
SOLUTION 2



2. Yahoo generates public and private keys and share the public key to the public?

- **Security and Efficiency?**

SOLUTION 3



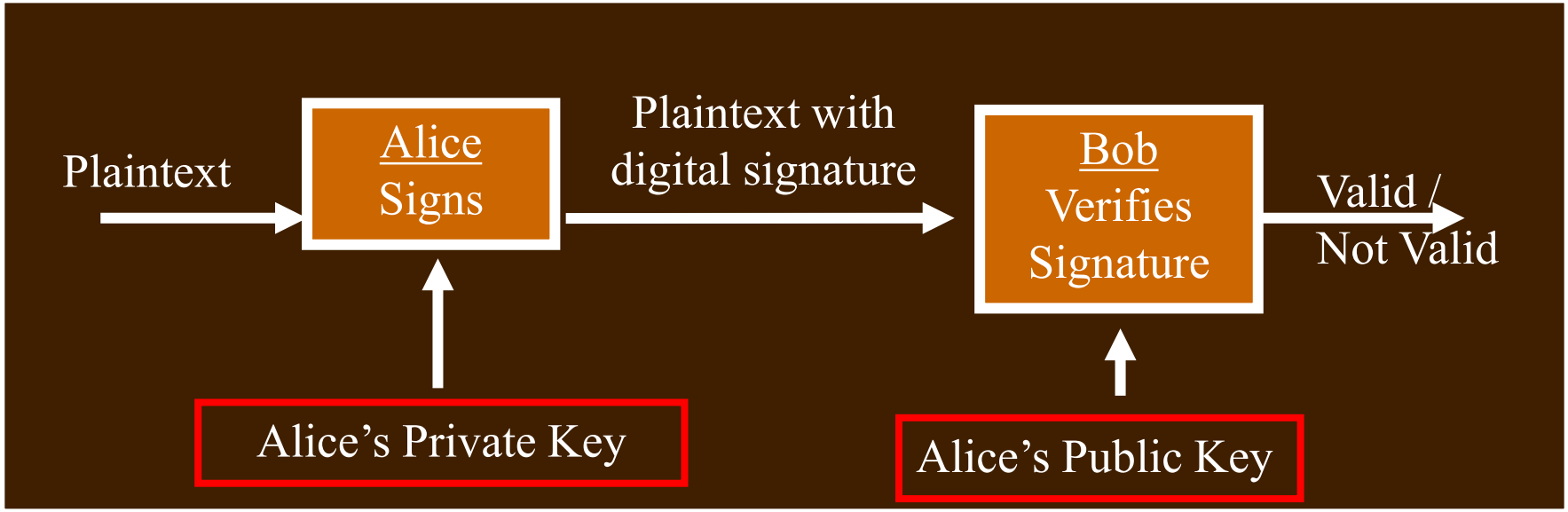
3. User generates public and private keys and share the public key to the public?

- **Security and Efficiency?**

RECALL: AUTHENTICATION IN PUBLIC KEY CRYPTO

Message integrity with *digital signatures*

- Alice computes hash, signs with her private key (no one else can do this without her key)
- Bob verifies hash on receipt using Alice's public key using the verification equation



RECALL: AUTHENTICATION (CONT'D)

- Authentication in public key crypto:
  - Hash function to hash the message into a digest
  - The action of sign the digest with (private key)
  - The action of verify the digest with (public key)

TRUSTED KEY SERVERS

- How do a large number of users authenticate each other?
  - inefficient / impractical for every pair of users to negotiate a secret key or share passwords
- Alternative: everybody shares a key with (and authenticates to) a single trusted third-party

TRUSTED INTERMEDIARIES

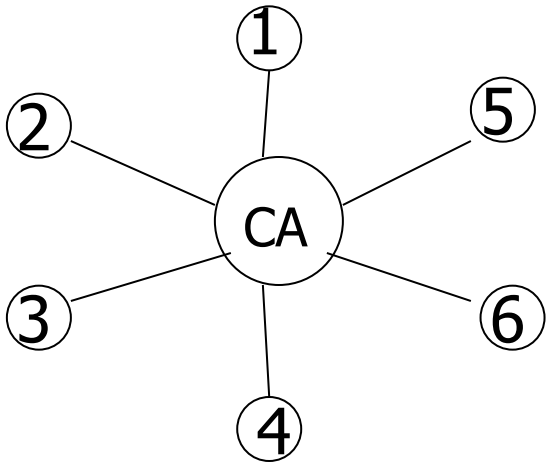
- Problem: authentication for large networks
- Solution #1
  - Public Key Infrastructure (PKI)
  - Based on public key cryptography
- Solution #2
  - Key Distribution Center (KDC)
    - Representative solution: Kerberos
  - Based on secret key cryptography

