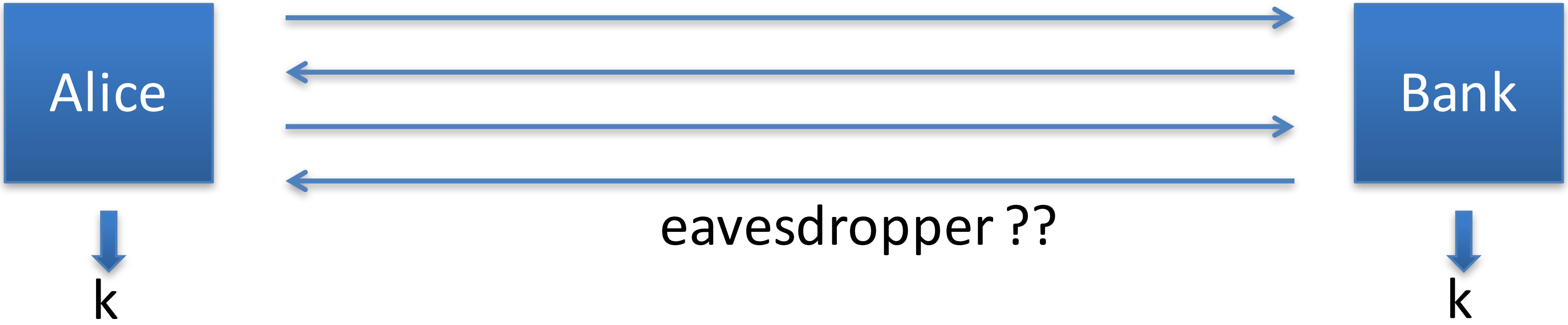
Auth. Key Exchange

Review: key exchange

Alice and Bank want to generate a secret key

- Saw key exchange secure against eavesdropping



- This lecture: **Authenticated Key Exchange (AKE)**

  key exchange secure against <u>active</u> adversaries

Active adversary

Adversary has complete control of the network:

- Can modify, inject and delete packets
- Example: man-in-the-middle

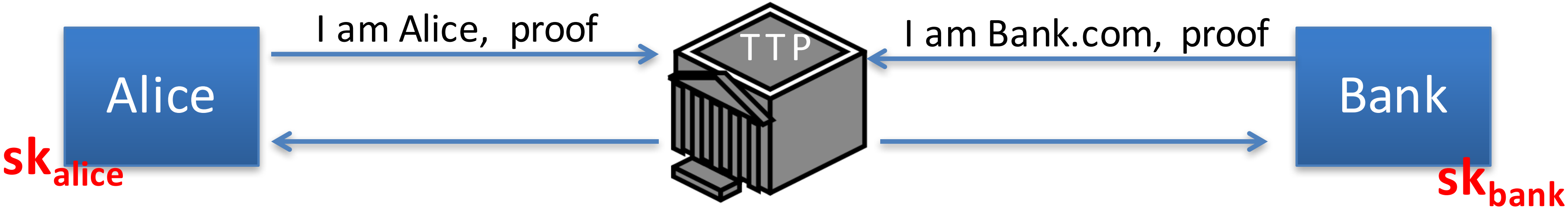
Moreover, some users are honest and others are corrupt

- Corrupt users are controlled by the adversary
  - Key exchange with corrupt users should not "affect" other sessions
- Adversary may corrupt an honest user at time T
  - We want sessions established at time $t < T$ to remain "secure"

Trusted Third Party (TTP)

All AKE protocols require a TTP to certify user identities.

Registration process:



Two types of TTP:

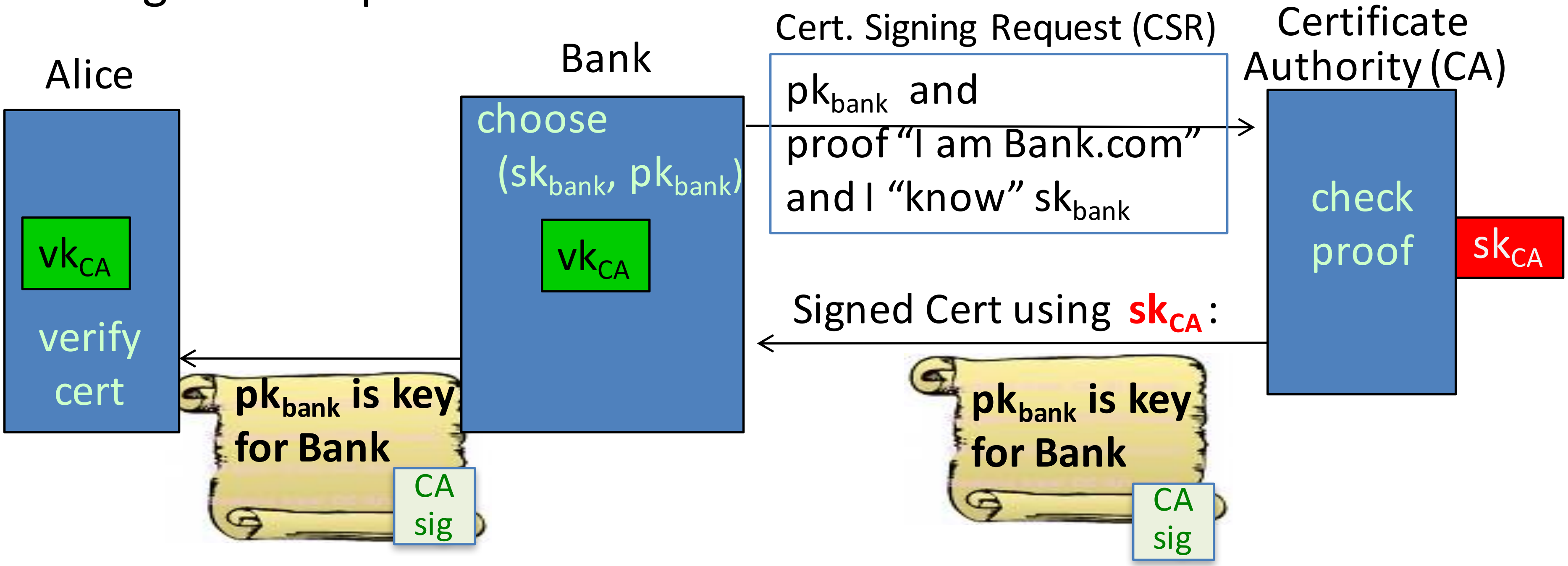
- **Online TTP**: actively participates in <u>every</u> key exchange (Kerberos)

  Benefit: security using only symmetric crypto

- **Offline TTP (CA)**: contacted only during registration (… not quite true)

Offline TTP: Certificate Authority (CA)

Registration process:



Assumptions: (for now)

- all parties have a certified pk (even corrupt users)
- only Bank can get a CA certificate for Bank

Certificates: example

Important fields:

Serial Number	5814744488373890497
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, July 31, 2013 4:59:24 AM Pacific Daylight Time
Not Valid After	Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 71 6C DD E0 0A C9 76 ...
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 8A 38 FE D6 F5 E7 F6 59 ...

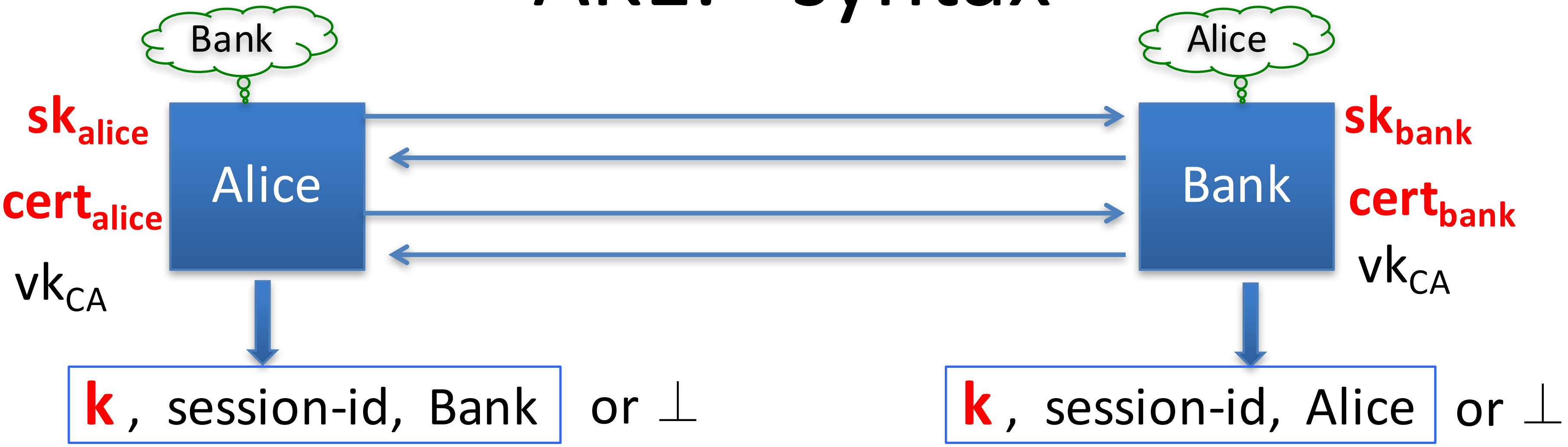
Equifax Secure Certificate Authority
↳ GeoTrust Global CA
↳ Google Internet Authority G2
↳ mail.google.com

mail.google.com
Issued by: Google Internet Authority G2
Expires: Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
This certificate is valid

Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2

AKE: syntax



Followed by Alice sending E(k, “data”) to Bank

Session-id (sid): public id of session

- Identifies instance of Alice that is talking to instance of Bank

$$\text{sid} = \text{sid}' \quad \Leftrightarrow \quad k = k'$$

AKE security (very informal)

Suppose Alice successfully completes an AKE to obtain **(k, sid, Bank)**

If Bank is not corrupt then:

<u>Authenticity</u> for Alice: (similarly for Bank)

