Key Management

CS 161: Computer Security

Prof. Vern Paxson

TAs: Paul Bramsen, Apoorva Dornadula,
David Fifield, Mia Gil Epner, David Hahn, Warren He,
Grant Ho, Frank Li, Nathan Malkin, Mitar Milutinovic,
Rishabh Poddar, Rebecca Portnoff, Nate Wang

http://inst.eecs.berkeley.edu/~cs161/

March 2, 2017

Digital Signatures

- Idea: as with public-key encryption, leverage a function that's easy to compute but intractable to invert … *unless* one possesses some private information
  - But instead, do this for a function that's hard to compute without private info, but easy to invert
- One way to produce such a function: use the inverse of a public-key encryption function
- *For example*, consider RSA ...

RSA Digital Signatures

- Alice generates public/private key pair, {n', e'} and {d'}
  - Prudent: ≠ her public/private keys for encryption
- … chooses, makes public a secure hash function H
- To sign a message M, she computes

  $\mathcal{S} = \text{SIGN}_{d'}(M) = H(M)^{d'} \bmod n'$
- *Anyone* (not just recipient Bob) can verify her signature on {M, $\mathcal{S}$} via

  $\text{VERIFY}_{n',e'}(M, \mathcal{S}) = \text{true}$ iff $H(M) = \mathcal{S}^{e'} \bmod n'$
- This follows from $(H(M)^{d'})^{e'} = (H(M)^{e'})^{d'} = H(M) \bmod n'$ (by previous analysis of RSA)

Considerations for Digital Signatures

- Any change to M will alter H(M), and therefore the computed S
  - Thus, detectable ⇒ provides integrity

- Security rests on difficulty of finding inverse of e, along with H being cryptographically strong

- Because anyone can confirm signature validity if Alice's public signature key is well-known, provides *non-repudiation*

Considerations for Digital Signatures, con't

- Non-repudiation:
  - Alice can't *deny to a third party* that she signed M (unless argues her private key was stolen)
  - Similar to a handwritten signature, but in fact *better* since can't be "digitized" and pasted into another document M*
    - Because {M*, *S*} won't validate
- Per previous example, to sign Firefox binaries Mozilla could simply just *once* publish a public key, and then "use it" to sign each release

Agreeing on Secret Keys

Without Prior Arrangement

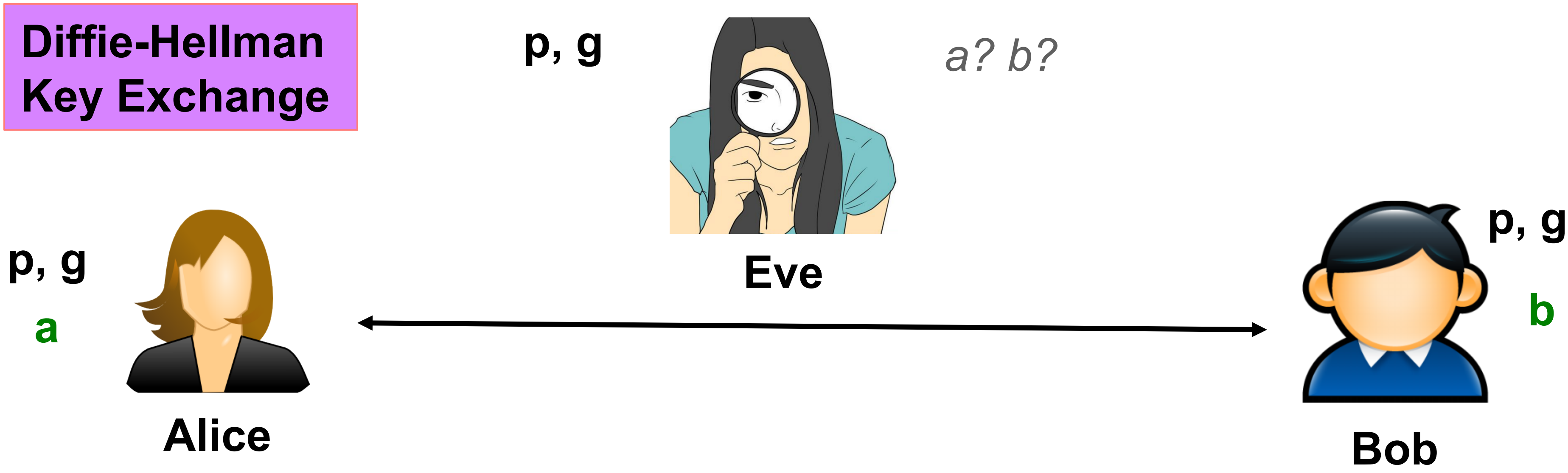
Diffie-Hellman Key Exchange

- While we have powerful symmetric-key technology, it requires Alice & Bob to agree on a secret key *ahead of time*
- What if instead they can somehow generate such a key *when needed*?
- Seems impossible in the presence of Eve observing all of their communication …
  - How can they exchange a key without her learning it?
- But: actually is possible using public-key technology
  - Requires that Alice & Bob **know** that their messages will reach one another without any meddling
  - So works for Eve-the-eavesdropper, but not Mallory-the-MITM
  - Protocol: *Diffie-Hellman Key Exchange* (DHE)

Diffie-Hellman Key Exchange

1. Everyone agrees in advance on a well-known (large) prime **p** and a corresponding **g**: 1 < g < p-1

Diffie-Hellman Key Exchange



2. Alice picks random secret ‘**a**’: 1 < a < p-1

3. Bob picks random secret ‘**b**’: 1 < b < p-1

Diffie-Hellman Key Exchange

p, g

A

B

Eve

a? b?

p, g

a

Alice

p, g

b

Bob

A = g^a mod p

g^b mod p = B

4. Alice sends A = g^a mod p to Bob

5. Bob sends B = g^b mod p to Alice

Diffie-Hellman Key Exchange

p, g

A

B

Eve

a? b?

p, g

a

K

Alice

p, g

b

K

Bob

$A = g^a \bmod p$

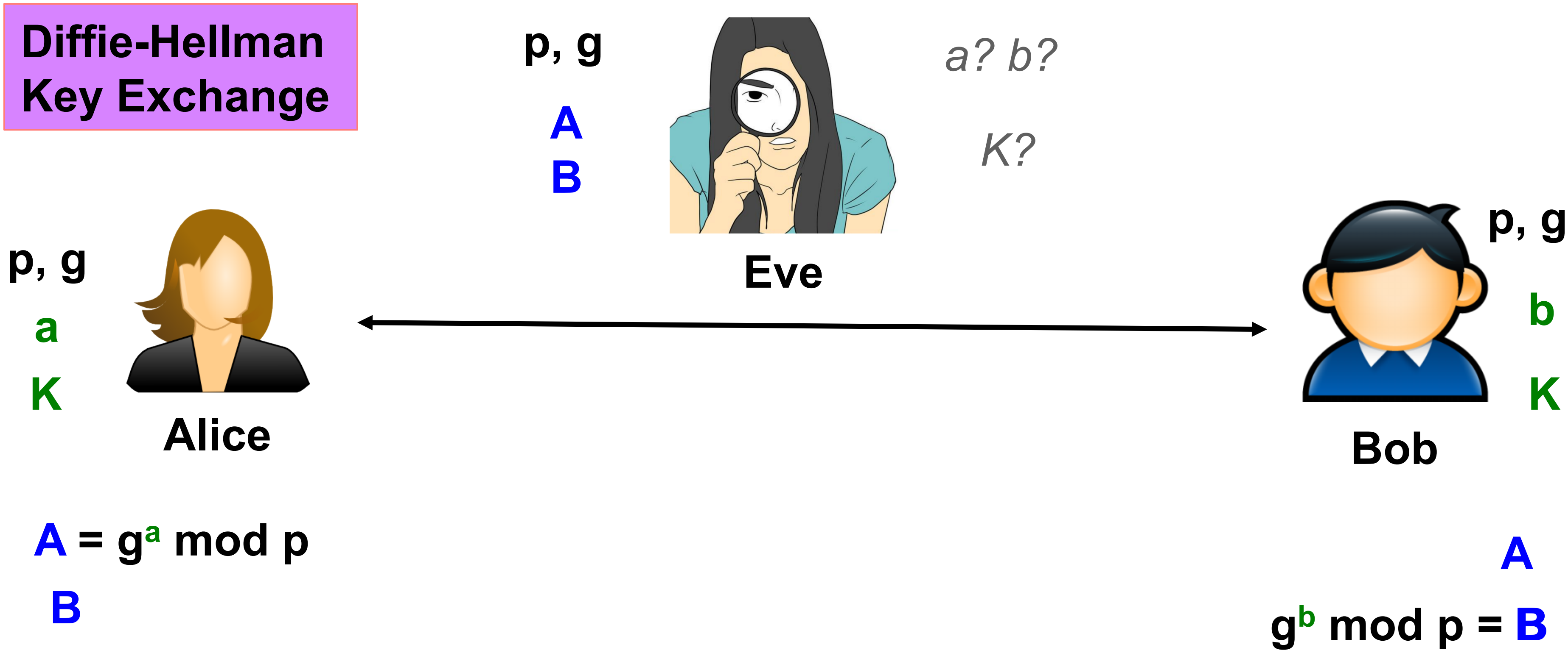
B

A

$g^b \bmod p = B$

6. Alice knows {a, A, B}, computes
   $K = B^a \bmod p = (g^b)^a = g^{ba} \bmod p$
7. Bob knows {b, A, B}, computes
   $K = A^b \bmod p = (g^a)^b = g^{ab} \bmod p$
8. K is now the shared secret key.

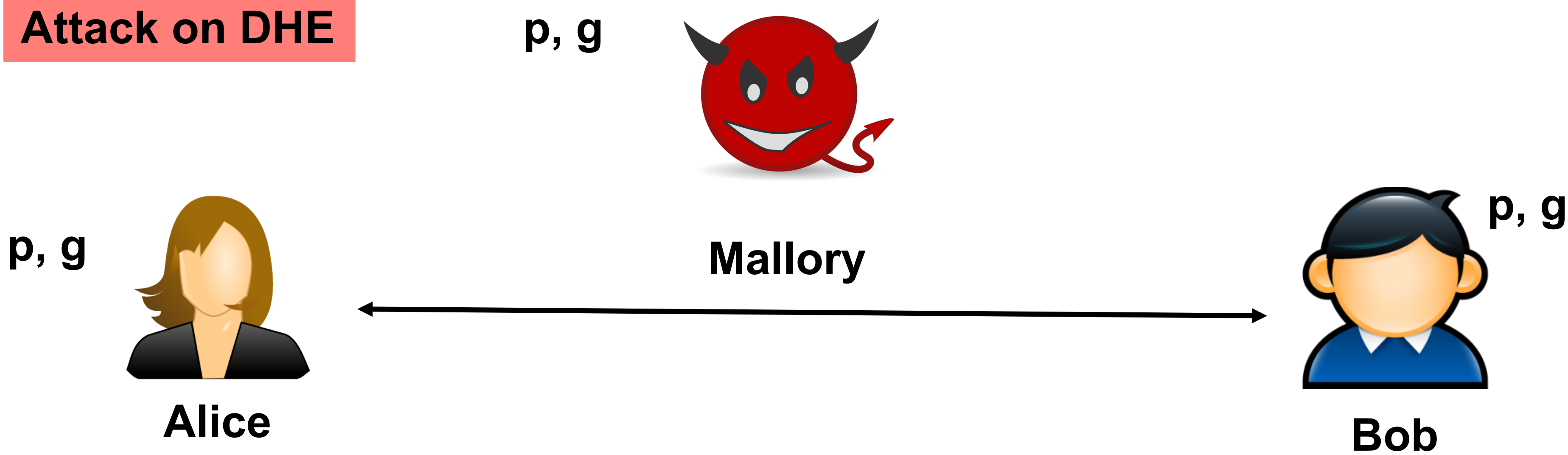
Diffie-Hellman Key Exchange



While Eve knows {p, g, g^a mod p, g^b mod p}, believed to be ***computationally infeasible*** for her to then deduce K = g^{ab} mod p.