WHAT IS PKI

- Informally, the infrastructure supporting the use of public key cryptography.
- A PKI consists of
  - Certificate Authority (CA)
  - Certificates
  - A repository for retrieving certificates
  - A method of revoking/updating certificates

CERTIFICATION AUTHORITIES (CA)

- A CA is a trusted node that maintains the public keys for all nodes (Each node maintains its own private key)



If a new node is inserted in the network, only that new node and the CA need to be configured with the public key for that node

CERTIFICATES

- A CA is involved in authenticating users' public keys by generating certificates
- A certificate is a signed message vouching that a particular name goes with a particular public key
- Example:
  1. [Alice's public key is 876234]$_{carol}$
  2. [Ted's public key is 676554]$_{Alice}$ & [Alice's public key is 876234]$_{carol}$

- Knowing the CA's public key, users can verify the certificate and authenticate Alice's public key

CERTIFICATES

- Certificates can hold expiration date and time
- Alice keeps the same certificate as long as she has the same public key and the certificate does not expire
- Alice can append the certificate to her messages so that others know for sure her public key

EXAMPLE

- CA – everyone knows CA's public key.
  - CA is trusted.
- Alice wants to communicate to the real Bob
  - She sends a request to CA
  - Obtains a digital certificate from CA:

Bob's public key is
1902A12B2318871BF1
Expiration: 1/1/2023
[signed by CA]

Bob's D-H g, p, and T are
129381,102A7182019284FF, 910A81213
Expiration: 1/1/2023
[signed by CA]

Q: digital certificate vs digital signature?

CA ADVANTAGES

1. The CA does not need to be online. [Why?]

2. If a CA crashes, then nodes that already have their certificates can still operate.

3. Certificates are not security sensitive (in terms of confidentiality).
   - Can a compromised CA decrypt a conversation between two parties?
   - Can a compromised CA fool Alice into accepting an incorrect public key for Bob, and then impersonate Bob to Alice?

CA PROBLEMS

- What if Alice is given a certificate with an expiration time and then is revoked (fired) from the system?
  - Alice can still use her certificate till the expiration time expires.
  - What kind of harm can this do?
  - Alice can still exchange messages with Bob using her un-expired certificate.

- Solution:
  - Maintain a Certificate Revocation List (CRL) at the CA. A Certificate is valid if (1) it has a valid CA signature, (2) has not expired, and (3) is not listed in the CA's CRL list.

TERMINOLOGY

- A CA signing a certificate for Alice's public key
  - CA → issuer Alice → subject
- Alice wants to find the Bob's public key
  - Bob → target
- Anyone with a public key is a principal
- Alice is verifying a certificate (or a chain of certificates)
  - Alice → verifier
- Trust anchor → A CA with a trusted public key

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

MONOPOLY MODEL

- One CA universally trusted by everyone
- Everyone must get certificates from this CA
- The public key to this organization is the only PKI trust anchor and is embedded in all software and hardware

PROBLEMS

1. There is NO universally trusted organization
2. Monopoly control. CA could charge any fees.
3. Once deployed, it is hard to switch to a different CA
4. Entire world's security relies on this CA
5. Inconvenient.

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

MONOPOLY + REGISTRATION AUTHORITIES (RA)

- RAs are affiliated with the single CA and are trusted by this CA.
- RAs check identities and provide the CA with relevant information (identity and public key information) to generate certificates.
- More convenient
- Still a monopoly. All the monopoly problems still hold.

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

DELEGATED CAS

- The trust anchor (known CA) issues certificates to other CAs (delegated CAs) vouching for their trustworthiness as CAs.

- Users can obtain their certificates from delegated CAs instead of the trust anchor CA.