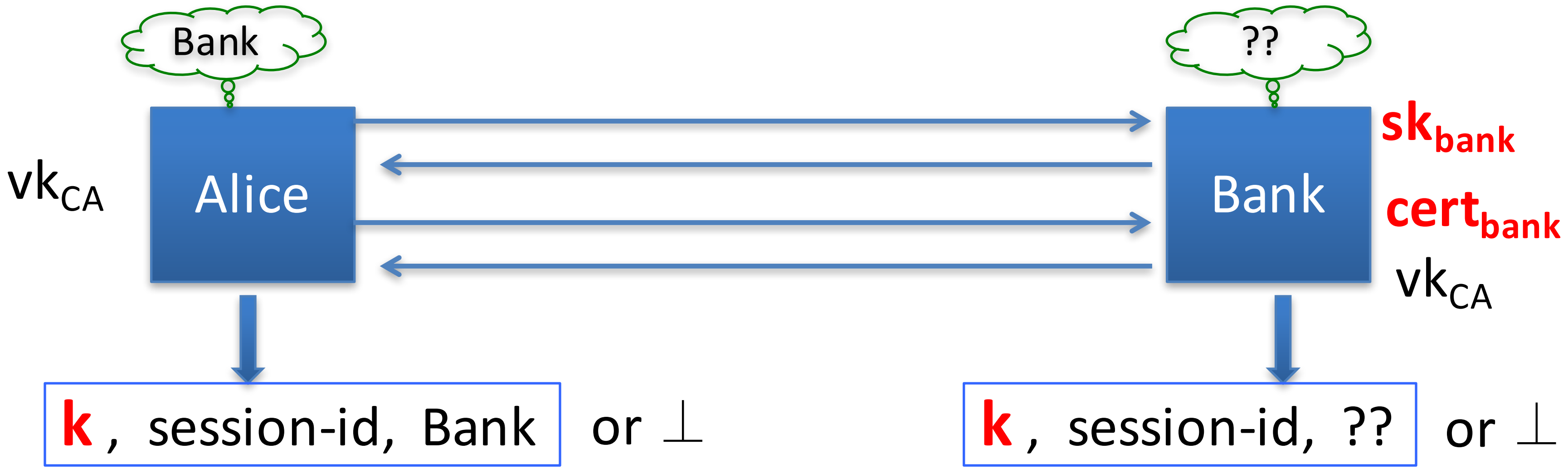
- If Alice's key k is shared with anyone, it is only shared with Bank

<u>Secrecy</u> for Alice: (similarly for Bank)

- To the adversary, Alice's key k is indistinguishable from random (even if adversary sees keys from other instances of Alice or Bank)

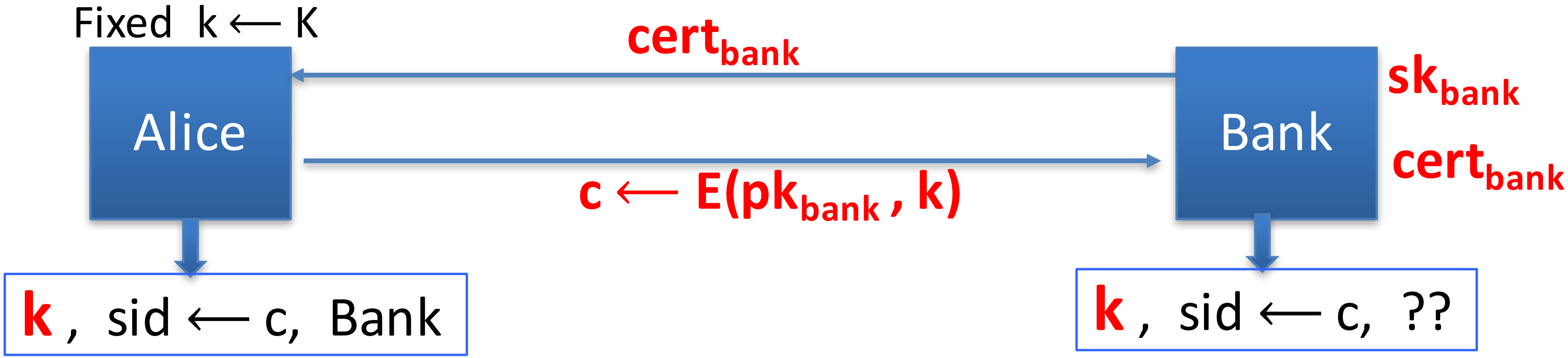
<u>Consistency</u>: if Bank completes AKE then it obtains **(k, sid, Alice)**

One-sided AKE



Security: authenticity for Alice and secrecy for Alice

- Bank has no guarantees for identity of peer (no consistency)
- Commonly used on the Web (often followed by ID protocol)



Alice uses same k for **<u>all</u>** key exchanges. Is this a secure one-sided AKE?

No, a single AKE with a corrupt user will reveal k

Lots of other problems with this protocol …

Things to remember …

Do not design AKE protocol yourself …

Just use latest version of TLS

(sid for TLS is called channel binding)

Building blocks

$cert_{bank}$: contains pk_{bank} . Bank has sk_{bank} .

$E_{bank}((m,r)) = E(pk_{bank}, (m,r))$ where E is *chosen-ciphertext secure*

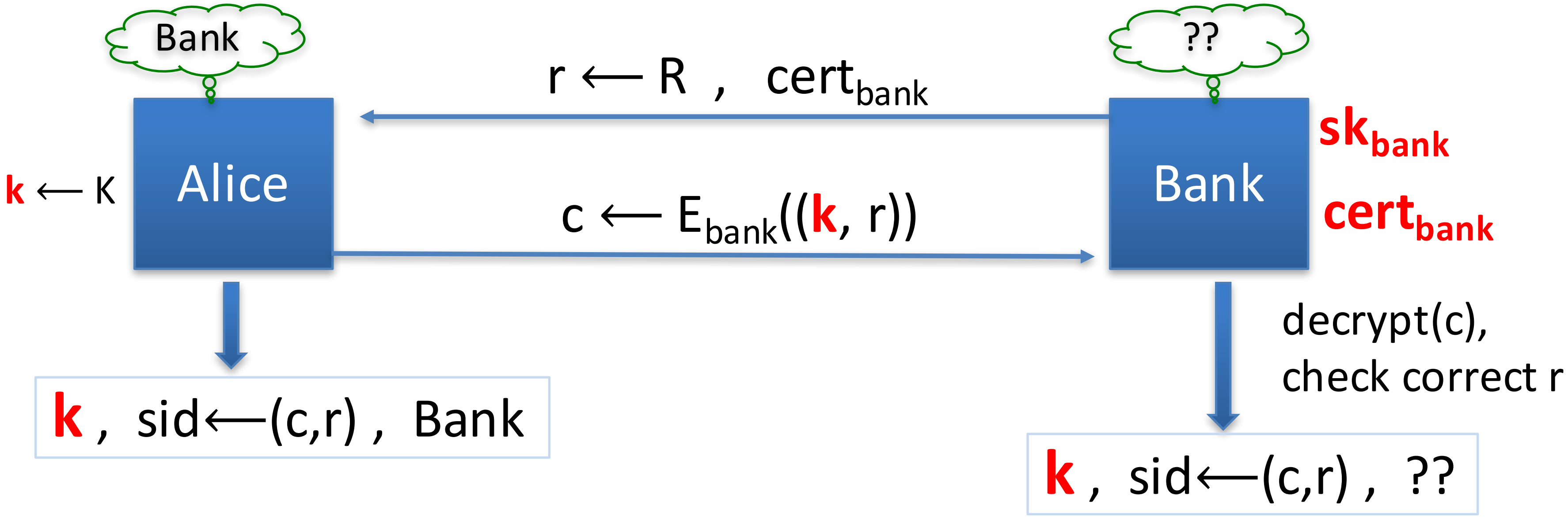
- Recall: from $E_{bank}((m,r))$ adv. cannot build $E_{bank}((m,r'))$ for $r' \neq r$

$S_{alice}((m,r)) = S(sk_{alice}, (m,r))$ where S is a signing algorithm

R: some large set, e.g. $\{0,1\}^{256}$

Protocol #1

Simple one-sided AKE protocol

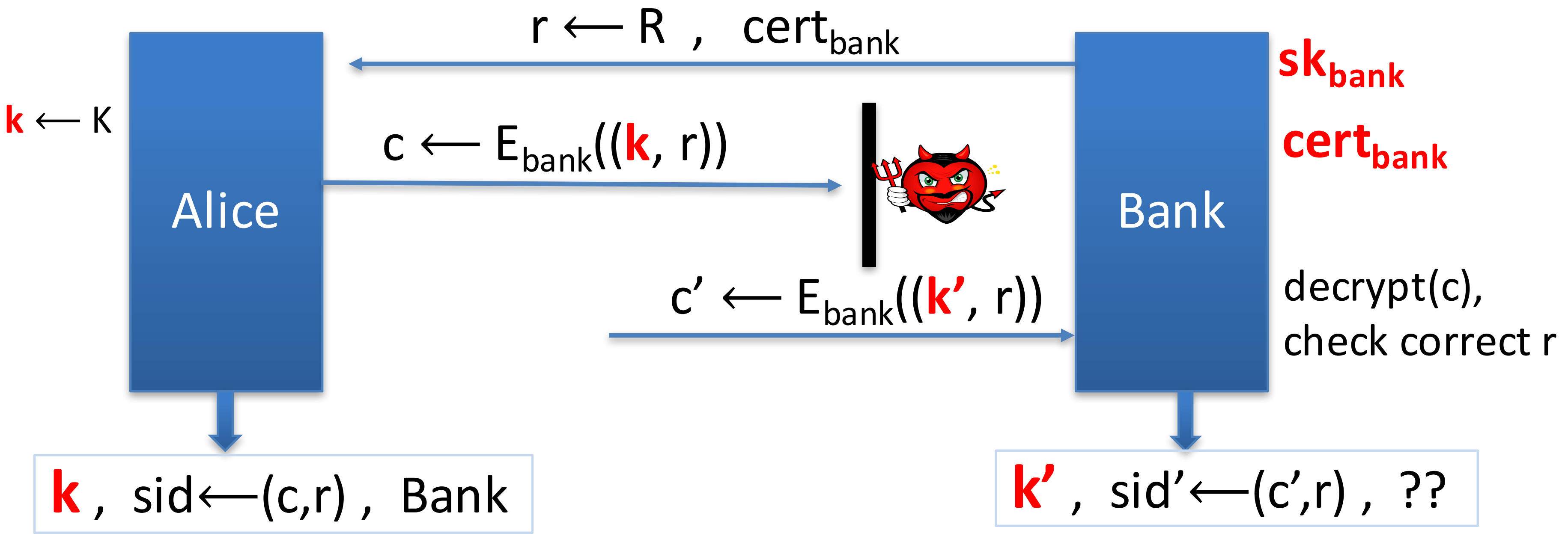


"Thm": this protocol is a secure one-sided AKE

Informally: if Alice and Bank are not corrupt then we have (1) secrecy for Alice and (2) authenticity for Alice

Did the AKE complete successfully?

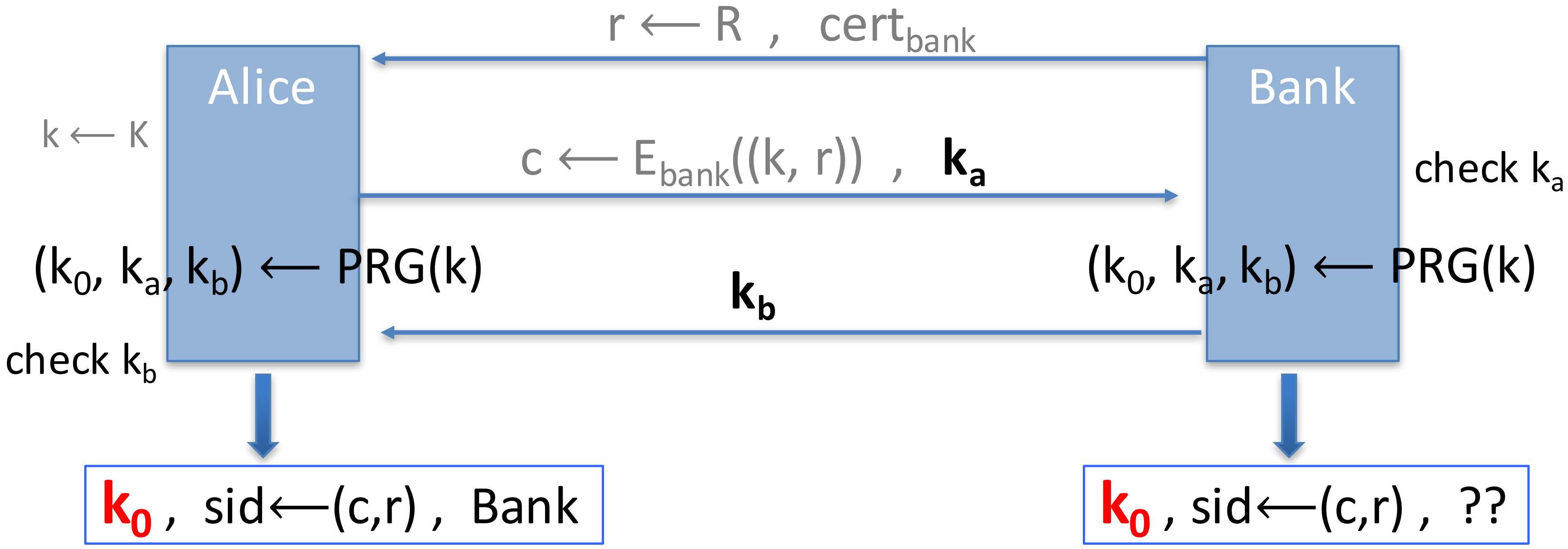
Problem: Alice does not know if Bank received k



One-sided: Bank does not know who k' is shared with

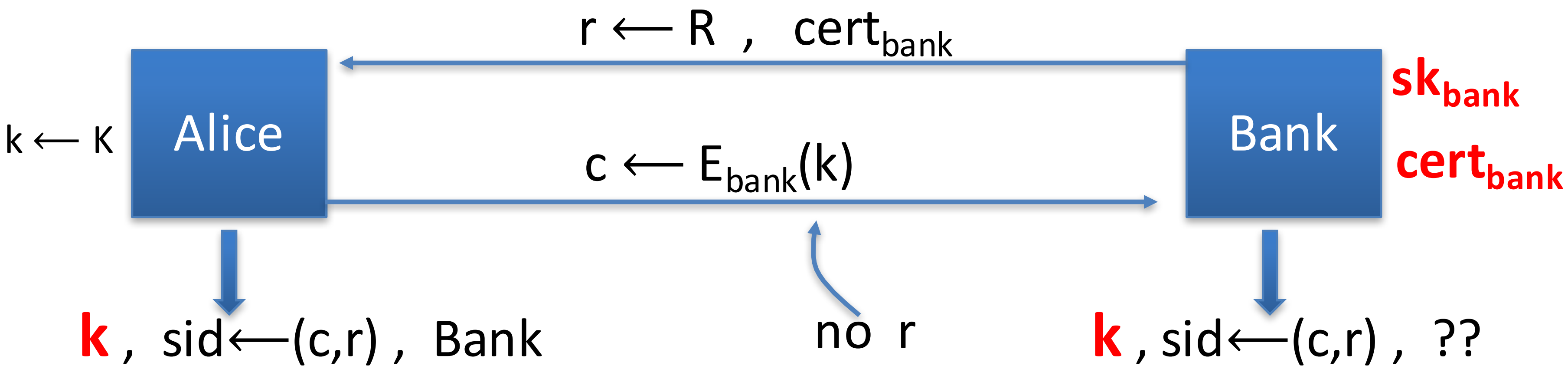
Explicit AKE via key confirmation

Key confirmation: if succeeds, Alice is assured Bank has k.



Note: k_0 used as session key (not k)

Insecure variant 1: r not encrypted



Problem: replay attack

Replay attack

$r \leftarrow R$, $cert_{bank}$

$k \leftarrow K$

Alice

$c \leftarrow E_{bank}(k)$

$c_1 \leftarrow E_{sym}$(k, "I am Alice, pay Bob 30$")

Bank

sk_{bank}

$cert_{bank}$

Later:

$r' \leftarrow R$, $cert_{bank}$

c

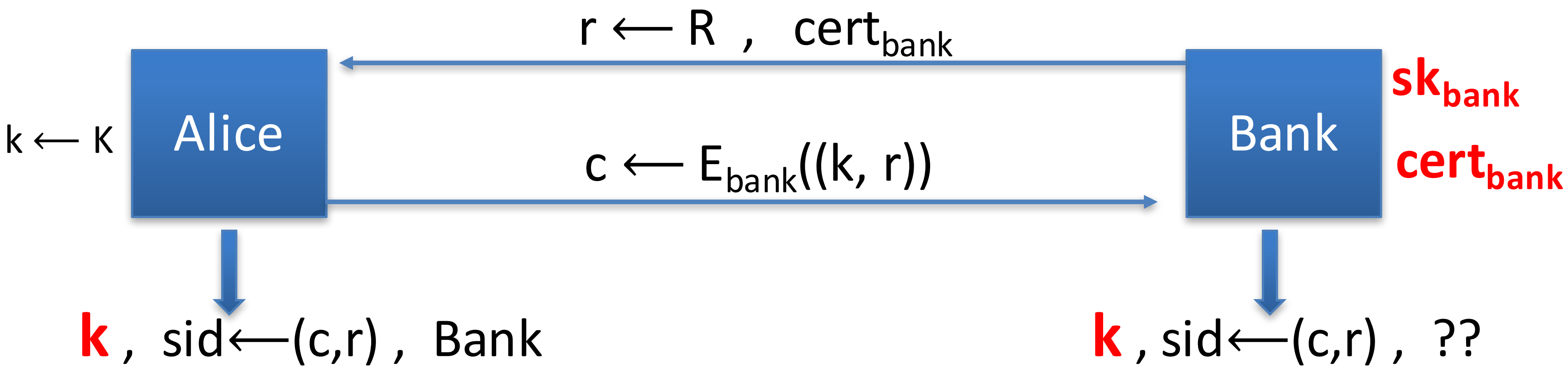
c_1

Bank

sk_{bank}

$cert_{bank}$

Insecure variant 2: E_{bank} not CCA-secure



Suppose: from $c = E_{bank}((k, r))$ can construct $c' = E_{bank}((k, r'))$ for any r'

$\Rightarrow$ replay attack

Replay attack

$r \longleftarrow R$, $cert_{bank}$

$k \longleftarrow K$ Alice

Bank **sk_{bank}** **$cert_{bank}$**

$c \longleftarrow E_{bank}((k, r))$

$c_1 \longleftarrow E_{sym}(k,$ "I am Alice, pay Bob 30$")

Later:

$r' \longleftarrow R$, $cert_{bank}$

Bank **sk_{bank}** **$cert_{bank}$**

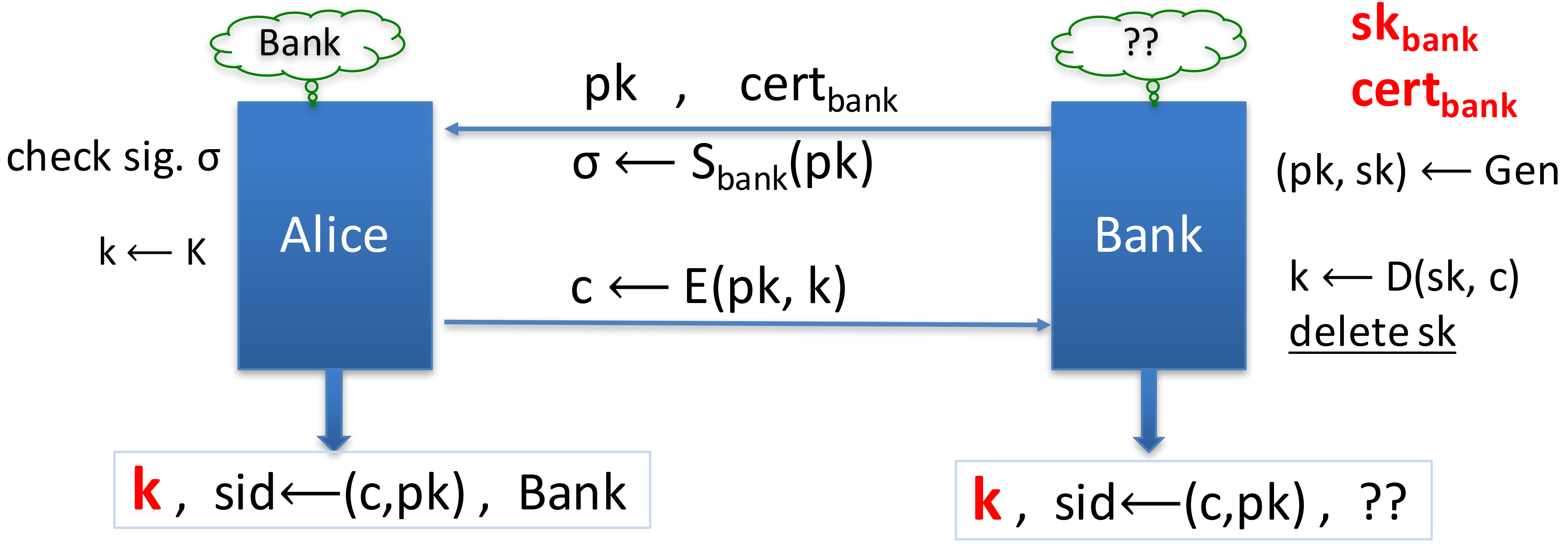
$c = E_{bank}((k, r)) \longrightarrow c' = E_{bank}((k, r'))$

c_1

accept c'

Protocol #2

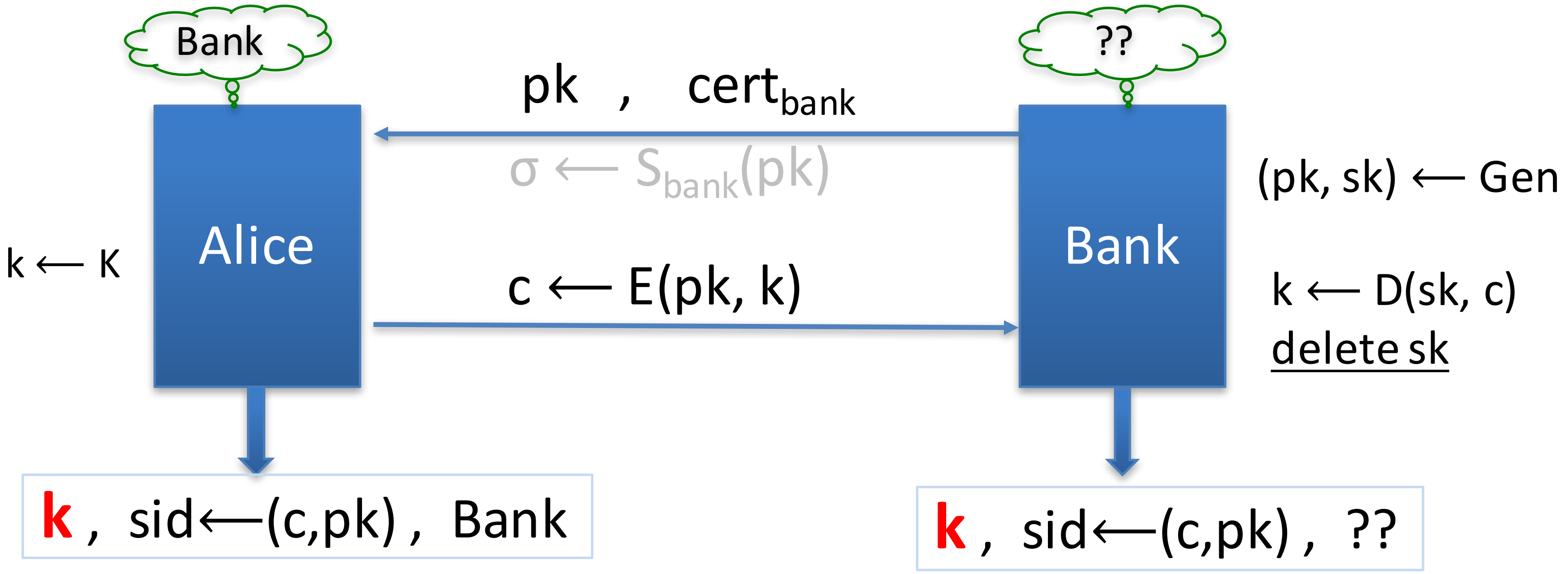
Simple one-sided AKE with forward-secrecy



(pk, sk) are ephemeral: sk is deleted when protocol completes

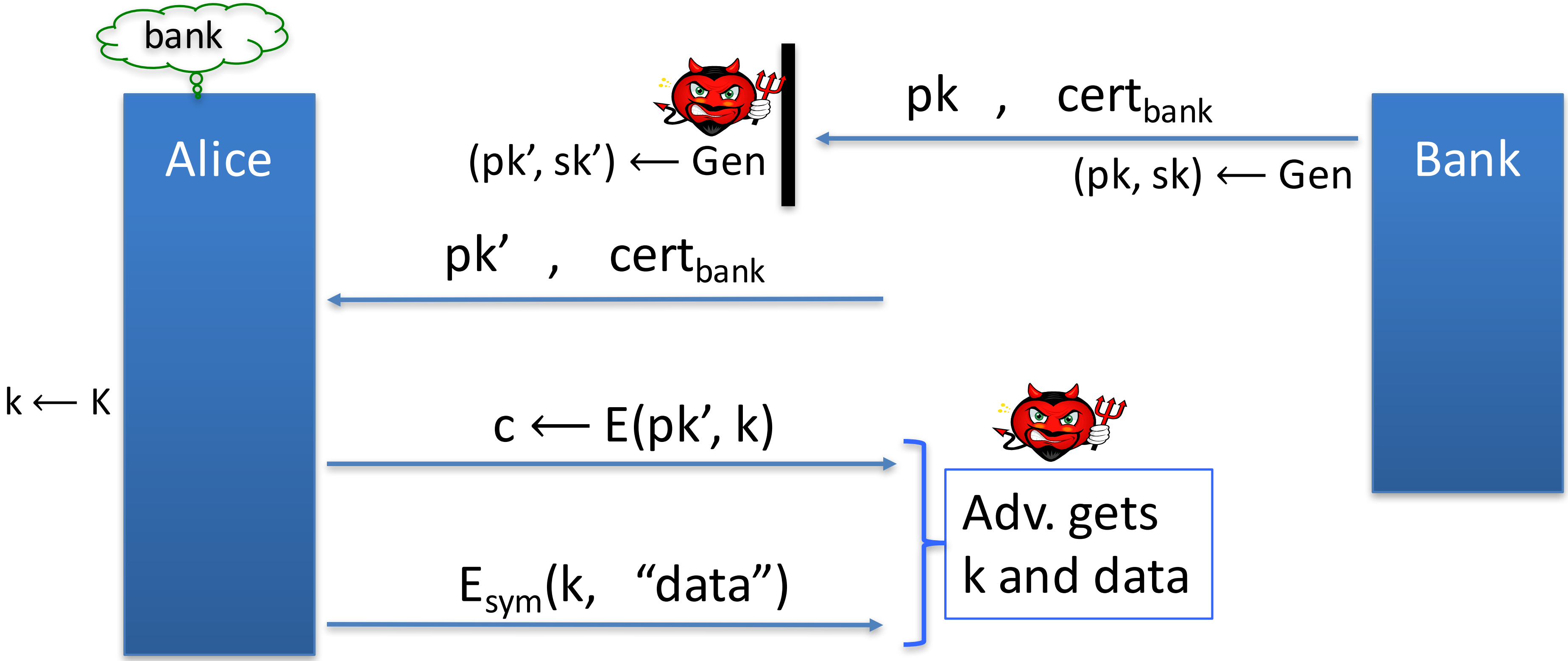
Compromise of Bank: past sessions are unaffected

Insecure variant: do not sign pk



Attack: complete key exposure

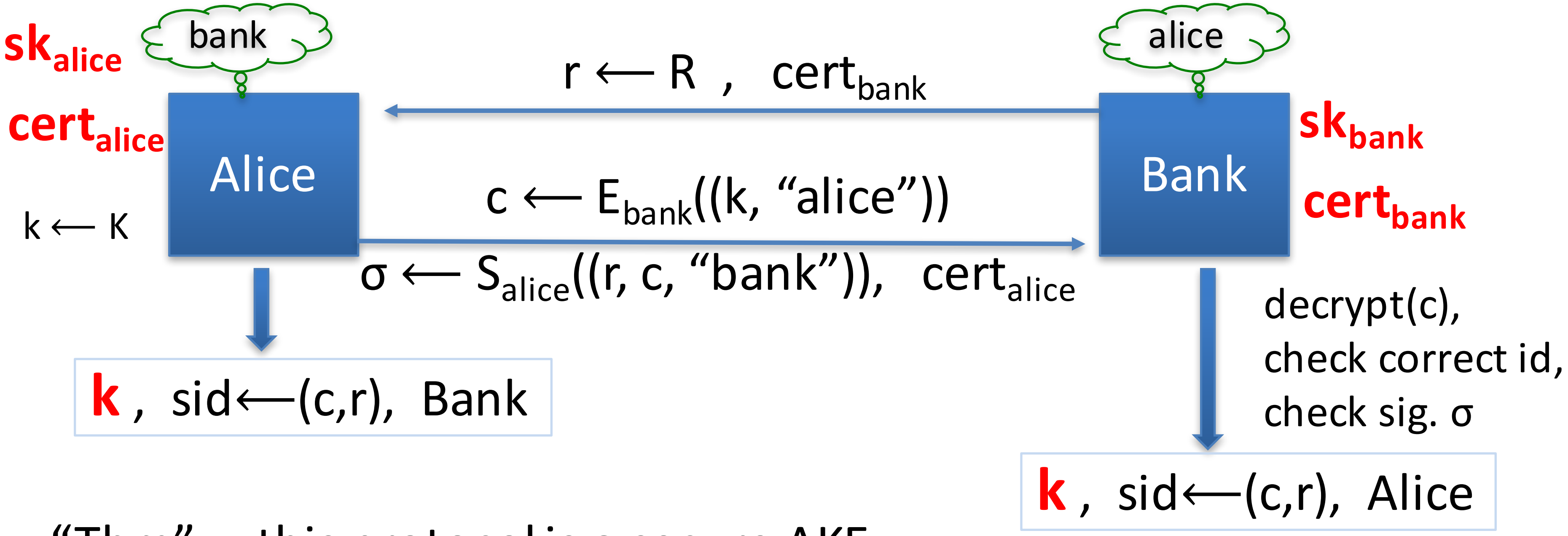
Attack: key exposure



Two-sided AKE

For now: no forward secrecy

Two-sided AKE (mutual authentication)



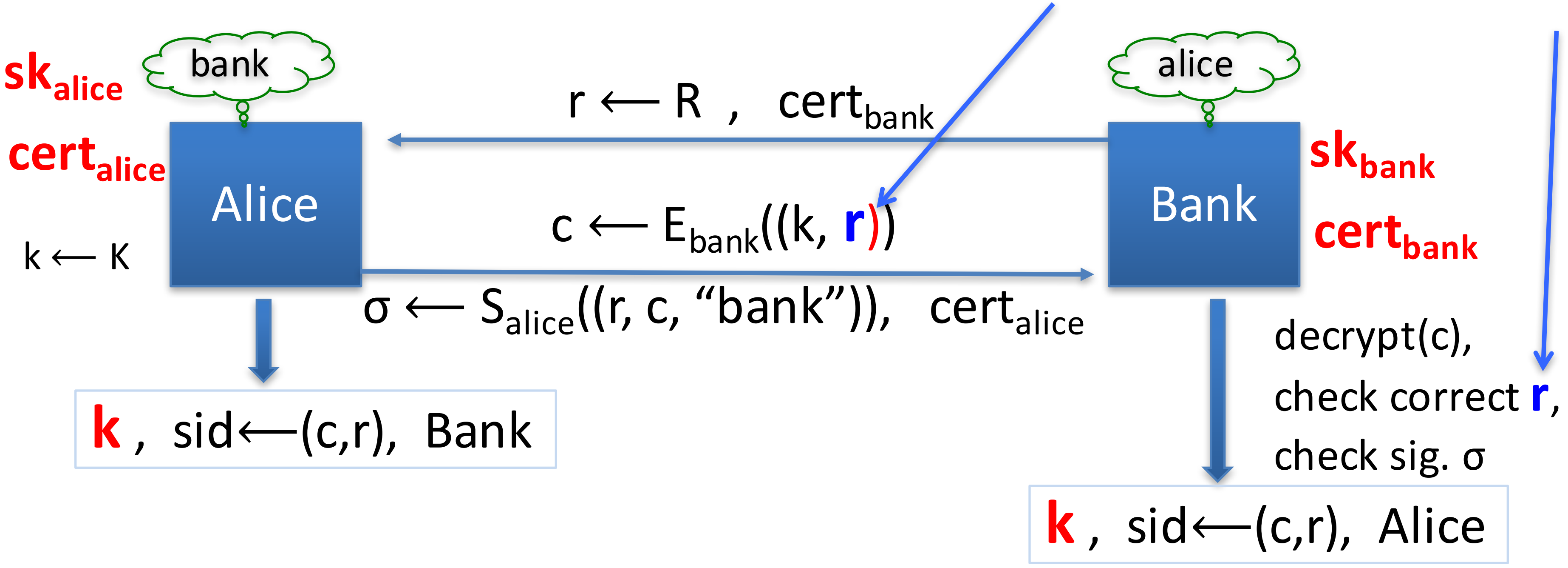
"Thm": this protocol is a secure AKE

Informally: if Alice and Bank are not corrupt then we have (1) secrecy and (2) authenticity for Alice and for Bank

Insecure variant: encrypt **r** instead of "Alice"

Any change to protocol makes it insecure, sometime in subtle ways

Example:



Attack: identity misbinding

bank

evil

$r \longleftarrow R$, $\text{cert}_{\text{bank}}$

Alice

Bank

$c \longleftarrow E_{\text{bank}}((k, \mathbf{r}))$

$\sigma \longleftarrow S_{\text{alice}}((r, c, \text{“bank”}))$, $\text{cert}_{\text{alice}}$

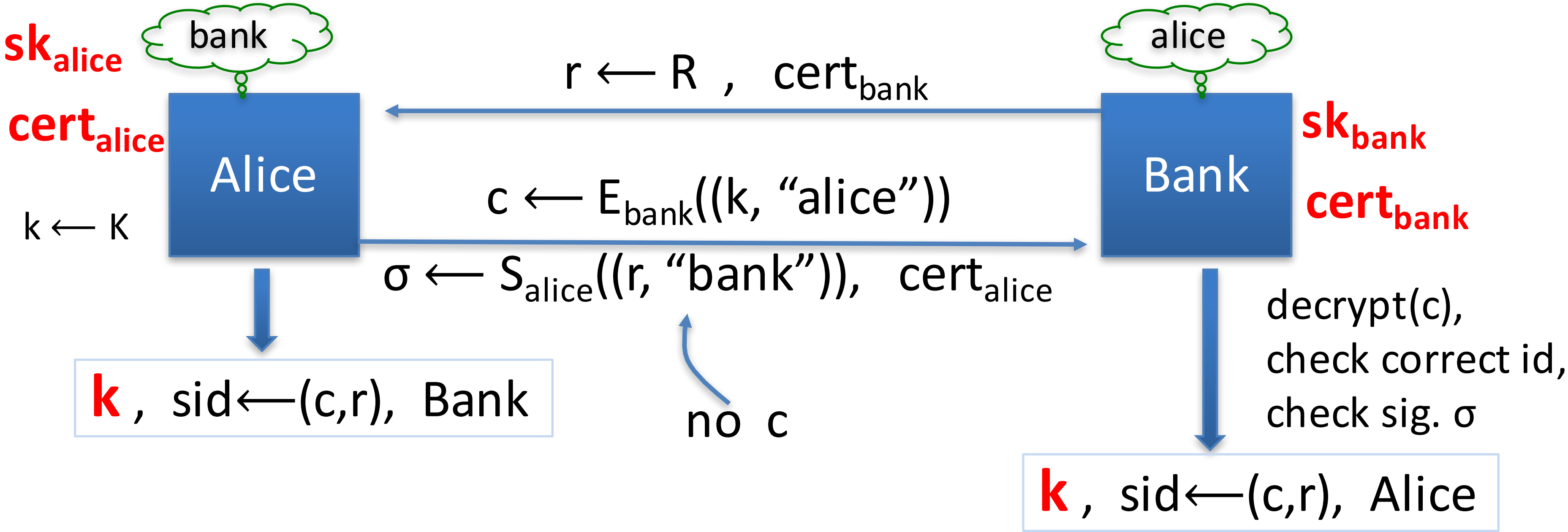
c

$\sigma' \longleftarrow S_{\text{evil}}((r, c, \text{“bank”}))$, $\text{cert}_{\text{evil}}$

decrypt(c),
check r,
check sig. σ'

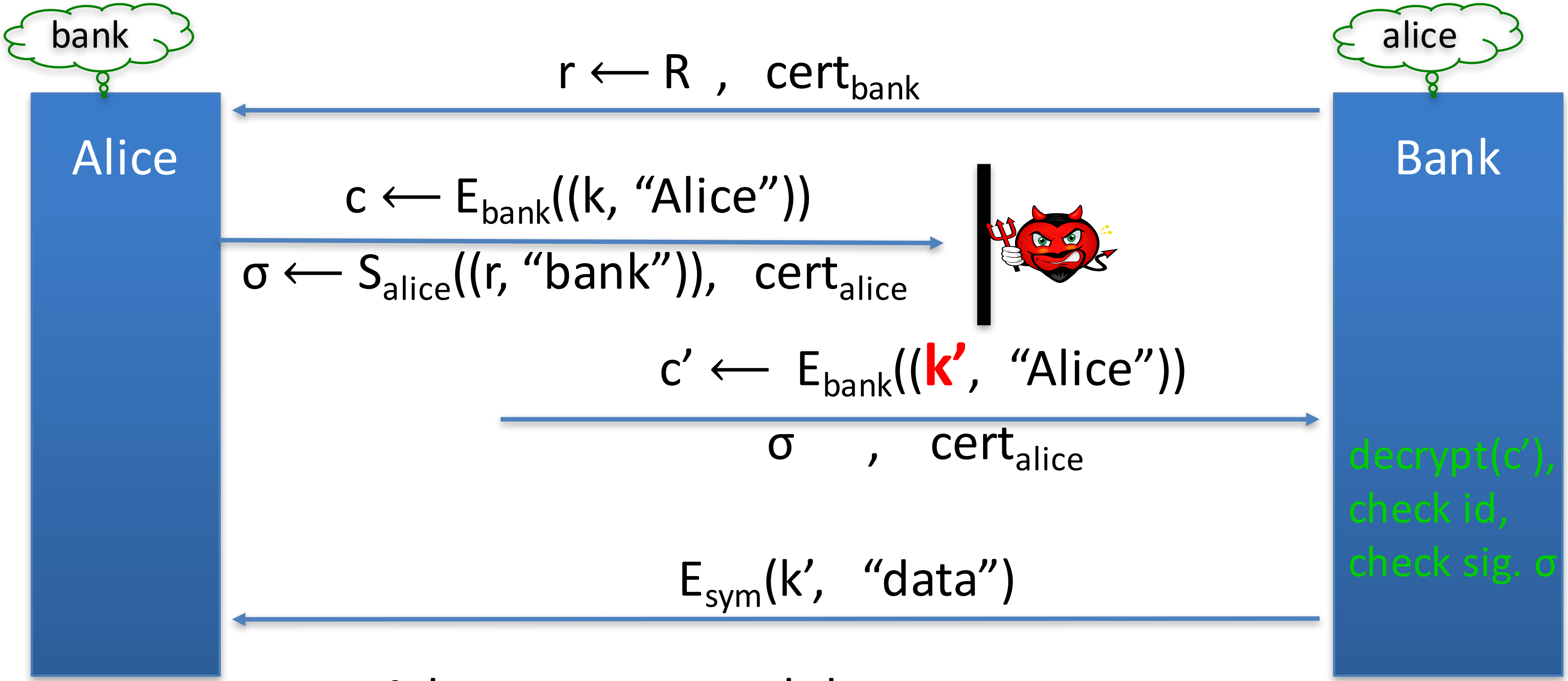
$E_{\text{sym}}(k,$ “deposit this check into my account”)

Insecure variant: do not sign c



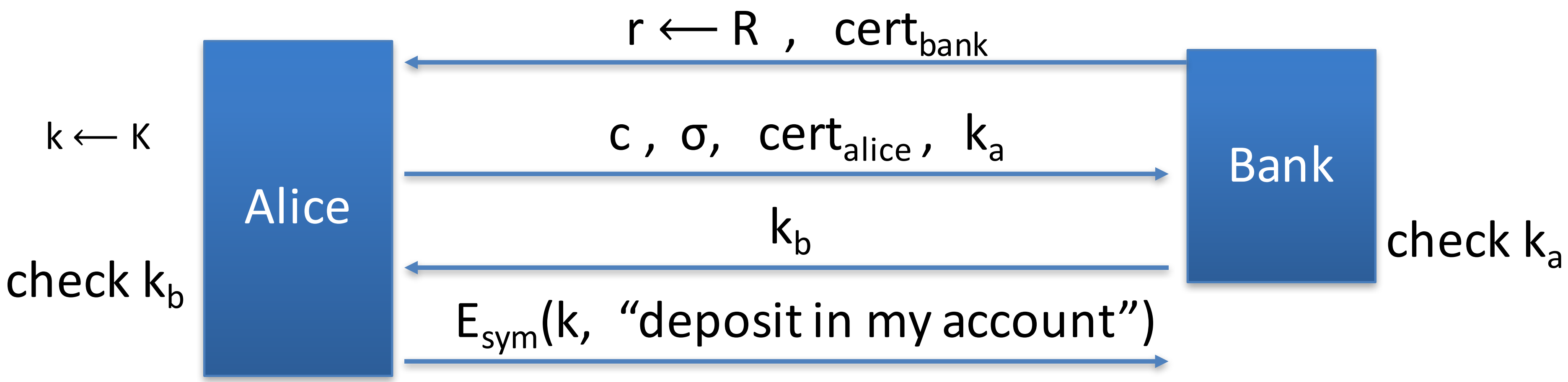
Attack: key exposure

Attack: key exposure



Adversary can read data

Can we defeat the attack on previous slide with key confirmation?



Yes, the attacker does not know k and cannot send a valid k_b

Many more AKE variants

Two-sided AKE with forward secrecy:

AKE with end-point privacy:

- Goal: certificates are not visible to adversary (TLS 1.3)

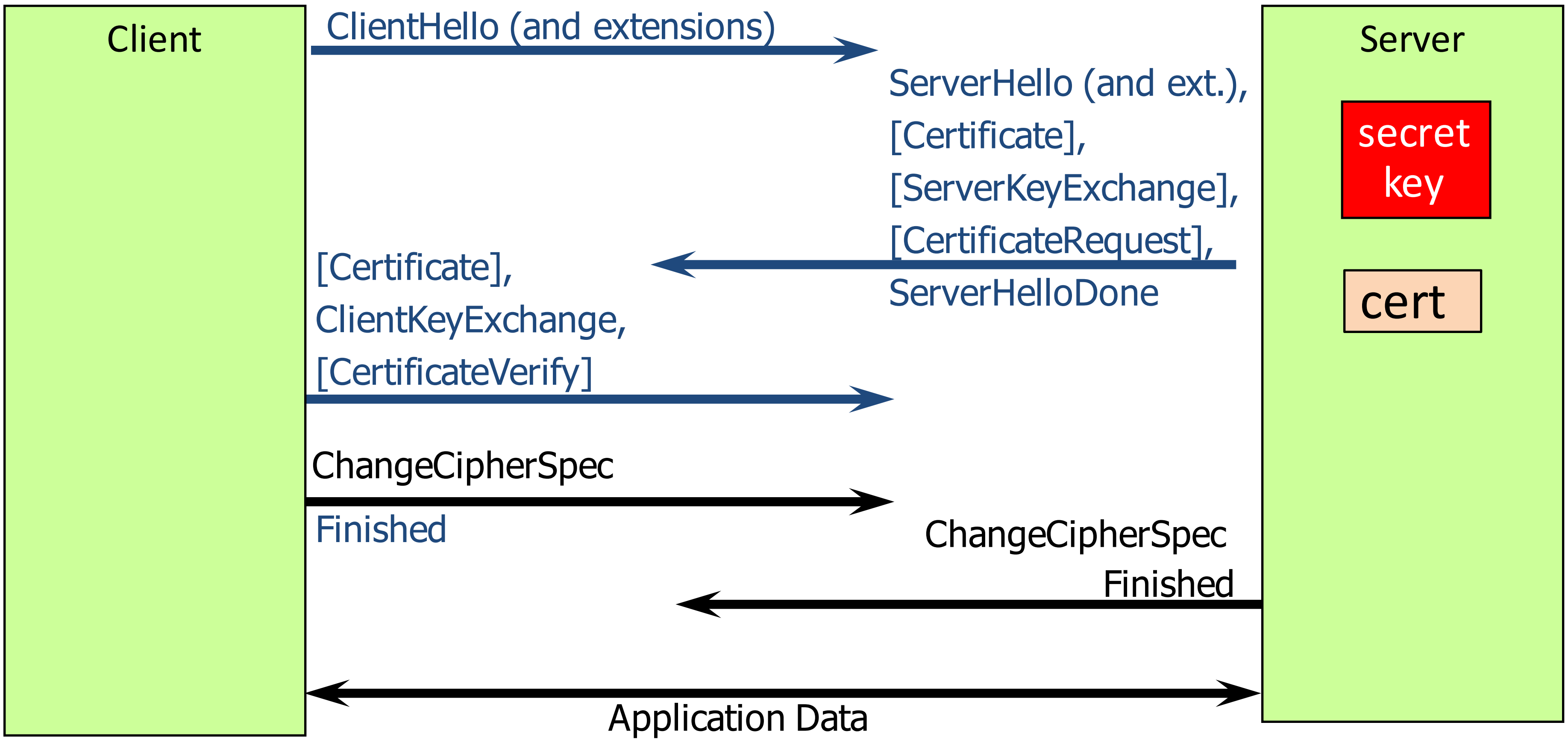
AKE based on a shared secret between Alice and Bank:

- High entropy shared secret: want forward secrecy
- Password: ensure no offline dictionary attack (PAKE)

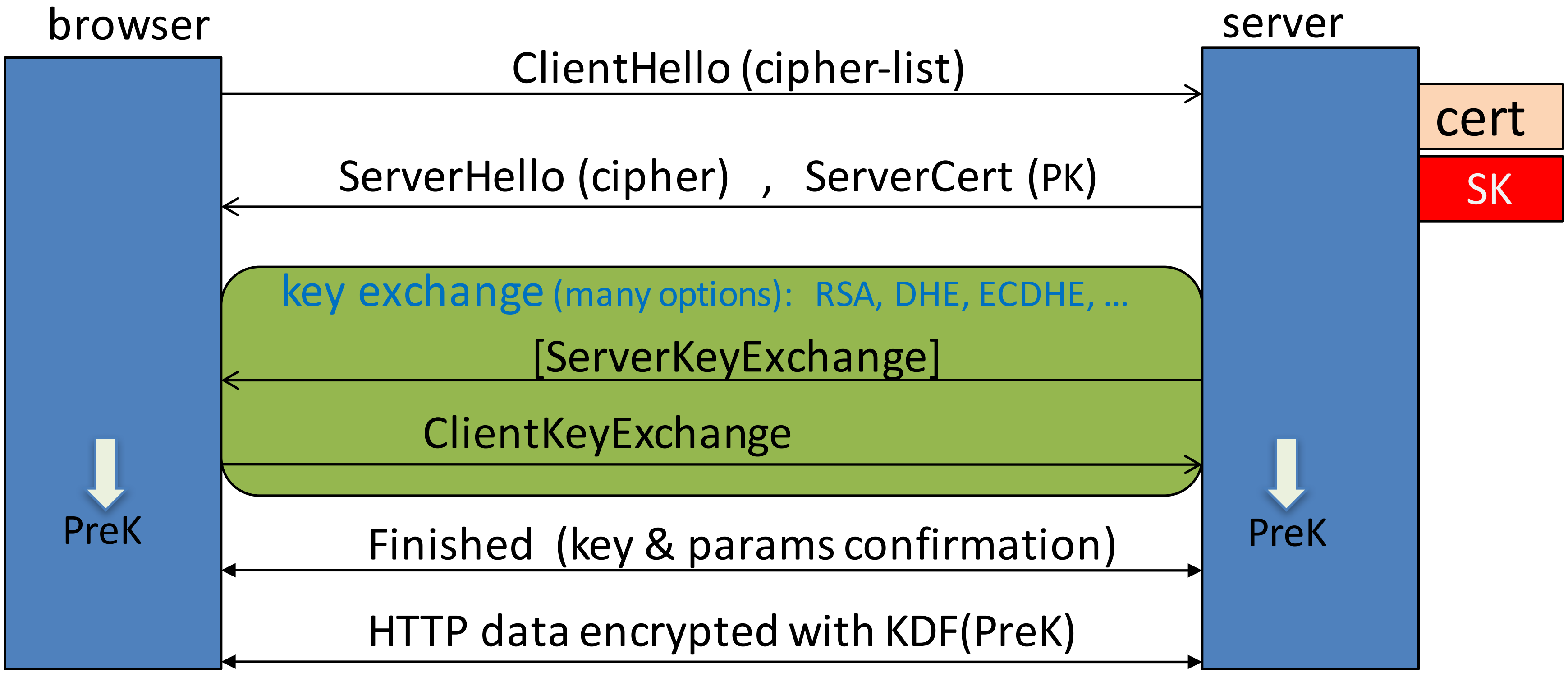
Auth. key exchange

TLS v1.2 key exchange

TLS session setup (handshake)

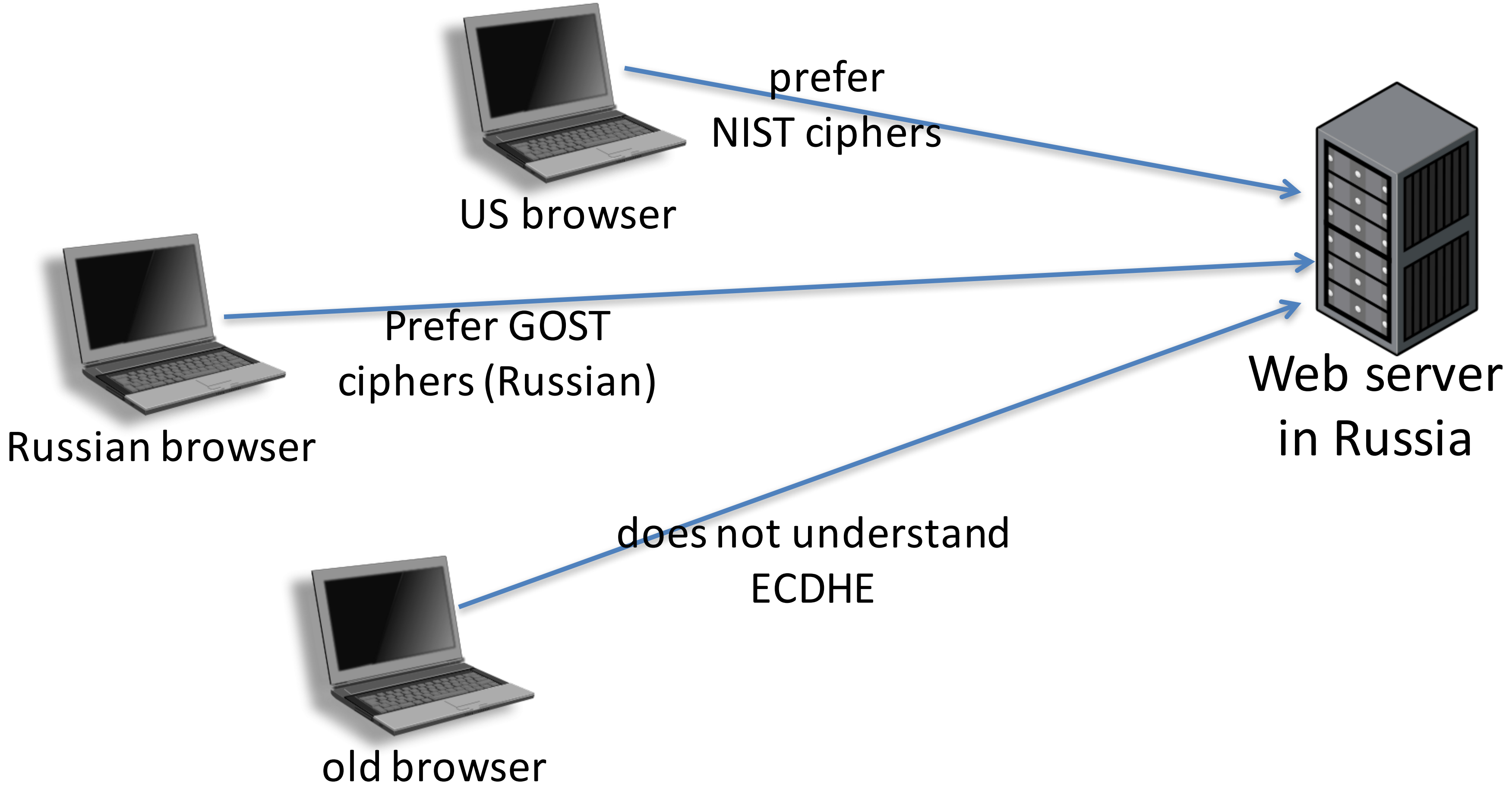


Brief overview of SSL/TLS

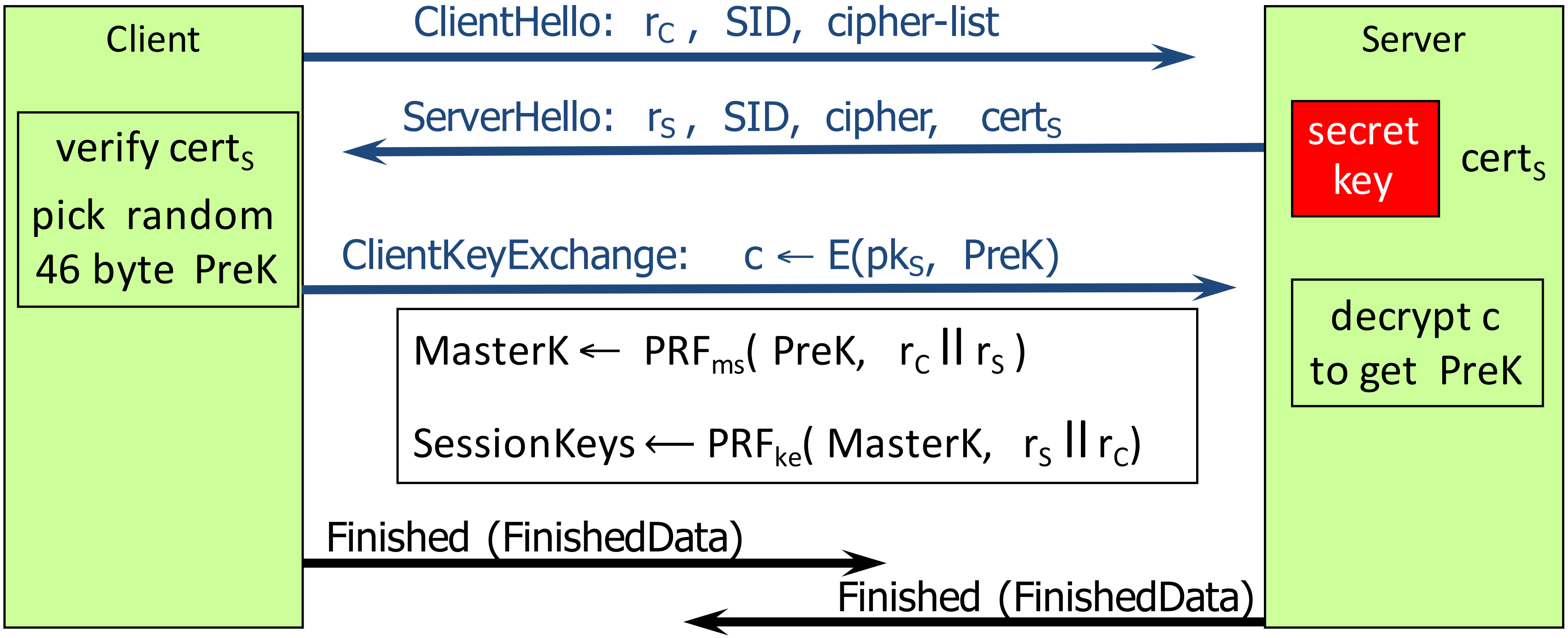


In this diagram: one sided authentication (no client authentication)

The need for negotiating ciphers



Abstract TLS: RSA exchange (simplified)



Key Confirmation: $\text{FinishedData} = \text{PRF}_{vd}(\text{MasterK}, \ \text{hash(HandshakeMessages)})$

Dan Boneh

Properties

r_C, r_S: prevent replay of of old session

RSA key exchange: no forward secrecy

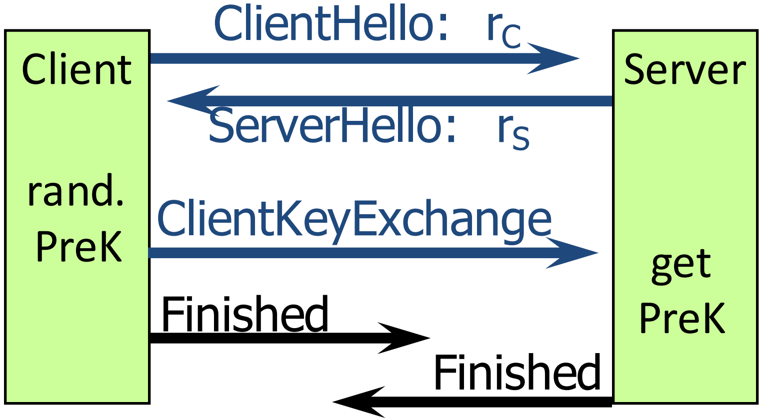
- Compromise of server secret key exposes old sessions
- Costly RSA decryption on server, easier RSA enc. on client

One sided identification:

- Browser identifies server using server-cert
- Server has no guarantees about client's identity
  - TLS has support for mutual auth. (client needs sk_C and $cert_C$)

Suppose always $r_C = 0$, but r_S is random

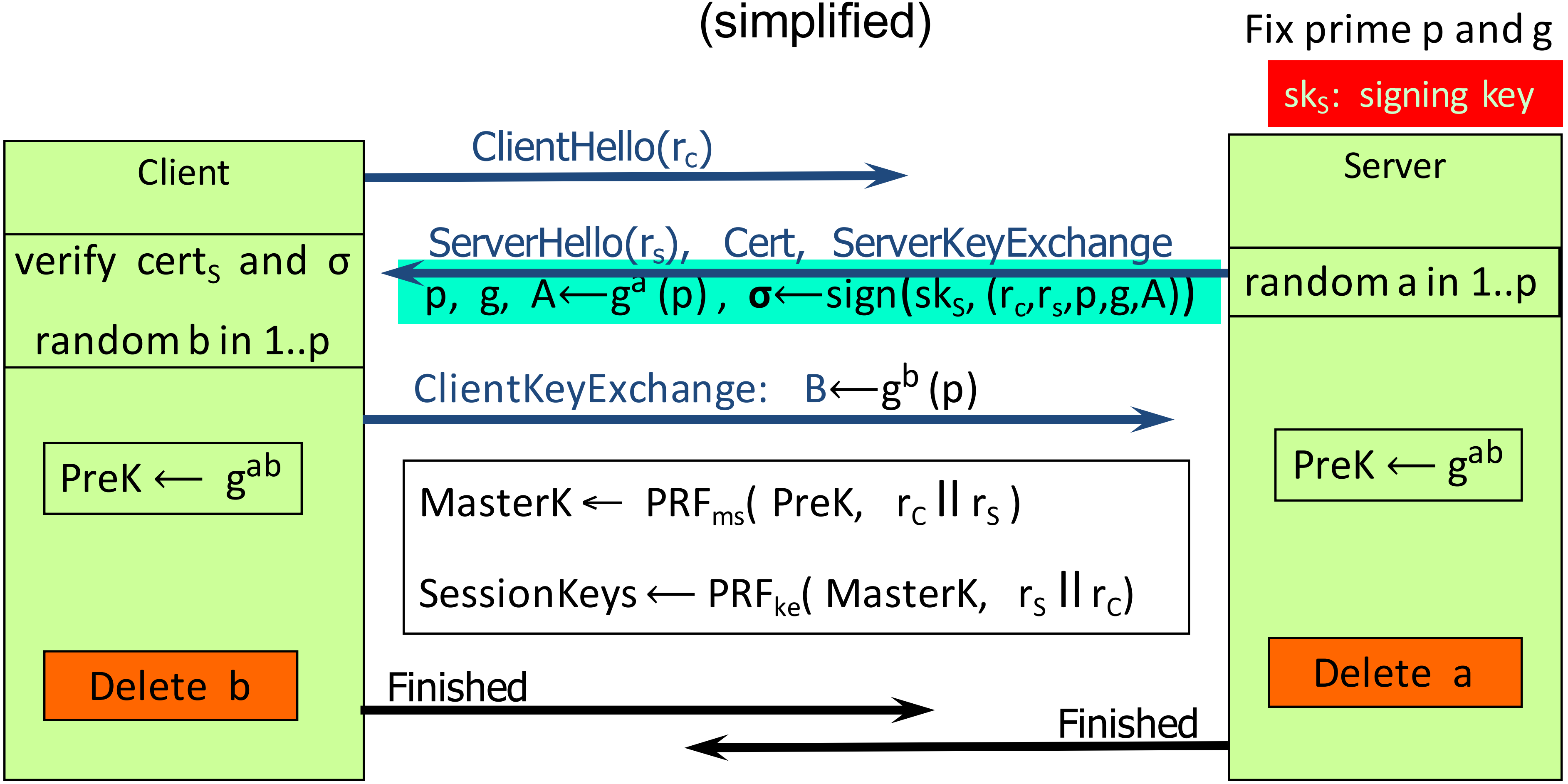
Would this be a secure one-sided AKE?

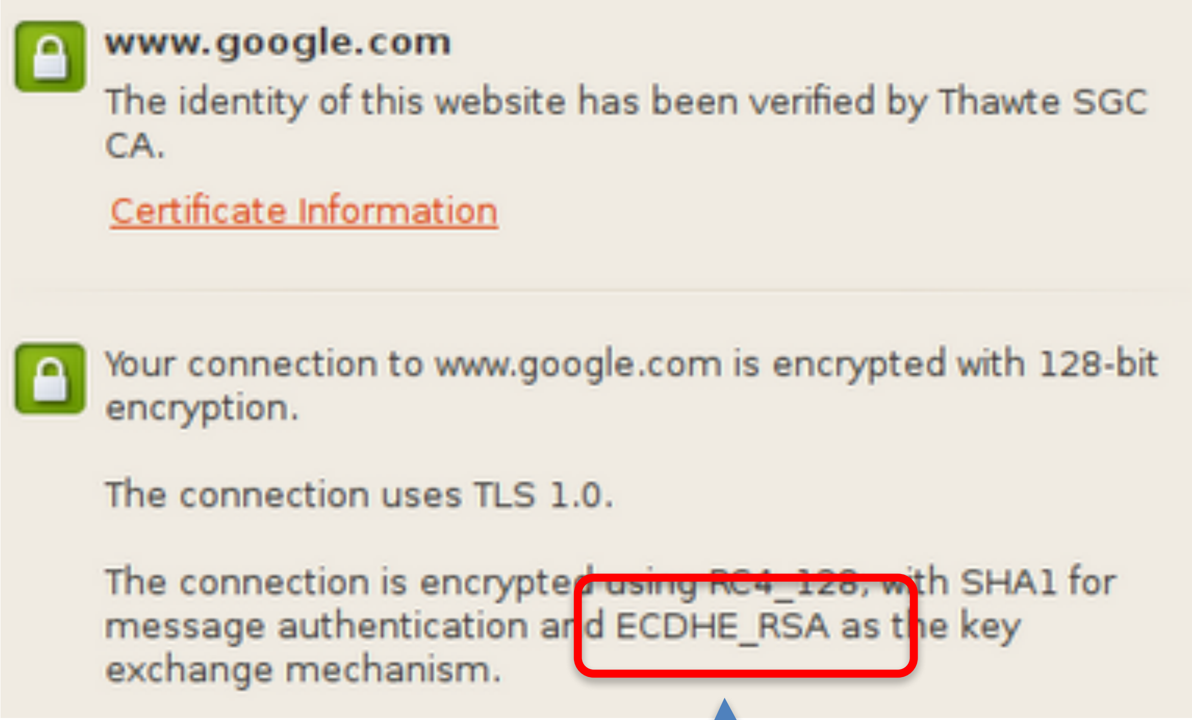


- ○ No, an attacker can replay an old session to the server
- ○ No, an attacker can replay an old session to the client
- ○ Yes, it would be a secure one-sided AKE
- ○ No, a man in the middle can expose PreK

TLS key exchange with forward-secrecy (DHE)

(simplified)





Elliptic curve
Diffie-Hellman

Prefer ECDHE over DHE

Performance: RSA vs. forward-secrecy

Cost of crypto operations on server per handshake:

- RSA key exchange: one RSA-2048 decryption (deprecated in TLS 1.3)
- ECDHE: Diffie-Hellman in group G with generator $g \in G$

1. One exp. to compute $A \longleftarrow g^a \in G$
2. One sig. on Diffie-Hellman parameters (G,g,A)

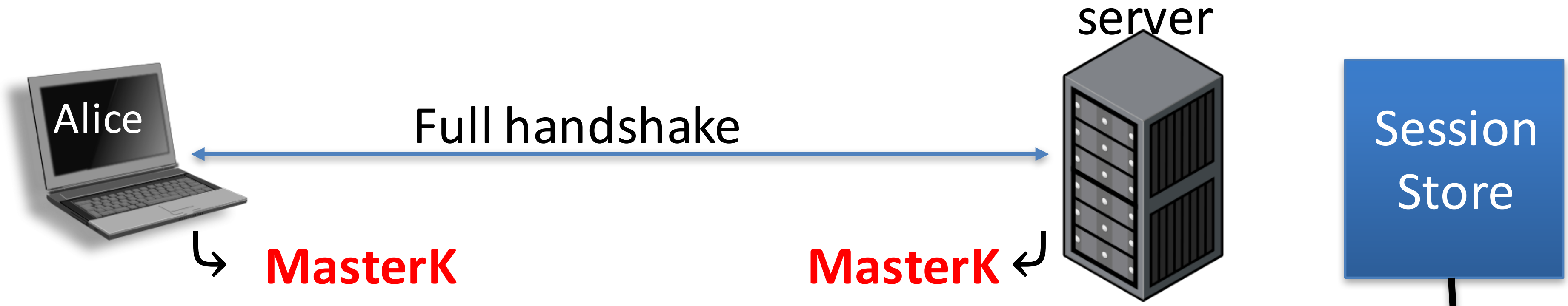
must be done for every handshake

3. One exp. to compute DH secret: $\text{PreK} \longleftarrow g^{ab} \in G$

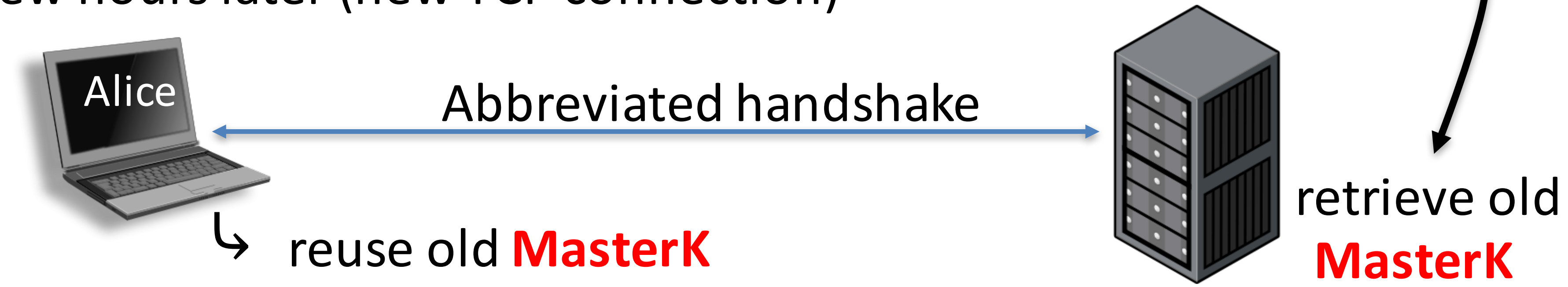
Server support (2014): RSA (99.9%) , DHE (60%) , ECDHE(18%)

Session Resume

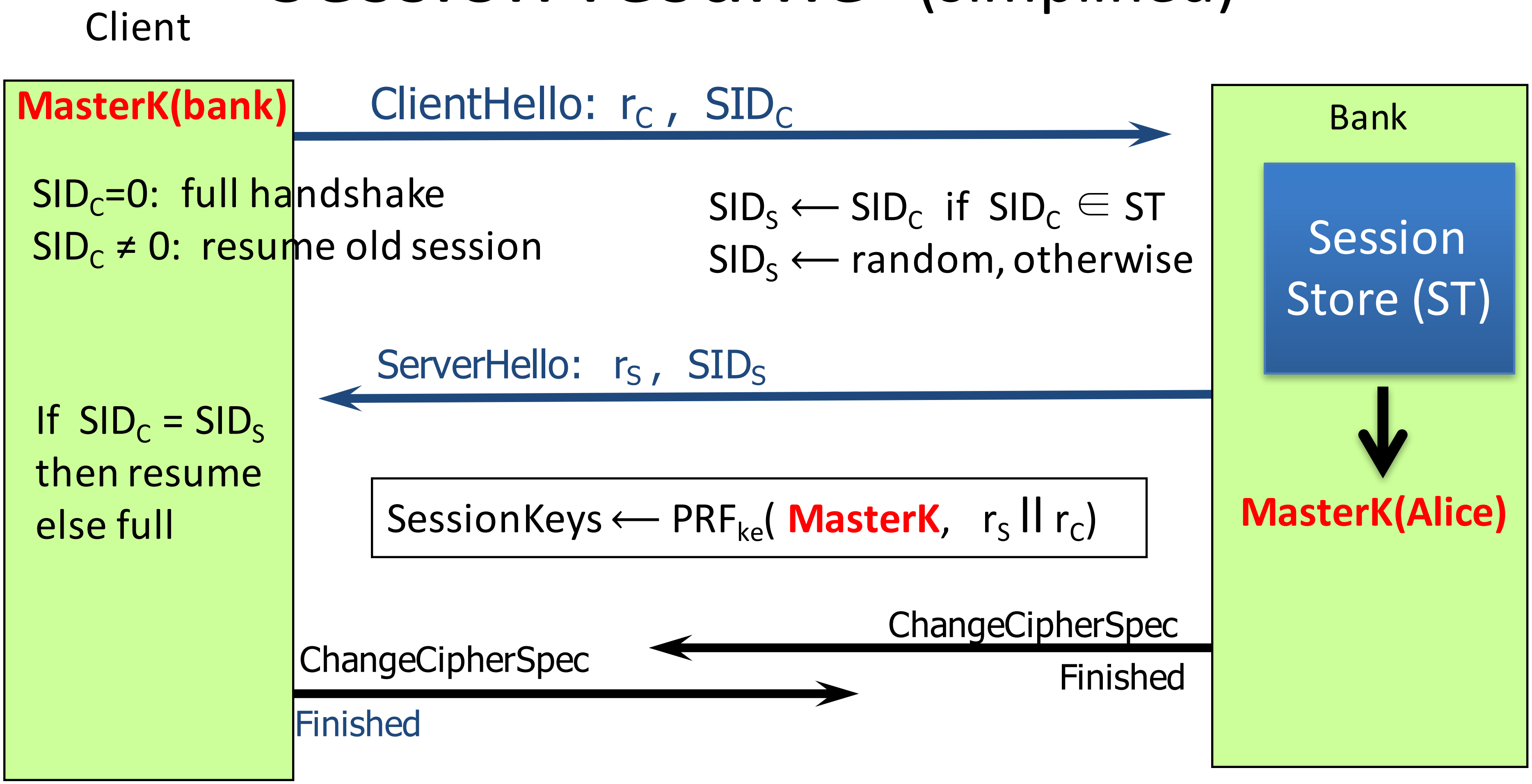
Goal: reduce # of full handshakes

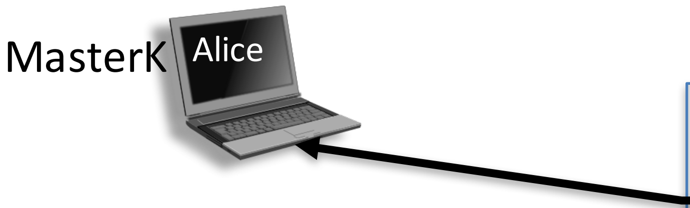


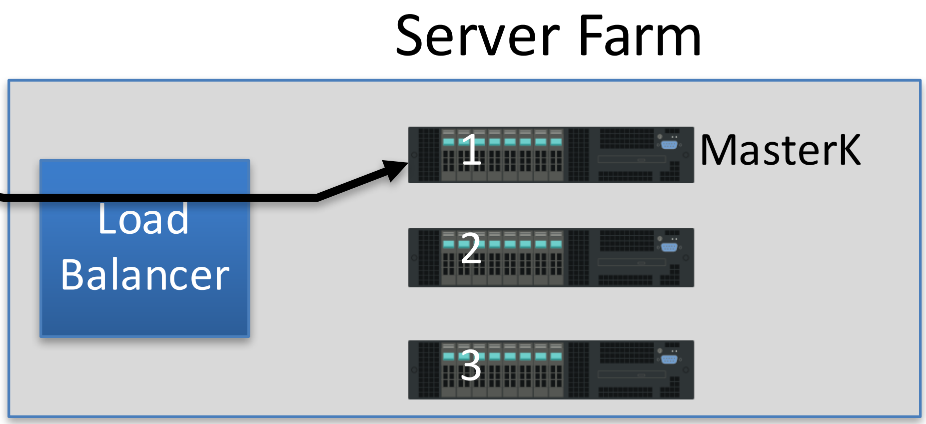
Few hours later (new TCP connection)



Session resume (simplified)







On a first connection Alice is connected to Server #1.

What must the site do to avoid a full handshake on subsequent connections from Alice?

- ○ Ensure that Alice always connects to server #1
- ○ Ensure that SessionStore is shared among all servers in farm
- ○ Do all TLS processing in the load balancer
- ○ All of the above are reasonable solutions

THE END