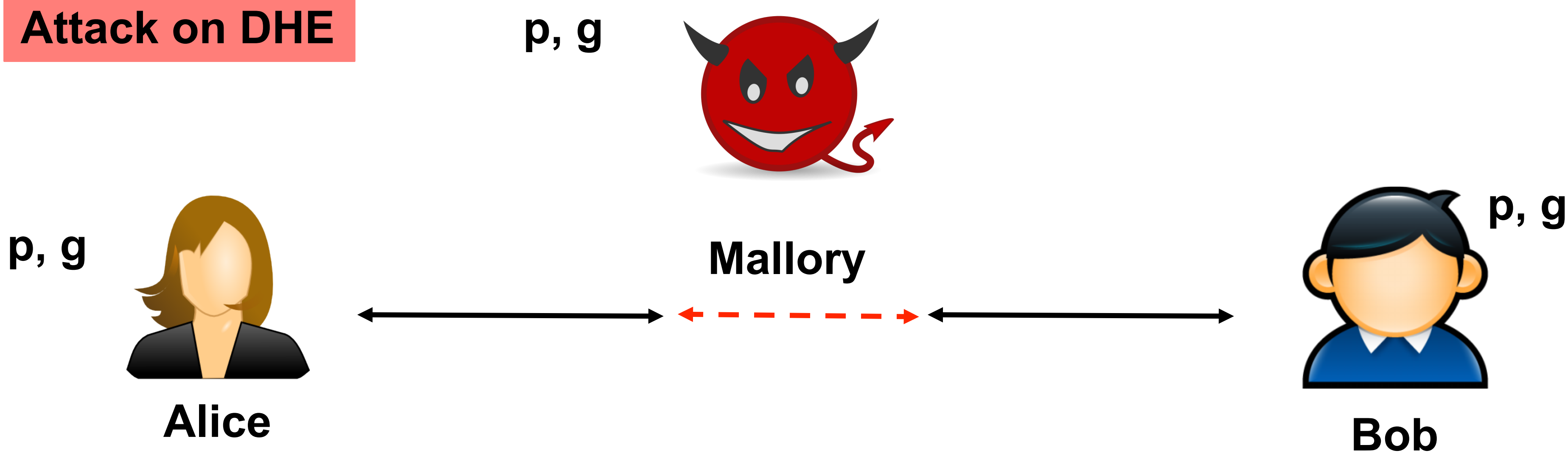
She can easily construct A·B = $g^a \cdot g^b$ mod p = g^{a+b} mod p.
But computing g^{ab} requires ability to take *discrete logarithms* mod p.

Attack on DHE



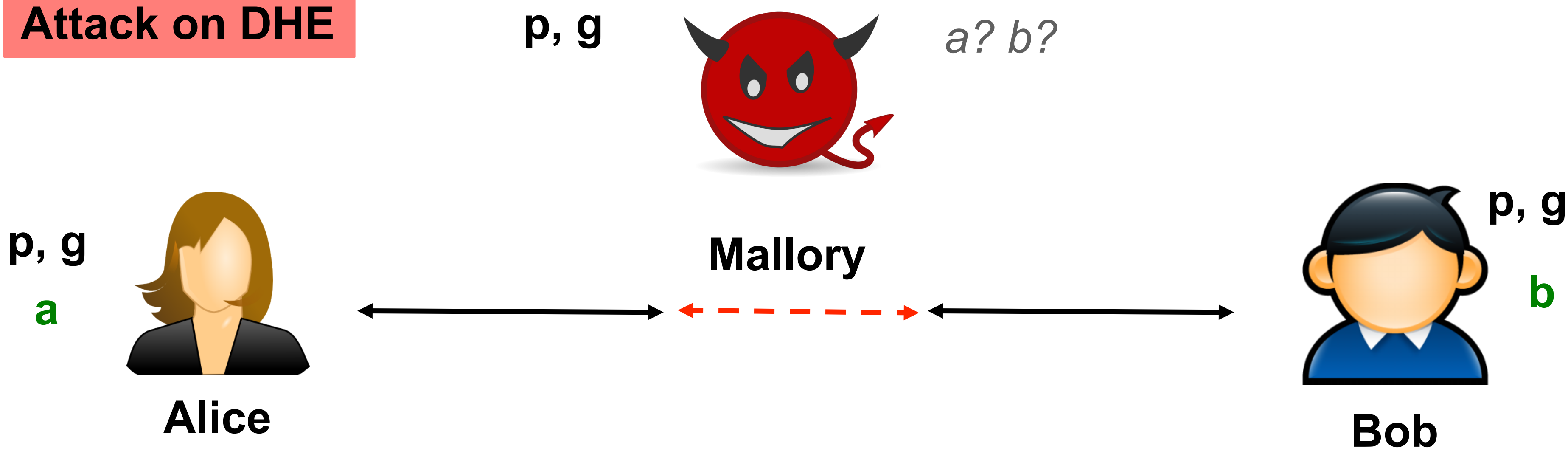
What happens if instead of Eve watching, Alice & Bob face the threat of a hidden Mallory (MITM)?

Attack on DHE



What happens if instead of Eve watching, Alice & Bob face the threat of a hidden Mallory (MITM)?

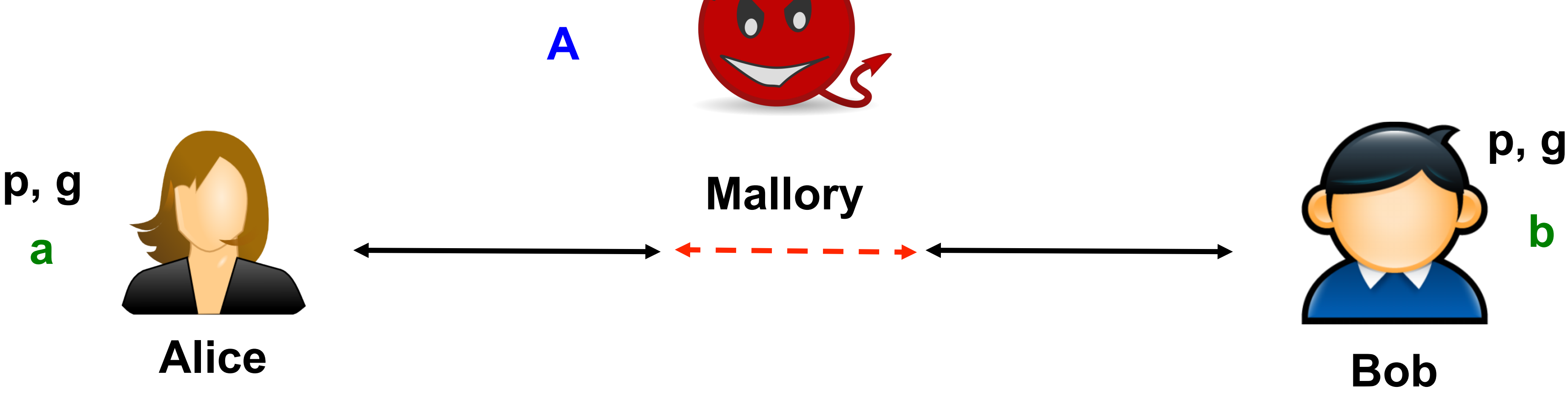
Attack on DHE



2. Alice picks random secret ‘**a**’: 1 < a < p-1

3. Bob picks random secret ‘**b**’: 1 < b < p-1

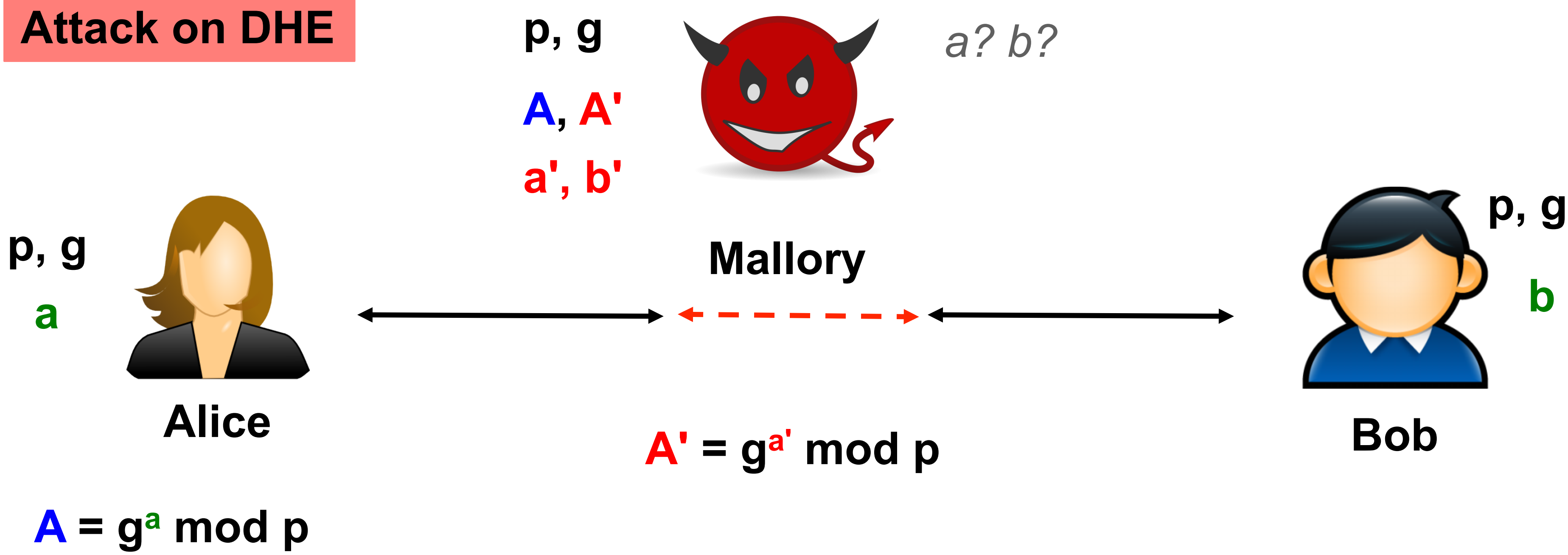
Attack on DHE



$A = g^a \bmod p$

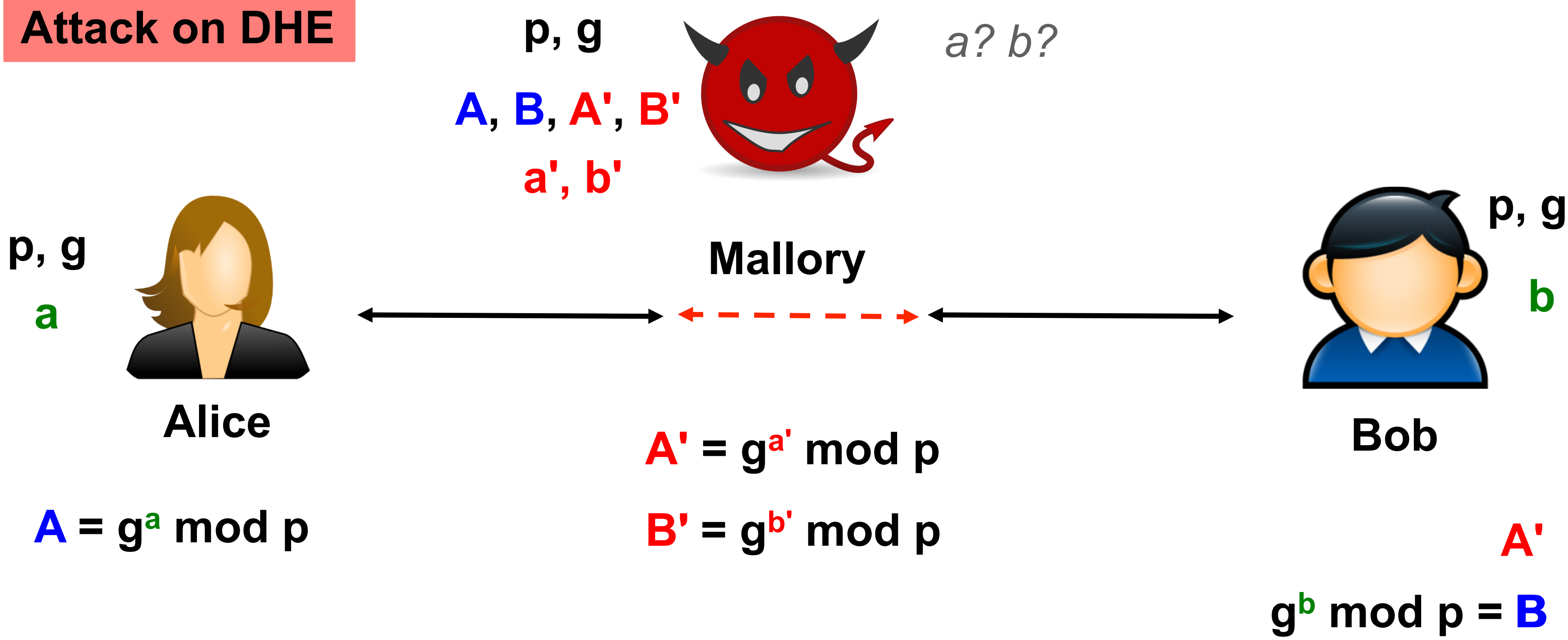
4. Alice sends $A = g^a \bmod p$ to Bob
5. Mallory prevents Bob from receiving A

Attack on DHE



6. Mallory generates her own a', b'
7. Mallory sends $A' = g^{a'} \bmod p$ to Bob

Attack on DHE



8. The same happens for Bob and B/B'

Attack on DHE

p, g

A, B, A', B'

a', b'

a? b?

Mallory

p, g

a

Alice

p, g

b

Bob

$A' = g^{a'} \bmod p$

$B' = g^{b'} \bmod p$

$K'_1 = A^{b'} \bmod p = g^{ab'} \bmod p$

$K'_2 = B^{a'} \bmod p = g^{ba'} \bmod p$

$A = g^a \bmod p$

B'

$K'_1 = (B')^a \bmod p = (g^{b'})^a = g^{b'a} \bmod p$

A'

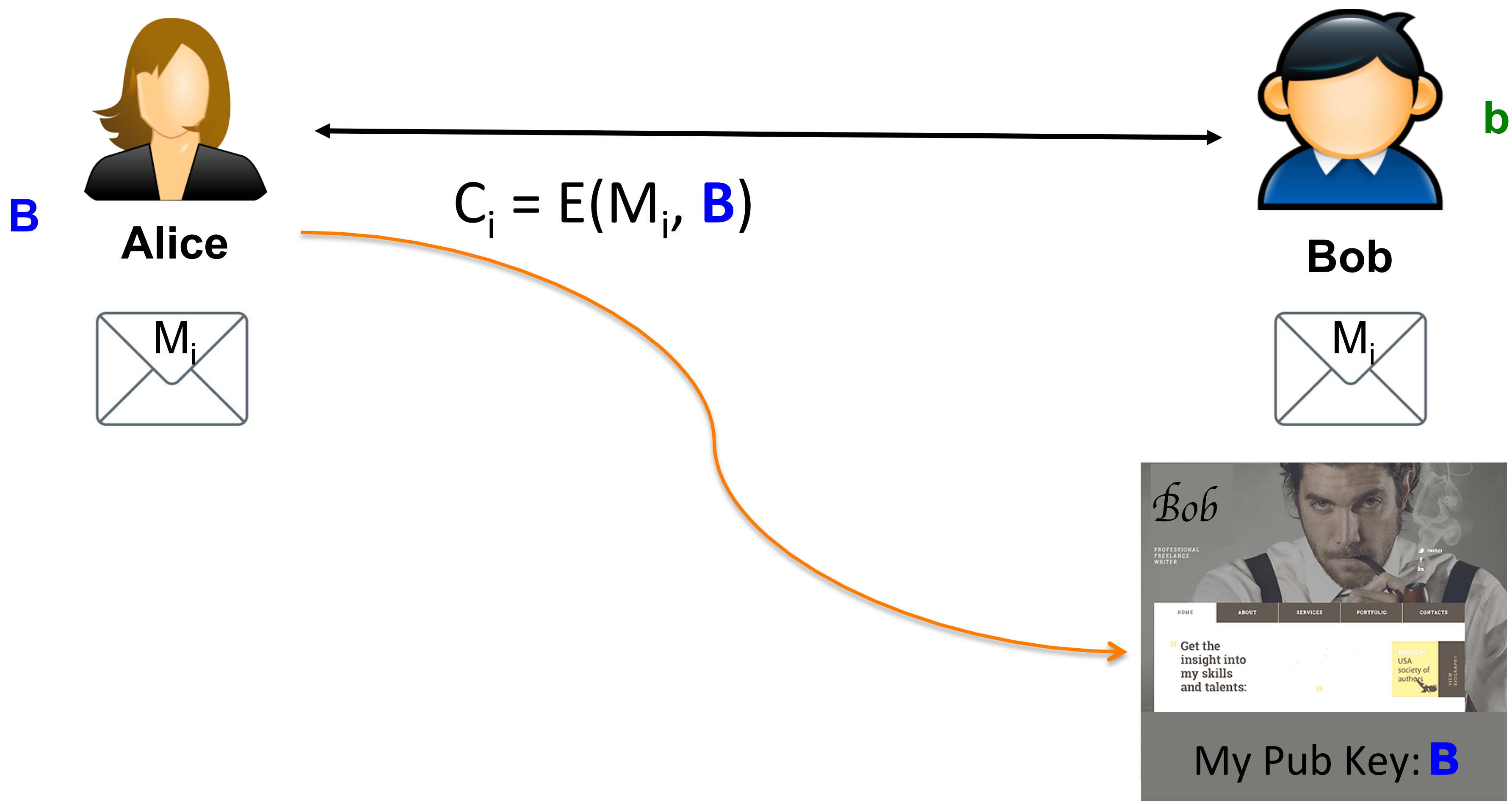
$g^b \bmod p = B$

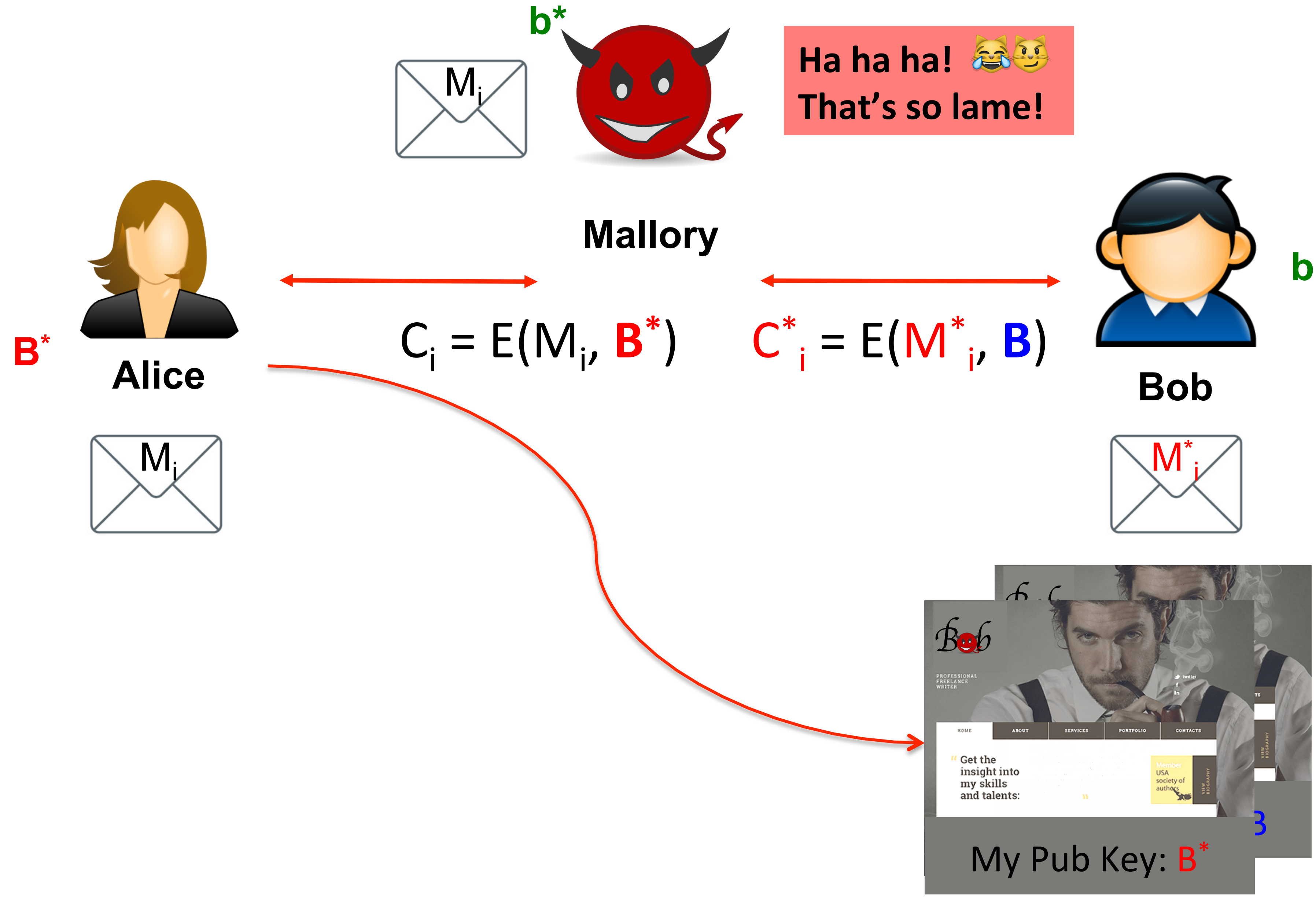
$K'_2 = (A')^b \bmod p = (g^{a'})^b = g^{a'b} \bmod p$

9. Alice and Bob now compute keys they share with … Mallory!
10. Mallory can relay encrypted traffic between the two …

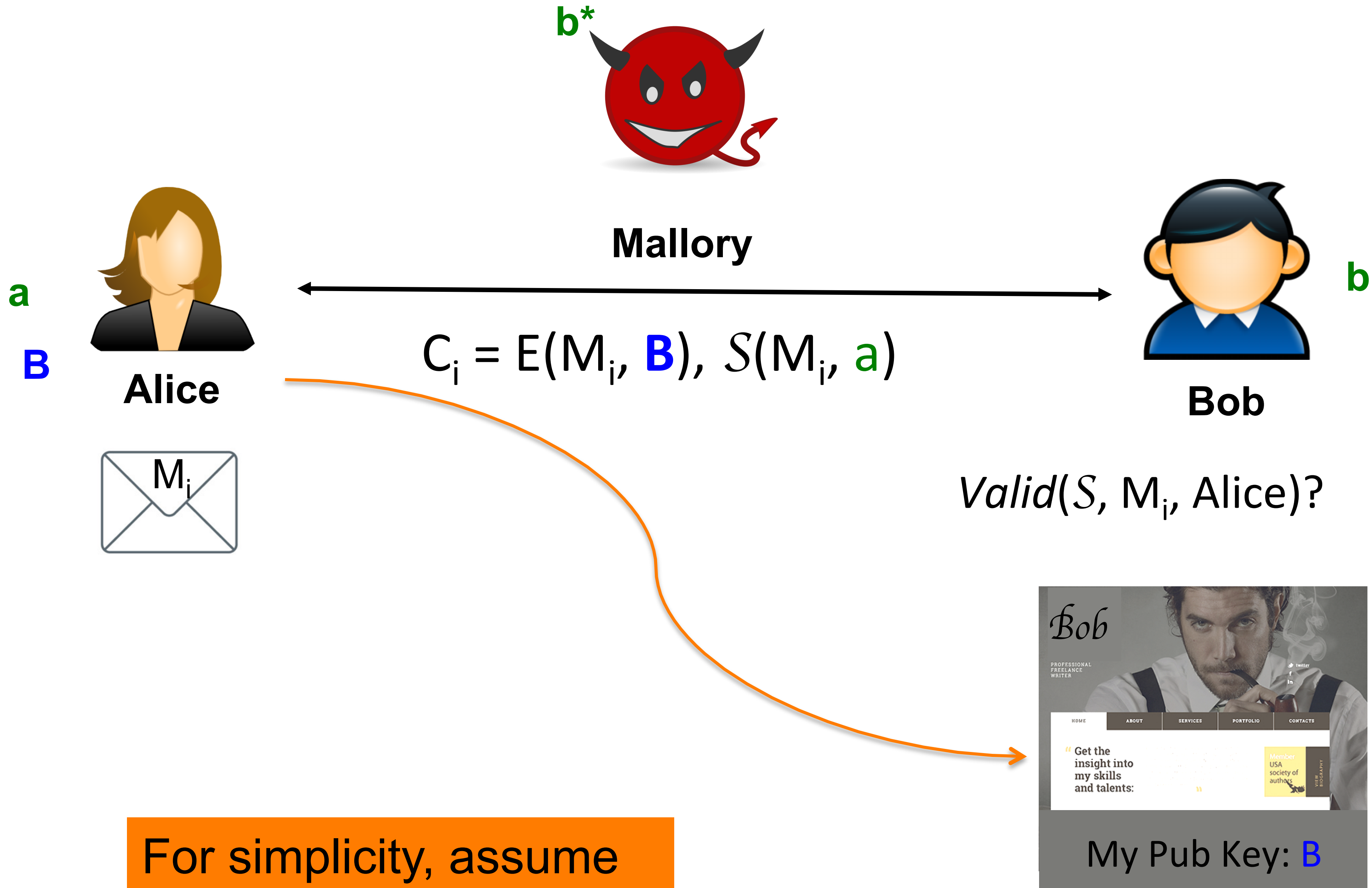
    10'. Modifying it or making stuff up *however she wishes*

Distributing Public Keys

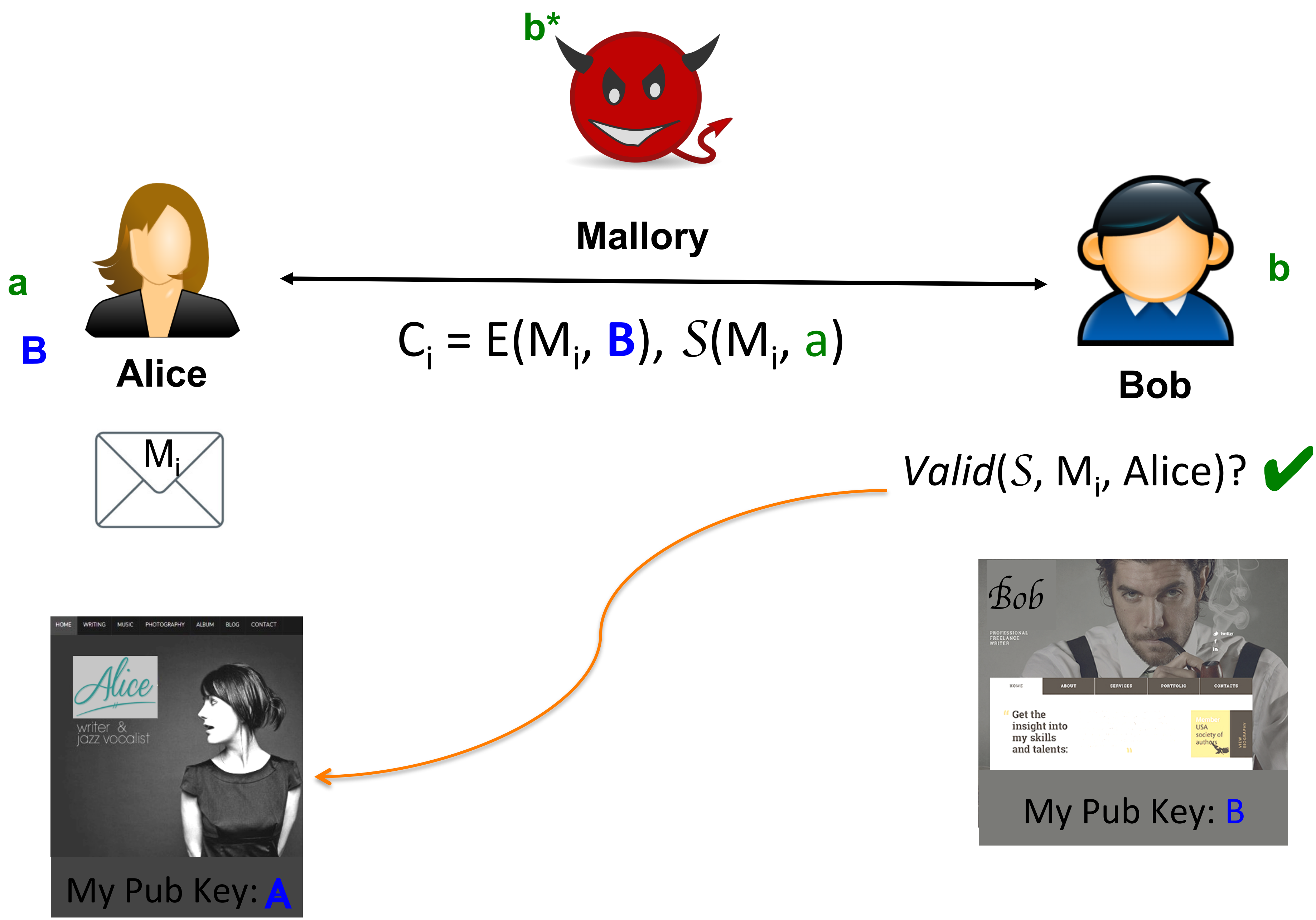

B
Alice
$C_i = E(M_i, B)$
b
Bob
M_i
M_i
Bob
HOME
ABOUT
SERVICES
PORTFOLIO
CONTACTS
Get the insight into my skills and talents:
My Pub Key: B



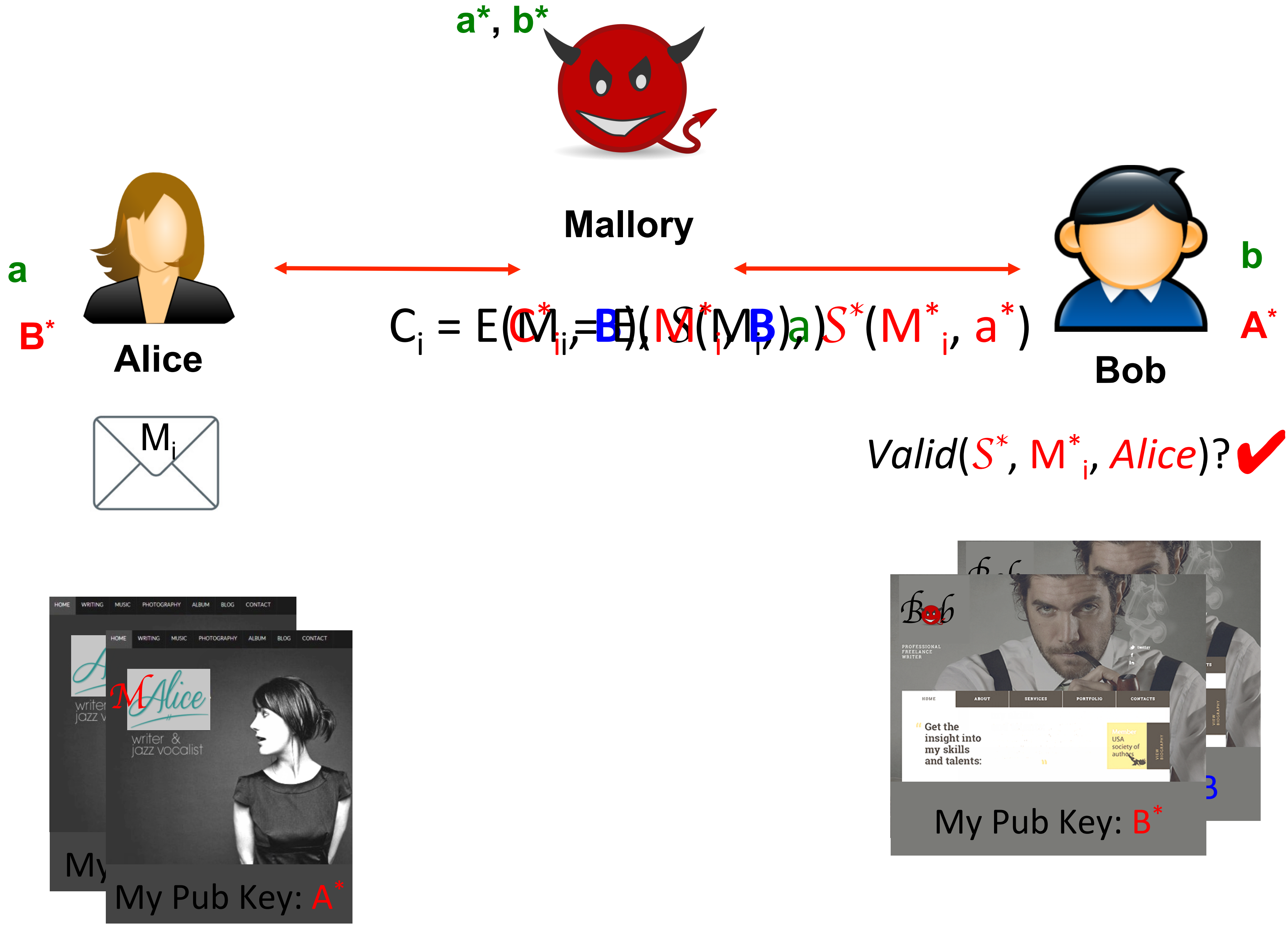
b*
M_i
Ha ha ha!
That's so lame!
Mallory
b
Alice
B^*
$C_i = E(M_i, B^*)$
$C^*_i = E(M^*_i, B)$
Bob
M_i
M^*_i
Get the insight into my skills and talents:
My Pub Key: B^*

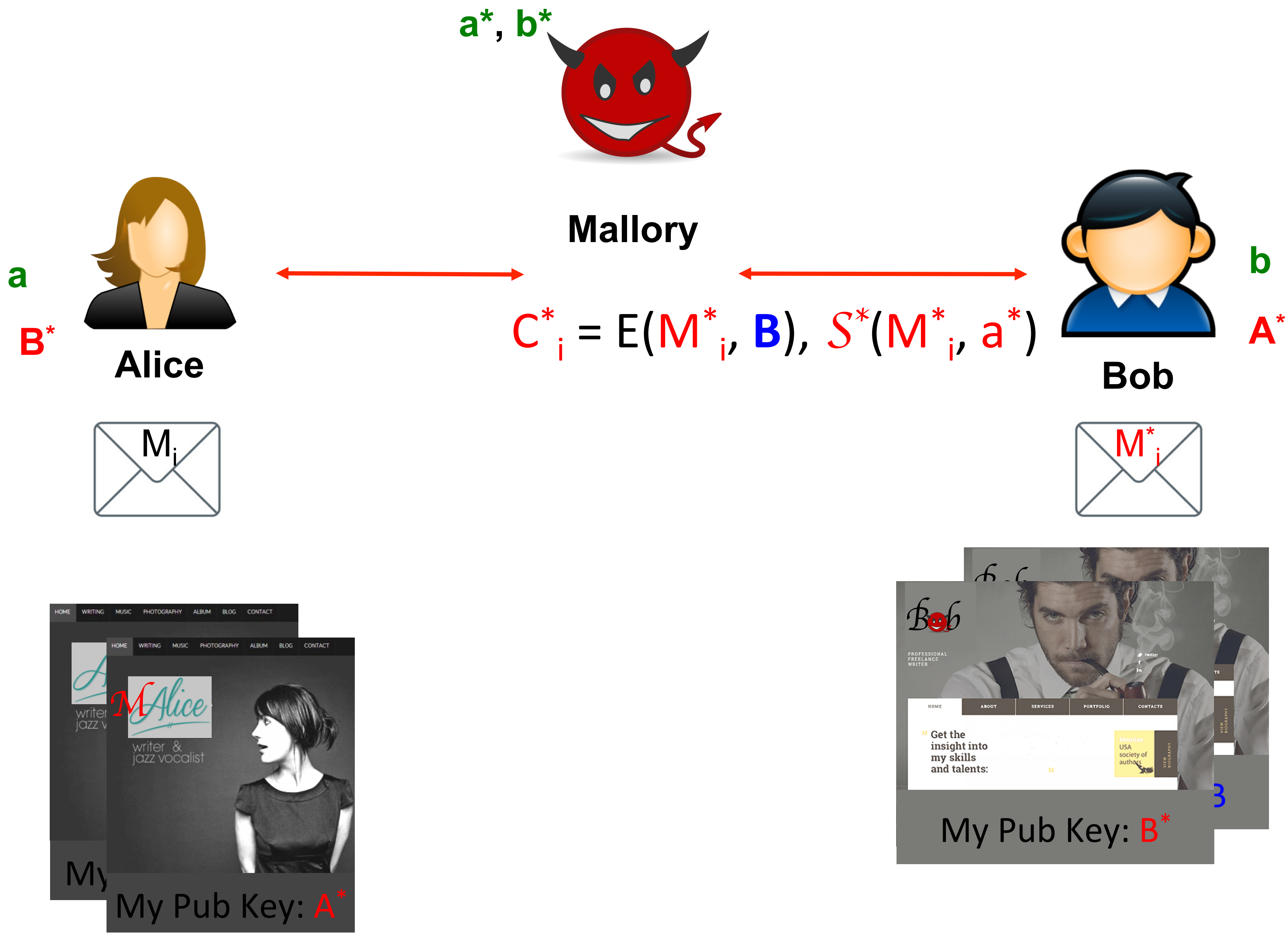


For simplicity, assume Alice uses same key for encryption & signing



b*
Mallory
a
B
Alice
b
Bob
C_i = E(M_i, B), S(M_i, a)
M_i
Valid(S, M_i, Alice)?
HOME WRITING MUSIC PHOTOGRAPHY ALBUM BLOG CONTACT
Alice
writer & jazz vocalist
My Pub Key: A
Bob
PROFESSIONAL FREELANCE WRITER
HOME ABOUT SERVICES PORTFOLIO CONTACTS
Get the insight into my skills and talents:
My Pub Key: B


a*, b*
Mallory
a
B*
Alice
b
A*
Bob
S*(M*_i, a*)
M_i
Valid(S*, M*_i, Alice)?
HOME WRITING MUSIC PHOTOGRAPHY ALBUM BLOG CONTACT
Alice
writer & jazz vocalist
My Pub Key: A*
Bob
PROFESSIONAL FREELANCE WRITER
HOME ABOUT SERVICES PORTFOLIO CONTACTS
Get the insight into my skills and talents:
My Pub Key: B*



a*, b*
Mallory
a
B*
Alice
b
A*
Bob
$C^*_i = E(M^*_i, B), S^*(M^*_i, a^*)$
M_i
M^*_i
HOME WRITING MUSIC PHOTOGRAPHY ALBUM BLOG CONTACT
Alice
writer & jazz vocalist
My Pub Key: A*
Bob
PROFESSIONAL FREELANCE WRITER
HOME ABOUT SERVICES PORTFOLIO CONTACTS
Get the insight into my skills and talents:
My Pub Key: B*

How Can We Communicate With Someone New?

- Public-key crypto gives us amazing capabilities to achieve confidentiality, integrity & authentication *without shared secrets* …
- But how do we solve MITM attacks?
- How can we trust we have the true public key for someone we want to communicate with?

- Ideas?

Trusted Authorities

- Suppose there's a party that everyone agrees to trust to confirm each individual's public key
  - Say the Governor of California

- Issues with this approach?
  - How can everyone agree to trust them?
  - Scaling: huge amount of work; single point of failure ...
    - ... and thus *Denial-of-Service* concerns
  - *How do you know you're talking to the right authority??*

Trust Anchors

- Suppose the trusted party distributes their key so *everyone has it* …


UNITED STATES OF AMERICA
IN GOD WE TRUST
LIBERTY
D
QUARTER DOLLAR


CALIFORNIA
1850
JOHN MUIR
YOSEMITE VALLEY
2005
E PLURIBUS UNUM


CALIFORNIA REPUBLIC


Jerry Brown's Public Key is 0x6a128b3d3dc67edc74d690b19e072f64
CALIFORNIA
1850
JOHN
MUIR
DE
YOSEMITE
VALLEY
2005
E PLURIBUS UNUM


CALIFORNIA REPUBLIC

Trust Anchors

- Suppose the trusted party distributes their key so *everyone has it* …
- We can then use this to *bootstrap trust*
  - As long as we have confidence in the decisions that that party makes

Digital Certificates

- Certificate (“cert”) = signed claim about someone’s key
  - More broadly: a signed *attestation* about some claim

- Notation:

  $\{ M \}_K$ = “message M encrypted with public key k”

  $\{ M \}_{K^{-1}}$ = “message M signed w/ private key for K”

- E.g. M = “Grant’s public key is K_{Grant} = `0xF32A99B...`”

  Cert: M,

  $\{$ “Grant’s public key ... `0xF32A99B...`” $\}_{K^{-1}_{Jerry}}$

  = `0x923AB95E12...9772F`

Certificate

Jerry Brown hearby asserts:
Grant's public key is K_{Grant} = 0xF32A99B...

The signature for this statement using K^{-1} *is* 0x923AB95E12...9772F

This

Certificate

Jerry Brown hearby asserts:

Grant's public key is K_{Grant} = 0xF32A99B...

The signature f is computed over all of this

K^{-1}_{Jerry} *is* 0x923AB95E12...9772F

Certificate

Jerry Brown hearby asserts:
Grant's public key is K_{Grant} = 0xF32A99B...

The signature for this statement using K^{-1}_{Jerry} *is* 0x923AB95E12...9772F

and can be *validated* using:

Certificate

This:

Jerry Brown hearby asserts:
Grant's public key is K

The signature for this st
K^{-1}_{Jerry} *is* 0x923AB95

If We Find This Cert Shoved Under Our Door …

- What can we figure out?
  - If we know Jerry's key, then whether he indeed signed the statement
  - If we trust Jerry's decisions, then we have confidence we really have Grant's key
- Trust = ?
  - Jerry won't willy-nilly sign such statements
  - Jerry won't let his private key be stolen

Analyzing Certs Shoved Under Doors …

- **How** we get the cert doesn't affect its utility
- **Who** gives us the cert doesn't matter
  - They're not any more or less trustworthy because they did
  - Possessing a cert doesn't establish any identity!
- **However:** if someone demonstrates they can decrypt data encrypted with K_{Grant}, then we have high confidence they possess K^{-1}_{Grant}
  - Same for if they show they can sign "using" K_{Grant}

Scaling Digital Certificates

- How can this possibly scale? Surely Jerry can't sign *everyone's* public key!
- Approach #1: Introduce hierarchy via delegation
  - { "Janet Napolitano's public key is 0x... and I trust her to vouch for UC" }$_{K^{-1}_{Jerry}}$
  - { "Nicholas Dirk's public key is 0x... and I trust him to vouch for UCB" }$_{K^{-1}_{Janet}}$
  - { "Jitendra Malik's public key is 0x... and I trust him to vouch for EECS" }$_{K^{-1}_{Nick}}$
  - { "Grant Ho's public key is 0x..." }$_{K^{-1}_{Jitendra}}$

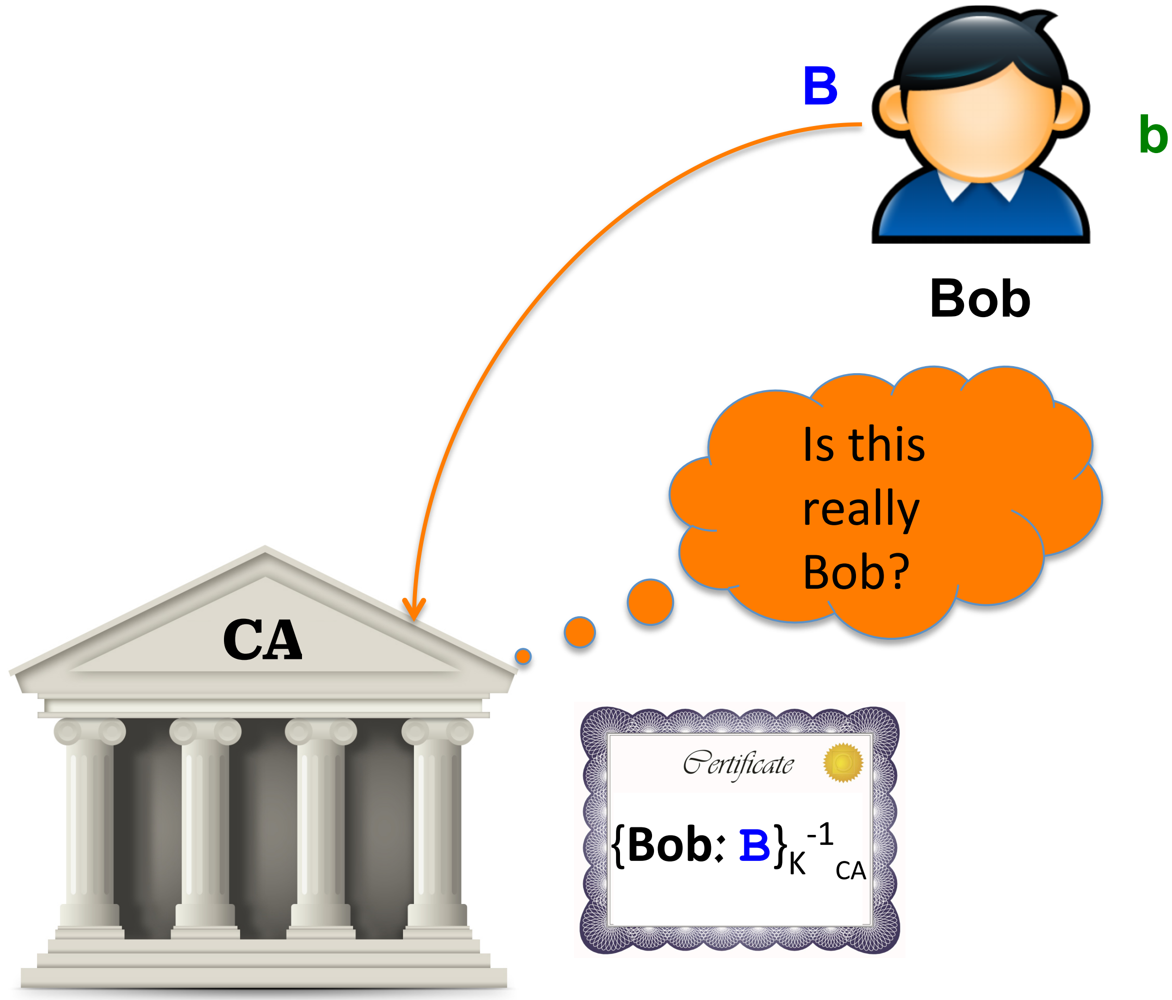
Scaling Digital Certificates, con't

- Grant puts this last on his web page
  - (or shoves it under your door)
- Anyone who can gather the intermediary keys can validate the chain
  - They can get these (other than Jerry's) from **anywhere** because they can validate them, too

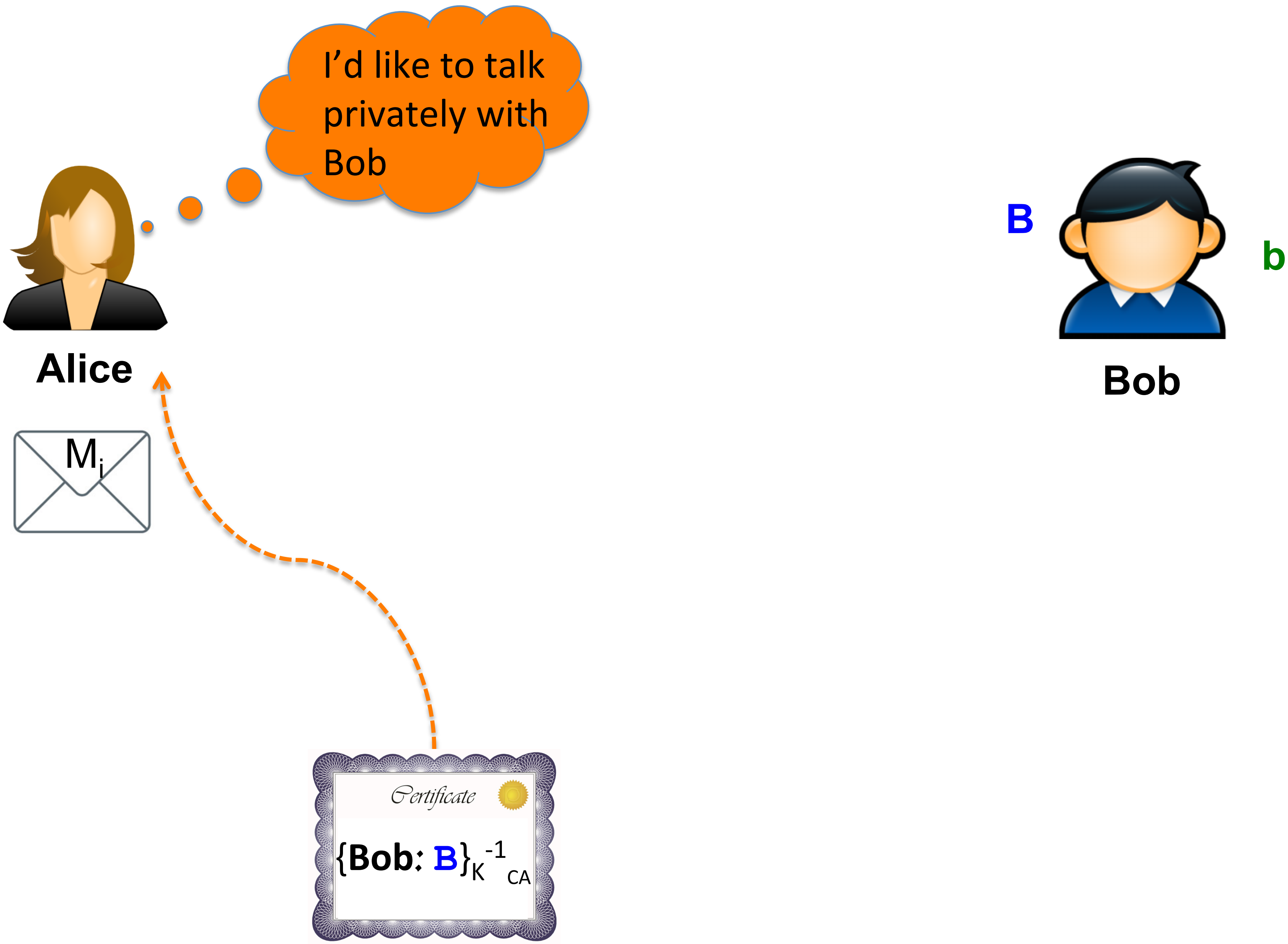
- Approach #2: have multiple trusted parties who are in the *business* of signing certs …
  - (The certs might also be hierarchical, per Approach #1)

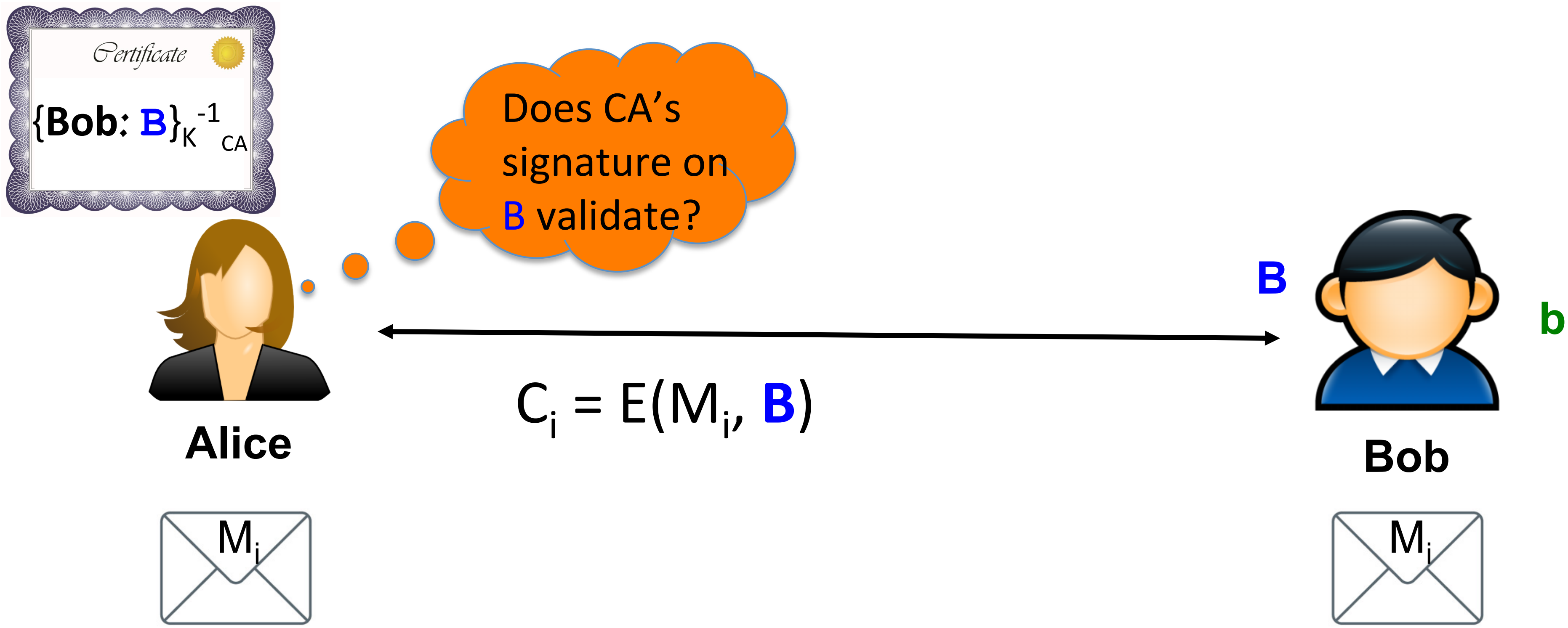
Certificate Authorities

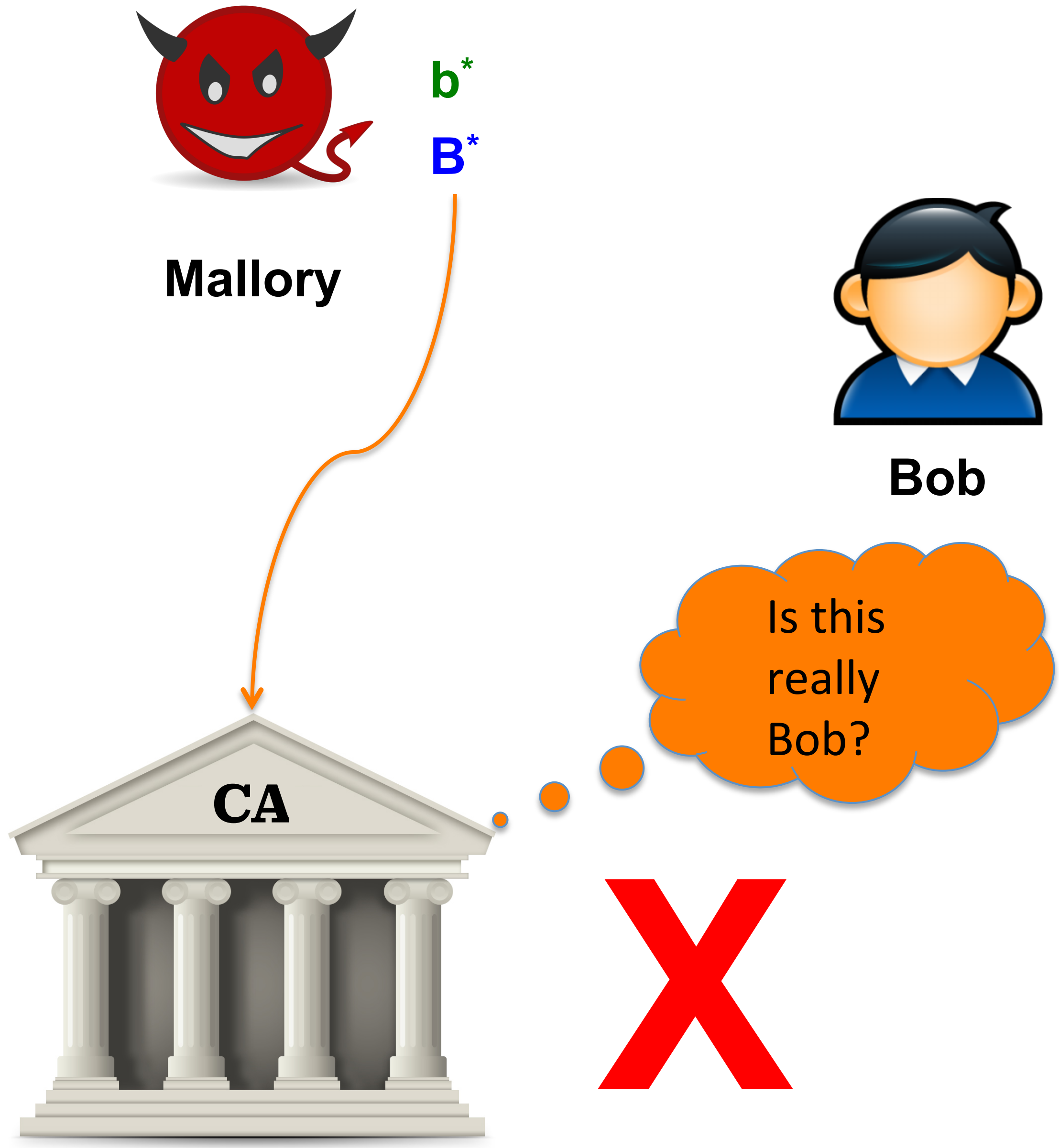
- CAs are trusted parties in a *Public Key Infrastructure* (PKI)
- They can operate offline
  - They sign ("cut") certs when convenient, not on-the-fly (… though see below …)
- Suppose Alice wants to communicate confidentially w/ Bob:
  - Bob gets a CA to issue {Bob's public key is B}$_{K^{-1}_{CA}}$
  - Alice gets Bob's cert any old way
  - Alice uses her known value of K_{CA} to verify cert's signature
  - Alice extracts B, sends $\{M\}_B$ to Bob



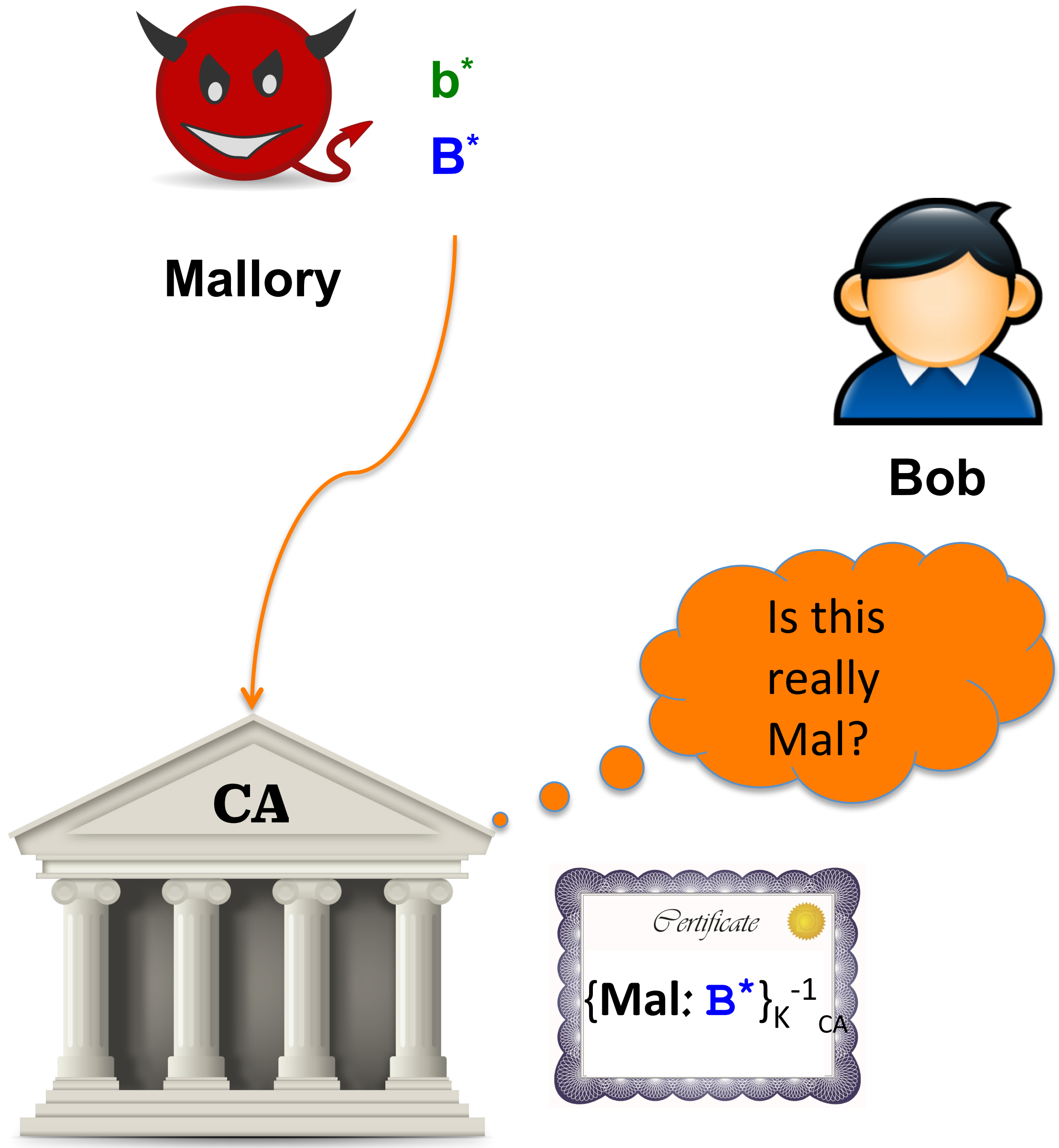
B
b
Bob
Is this really Bob?
CA
Certificate
{Bob: B}$_{K^{-1}_{CA}}$


I'd like to talk privately with Bob
Alice
M_i
B
b
Bob
Certificate
{Bob: B}$_{K_{CA}^{-1}}$

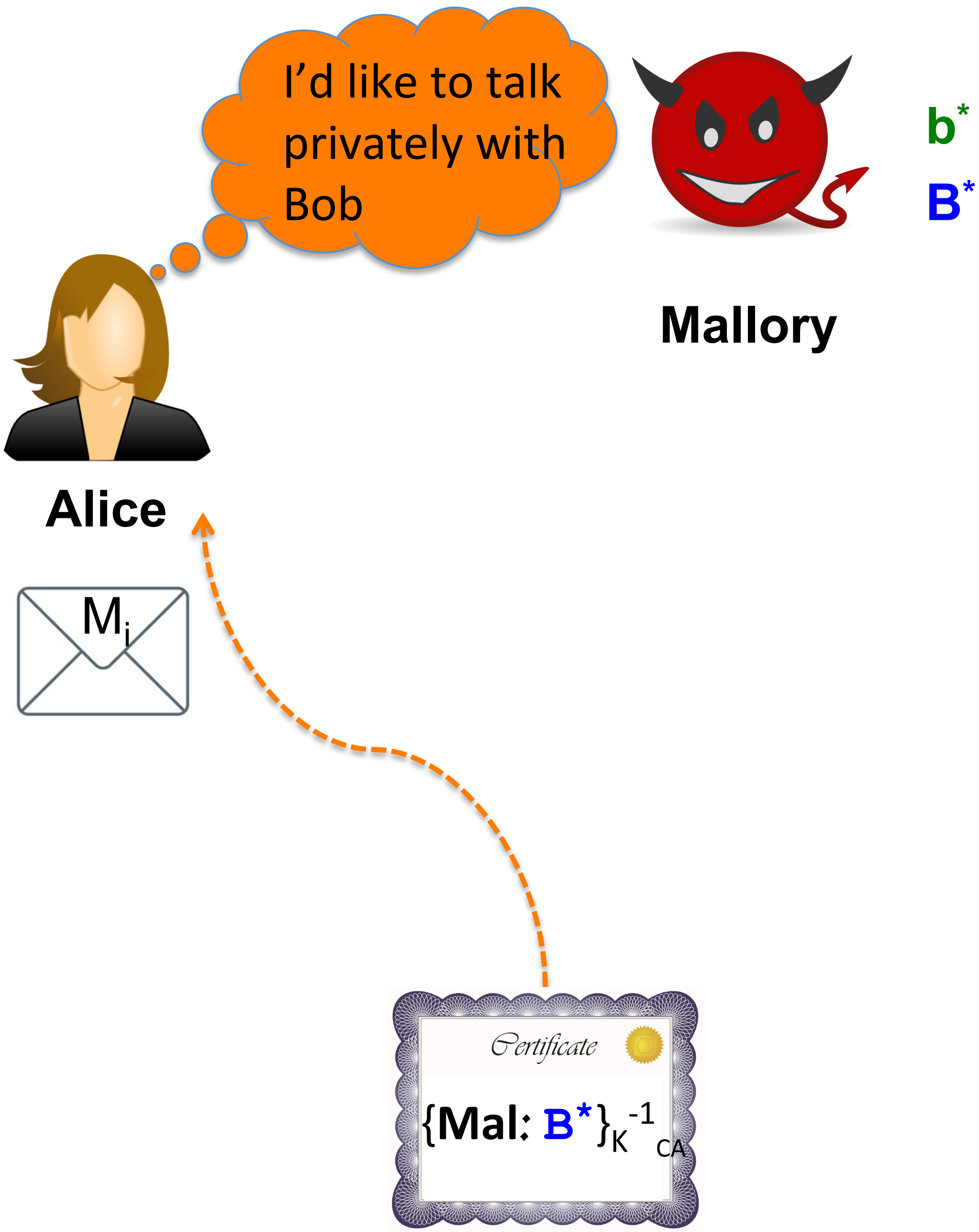

Certificate
{Bob: B}K-1 CA
Does CA's signature on B validate?
B
b
Alice
Bob
Ci = E(Mi, B)
Mi
Mi



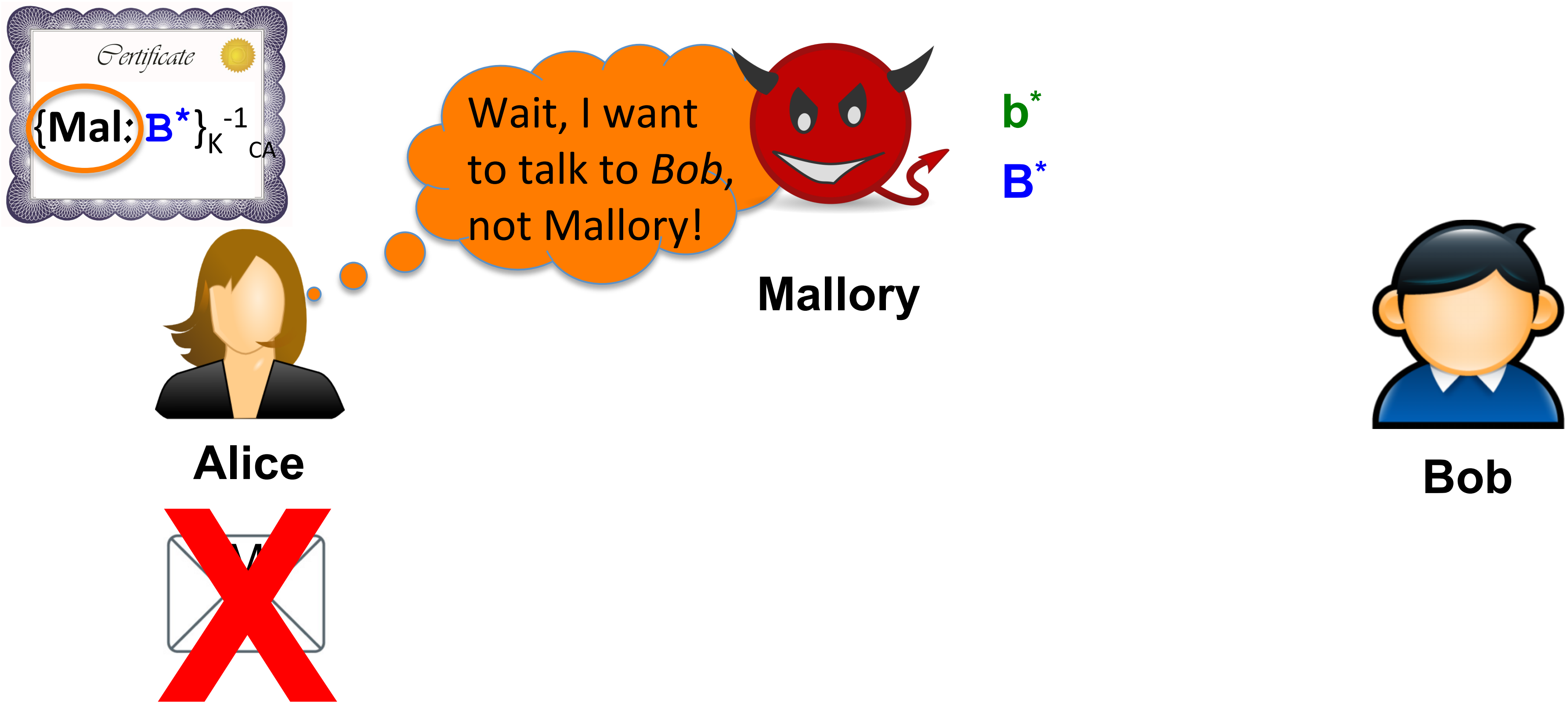
b*
B*
Mallory
Bob
Is this really Bob?
CA
X



b*
B*
Mallory
Bob
Is this really Mal?
CA
Certificate
{Mal: B*}K-1CA

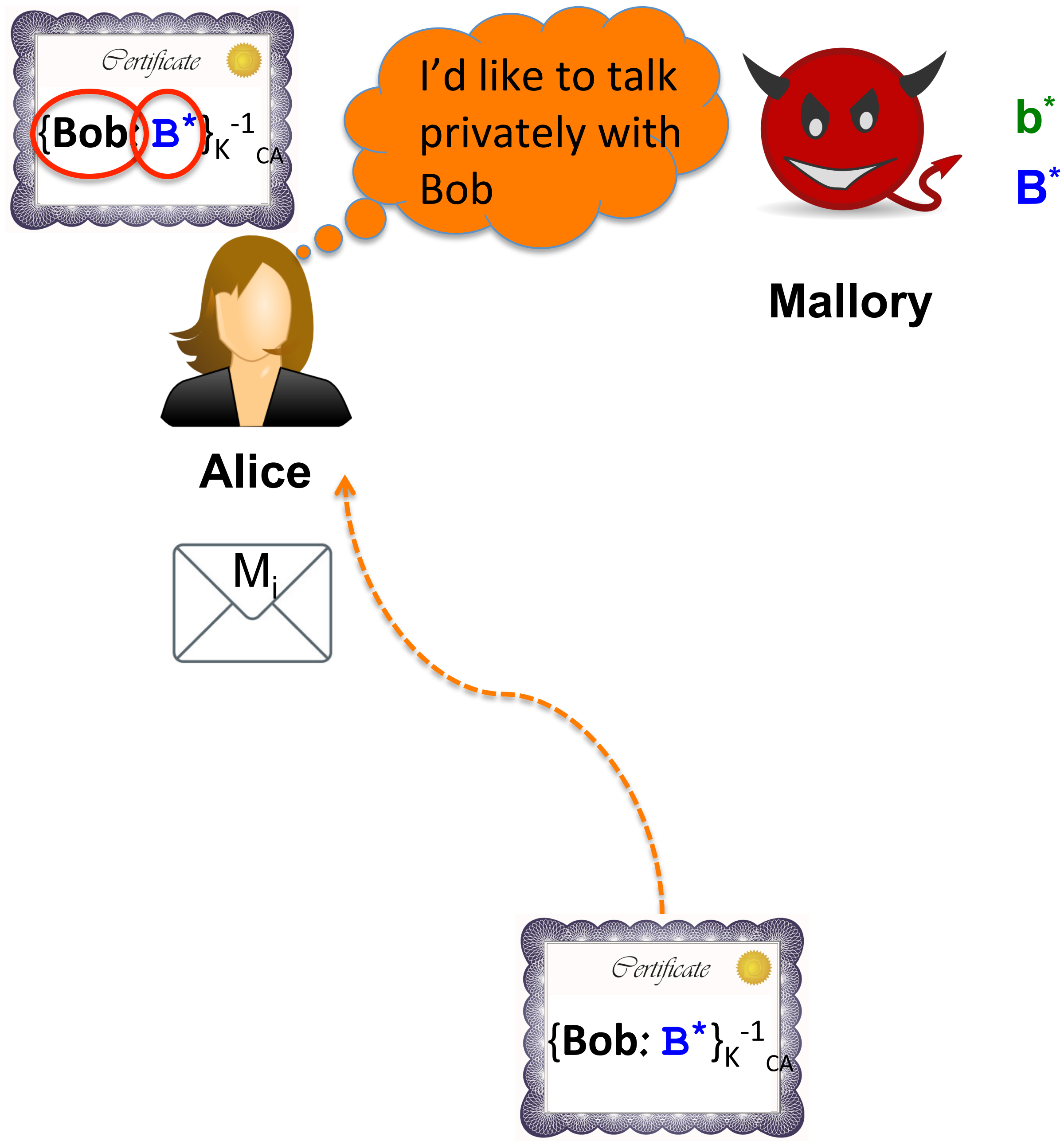


I'd like to talk privately with Bob
Mallory
b*
B*
Alice
Bob
M_i
Certificate
$\{Mal: B^*\}_{K^{-1}_{CA}}$


Certificate
{Mal: B*}K-1CA
Wait, I want to talk to *Bob*, not Mallory!
b*
B*
Mallory
Alice
Bob

Revocation

- What do we do if a CA screws up and issues a cert in Bob's name to Mallory?



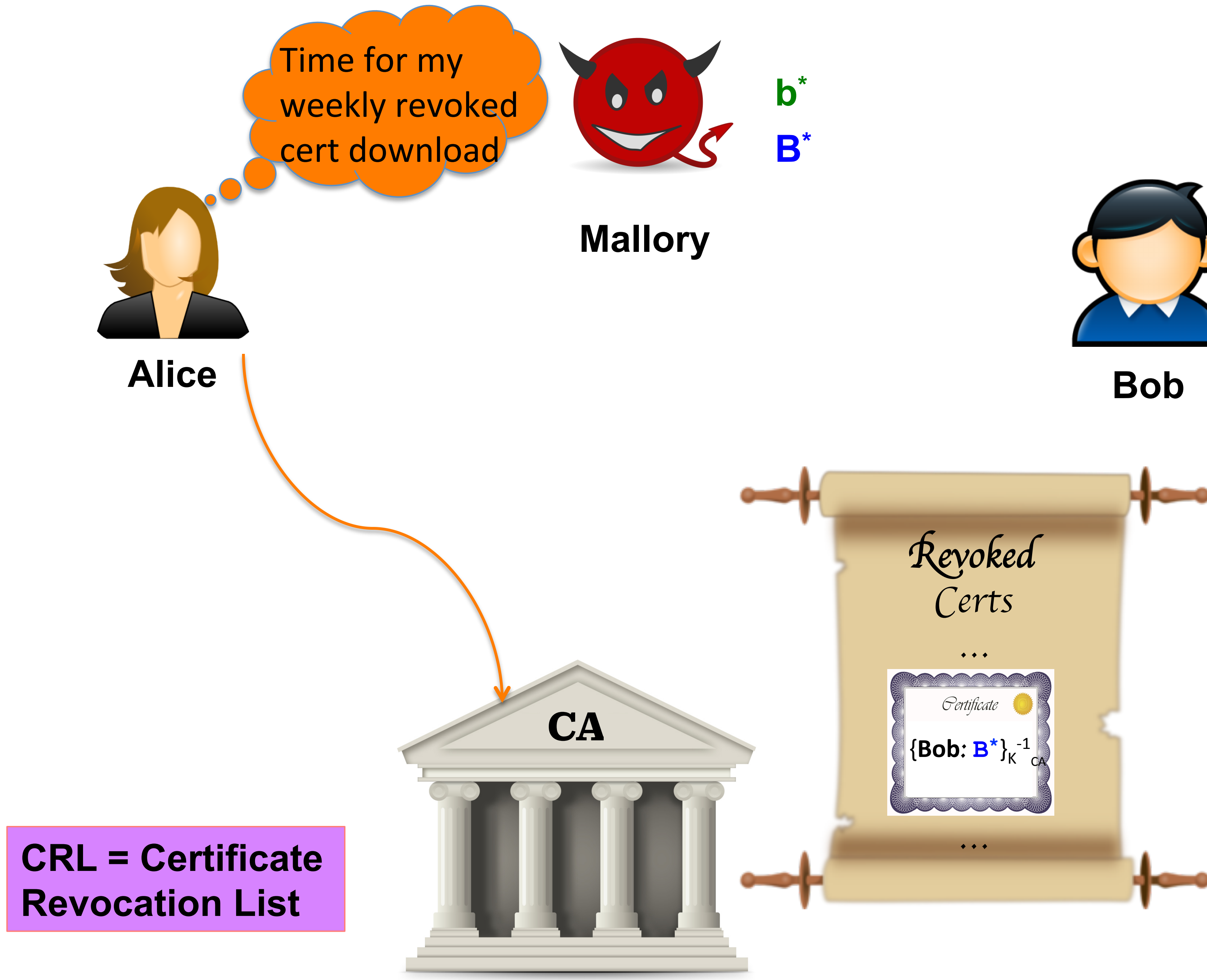
{Bob: B*}K-1 CA
I'd like to talk privately with Bob
Mallory
b*
B*
Alice
Bob
Mi
Certificate
{Bob: B*}K-1 CA

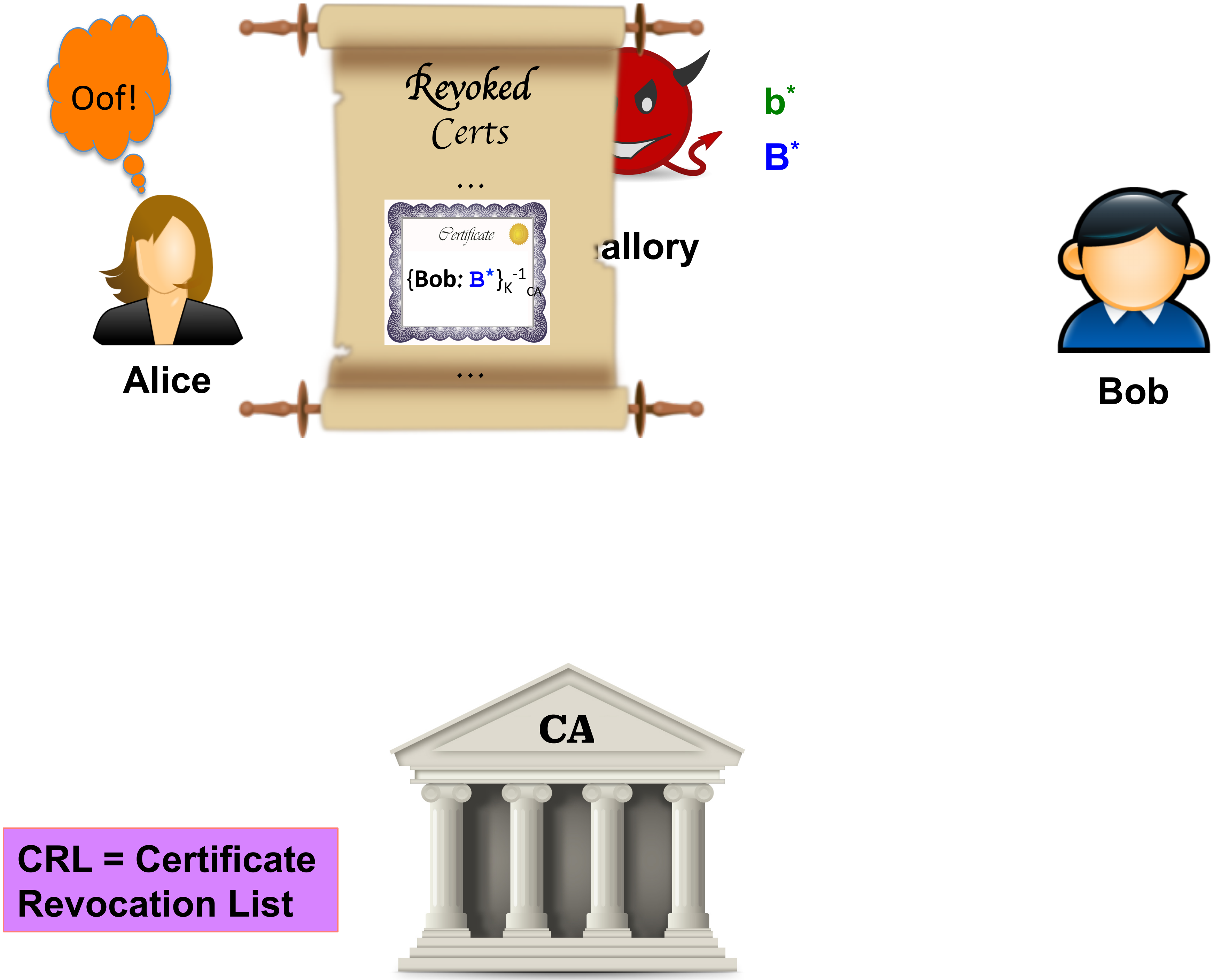
Revocation

- What do we do if a CA screws up and issues a cert in Bob's name to Mallory?
  - E.g. Verisign issued a `Microsoft.com` cert to a *Random Joe*
  - (Related problem: Bob realizes b has been stolen)
- *How do we recover from the error?*
- Approach #1: expiration dates
  - Mitigates possible damage
  - But adds management burden
    - Benign failures to renew will break normal operation

Revocation, con't

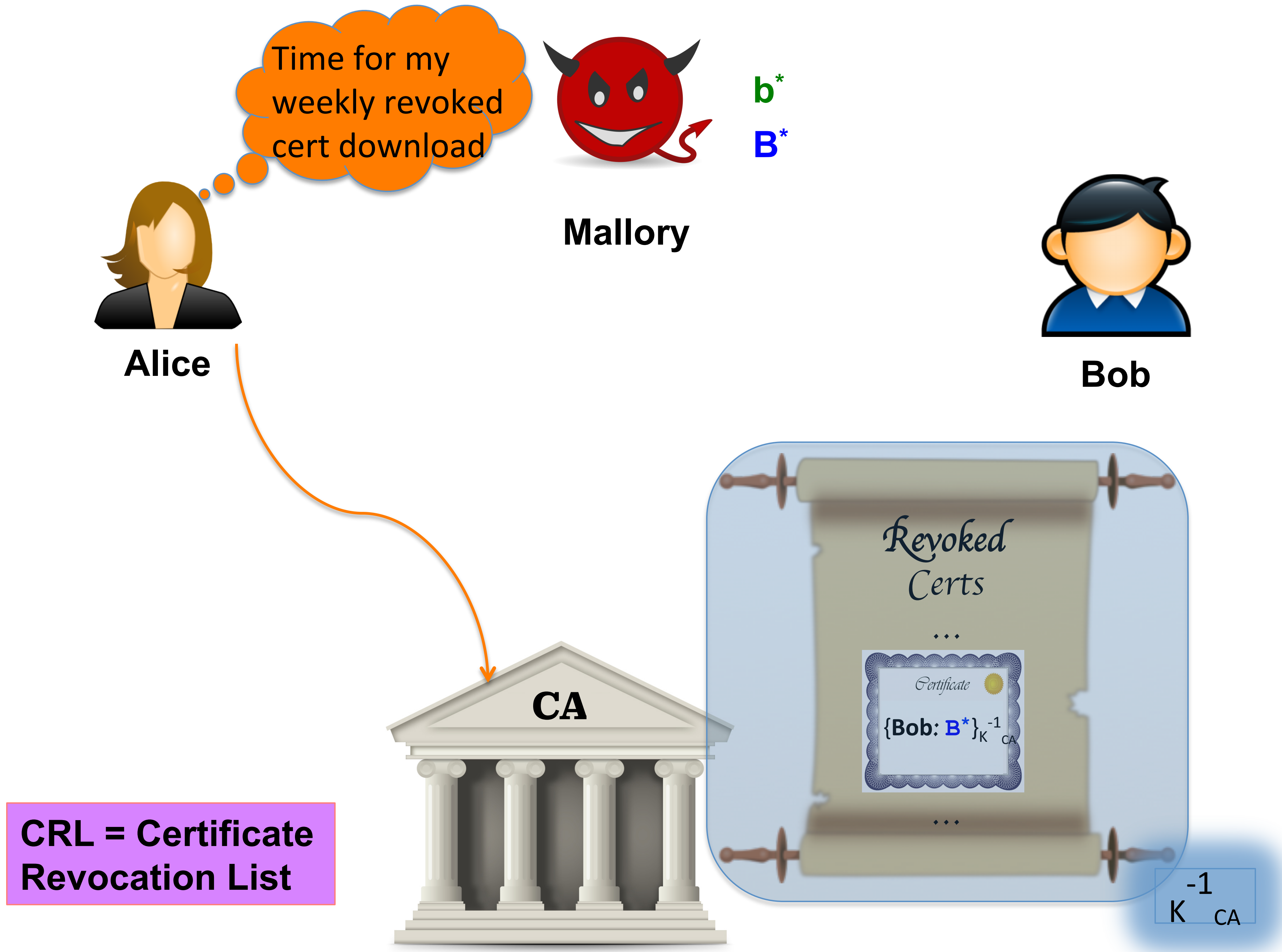
- Approach #2: announce revoked certs
  - Users periodically download *cert revocation list* (CRL)


Time for my weekly revoked cert download
Mallory
b*
B*
Alice
Bob
CA
Revoked Certs
...
Certificate
{Bob: B*}K-1CA
...
CRL = Certificate Revocation List


Oof!
Alice
Revoked Certs
...
Certificate
{Bob: B*}K-1CA
...
allory
b*
B*
Bob
CA
CRL = Certificate Revocation List

Revocation, con't

- Approach #2: announce revoked certs
  - Users periodically download *cert revocation list* (CRL)
- Issues?
  - Lists can get large
  - Need to *authenticate the list* itself – how?



Time for my weekly revoked cert download
Mallory
b*
B*
Alice
Bob
CA
Revoked Certs
...
Certificate
{Bob: B*}K-1CA
...
K-1 CA
CRL = Certificate Revocation List

Revocation, con't

- Approach #2: announce revoked certs
  - Users periodically download *cert revocation list* (CRL)
- Issues?
  - Lists can get large
  - Need to authenticate the list itself – how? Sign it!
  - Mallory can exploit download lag
  - What does Alice do if can't reach CA for download?
    1. Assume all certs are invalid (*fail-safe defaults*)
       - Wow, what an unhappy failure mode!
    2. Use old list: widens exploitation window if Mallory can "DoS" CA (DoS = denial-of-service)