- Example:
  - [Carol's public key is 676554]$_{Ted}$ & [Alice's public key is 876234]$_{carol}$
  - Ted: trust anchor CA & Carol: delegated CA

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

OLIGARCHY MODEL

- A few trusted CAs and a certificate issued by any one of them is accepted
- Competition between CAs is good
- Problems: Not as secure as the monopoly case
  - Need to protect more CAs (instead of only one)
  - Might be easier to trick a naïve user by inserting a bogus trust anchor in the list of trusted CAs
    - How do you trust a give list of trusted CAs?
  - It is hard to examine the set of trust anchors and determine whether some has modified the set

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

ANARCHY MODEL (WEB OF TRUST)

- Fully distributed approach. No CA or list of CA provided to the users. Anyone can sign certificates for anyone else.
- Each user is responsible for configuring some trust anchors .
- A database maintains these certificates.
- Unworkable on a large scale.

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

NAME CONSTRAINTS

- A CA is responsible for certifying users in his domain only
  - OU CA certifies OU students/faculty/staff
- Provides complete autonomy
- CAs need to be able to identify each other.
  - How?

PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

TOP-DOWN WITH NAME CONSTRAINTS

- Everyone agrees on a root organization and the root CA delegates to other CA. (A centralized trust anchor (CA) + delegated CAs).
- To get a certificate, contact the root.
- You will be redirected to an appropriate delegated CA.
- Delegated CAs can only issue certificates for users in their domain.

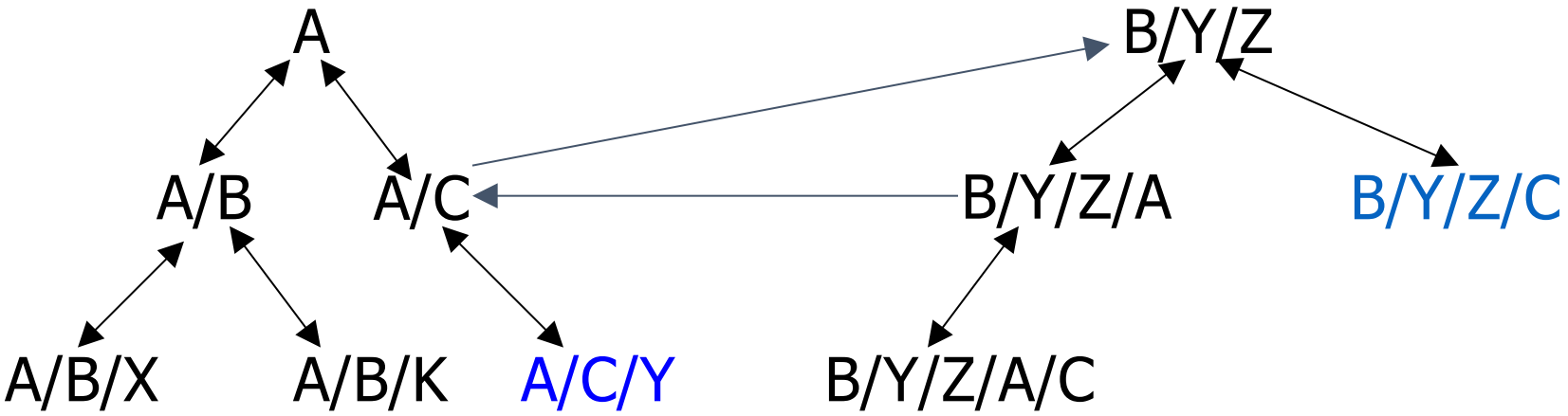
PKI MODELS

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

BOTTOM-UP WITH NAME CONSTRAINTS

- Each organization maintains its own CA, and CAs link to others.
  - A parent certifies its children and the children certify their parent
- The hierarchy is traversed in a bottom-up fashion.
  - In addition to up and down links, cross links are allowed

BOTTOM-UP WITH NAME CONSTRAINTS



How can A/C/Y verify the certificate of B/Y/Z/C?

How can B/Y/Z/C verify the certificate of A/C/Y?

Solution: Follow up-links until you encounter an ancestor of the target, then follow at most one cross-link, and then follow down-links from there

YAHOO'S CERTIFICATE

Certificate Hierarchy

- VeriSign Class 3 Public Primary Certification Authority - G5
  - Symantec Class 3 Secure Server CA - G4
    - www.yahoo.com

If the browser cannot verify the certificate: