Security in networks

Damien Saucez
Inria Sophia Antipolis

UBINET/CSSR 02/01/2013
Evolving Internet II

Contact information

- Damien Saucez
  - Office: Inria, Lagrange L.145
  - Email: damien.saucez@inria.fr
  - Phone: +33 4 89.73.24.18

Outline of the course

- 12/14/2012: naming and addressing
- 12/21/2012: routing and forwarding
- 01/25/2013: interior gateway protocols
- 02/01/2013: exterior gateway protocols
- 02/08/2013: security in networks
- 02/15/2013: final examination

Table of Content

- The basics
- Securing communications
- Operational corner
- Research corner

The basics

Security threats

- Intrusion
  - an attacker gains remote access to some resources that are normally denied to her
    - e.g., steal processing power, botnets
- Eavesdropping
  - an attacker collects traffic of a target in order to gain access to restricted sensitive information
    - e.g., steal passwords by sniffing wireless traffic
- Denial of Service (DoS)
  - an attacker disrupts a specific targeted service
    - e.g., block the youtube website

The attackers

- Hackers
  - look for challenge, notoriety, and fun
    - e.g., hackers, script kiddies, students :-D
- Spies
  - look for political/business gains
    - e.g., intelligence, police, industrial spies
- Criminals
  - look for financial gains, religious/political visibility, or just to break something
  - e.g., criminals, terrorists, vandals

Definitions

- Key
  - input of cryptographic functions to determine its output
- Authentication
  - proof that the message is coming from the one claiming to be at the origin of the message
- Integrity
  - proof that the message has not been altered since its creation
- Non-repudiation of origin
  - an entity that generated a message cannot deny have generated the message
- Encryption
  - action of encoding of a message such that an eavesdropper can’t read the message but legitimate destination can
- Decryption
  - action of decoding an encrypted message
- Signature
  - a mathematically constructed proof of authenticity of a message

Hall of fame

- Alice and Bob
  - are legitimate users, Alice and Bob exchange messages
- Chuck
  - is a malicious user that is not between Alice and Bob
- Eve
  - is a malicious user that can eavesdrop
- Trudy
  - is a malicious user that can perform (wo)man-in-the-middle attacks
- Trent
  - is a legitimate user that plays the role of a trusted arbitrator

Why is good security level so hard to obtain?

- The security level of a system equals the security level of the weakest part of the system
  - e.g., encrypting your HDD to avoid information leak if the laptop is stollen is useless if the password is written on a post-it attached on the laptop
- Digital system are complexes
  - interactions with many components, distribution, easily bugged...

Security is a tradeoff

- Compare cost and probability of an attack and cost of securing the system against this attack
  - e.g., is that necessary to make data unbreakable for 20 years if they are outdated after 1 hour?
- Explain the security systems and their reasons
  - if a user does not understand why he must follow a procedure, he will not follow it
    - e.g., how many of you already give their password to someone else?
- Never “over-secure” a system
  - if the system is too hard to use, people will find countermeasure
    - e.g., too hard to use corporate mails? Then use gmail to send corporate mails...

Procedures!

- Protection will never be perfect
- Prepare procedures
  - what to do BEFORE an attack?
    - what to do to limit the risk (e.g., passwords) of attack and to be ready if an attack happens (e.g., backup)
  - what to do DURING an attack?
    - the attack is on going, how to stop it
  - what to do AFTER an attack?
    - the attack succeeded, how to recover from it

The techniques

- fill me
- fill me
- fill me

Securing communications

Live work

- Construct a communication mechanism where Alice and Bob can safely exchange messages
  - you have 20 minutes

Live work (contd.)

- Break your neighbor’s mechanism
  - you have 5 minutes

Objective

- Construct a communication mechanism where Alice and Bob can exchange messages such that
  - only Alice and Bob can generate messages
  - nobody else than Alice or Bob can read messages
  - nobody can alter messages

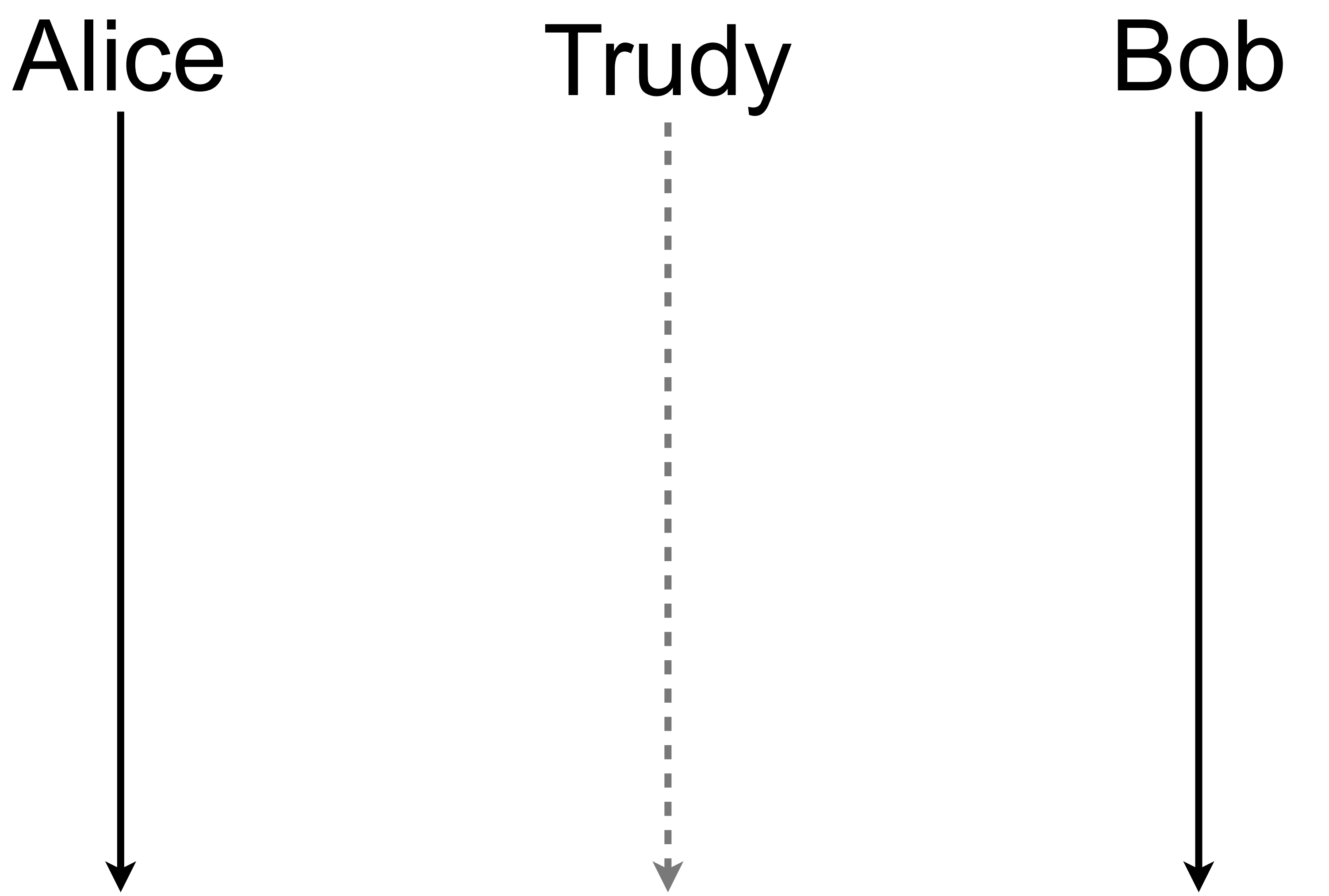
Steps

- fill me
- fill me
- fill me

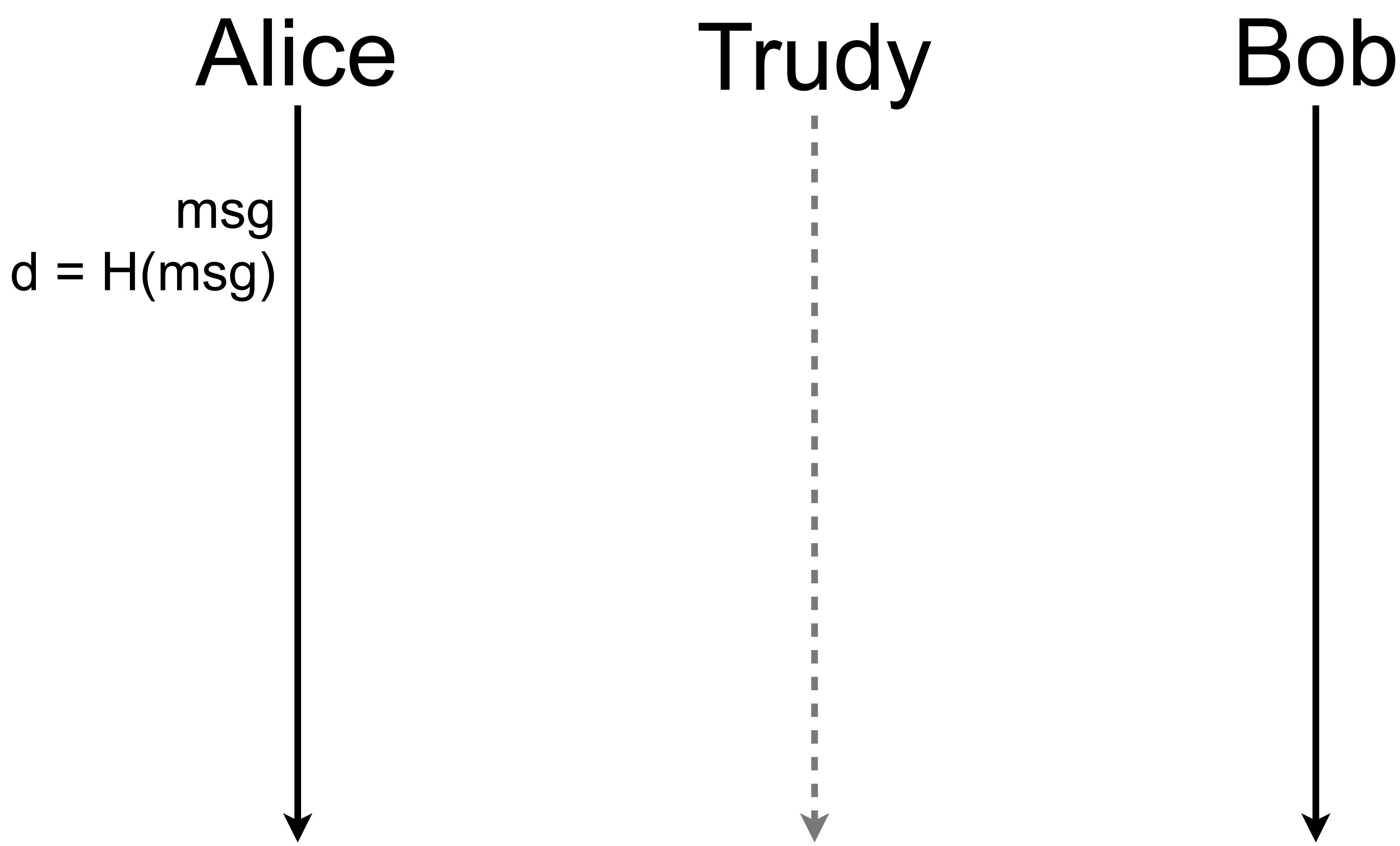
Hash function

- Validate that a message has not been altered on its way between Alice and Bob
- Hash functions map arbitrary large numbers of variable length to fixed-length numbers
  - h = H(m), h is called hash or digest
  - e.g., MD5, SHA-1, SHA-256
- Good hash functions for cryptography must be such that
  - H(m) is not complex to compute
  - but finding a m_2 such that $H(m_2) = H(m)$ is complex,
  - H(m) is deterministic,
  - H output must be evenly distributed over the output set
- Example
  - SHA-1 maps messages its input space on a 160-bits output
    - SHA-1(Message to validate) = 5e06ee754bda0d33cf65ec305ffc779404e66029
    - SHA-1(Message t**O** validate) = b1c306f8cb792fa14d4d1fdcf6f37d86c2fe6bb9

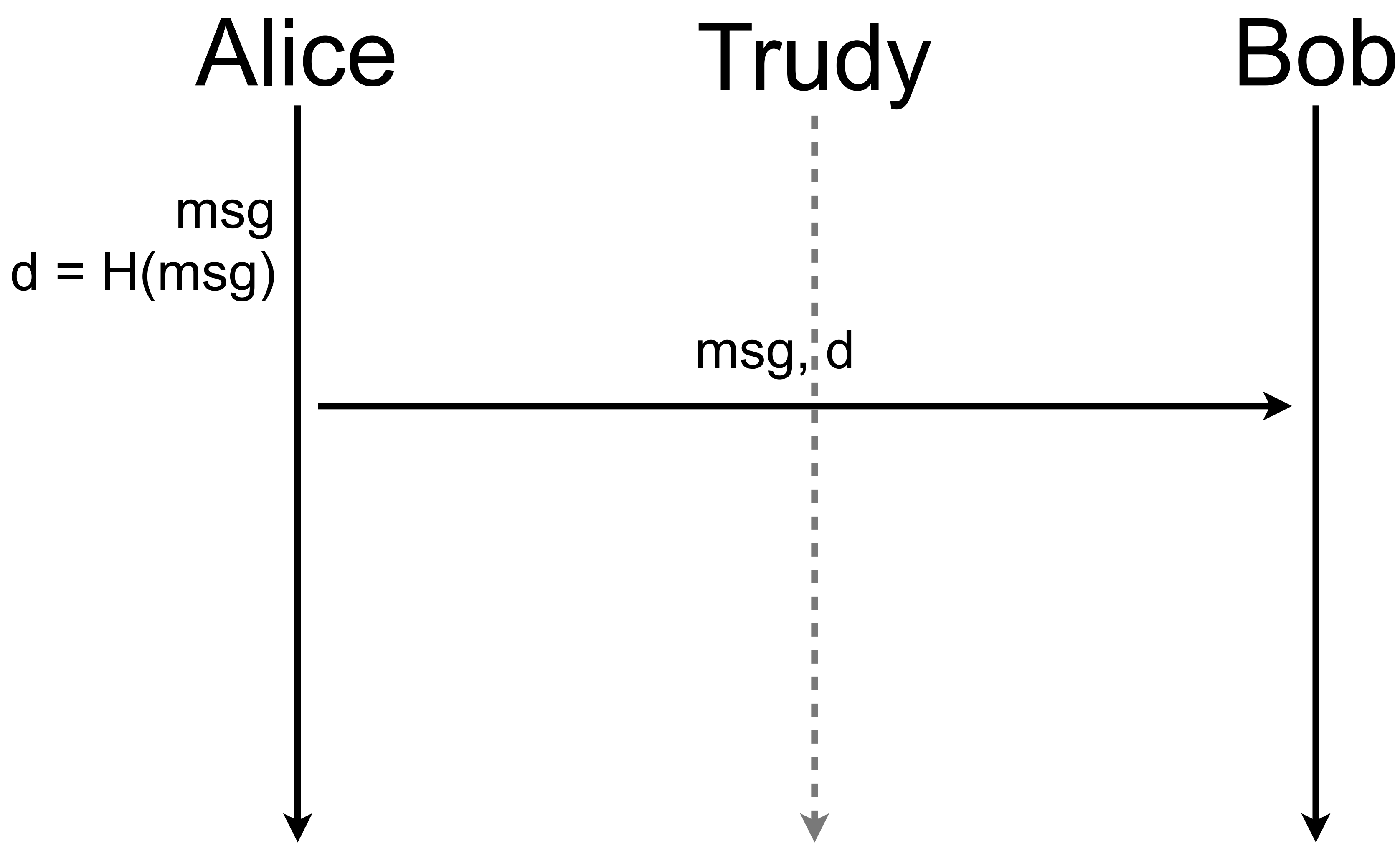
Is that enough?



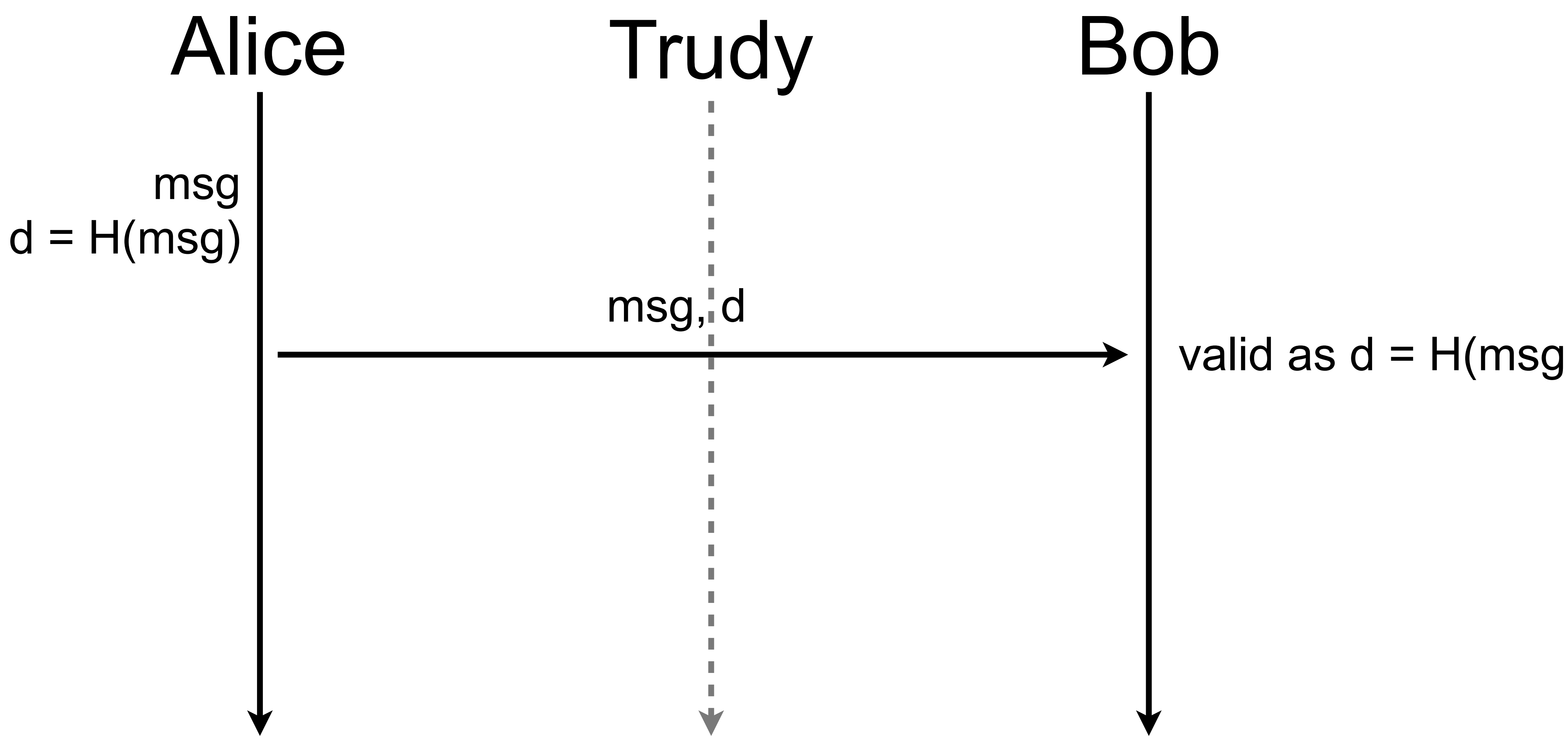
Is that enough?



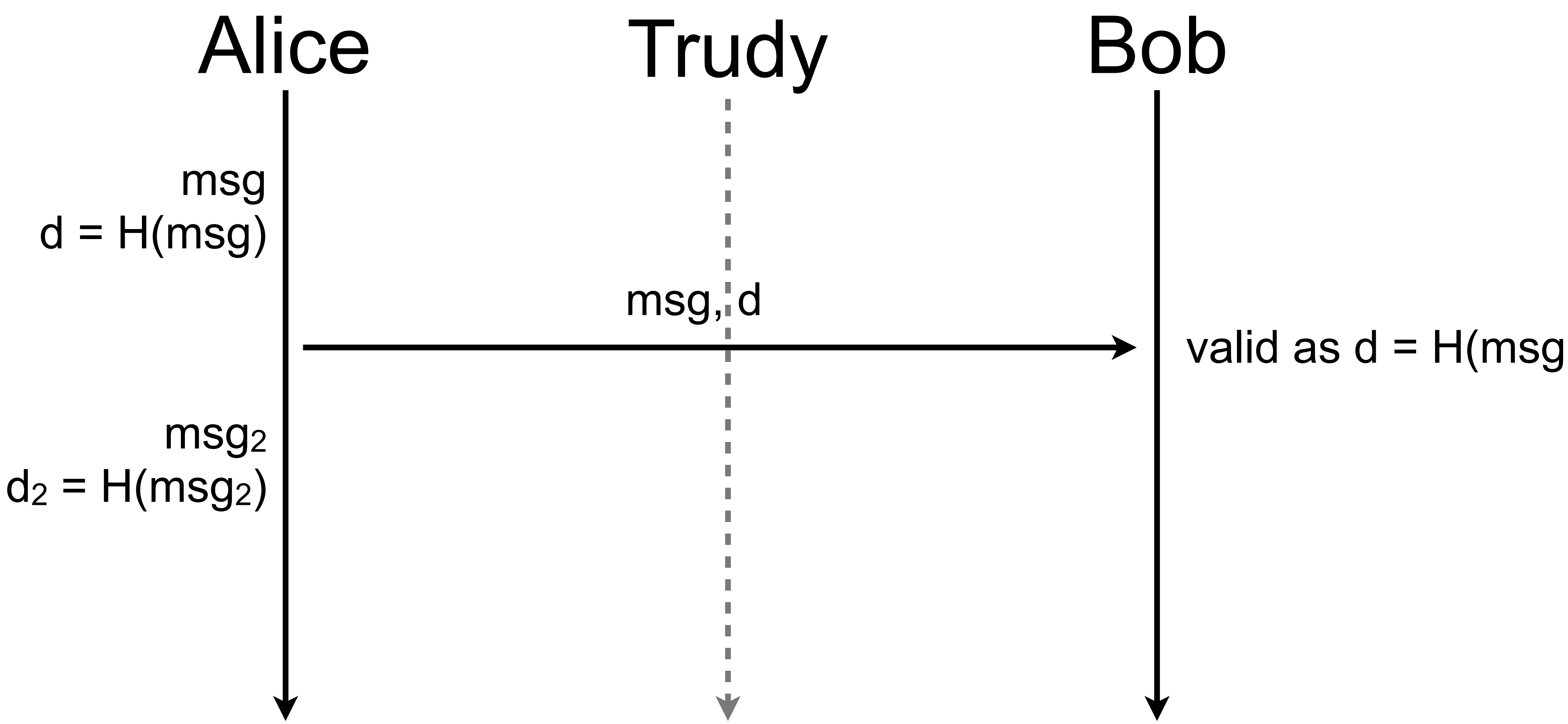
Is that enough?



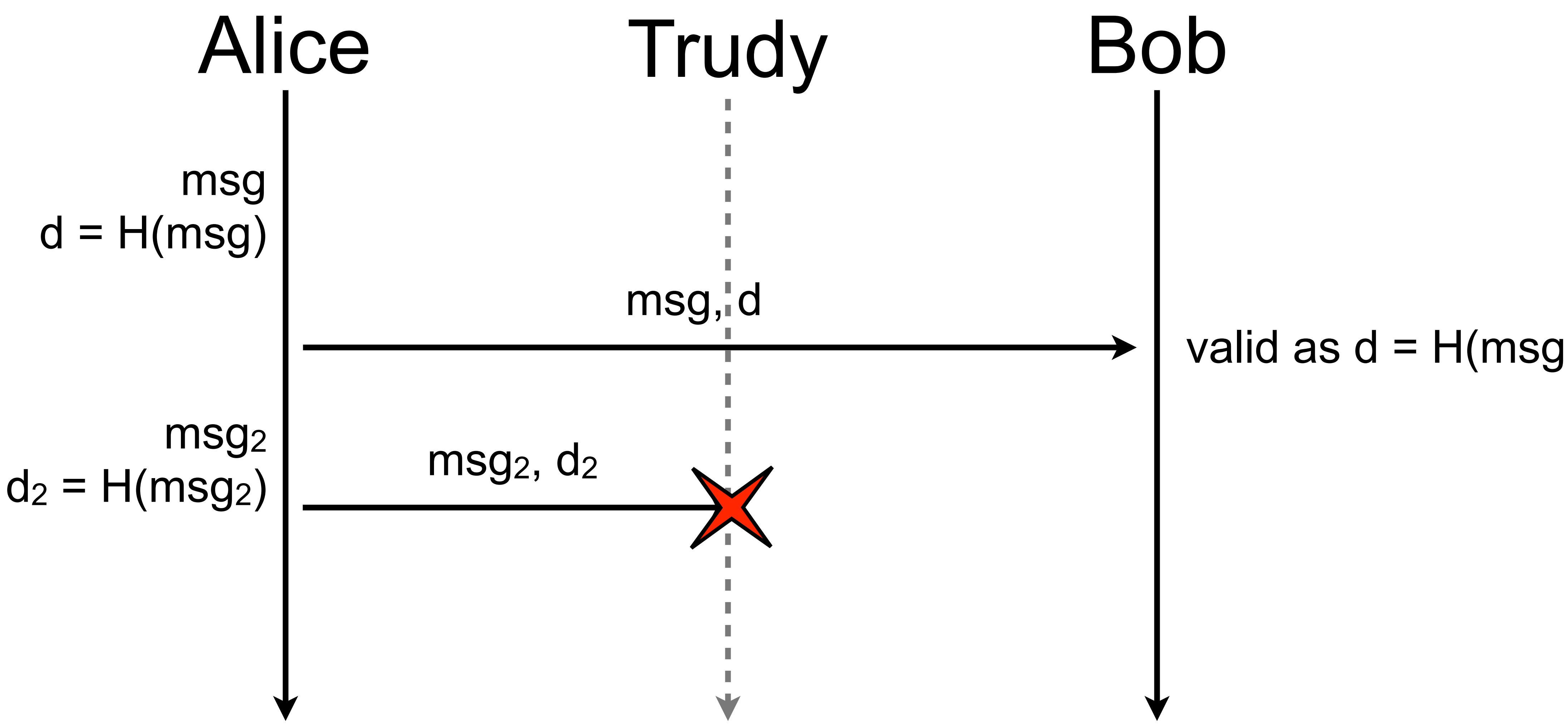
Is that enough?



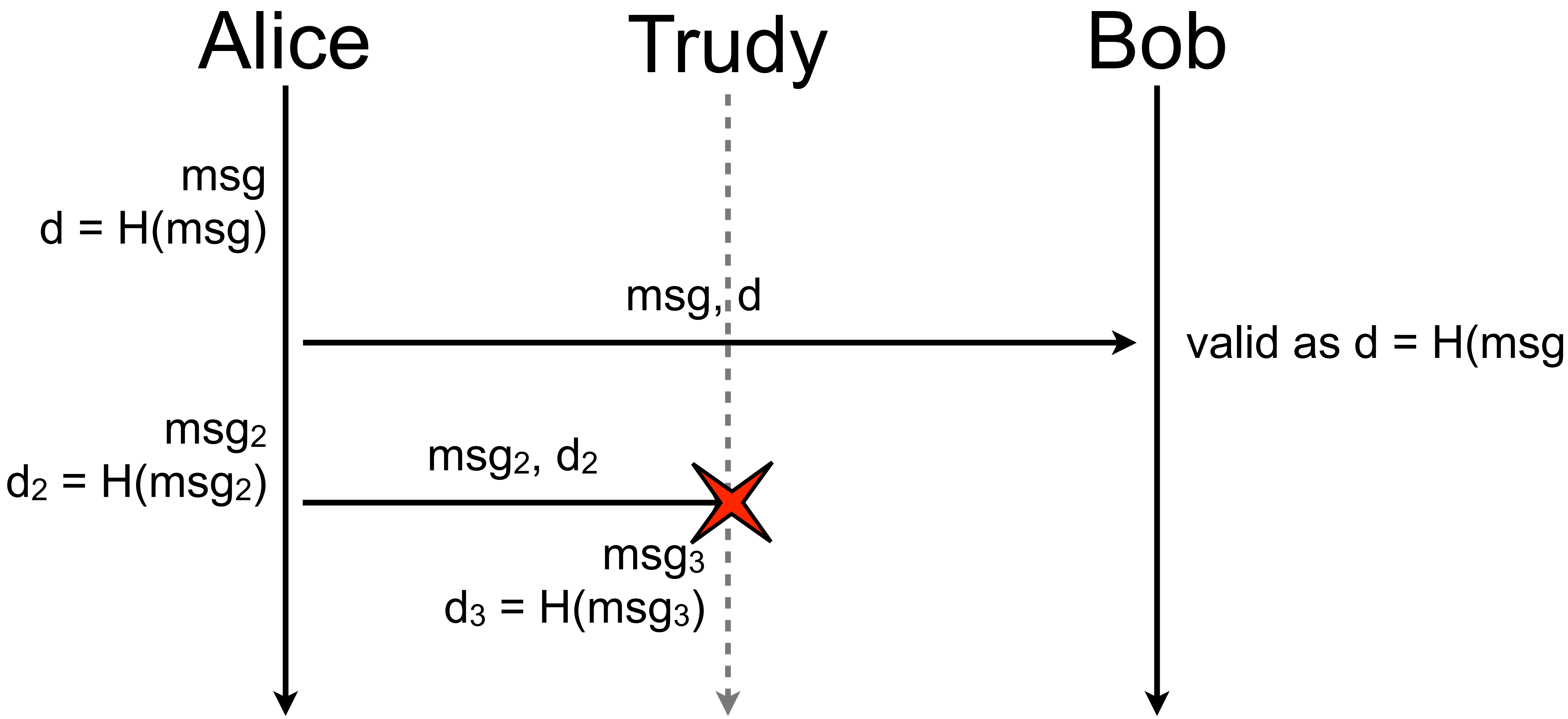
Is that enough?



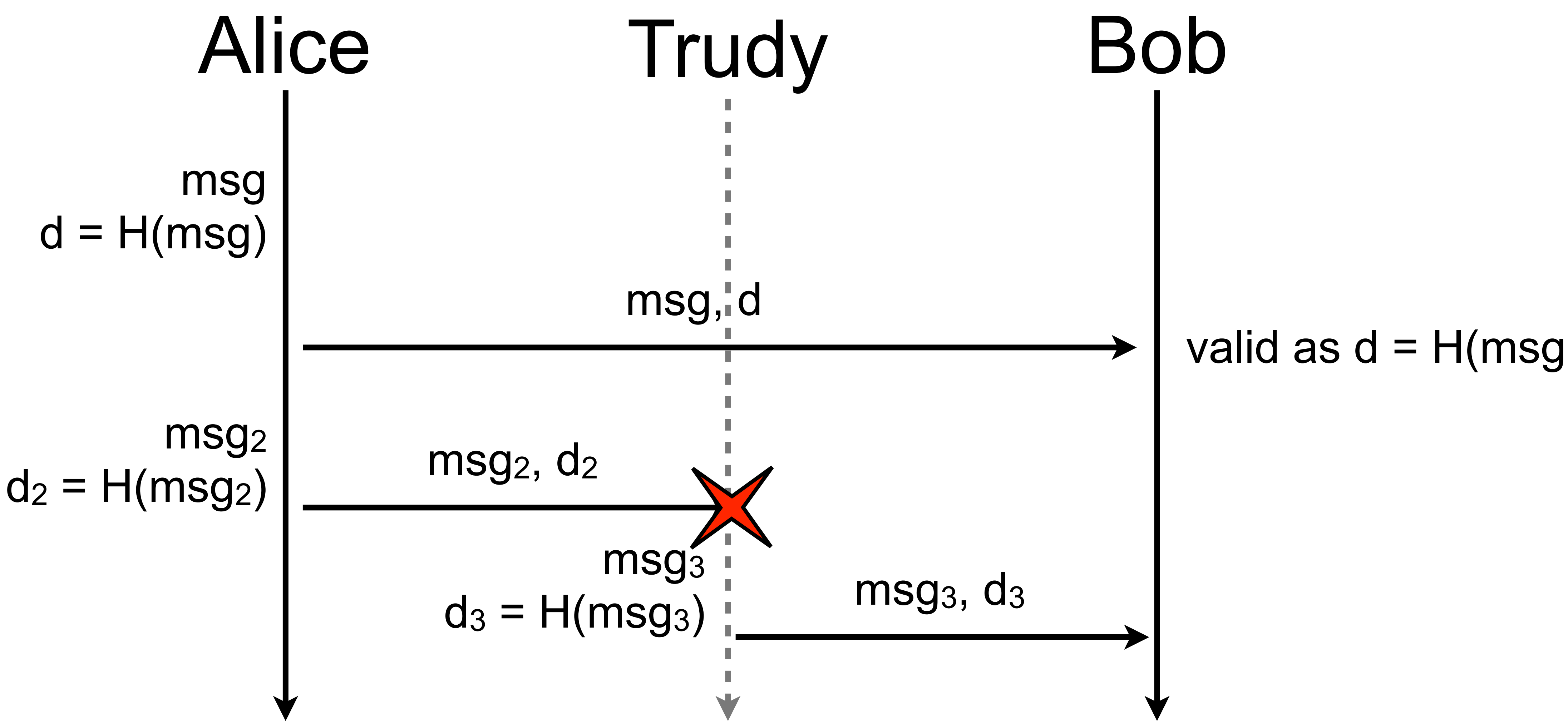
Is that enough?



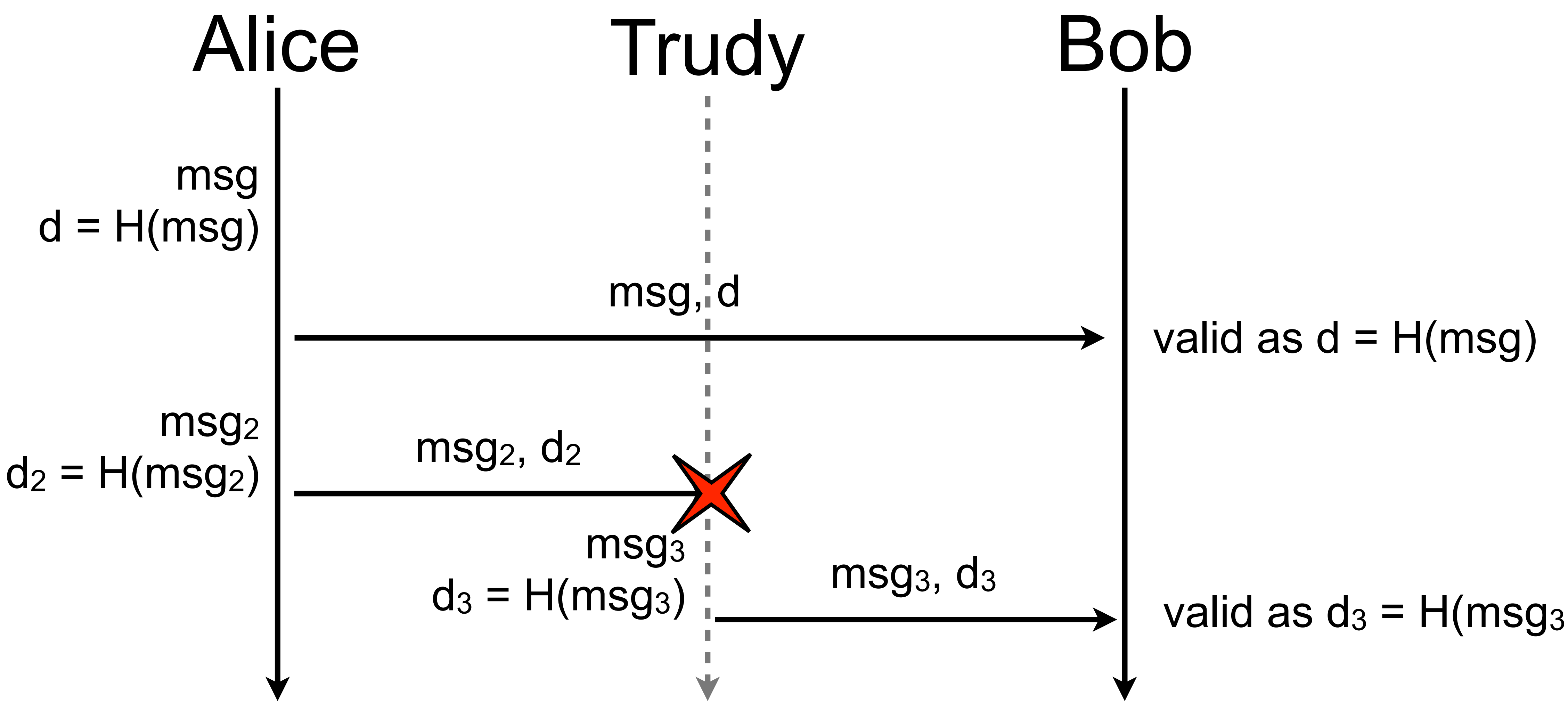
Is that enough?



Is that enough?



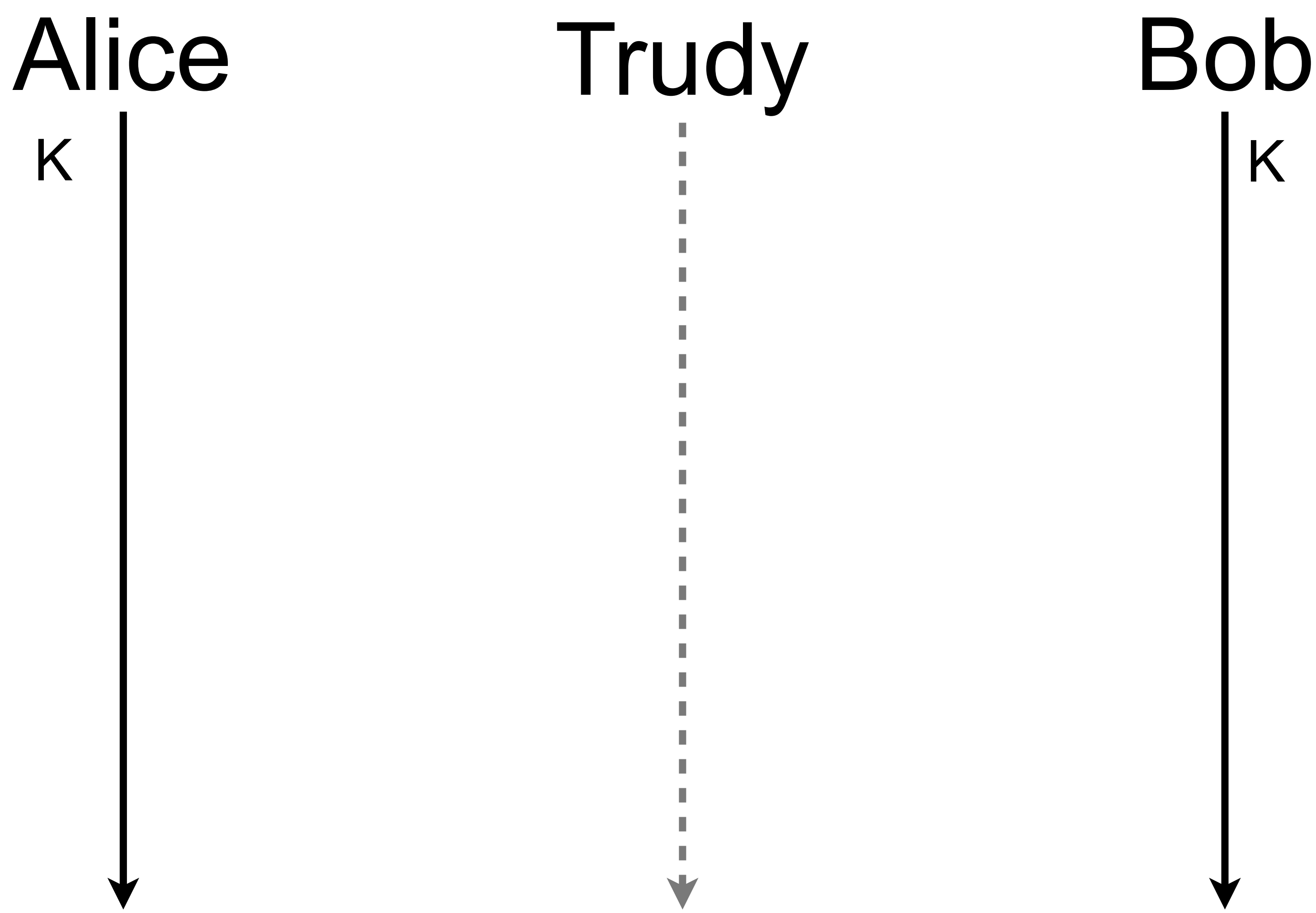
Is that enough?



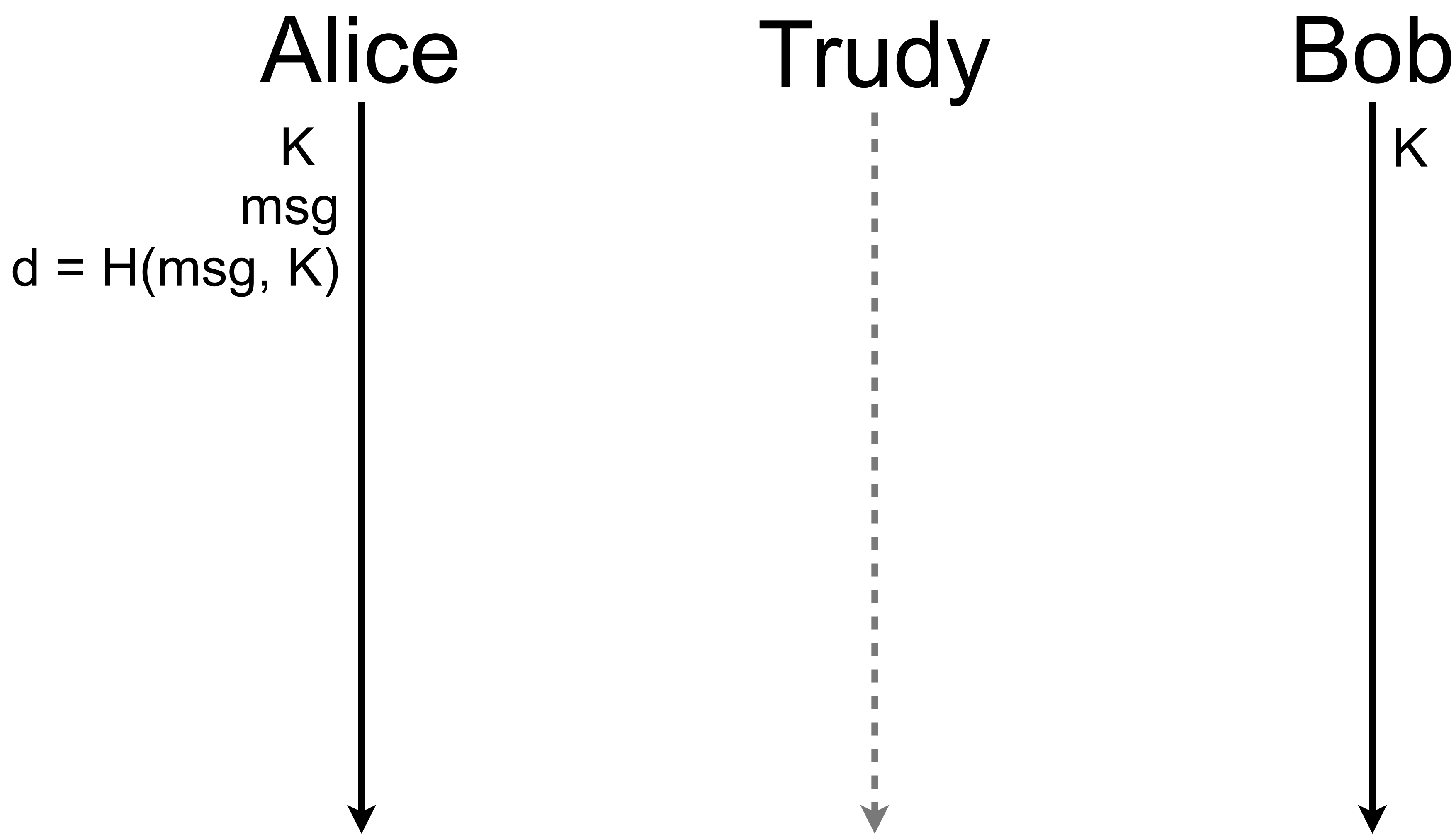
Hash function with salt

- Hash functions are deterministic
- Add a salt such that the output of the hash function is a function of the message and the salt
  - h = H(m, s) where s is the salt or key of the hash function
- As long as Trudy does not know the salt, she can't forge a valid digest

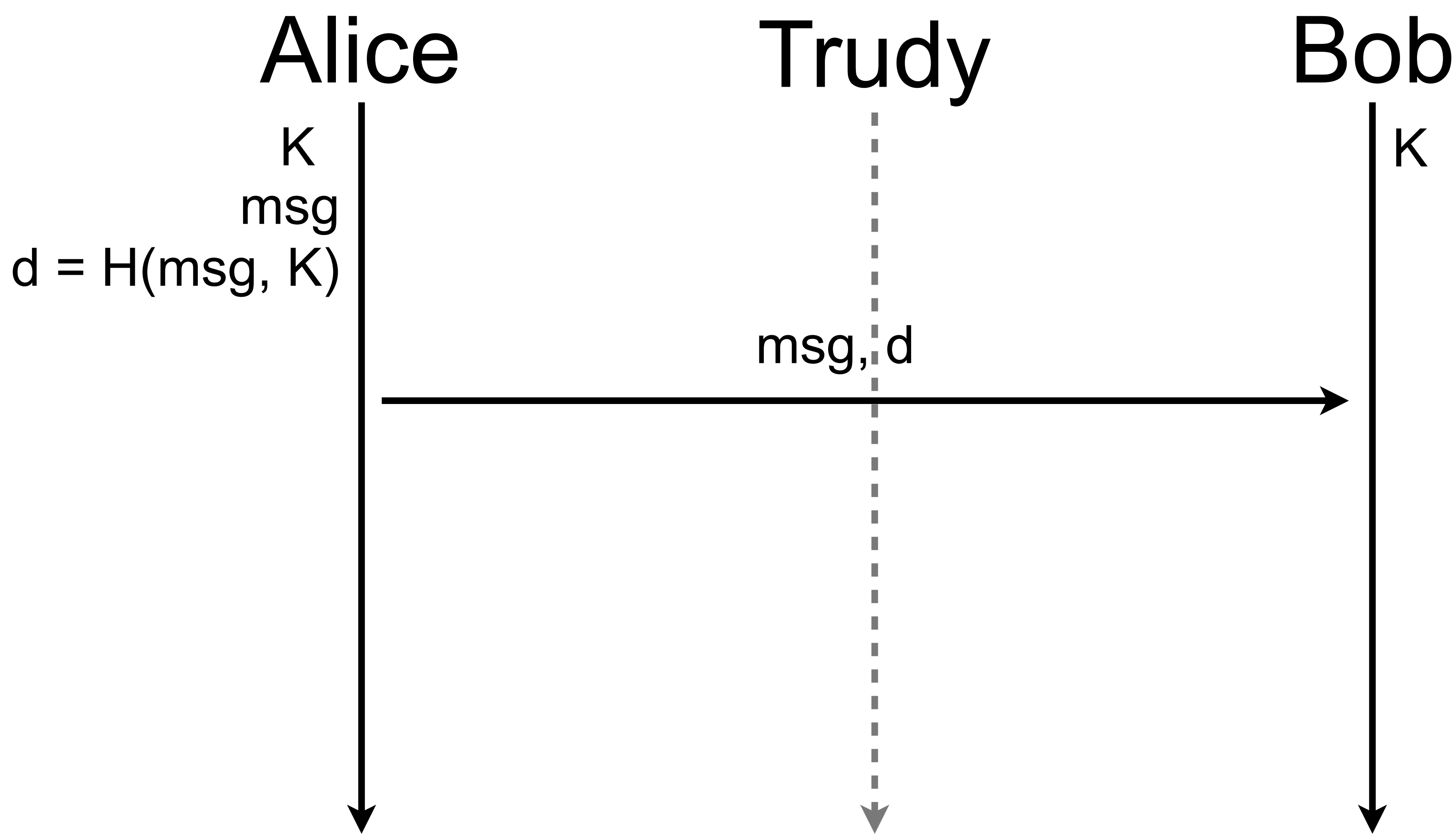
Hash function with salt (contd.)



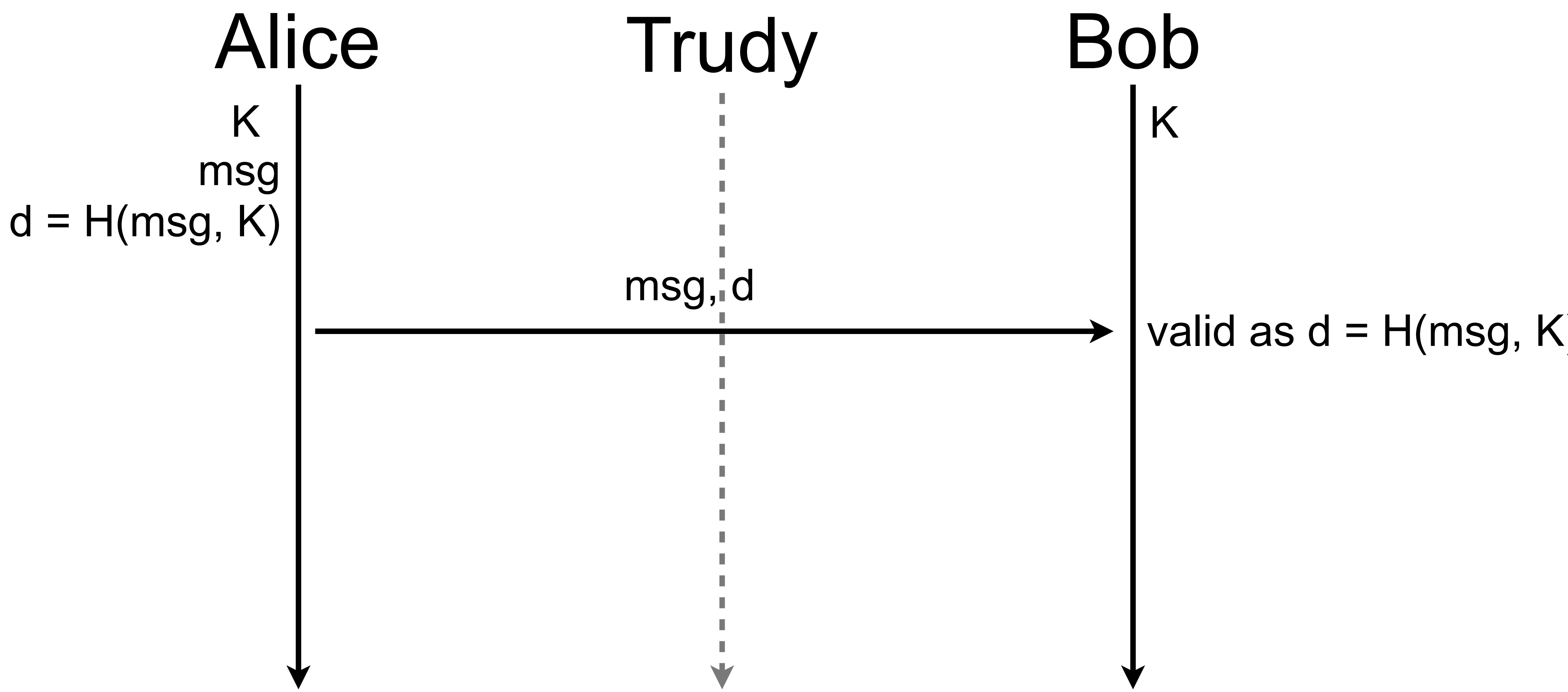
Hash function with salt (contd.)



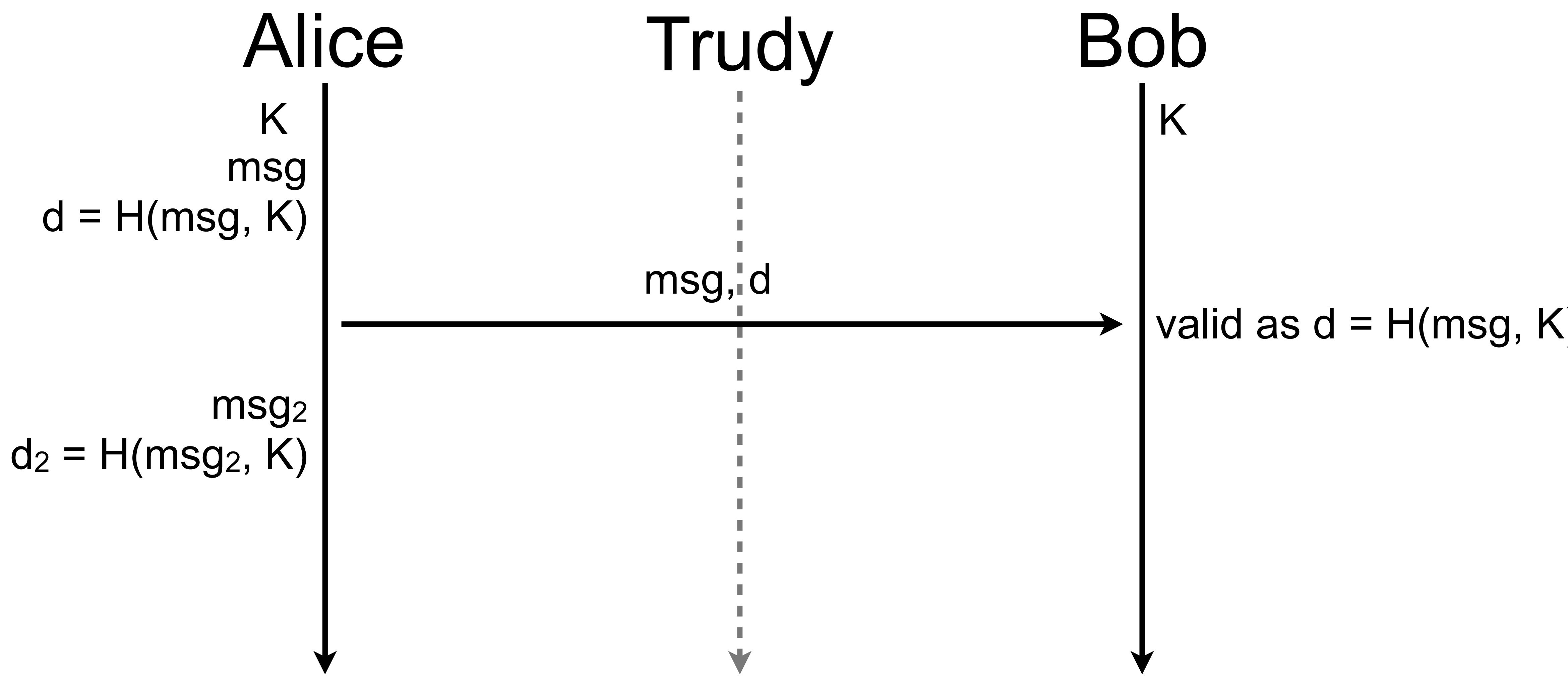
Hash function with salt (contd.)



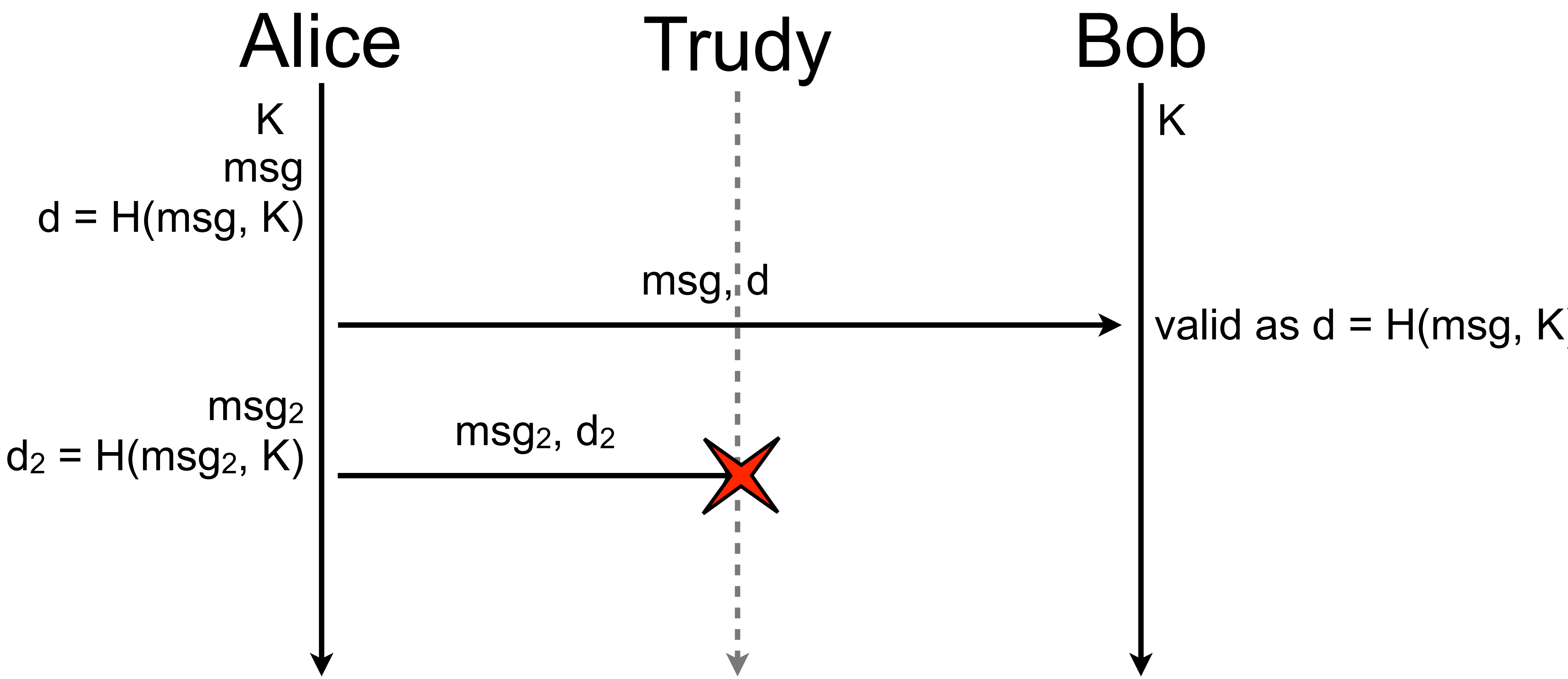
Hash function with salt (contd.)



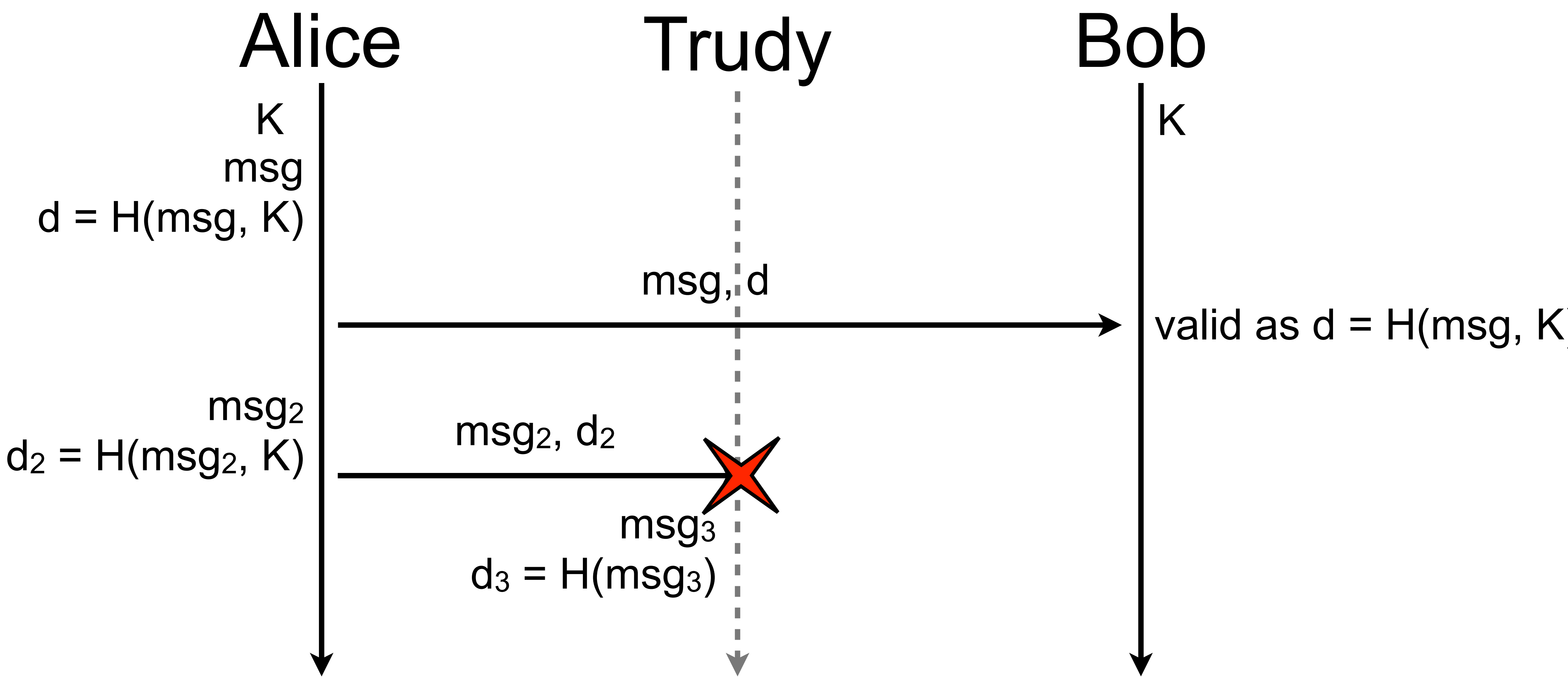
Hash function with salt (contd.)



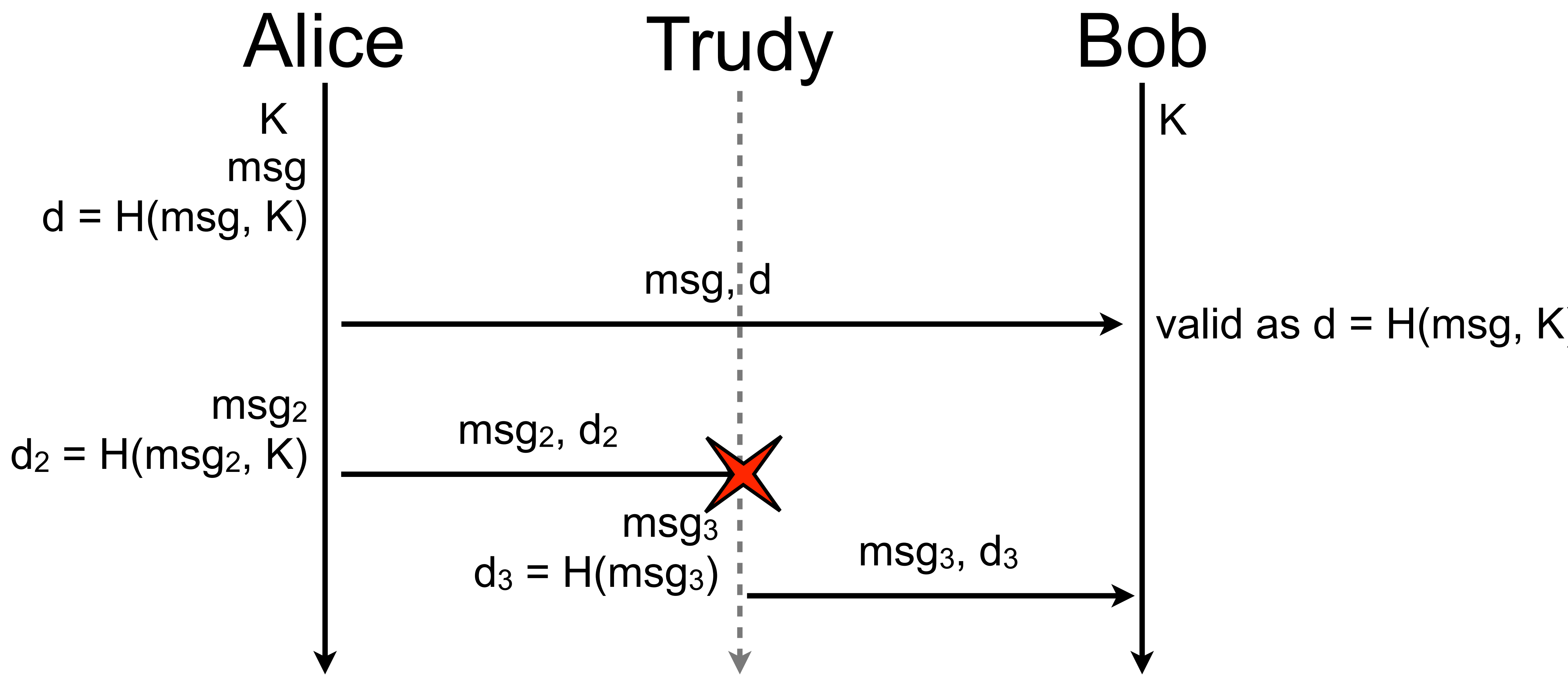
Hash function with salt (contd.)



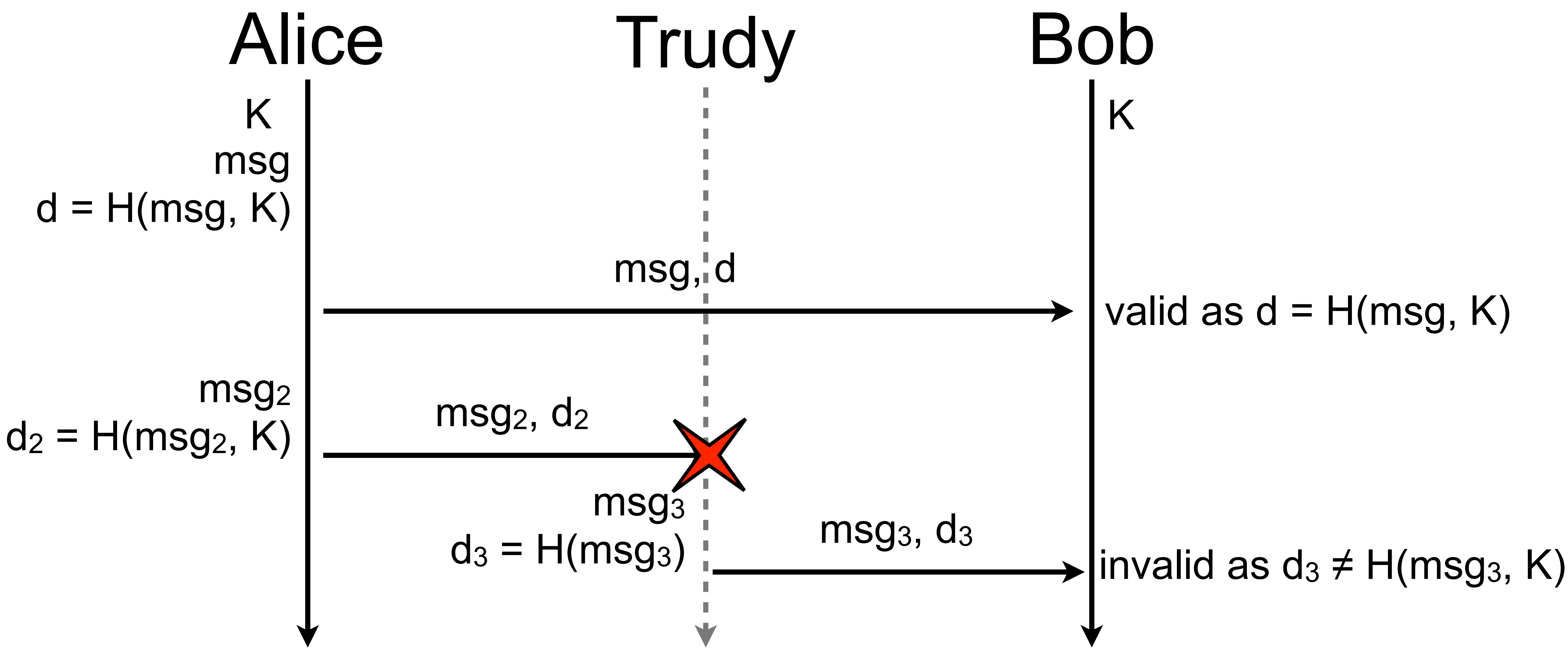
Hash function with salt (contd.)



Hash function with salt (contd.)



Hash function with salt (contd.)



Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

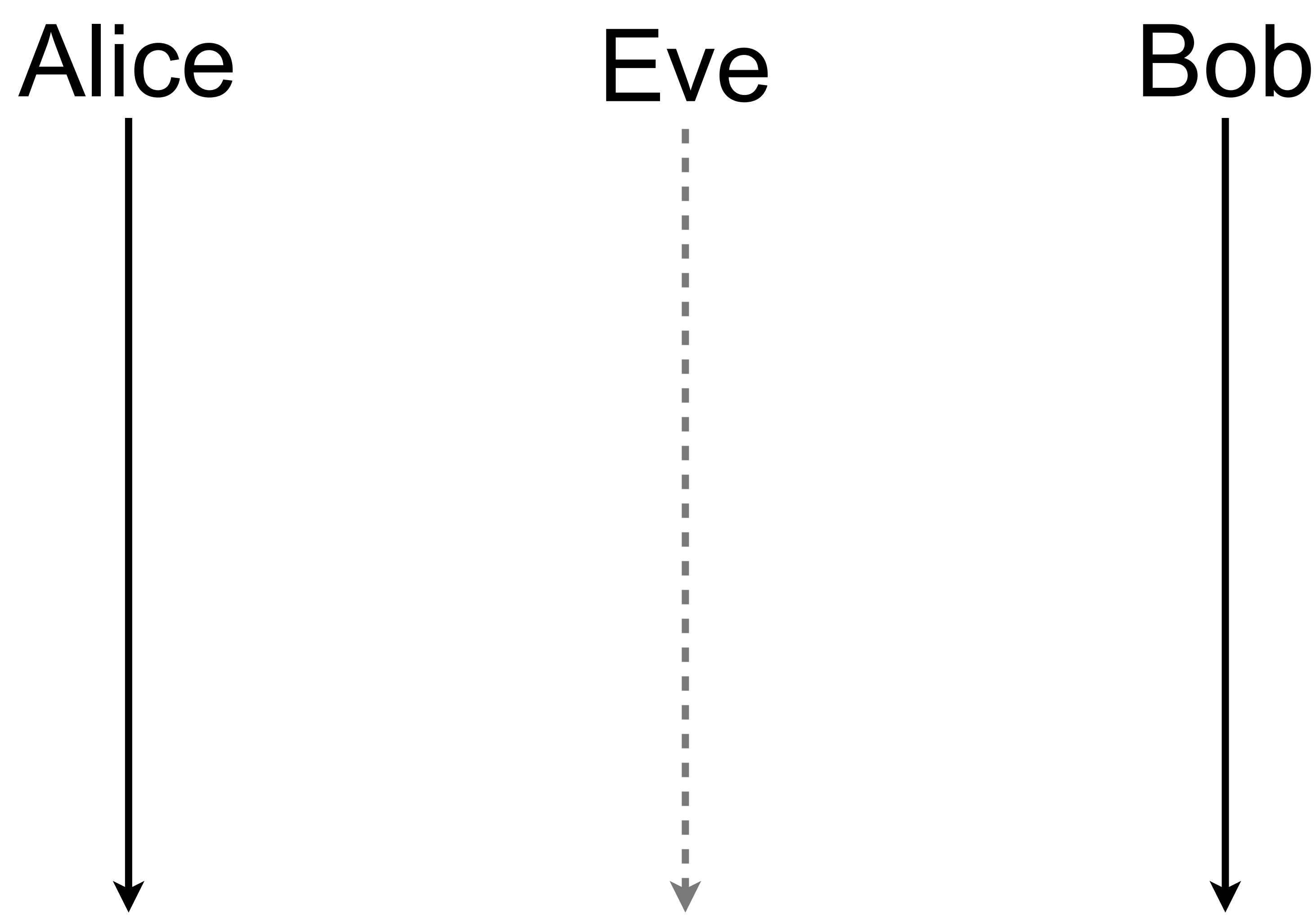
How can Alice and Bob agree on K?

Diffie-Hellman key exchange

- How can Alice and Bob agree on a secret number and be sure that Eve will not discover it?
- Principle
  - do not exchange the secret number but other numbers that are use to build up the secret

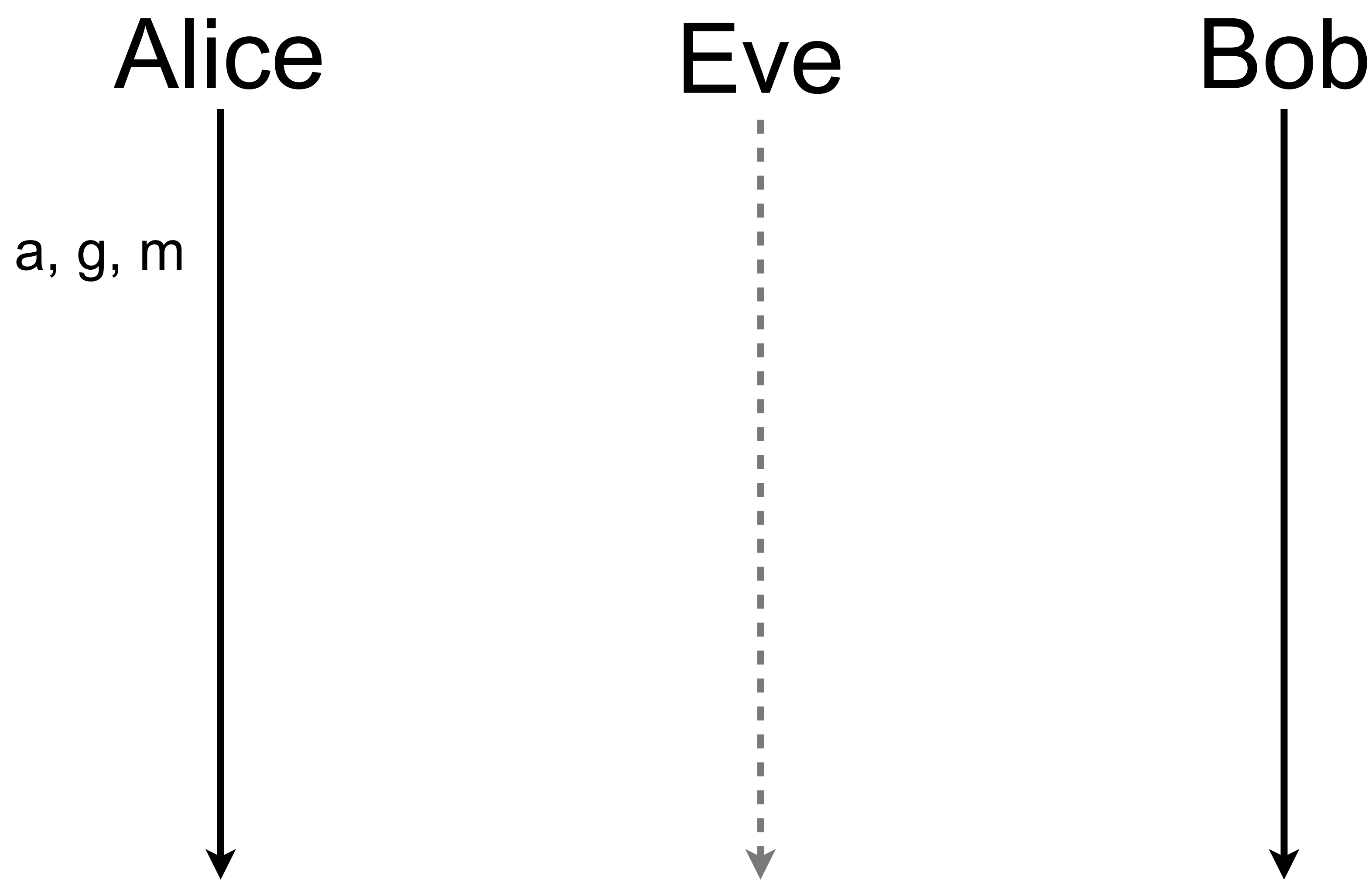
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



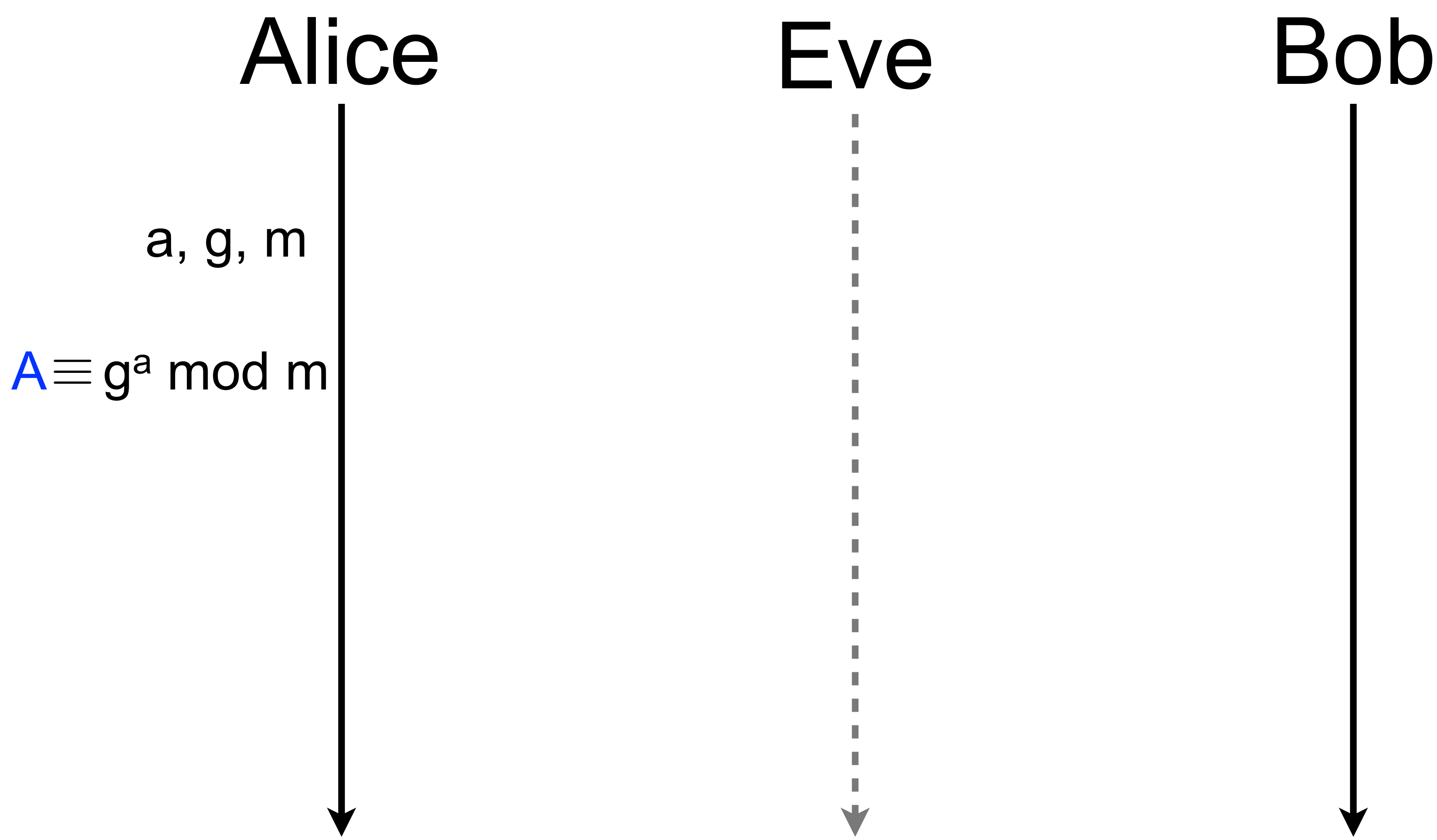
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



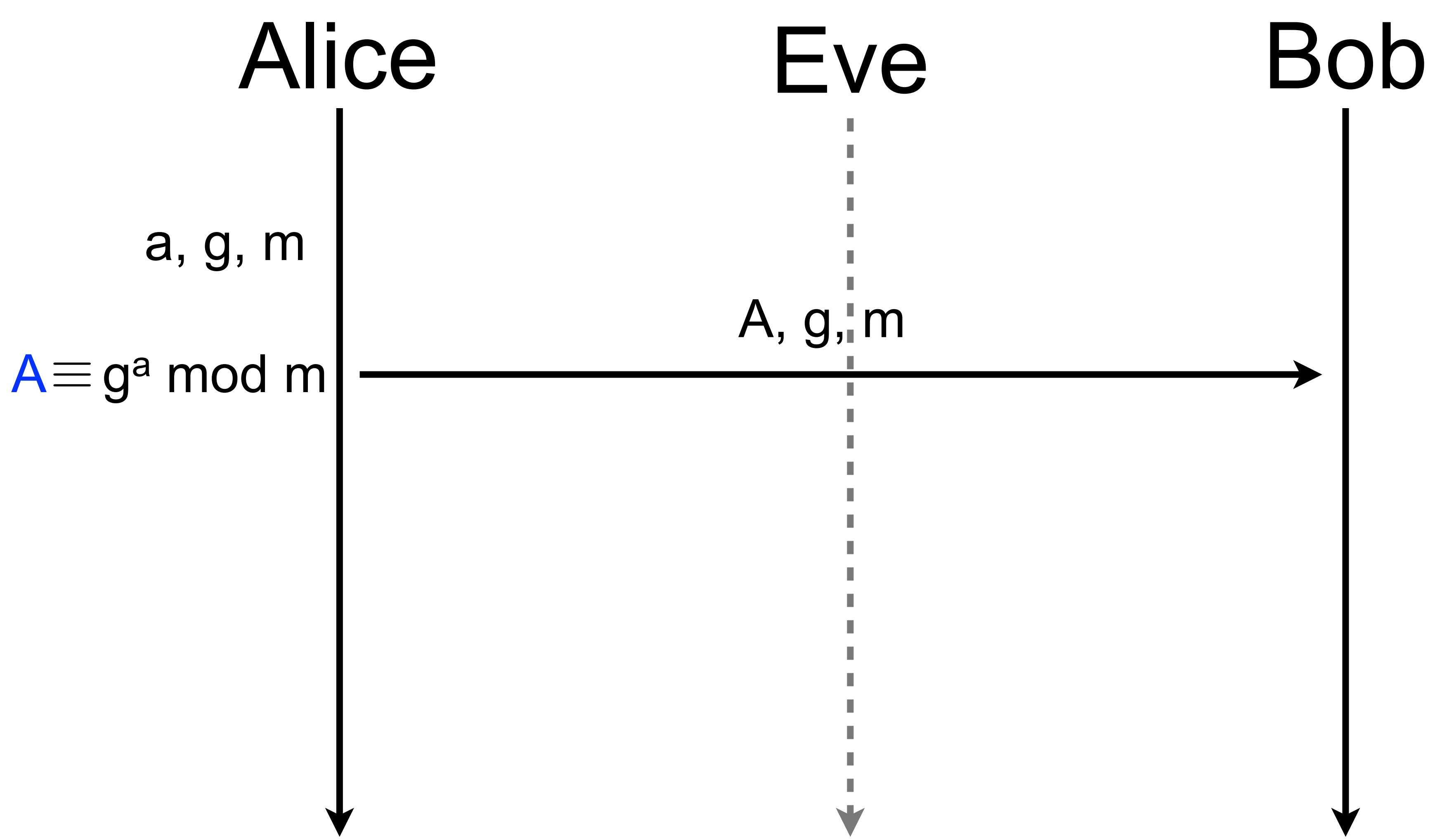
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



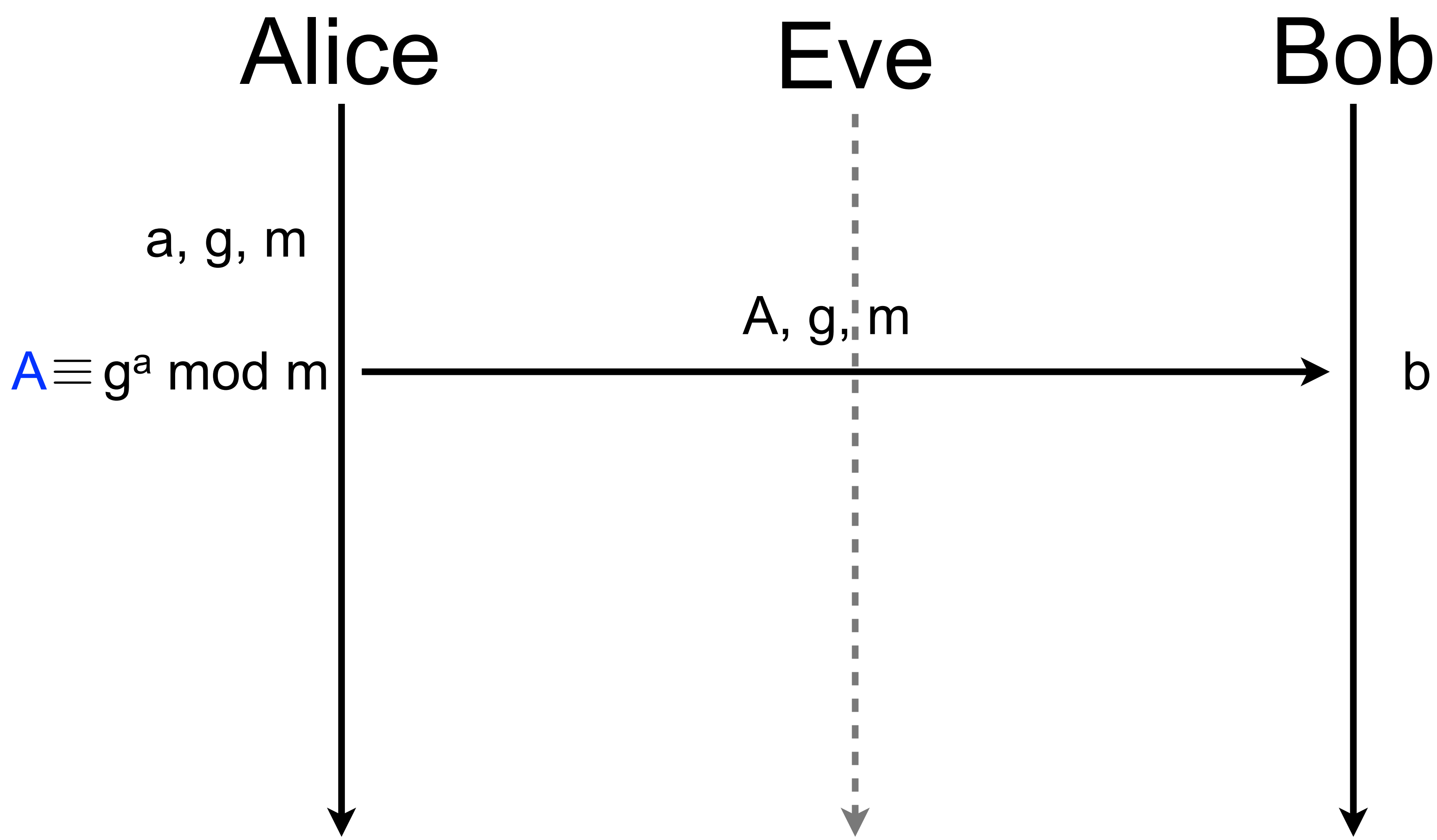
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



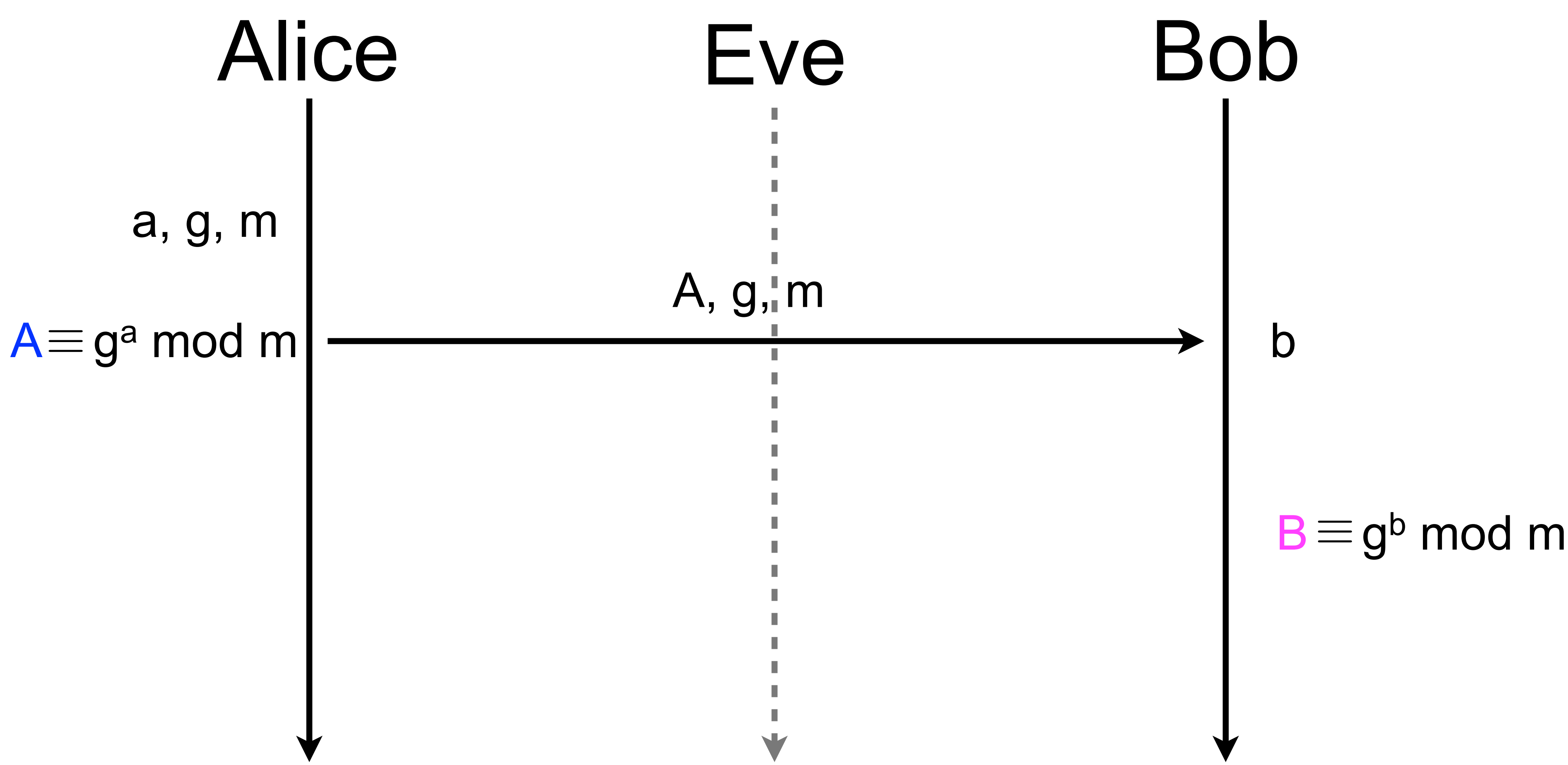
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



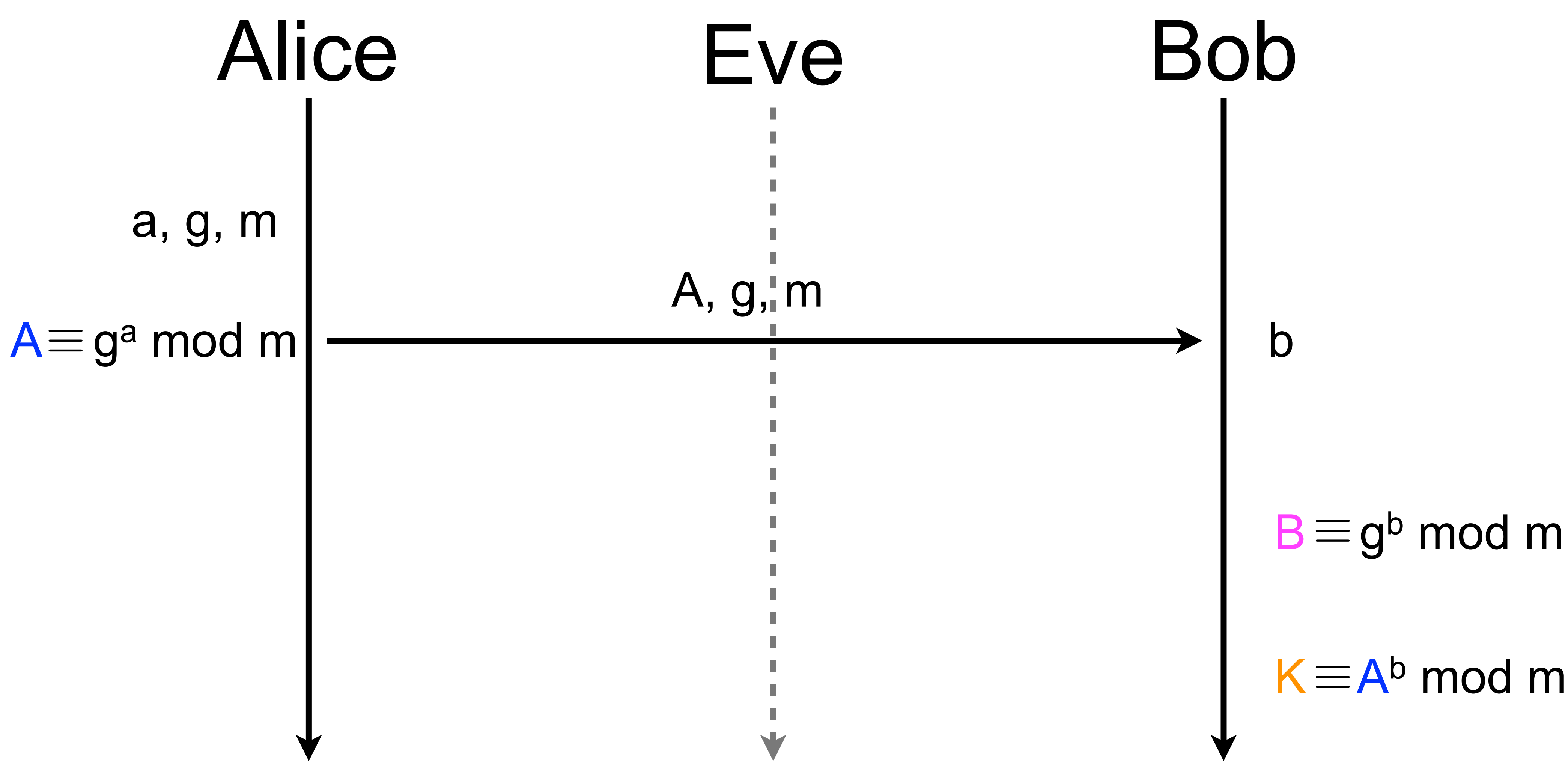
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



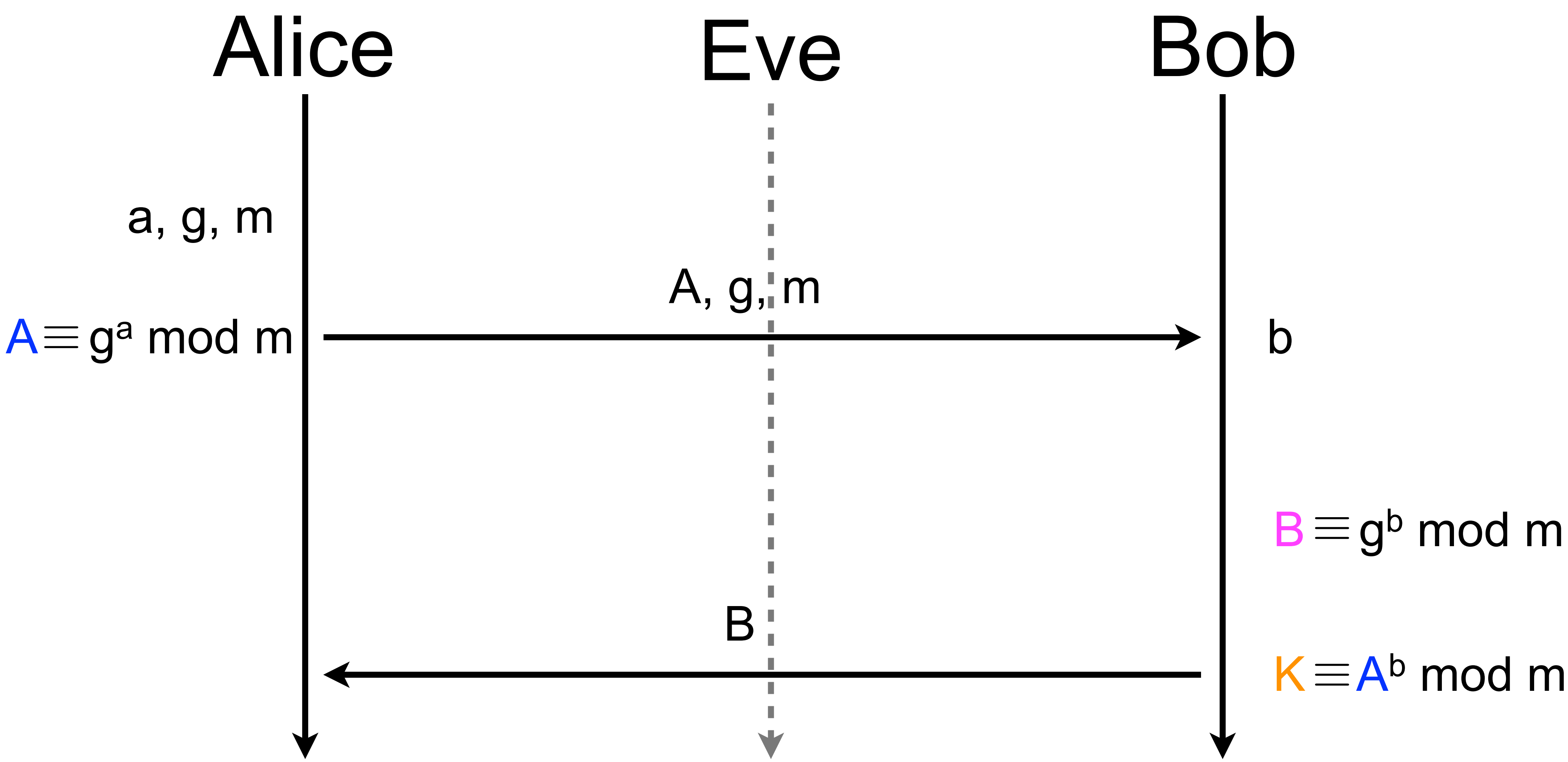
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



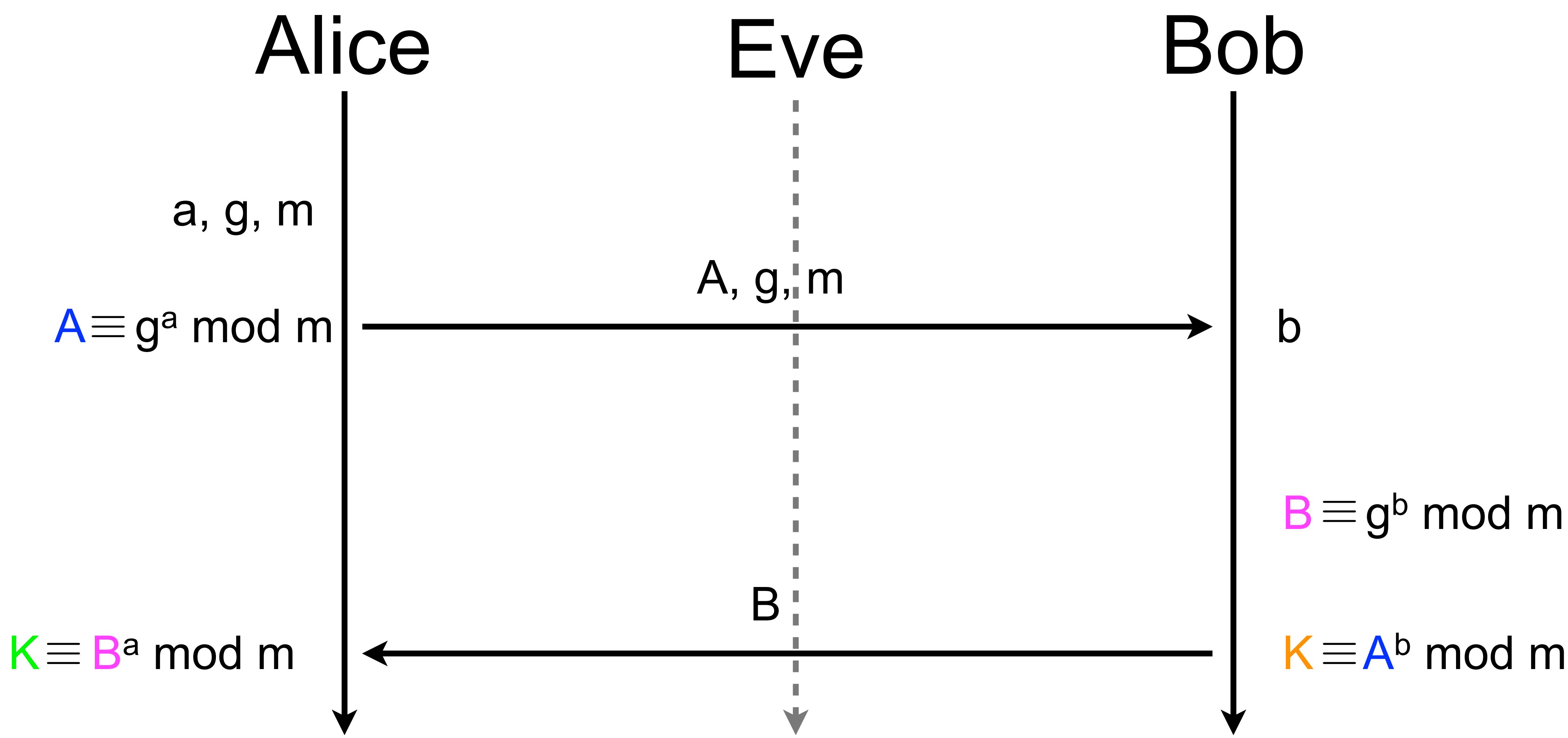
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



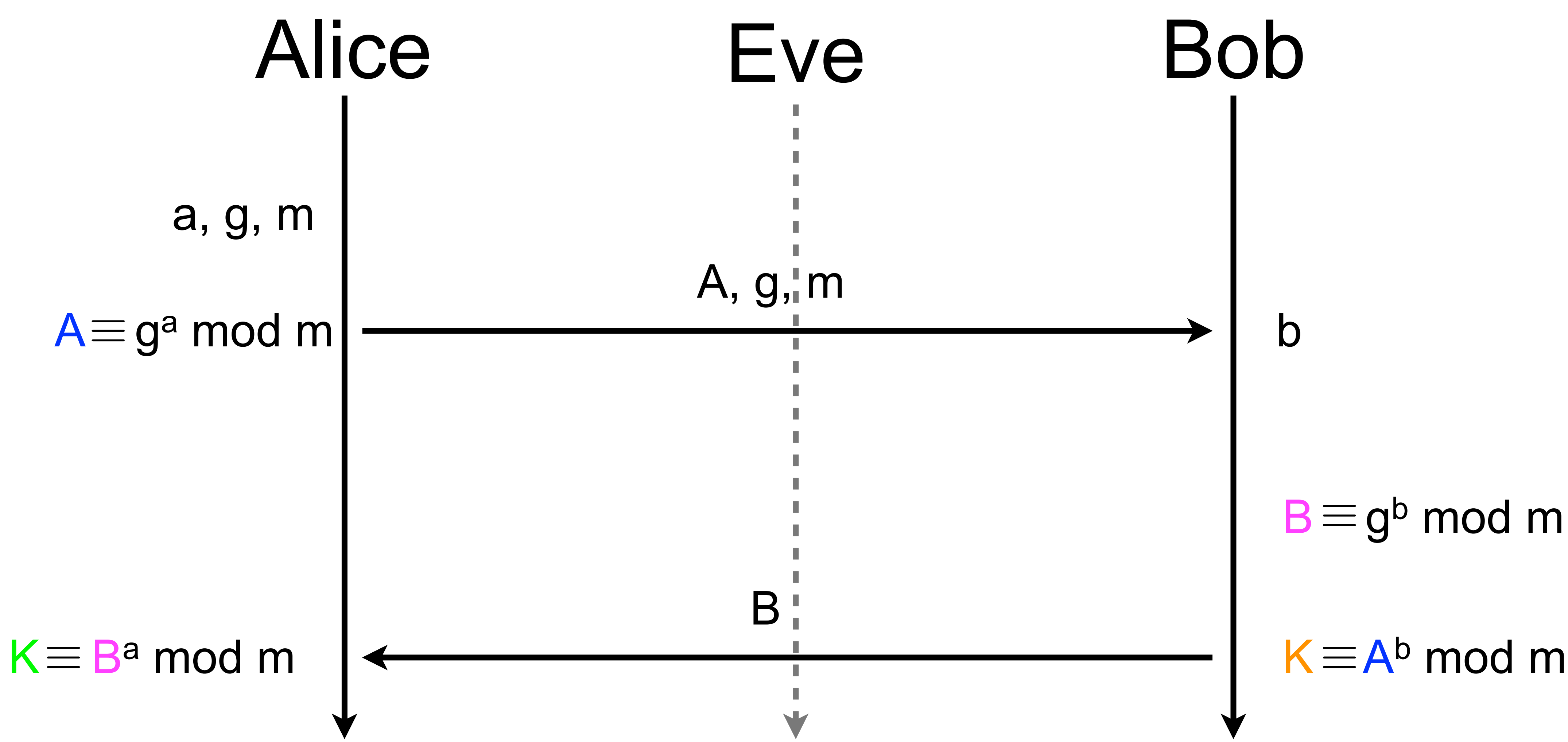
Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



Diffie-Hellman key exchange (contd.)

- Working on finite group and positive integers



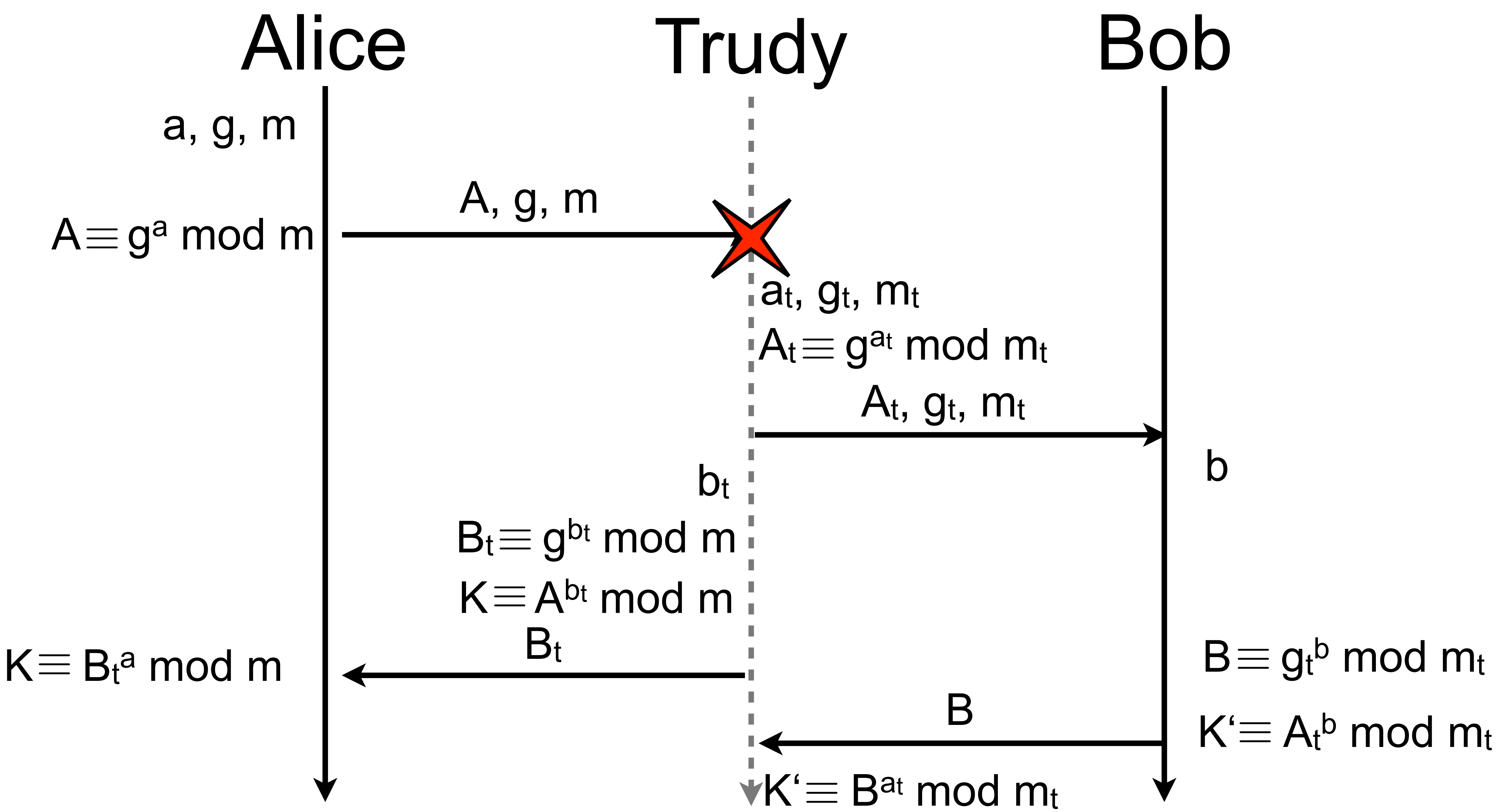
$$K \equiv A^b \bmod m \equiv (g^a \bmod m)^b \bmod p \equiv g^{ba} \bmod m \equiv (g^b \bmod m)^a \bmod m \equiv B^a \bmod m \equiv K$$

Diffie-Hellman key exchange (contd.)

- Why can't Eve guess K if she knows A, B, g, and m?
  - discrete exponentiation is linear with the size of the argument
    - easy to compute $x \equiv y^z \bmod p$
  - but for some discrete groups, no efficient algorithm is known to compute discrete logarithm
    - hard to determine natural z that ensures $x \equiv y^z \bmod p$
  - Eve knows A, B, g, and m but can't determine neither **a** nor **b** that are absolutely necessary to compute K
    - $K \equiv A^{\mathbf{b}} \bmod m \equiv (g^{\mathbf{a}} \bmod m)^{\mathbf{b}} \bmod p \equiv g^{\mathbf{ba}} \bmod m$
      $\equiv (g^{\mathbf{b}} \bmod m)^{\mathbf{a}} \bmod m \equiv B^{\mathbf{a}} \bmod m$

Diffie-Hellman key exchange (contd.)

- Trudy can break Diffie-Hellman



Diffie-Hellman key exchange (contd.)

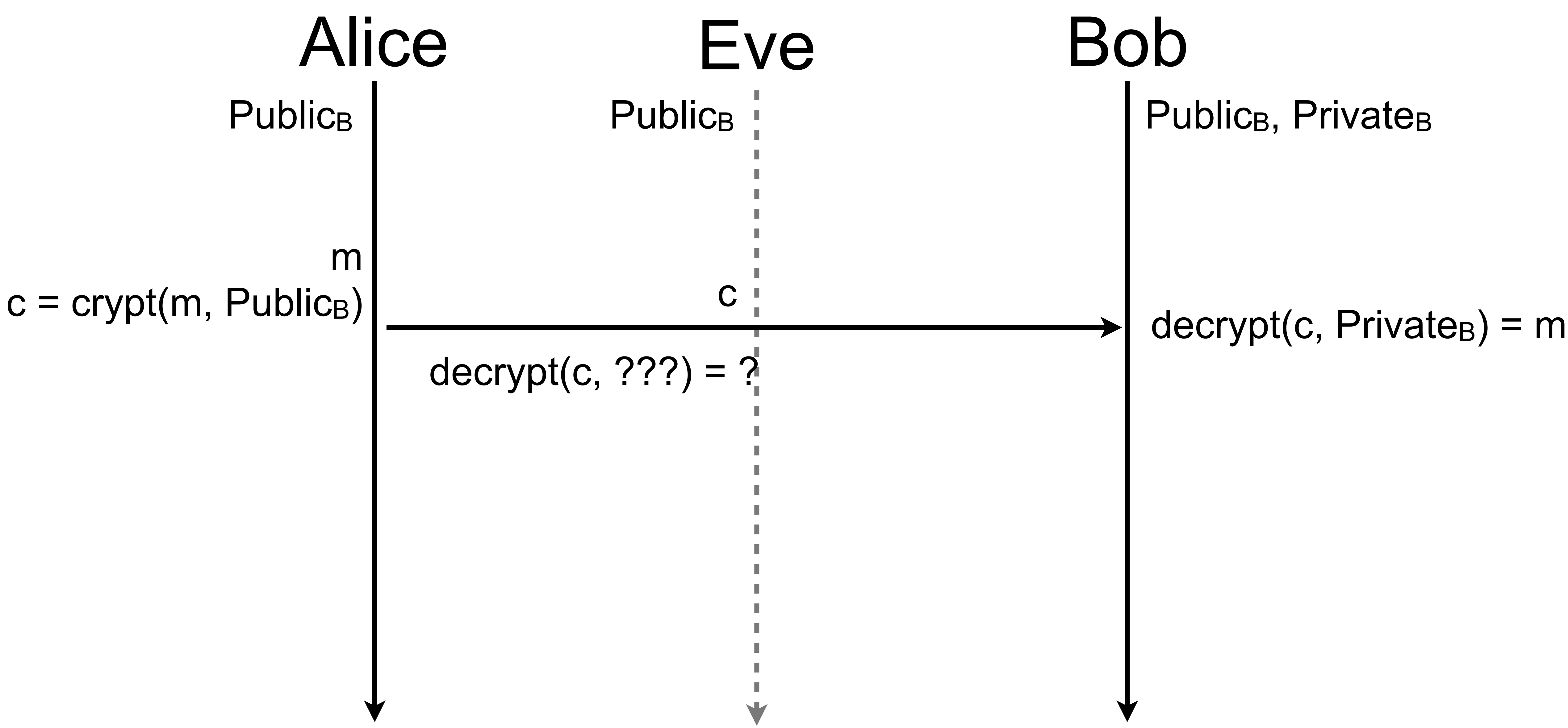
- How can we protect Diffie-Hellman from Trudy?
- Principle
  - Alice and Bob sign the messages exchanged in Diffie-Hellman (?!)

Asymmetric cryptography

- In asymmetric cryptography (aka public-key cryptography), two keys are used
    - public key
        - publicly available to anybody (even attackers)
        - used to encrypt a message
    - private key
        - known only by the legitimate owner of the public key
        - used to decrypt a message
- e.g., RSA, PGP, Diffie-Hellman
- Public-key cryptography is 10 to 100 times slower than symmetric-key cryptography
    - seldom (never?) used to encrypt communications

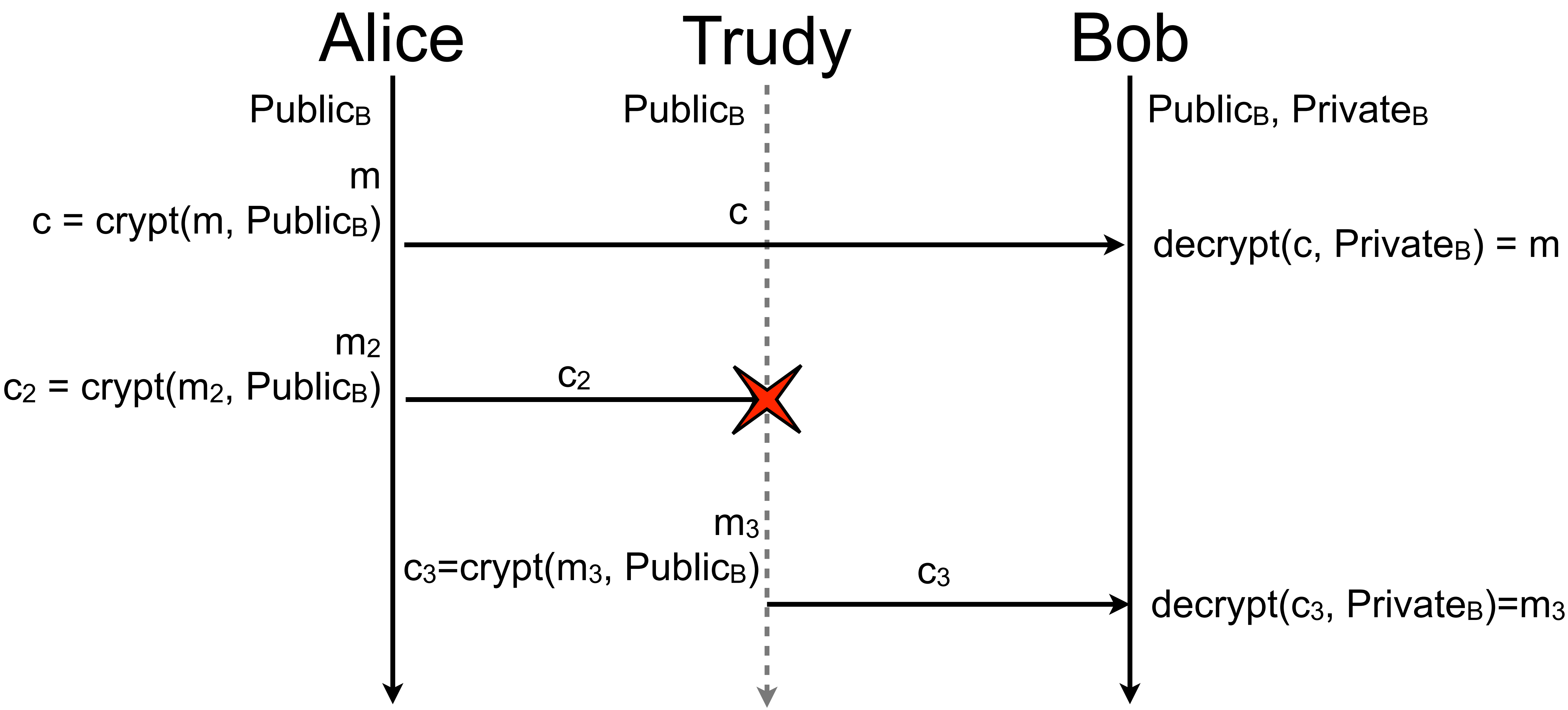
Asymmetric cryptography (contd.)

- Eve cannot determine the message



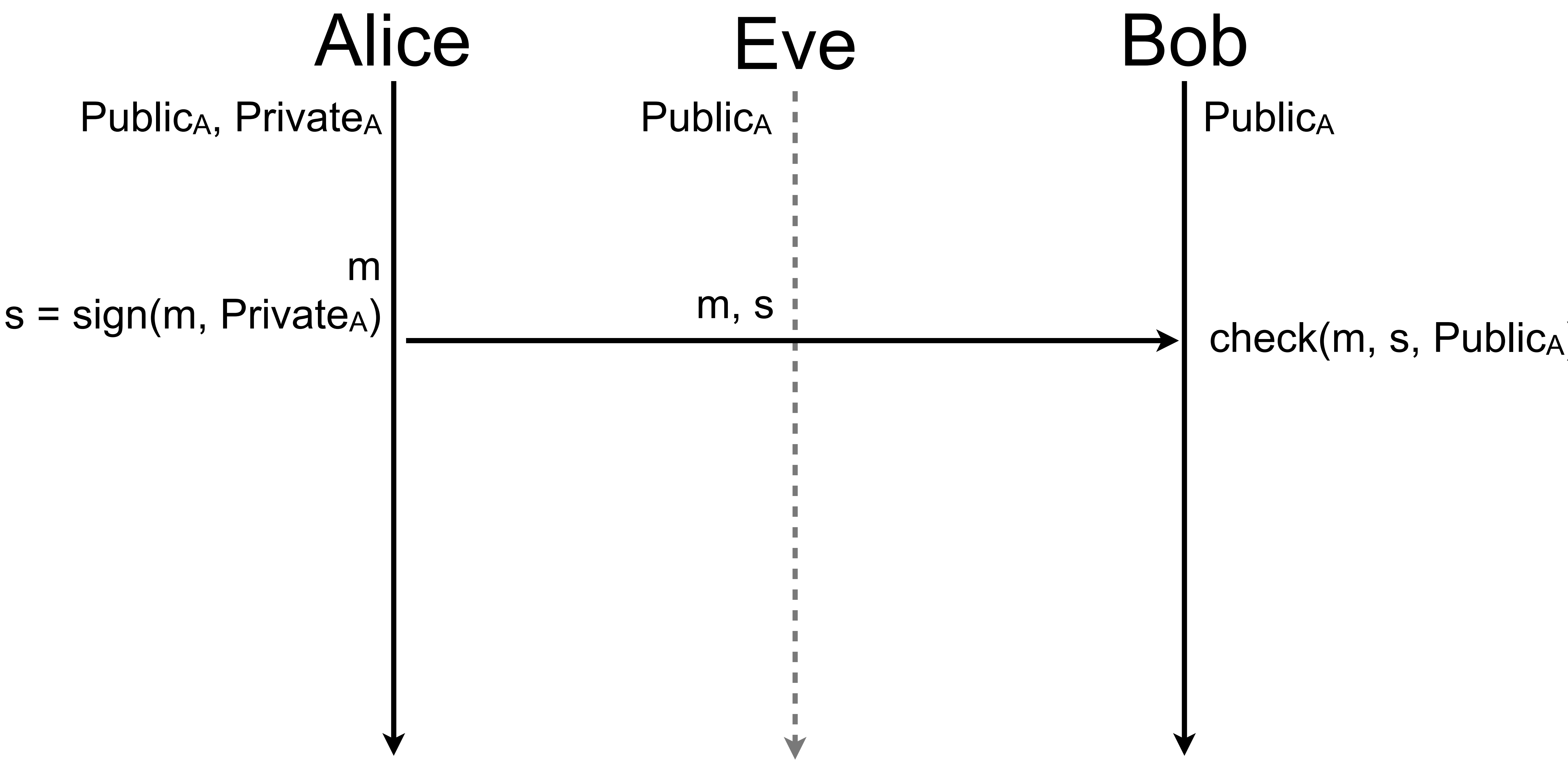
Asymmetric cryptography (contd.)

- Trudy can send a forged message



Asymmetric cryptography (contd.)

- Eve can read the message

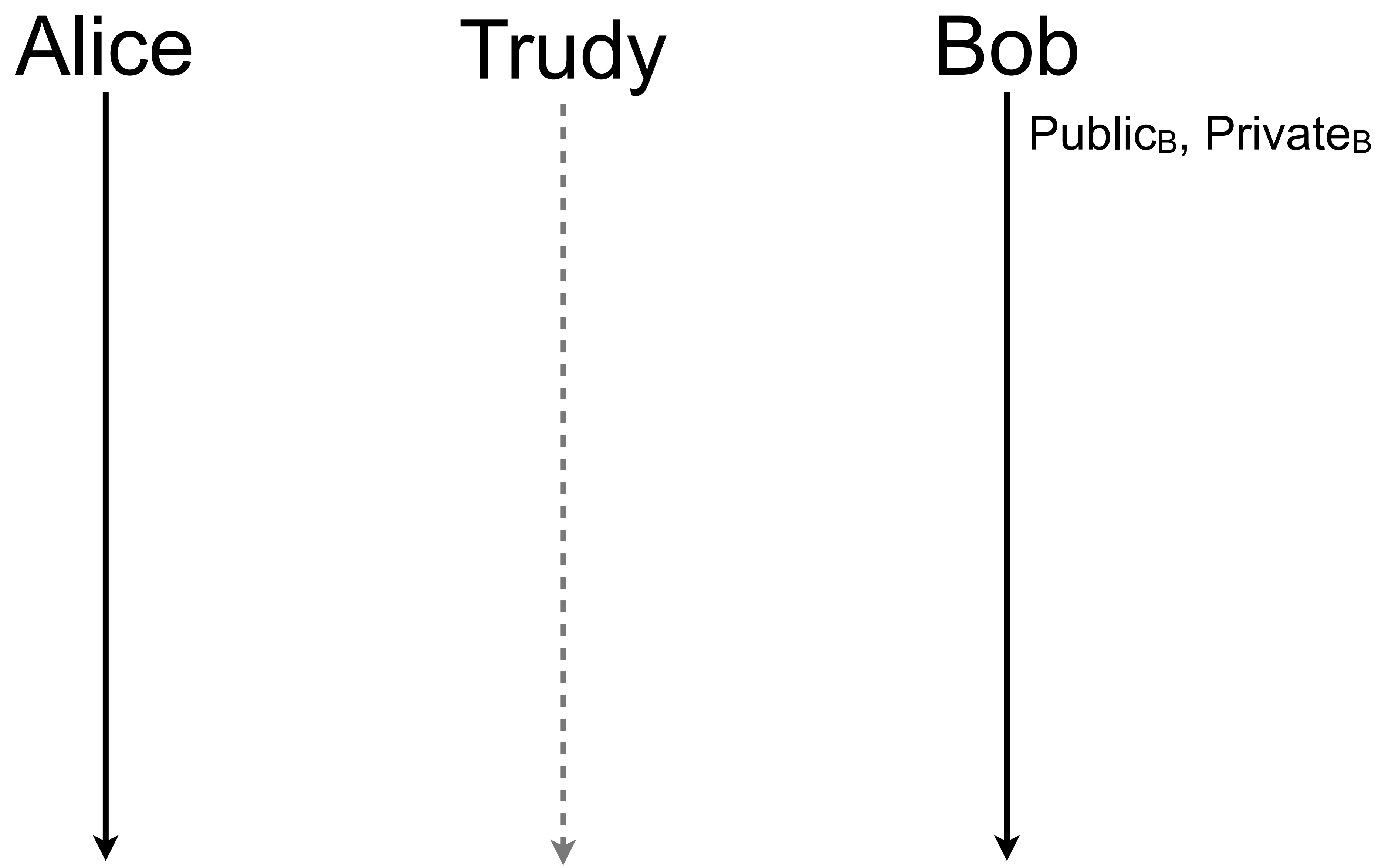


How to build sign and check?

- s = sign(m, k) = crypt(H(m), k)
- check(m, s, K) = (H(m)==decrypt(s, K))
  - where k is the private key of the signer and K is the public key
- Asymmetric cryptography is slow and m can be large
  - encrypting m would be too costly
  - solution: consider the digest of m while signing

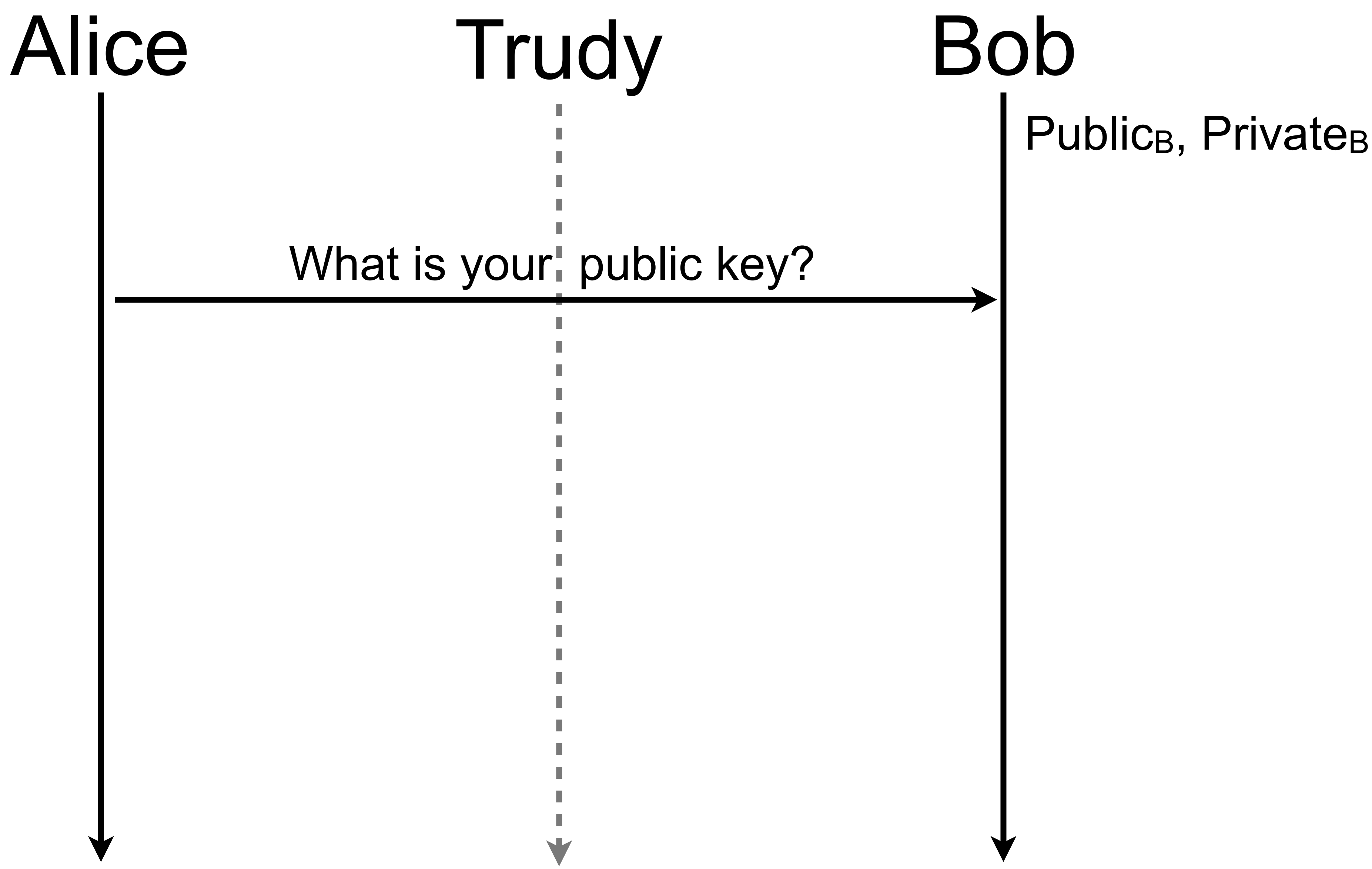
Public key infrastructure

- How to safely obtain Bob's public key?



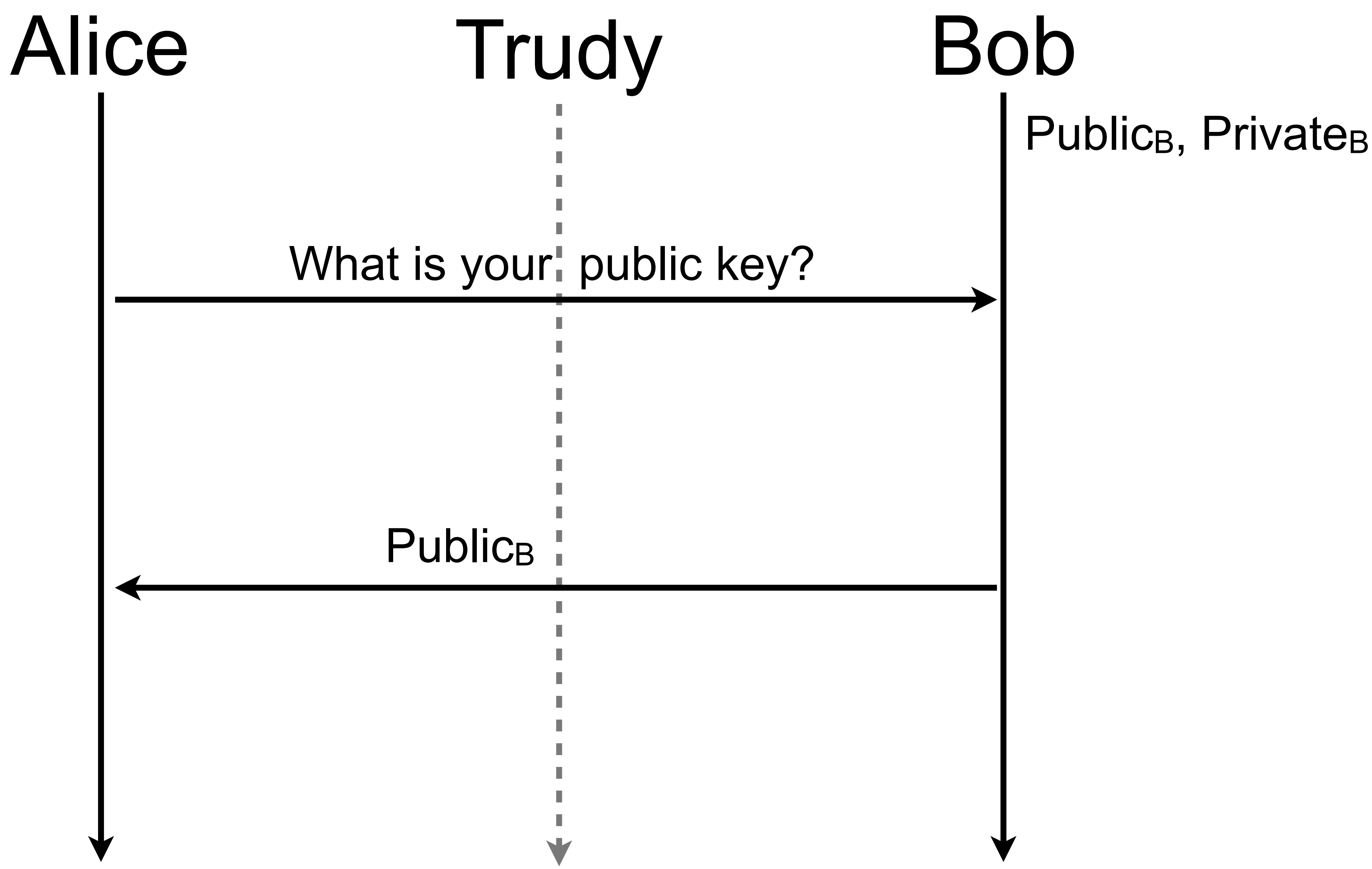
Public key infrastructure

- How to safely obtain Bob's public key?



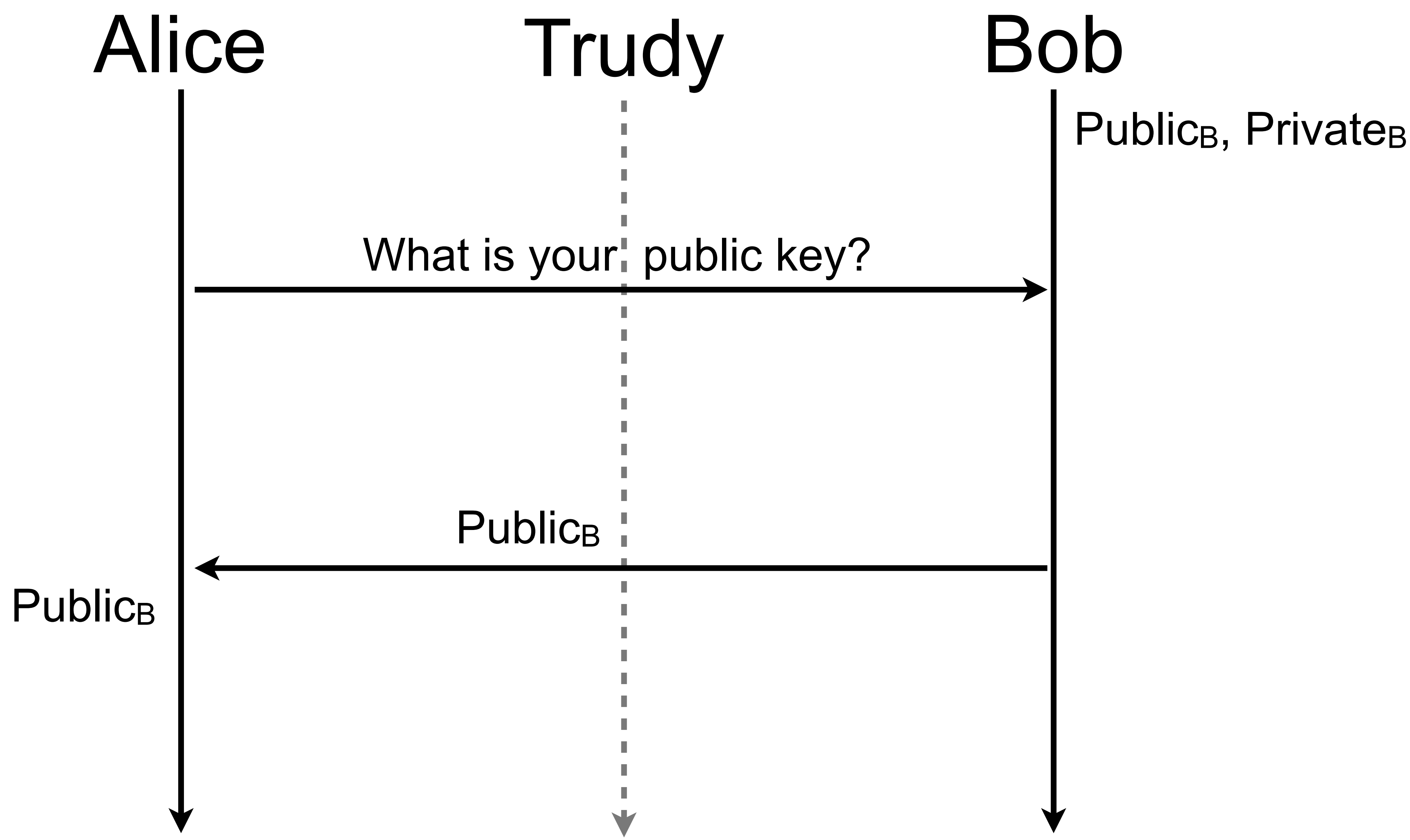
Public key infrastructure

- How to safely obtain Bob's public key?



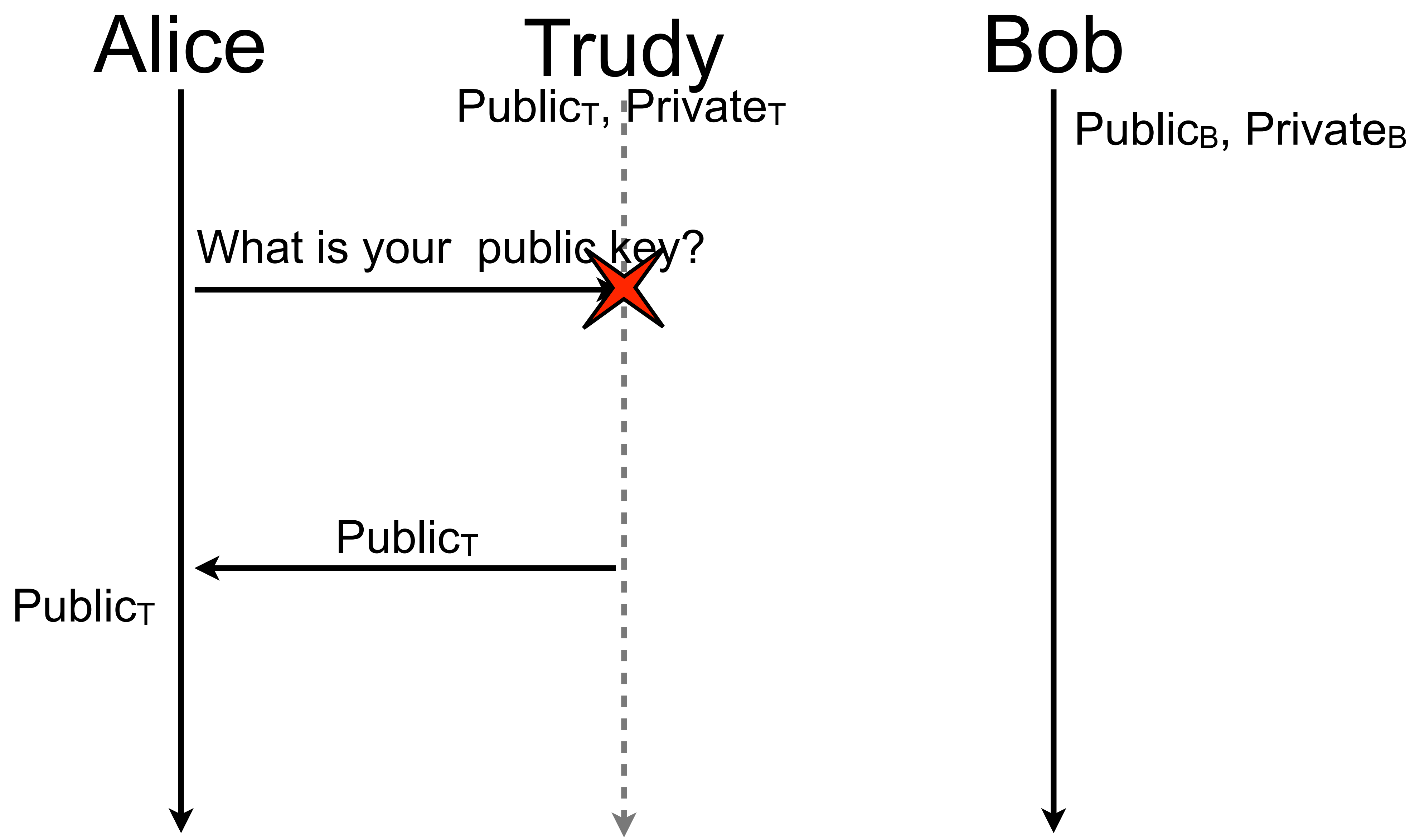
Public key infrastructure

- How to safely obtain Bob's public key?



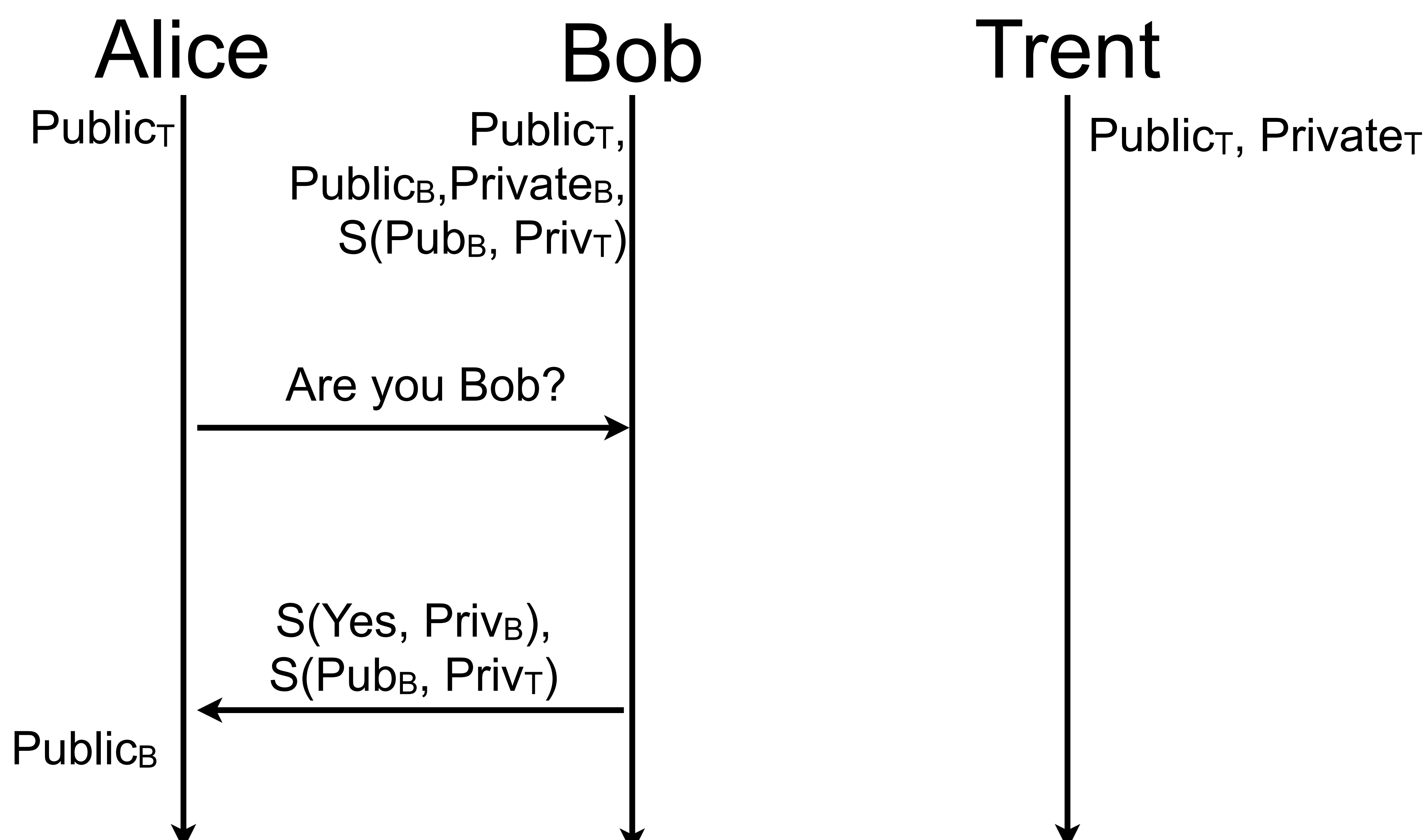
Public key infrastructure (contd.)

- Trudy can send a forged key



Public key infrastructure (contd.)

- Alice and Bob trust a third party (e.g., Trent) for authentication



Public key infrastructure (contd.)

- Practically, Bob sends a certificate (e.g., X.509), not only its public key and signature
- A certificate provides many information to be able to correctly identify and authenticate its subject (e.g., Bob)
  - the subject name and organization
  - the subject public key (and type)
  - the issuer name and organization
  - the certificate validity time (valid not before and not after)
  - the certificate signature and type, signature made by the issuer of the certificate
  - ...

Public key infrastructure (contd.)

- A certificate signed with the private key of the public key indicated into the certificate is said self-signed
  - prove nothing except that the issuer knows the private key of the subject
- Certificates can be chained, the subject is certified by its issuer, the issuer itself is certified by its own issuer, and so on until the root of the certification is reach
  - when a certificate is not self-signed, it indicates the chain of certificates used for its authentication
- The entity verifying the certificates backtracks the chain of certificate until is reaches the certificate of a certification authority (CA) he knows
- Trusted parties are installed separately (e.g., hardcoded, during OS updates)
  - assumption: the trusted party is not compromised

Public key infrastructure (contd.)

- Certificates are issued once and valid during a given time period, whatever the number of time it is used
- What if the subjects leaves its organization? The private key of the subject is stolen? The private key of the issuer is stolen?
- Keys are selected big enough to not be broken during validity time
- When a certified key is compromised, the certificate is revoked
  - the issuer maintains the list of revoked certificates
    - when a certificate is checked for validity, the verifying client should verify that the certificate is not in the revoked certificates list

Public key infrastructure (contd.)

- *"A public key infrastructure is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates" [1]*
- A certificate $Cert_1$ issued by a CA can be used to certify any certificate $Cert_2$
  - $Cert_2$ is authenticated if
    check($Cert_2$, $Cert_2$.signature, $Cert_2$.issuer.public_key) &
    check($Cert_1$, $Cert_1$.signature, $Cert_1$.*issuer*.public_key) &
    $Cert_2$ not in $Cert_2$.issuer.revoke_list &
    $Cert_1$ not in $Cert_1$.issuer.revoke_list
    - where $Cert_2$.issuer is identified with $Cert_1$ and $Cert_1$.issuer is identified by CA's certificate
    - assuming that the verifier knows CA's certificate

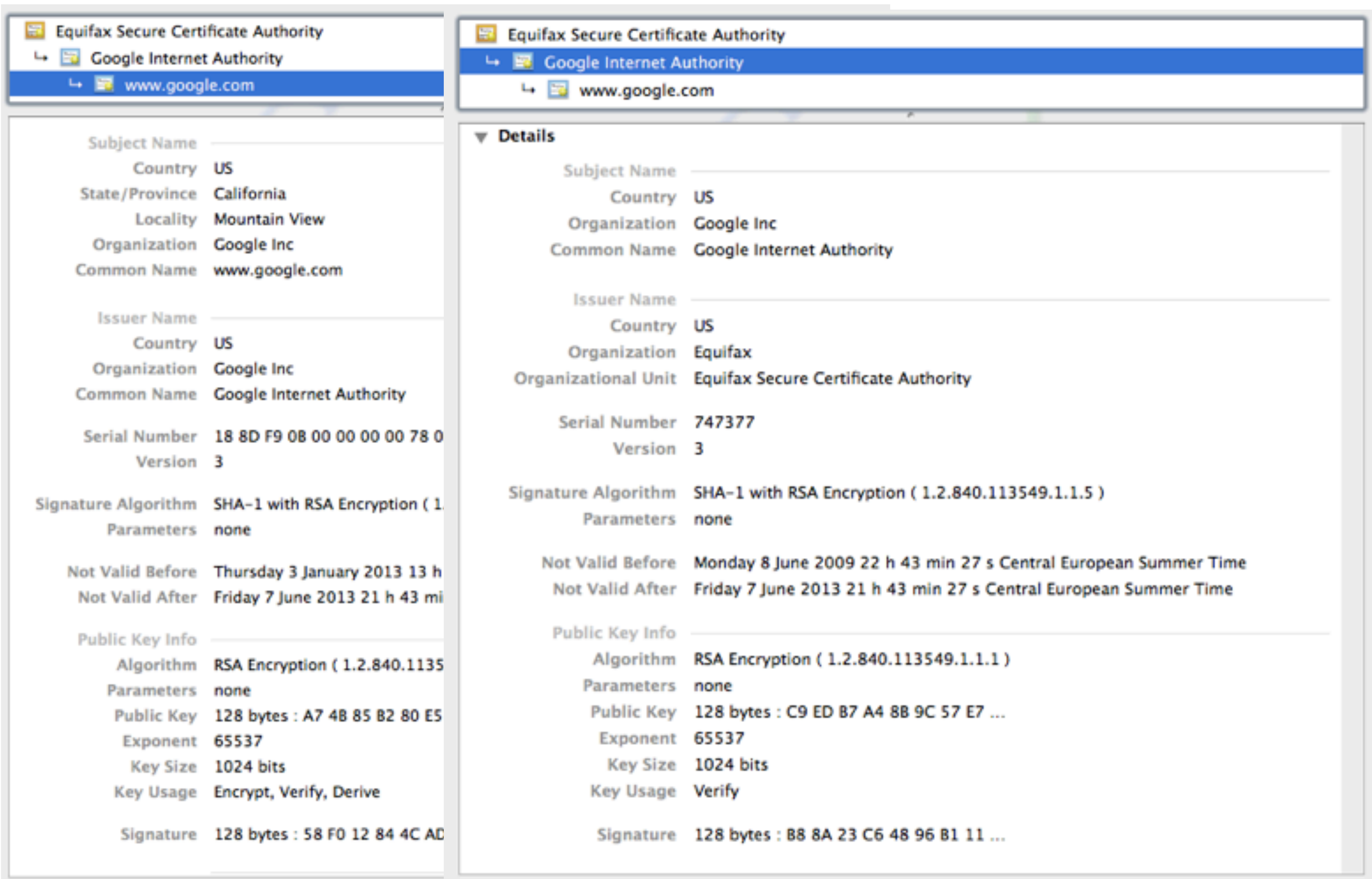
[1] http://en.wikipedia.org/wiki/Public-key_infrastructure

Public key infrastructure (contd.)

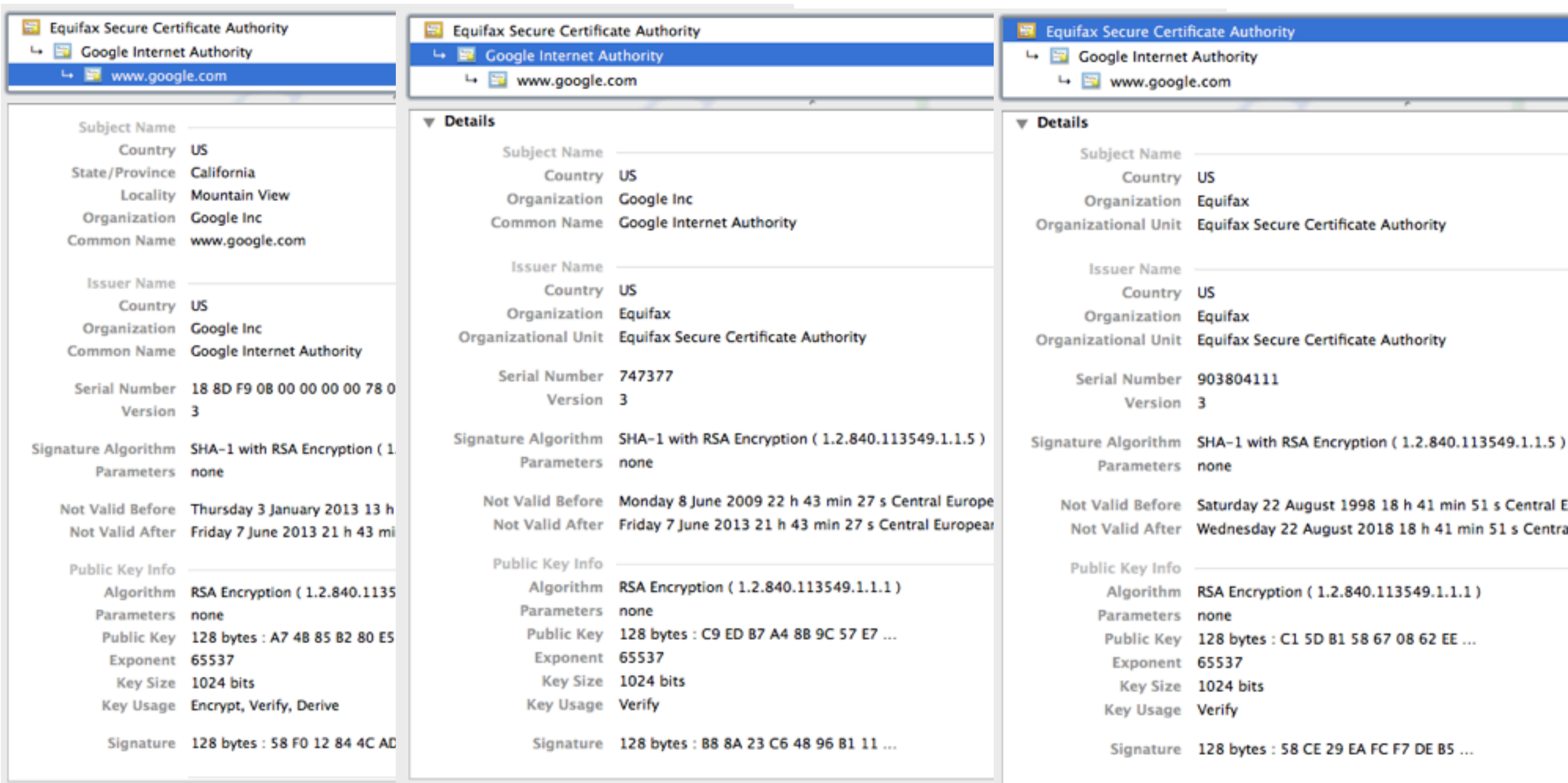
Equifax Secure Certificate Authority
↳ Google Internet Authority
↳ www.google.com

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	www.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority
Serial Number	18 8D F9 0B 00 00 00 00 78 0B
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Thursday 3 January 2013 13 h 15 min 52 s Central European Standard Time
Not Valid After	Friday 7 June 2013 21 h 43 min 27 s Central European Summer Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	128 bytes : A7 4B 85 B2 80 E5 94 03 ...
Exponent	65537
Key Size	1024 bits
Key Usage	Encrypt, Verify, Derive
Signature	128 bytes : 58 F0 12 84 4C AD A2 7E ...

Public key infrastructure (contd.)

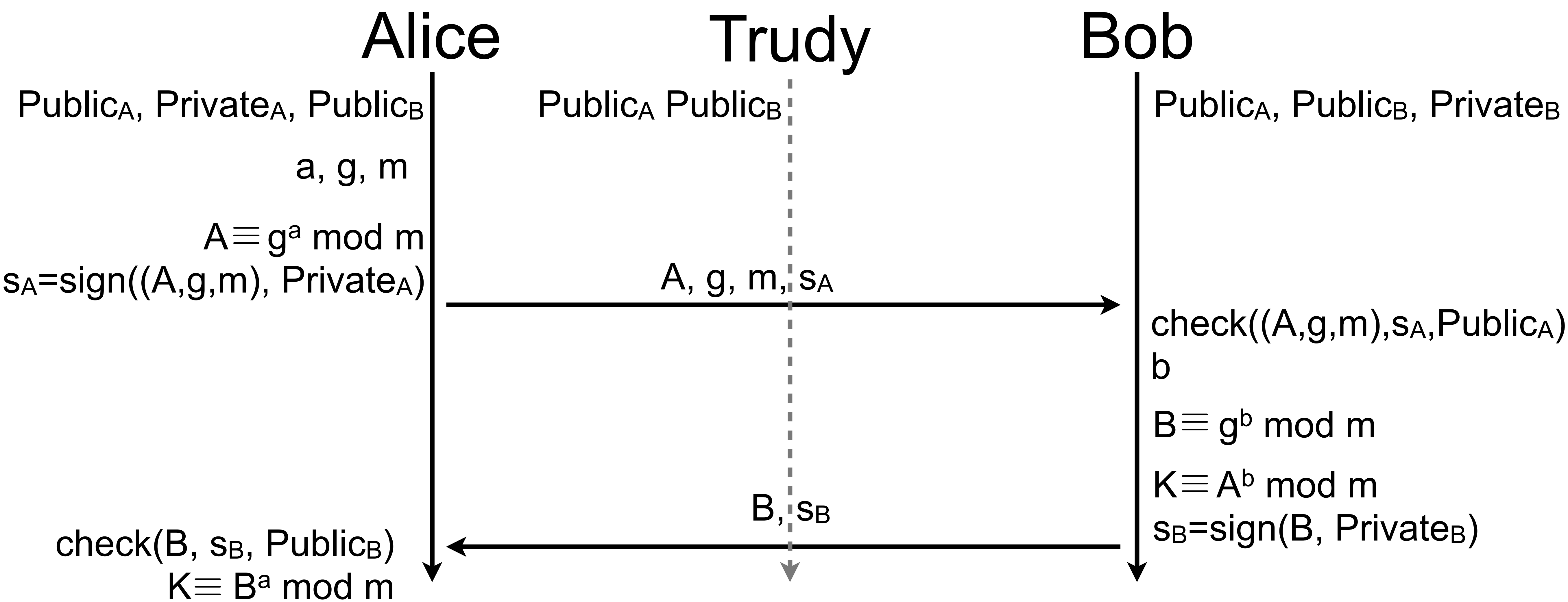


Public key infrastructure (contd.)



Diffie-Hellman key exchange (the return)

- Trudy cannot perform her attack anymore



Problem solved?

- fill me
- fill me
- fill me

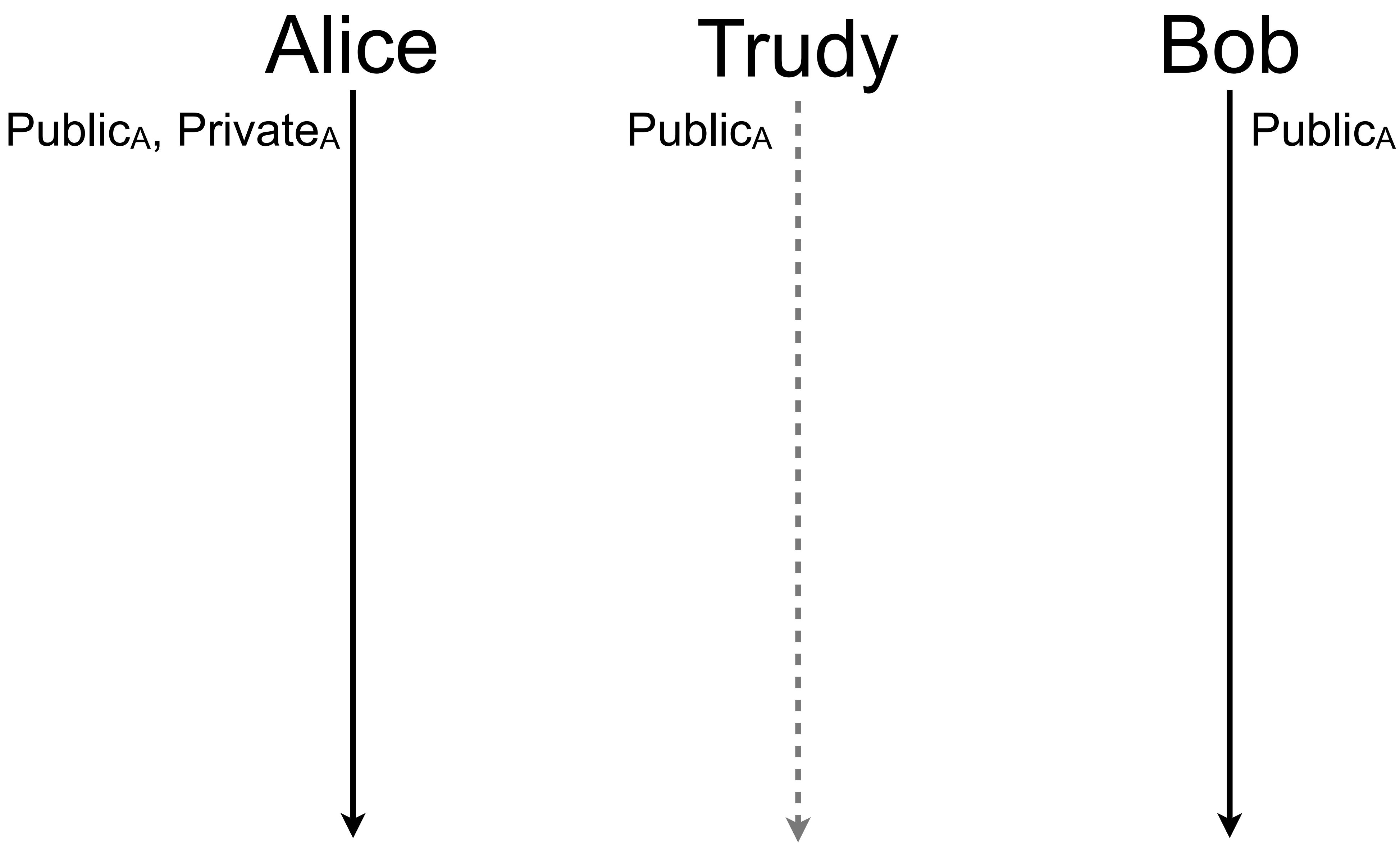
Problem solved?

- fill me
- fill me
- fill me

Replay attacks are still possible!

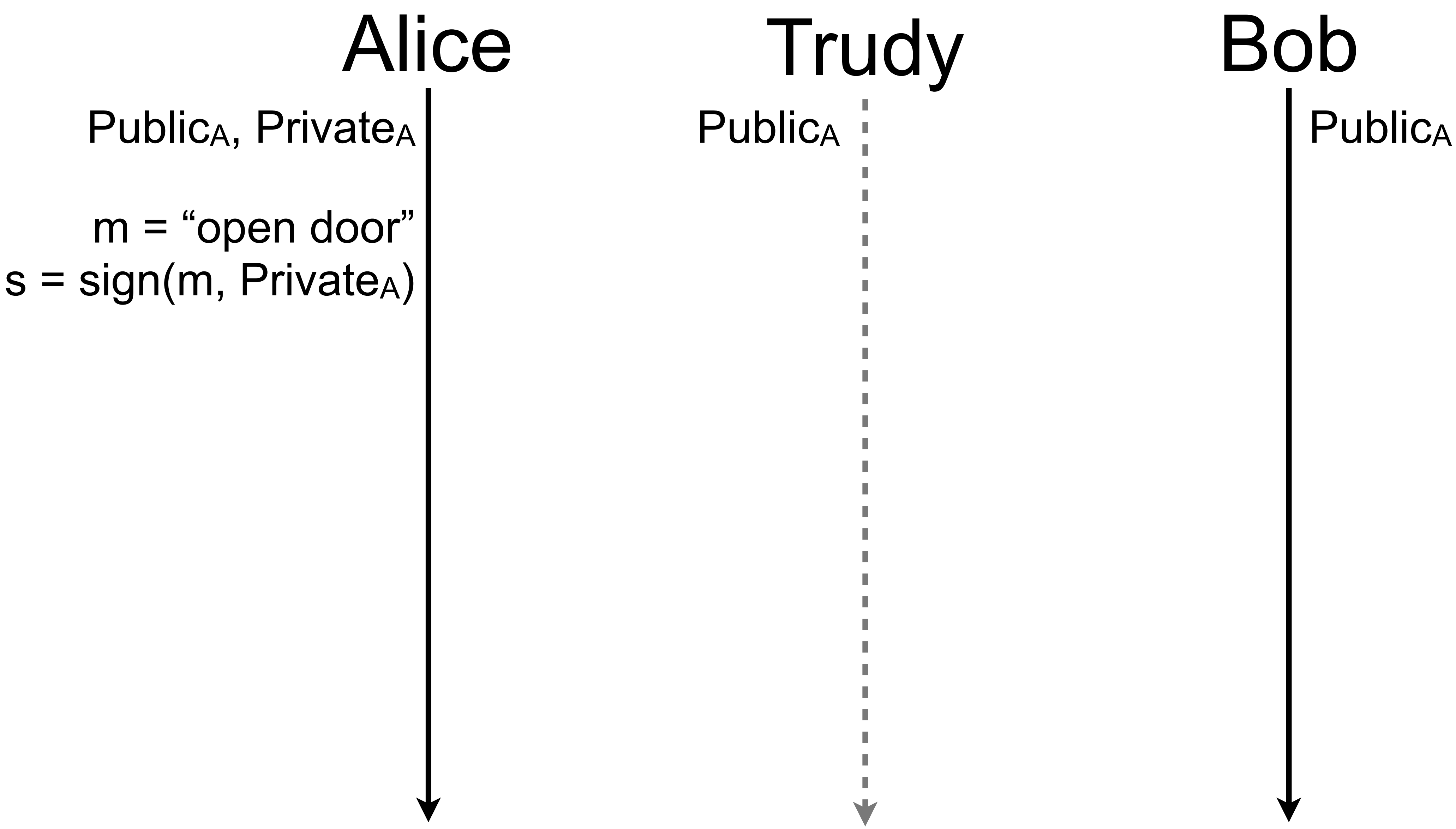
Nonce

- Trudy can replay a message



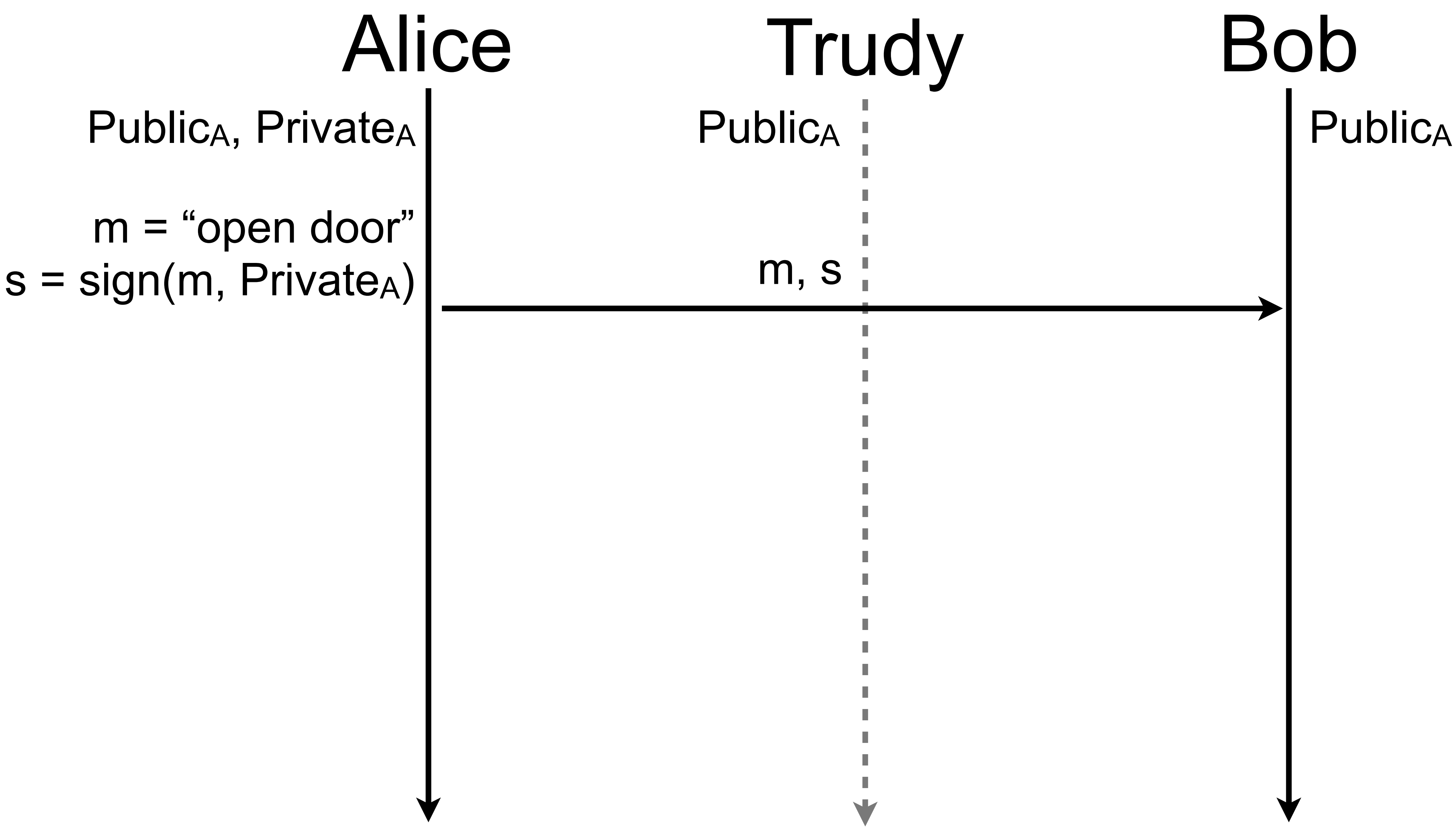
Nonce

- Trudy can replay a message



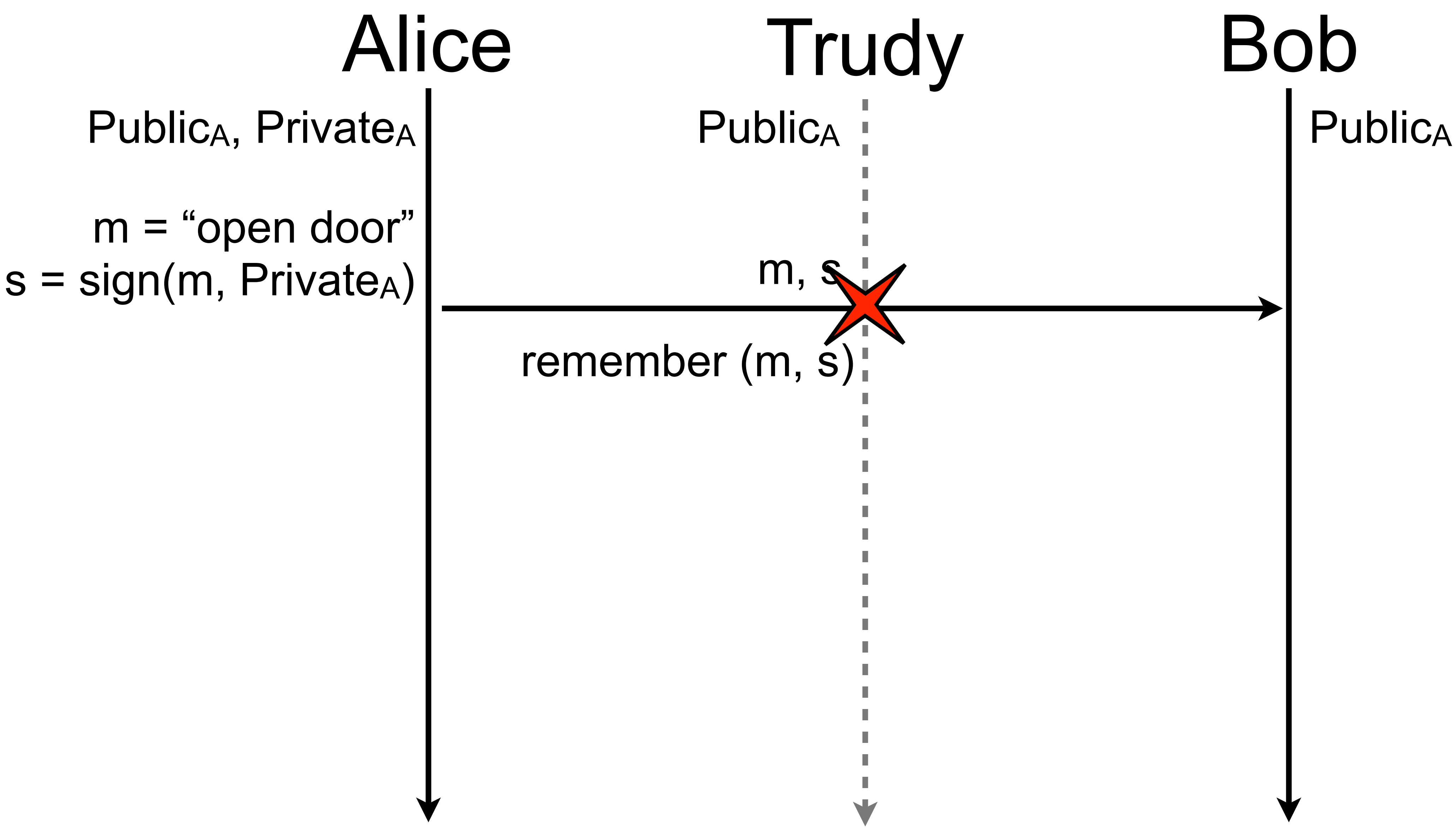
Nonce

- Trudy can replay a message



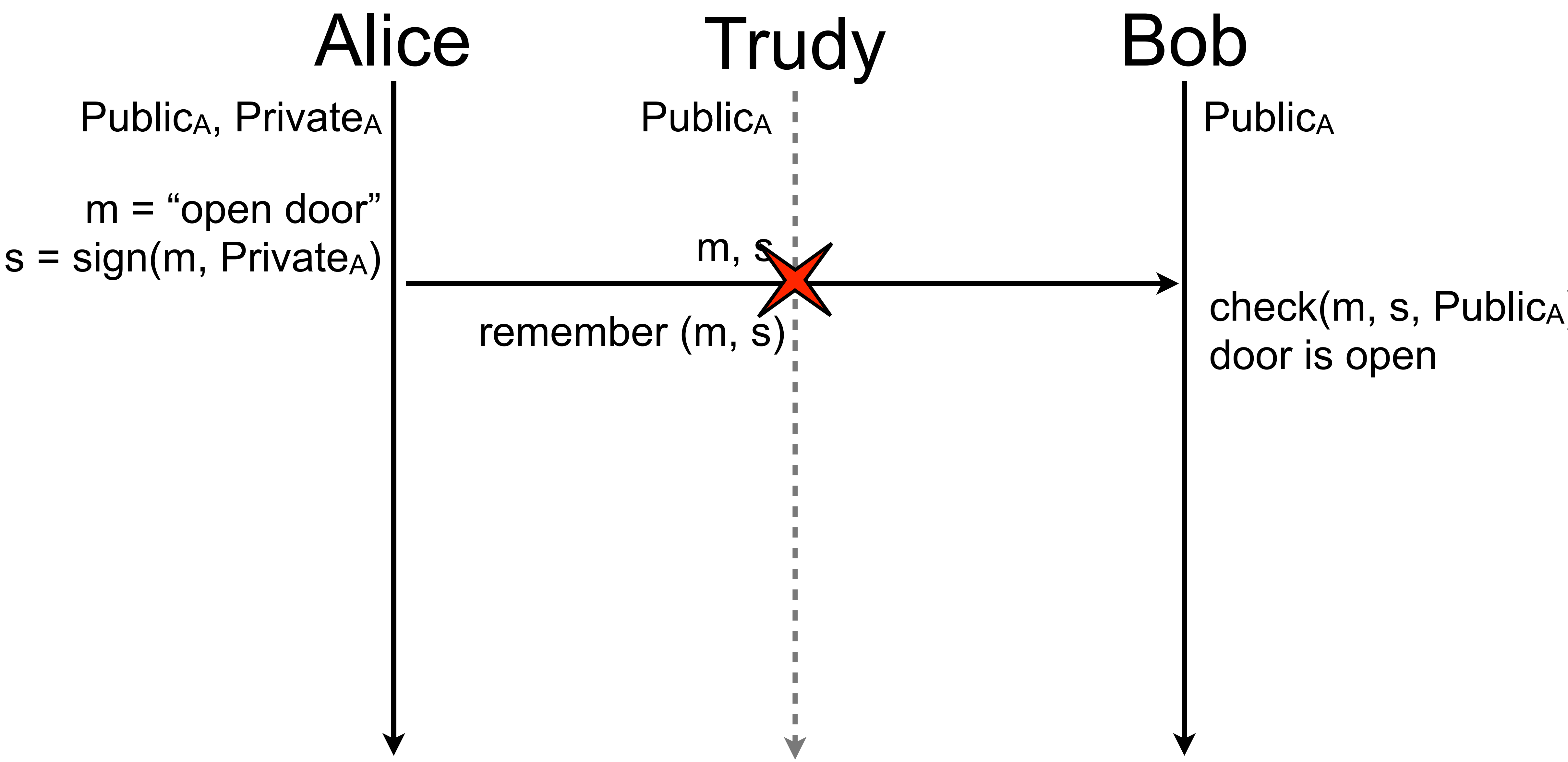
Nonce

- Trudy can replay a message



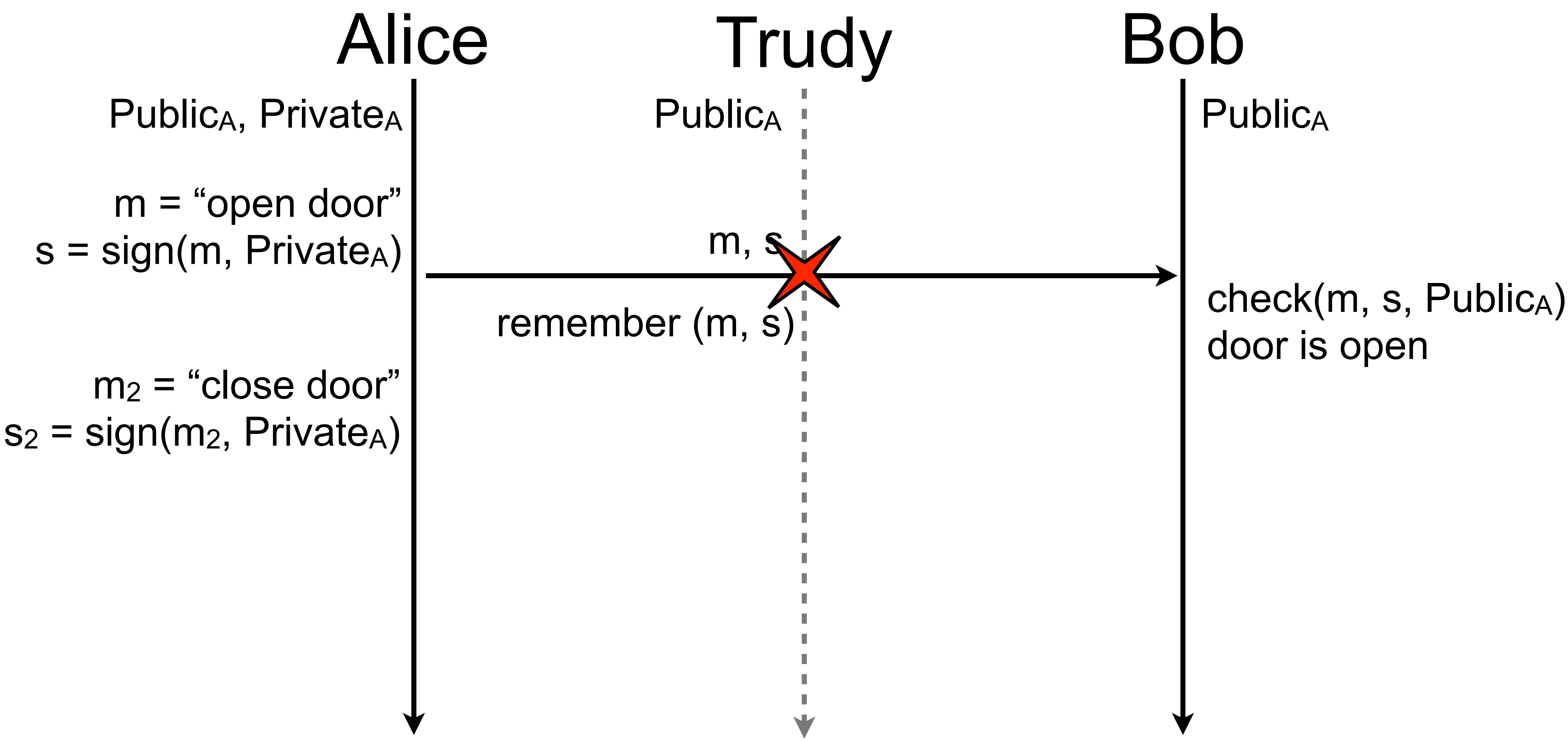
Nonce

- Trudy can replay a message



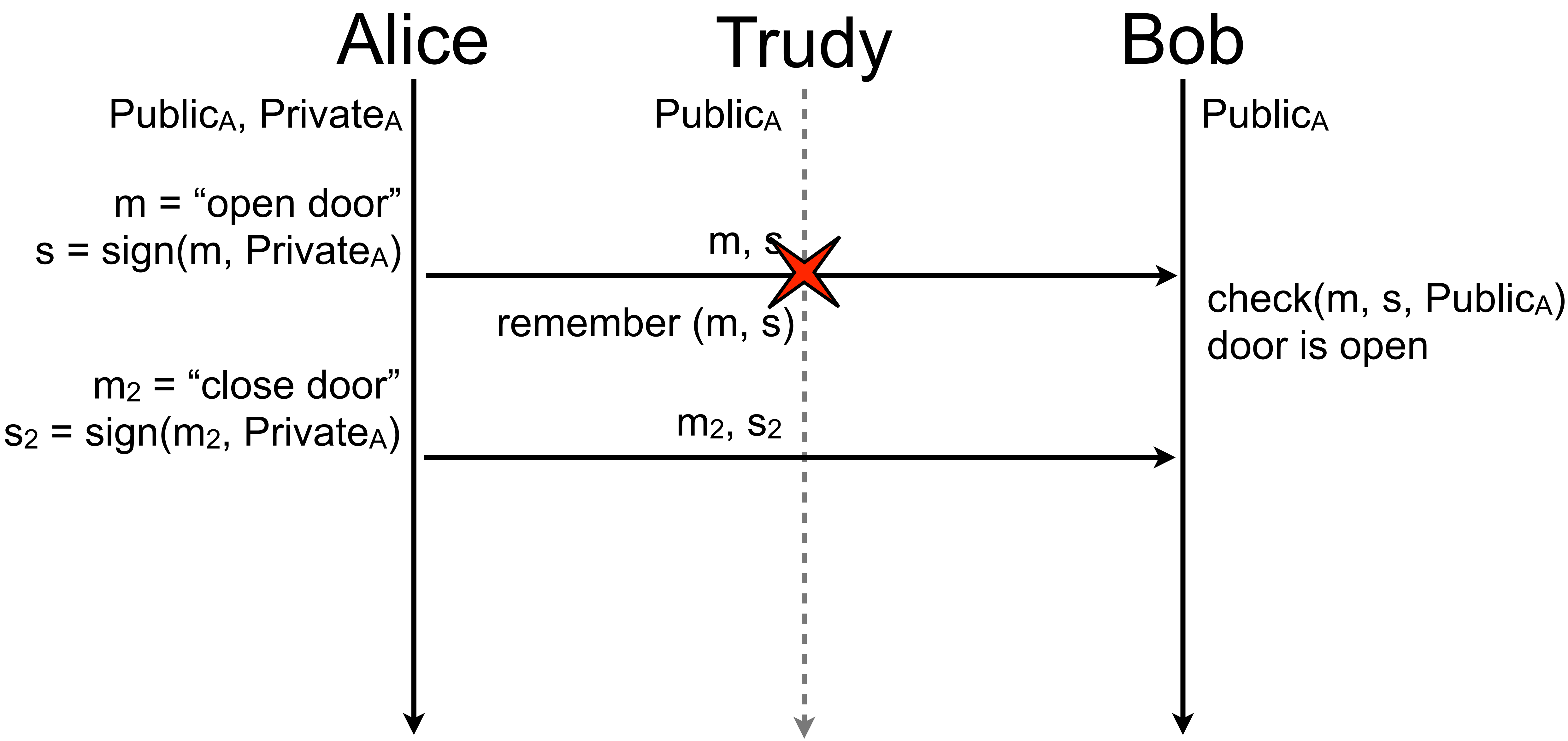
Nonce

- Trudy can replay a message



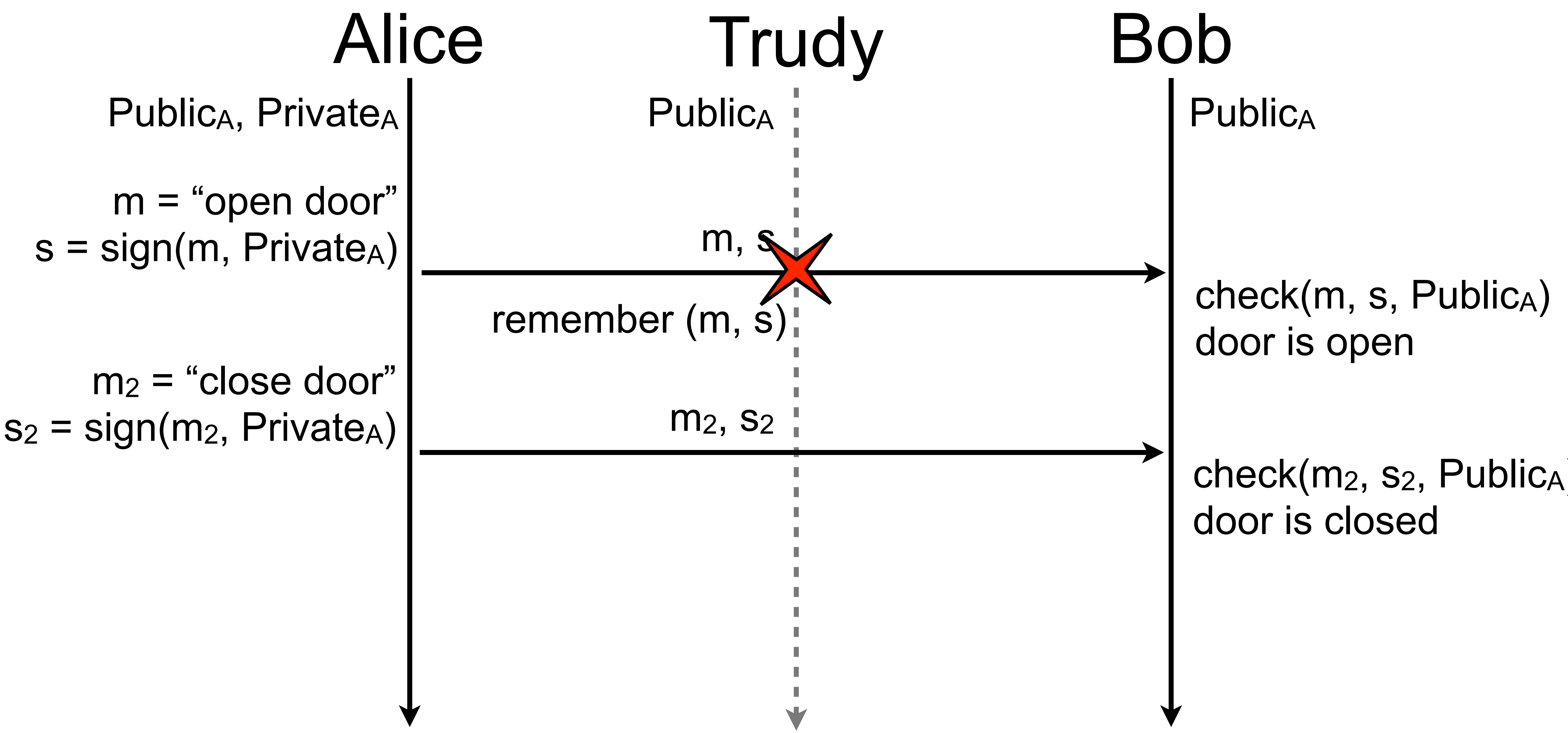
Nonce

- Trudy can replay a message



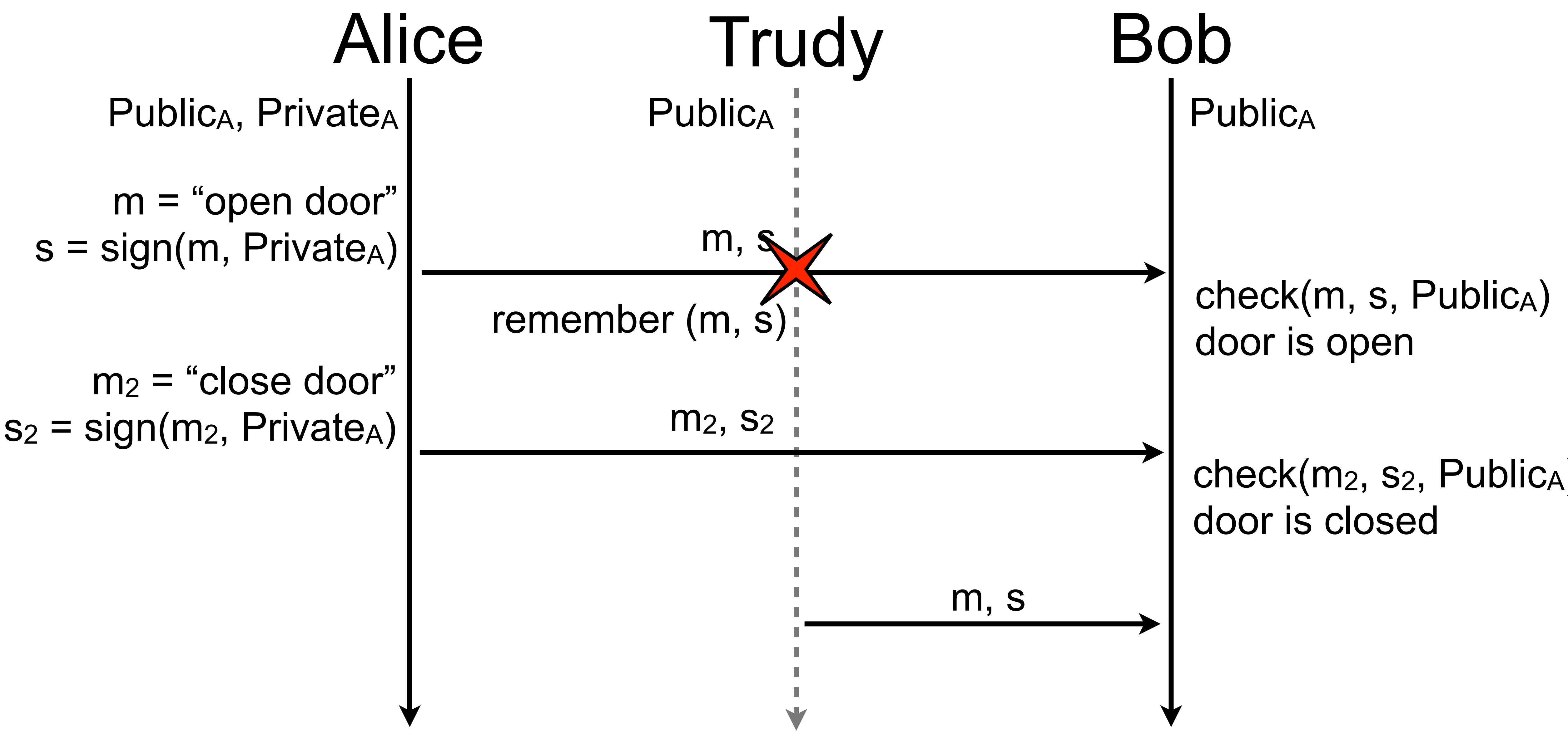
Nonce

- Trudy can replay a message



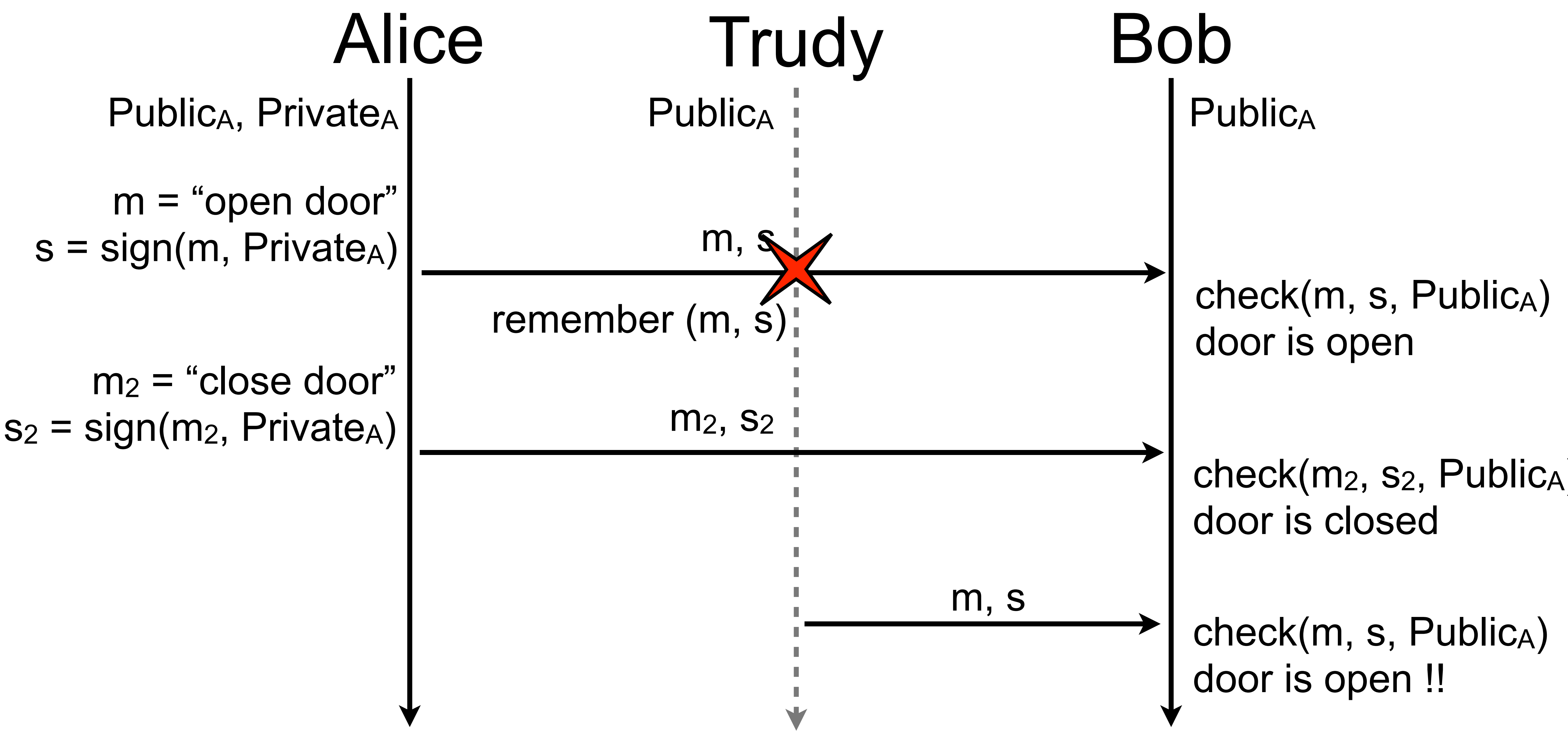
Nonce

- Trudy can replay a message



Nonce

- Trudy can replay a message

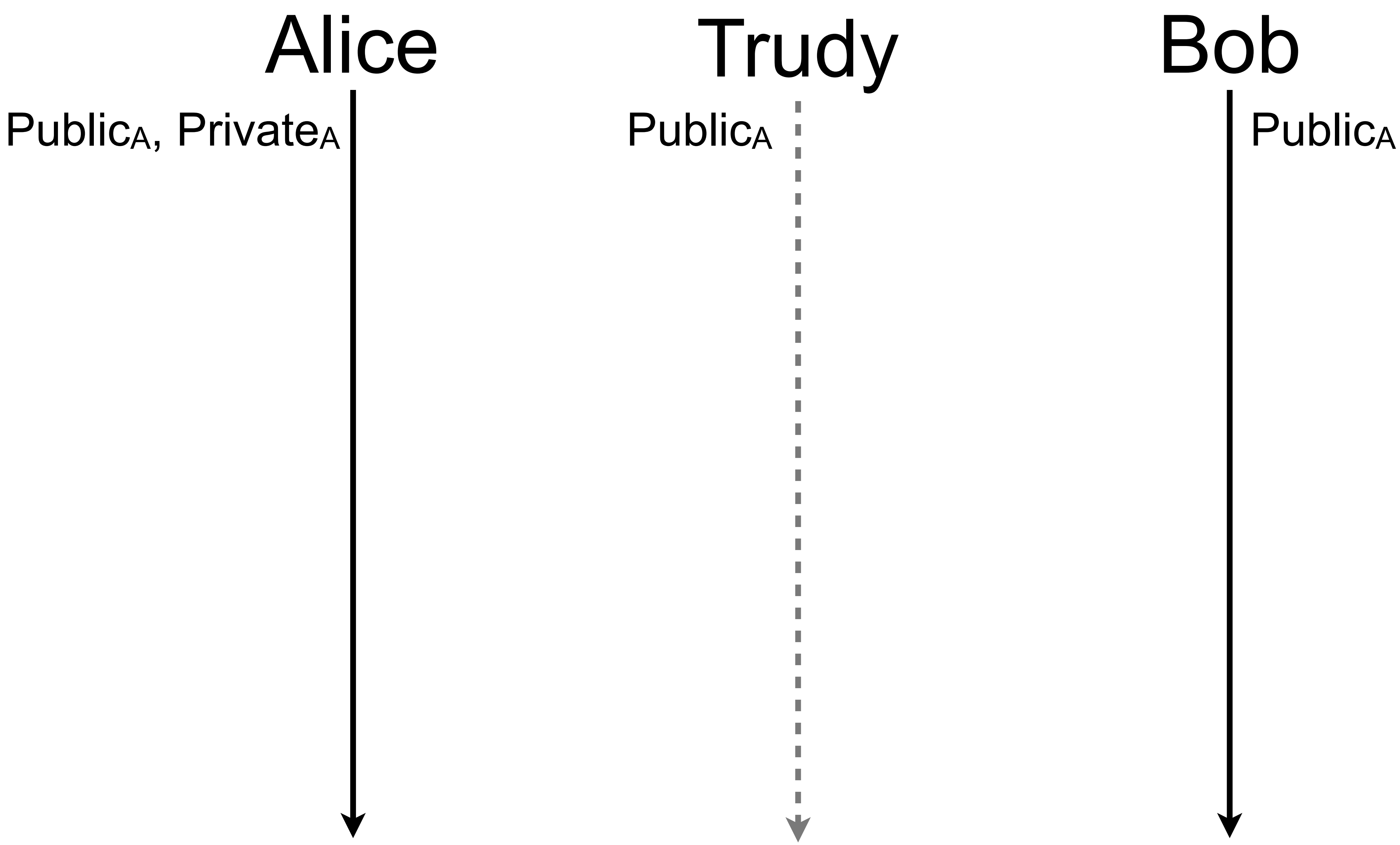


Nonce (contd.)

- A nonce is a number used only once
- Three general methods to create nonces
  - sequential number
    - increment after each use
    - keep it in non-volatile storage in case of reboot
  - timestamp
    - current time of the nonce generation
    - be sure clock is not going backward (e.g., winter time)
  - random number
    - low collision probability if the pseudo random number generator is good and random number is big enough (e.g., more than 128 bits)
- Nonce alone is rarely enough to have a good protection
  - not robust to eavesdropping or man-in-the-middle attack

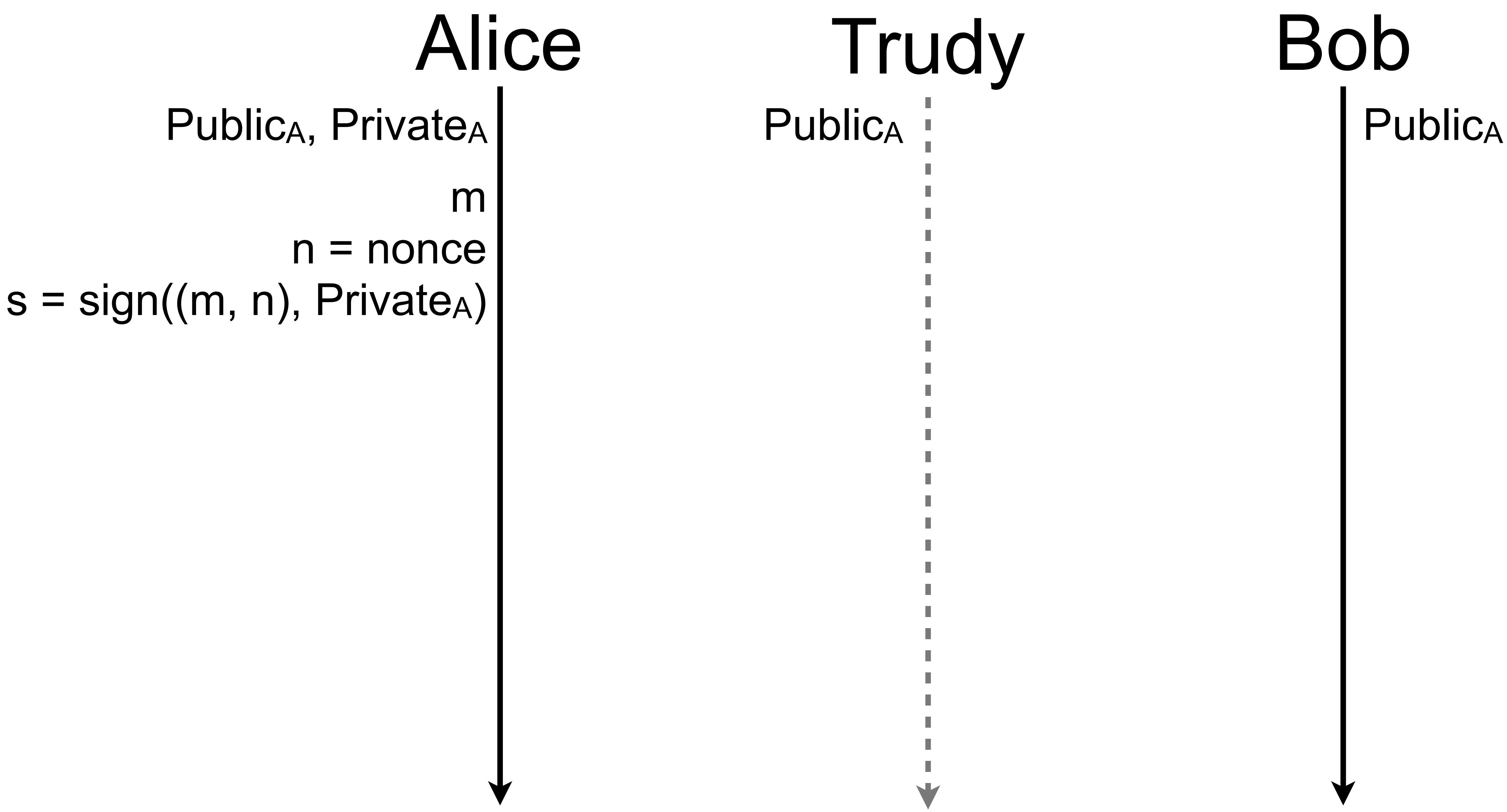
Nonce (contd.)

- Each message is make unique thanks to the nonce



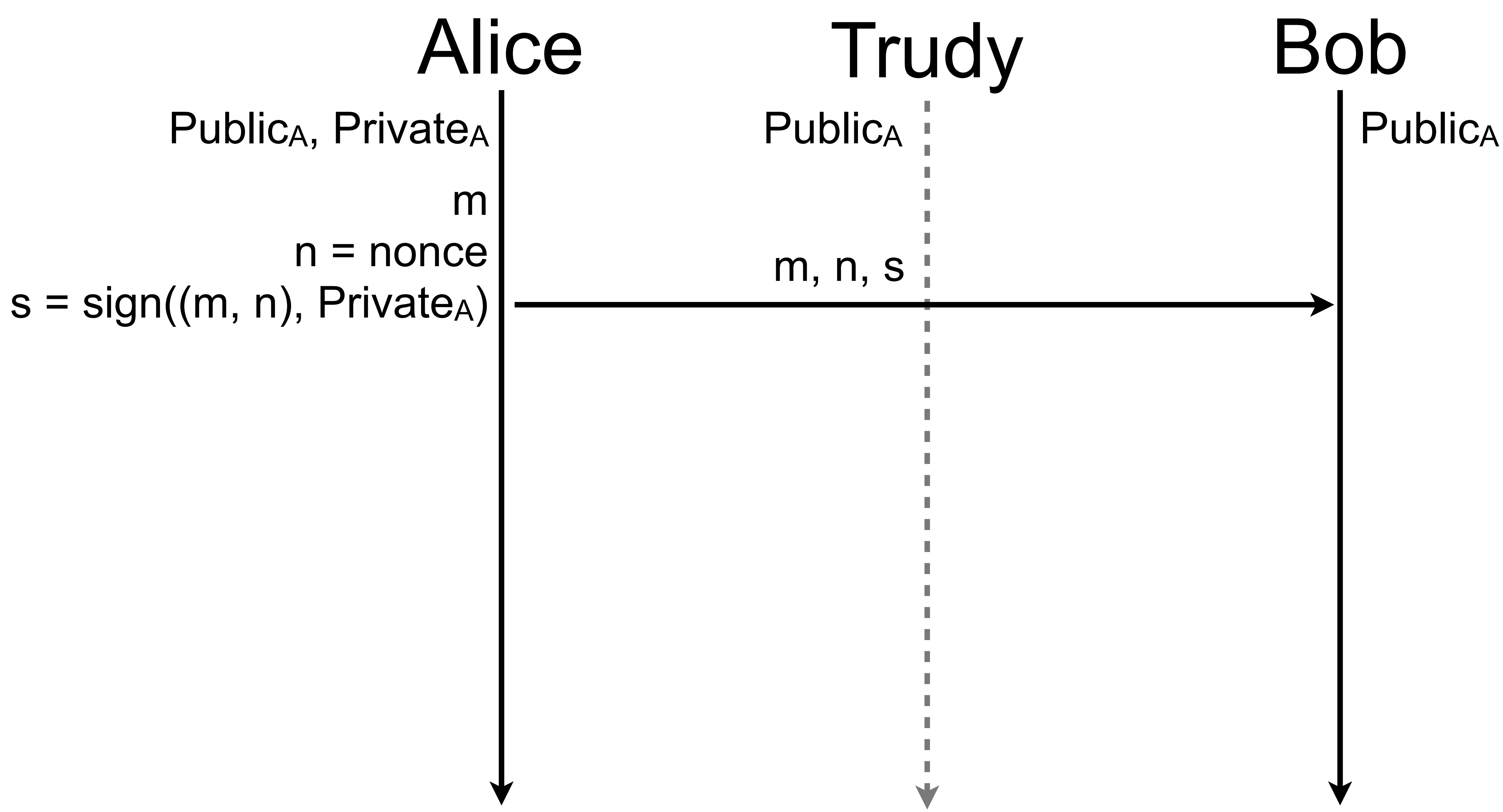
Nonce (contd.)

- Each message is make unique thanks to the nonce



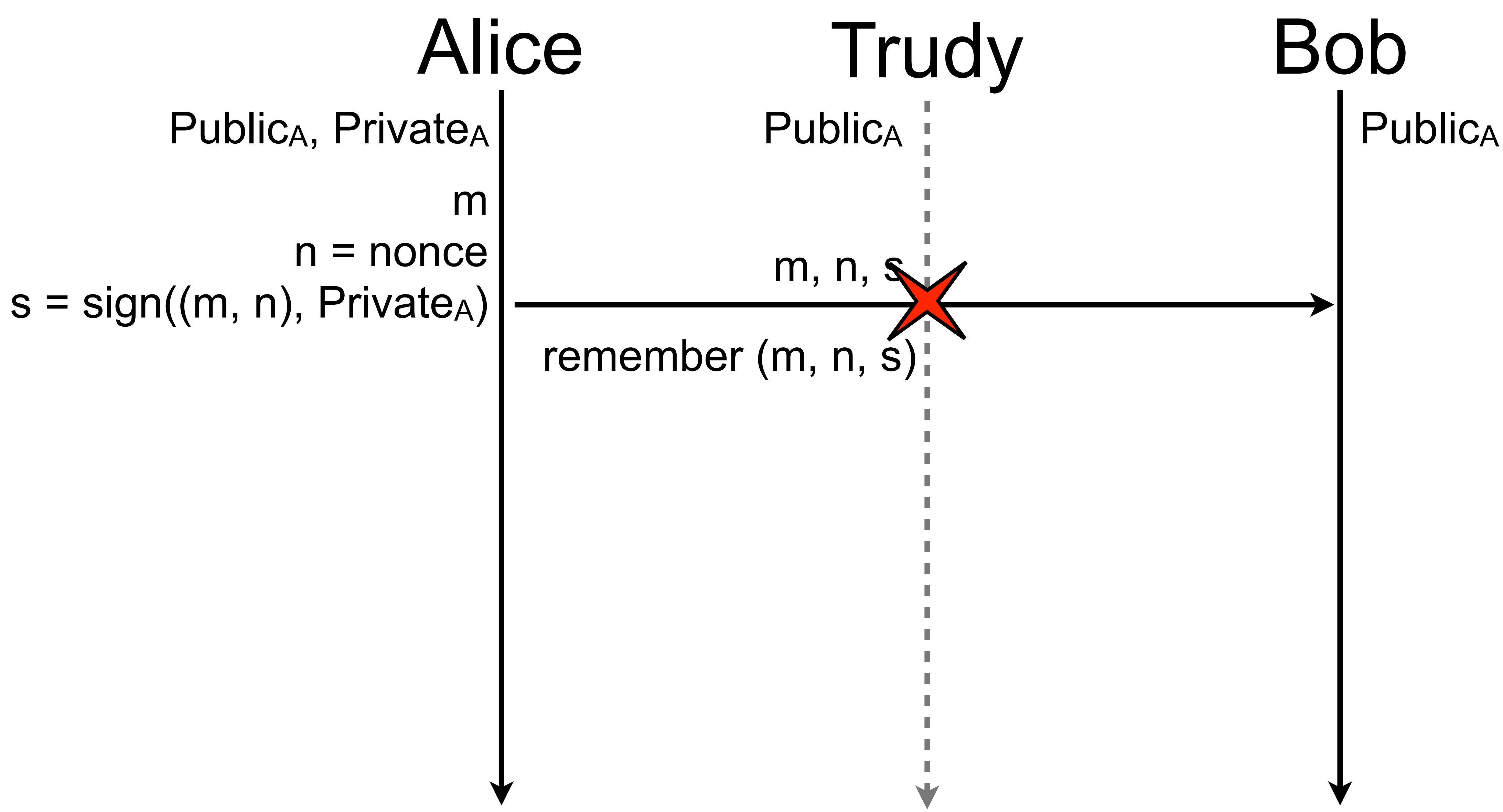
Nonce (contd.)

- Each message is make unique thanks to the nonce



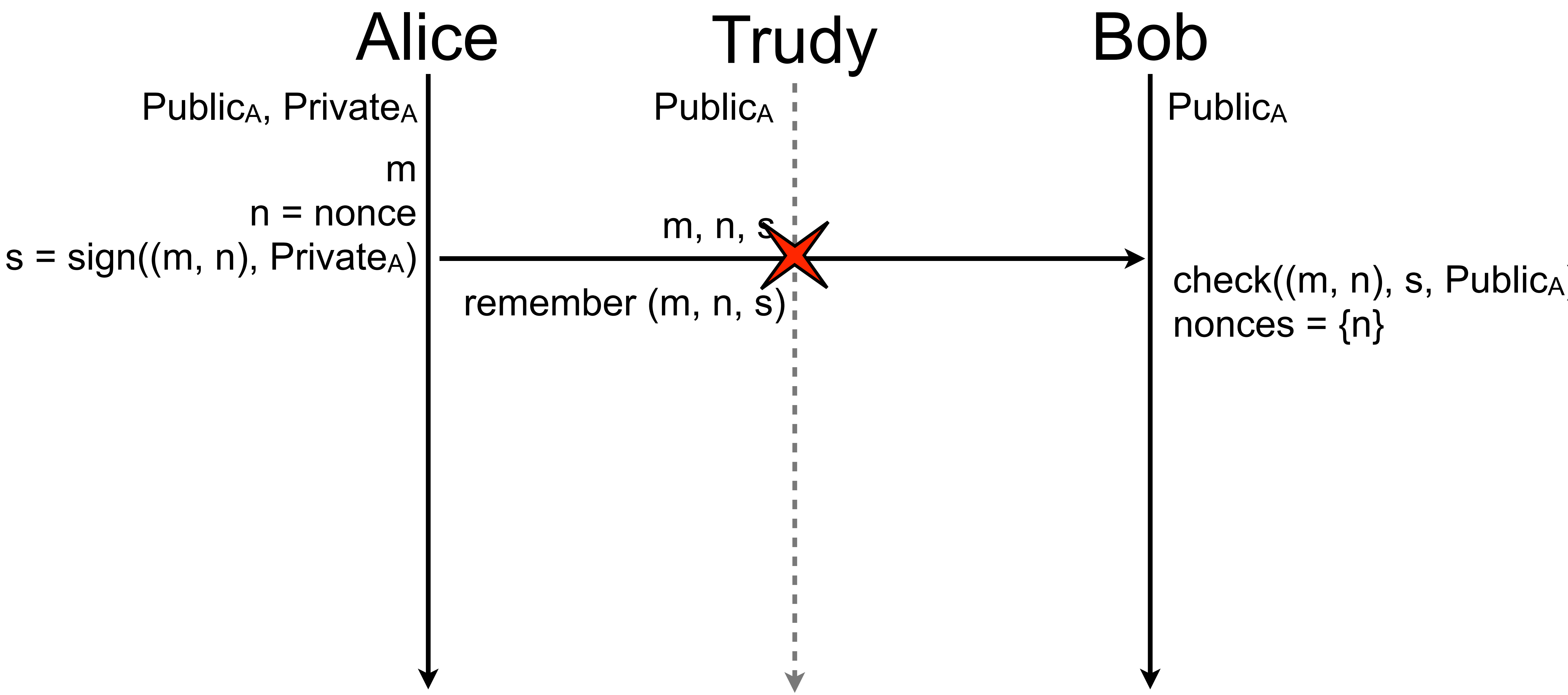
Nonce (contd.)

- Each message is make unique thanks to the nonce



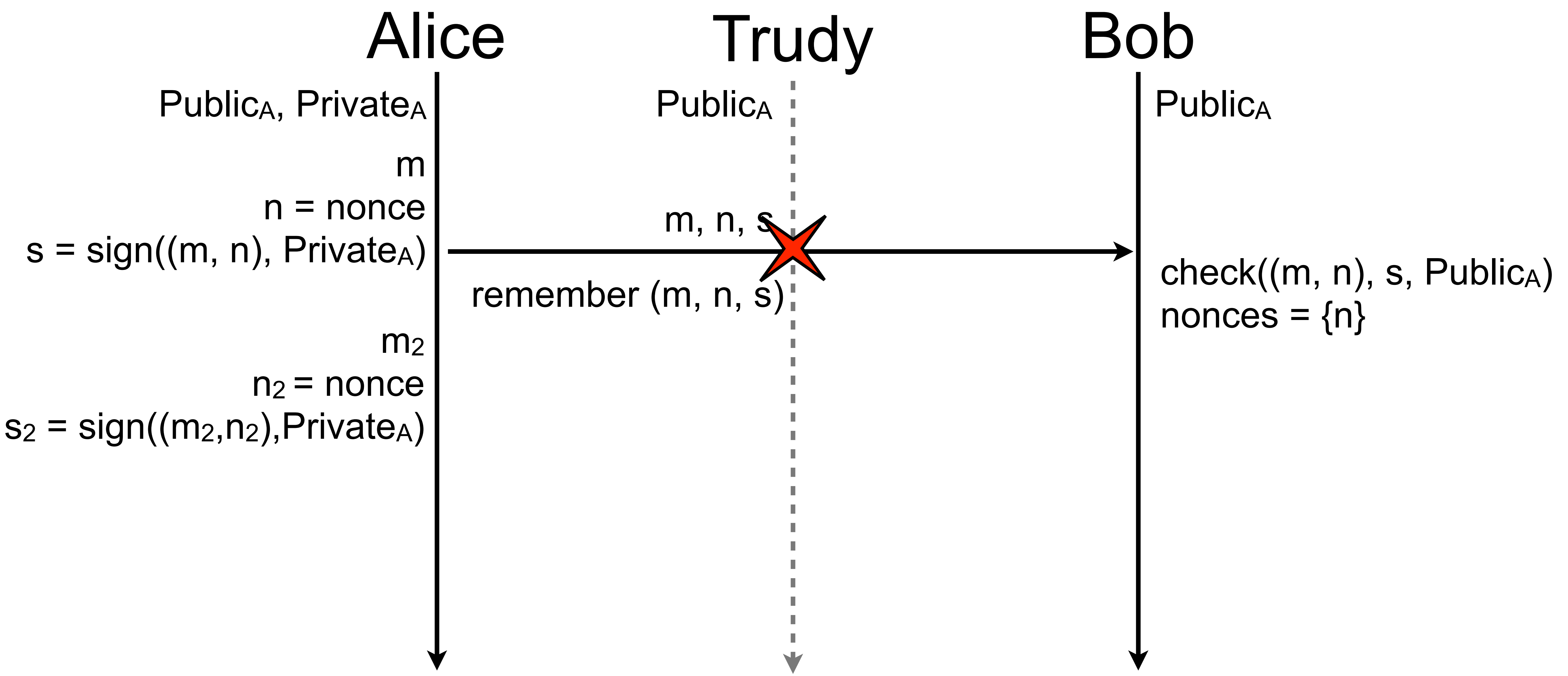
Nonce (contd.)

- Each message is make unique thanks to the nonce



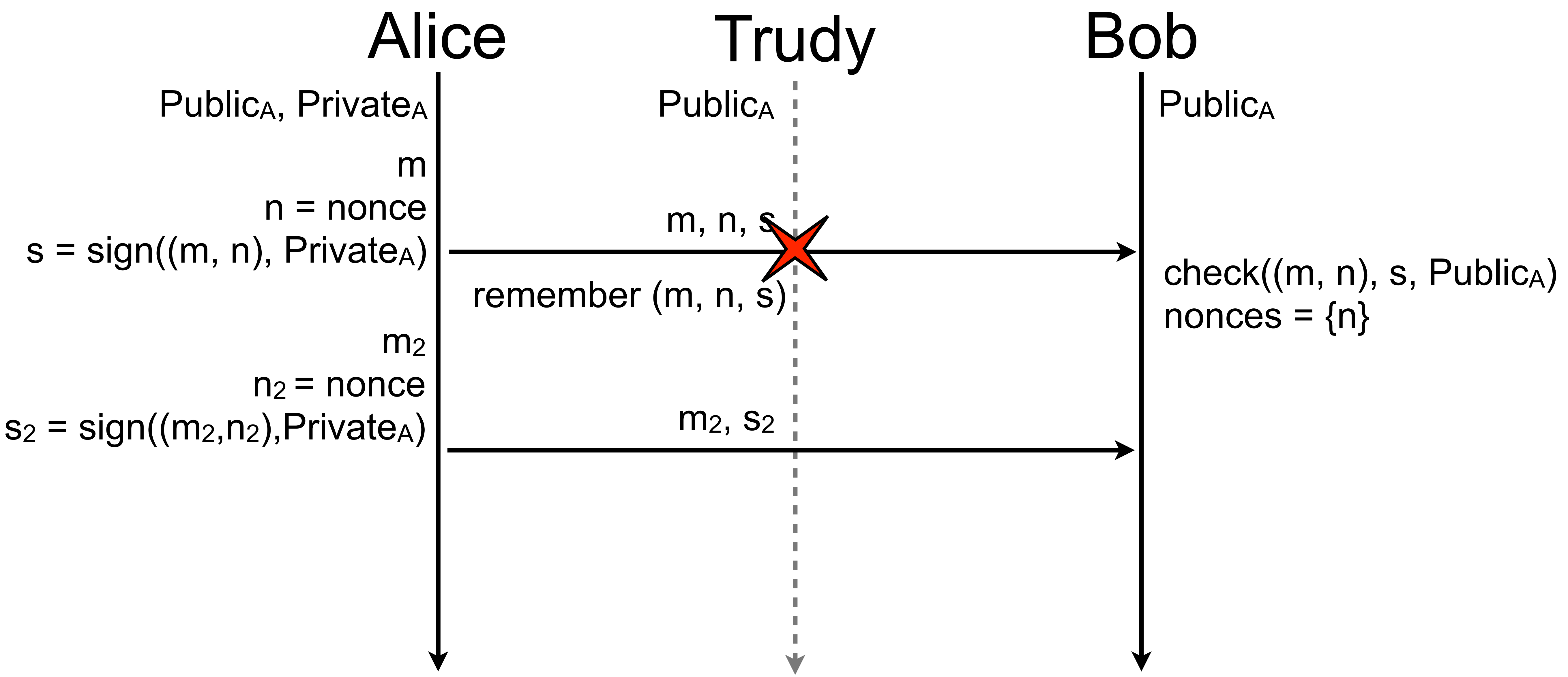
Nonce (contd.)

- Each message is make unique thanks to the nonce



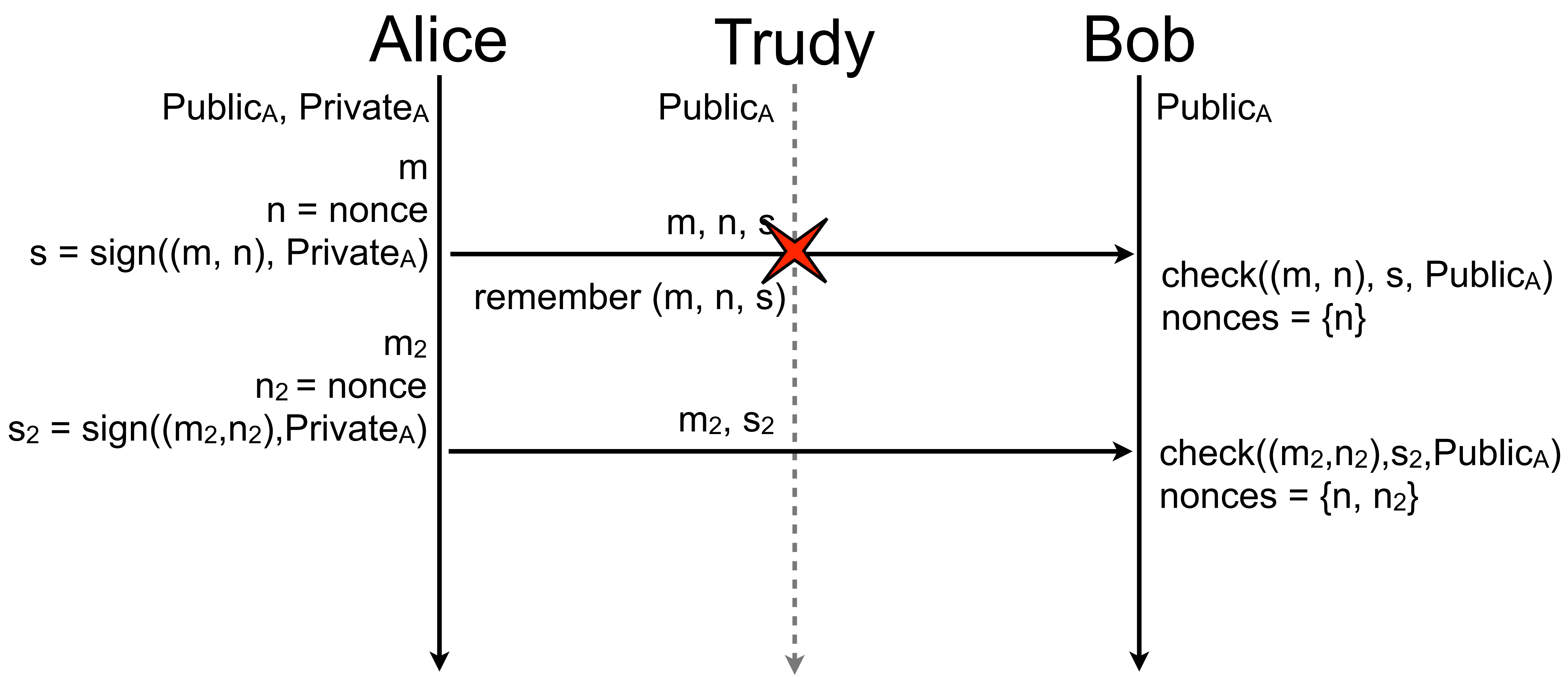
Nonce (contd.)

- Each message is make unique thanks to the nonce



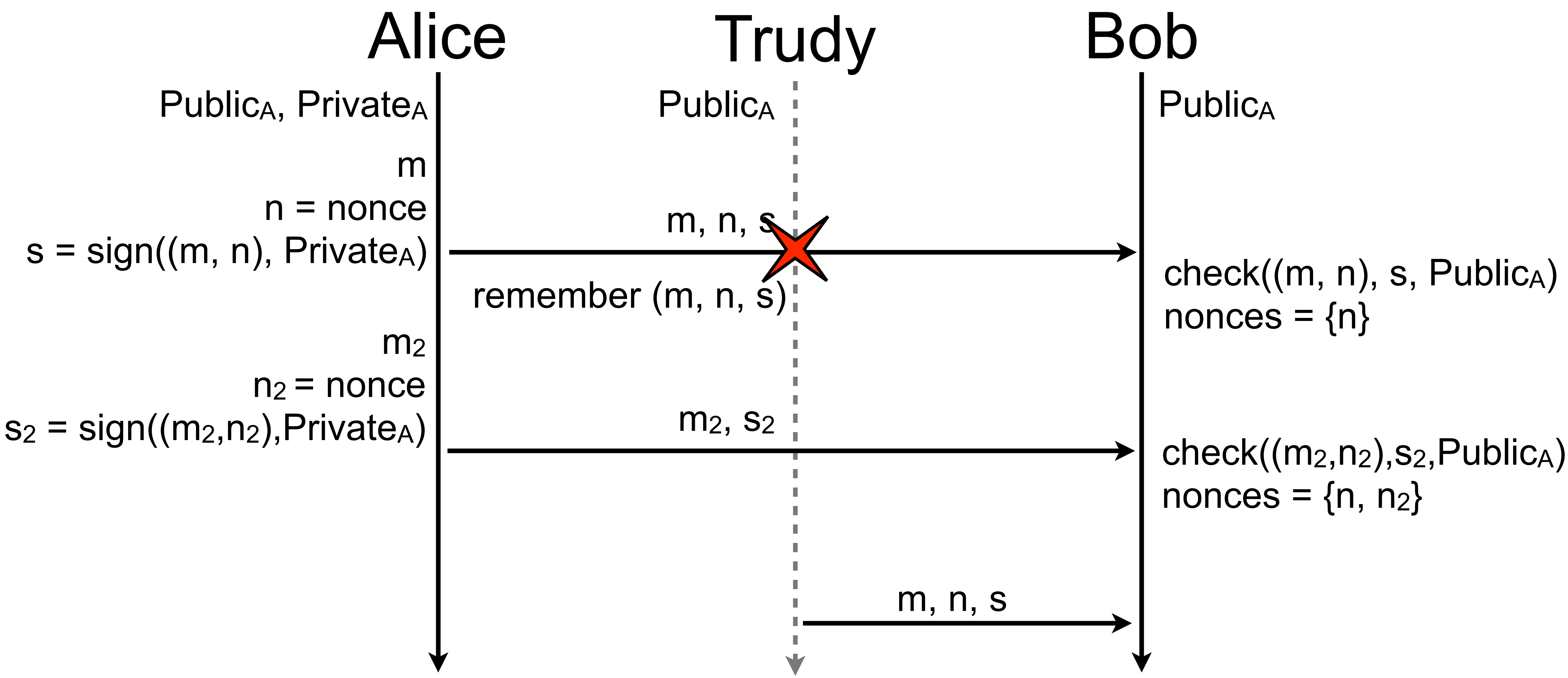
Nonce (contd.)

- Each message is make unique thanks to the nonce



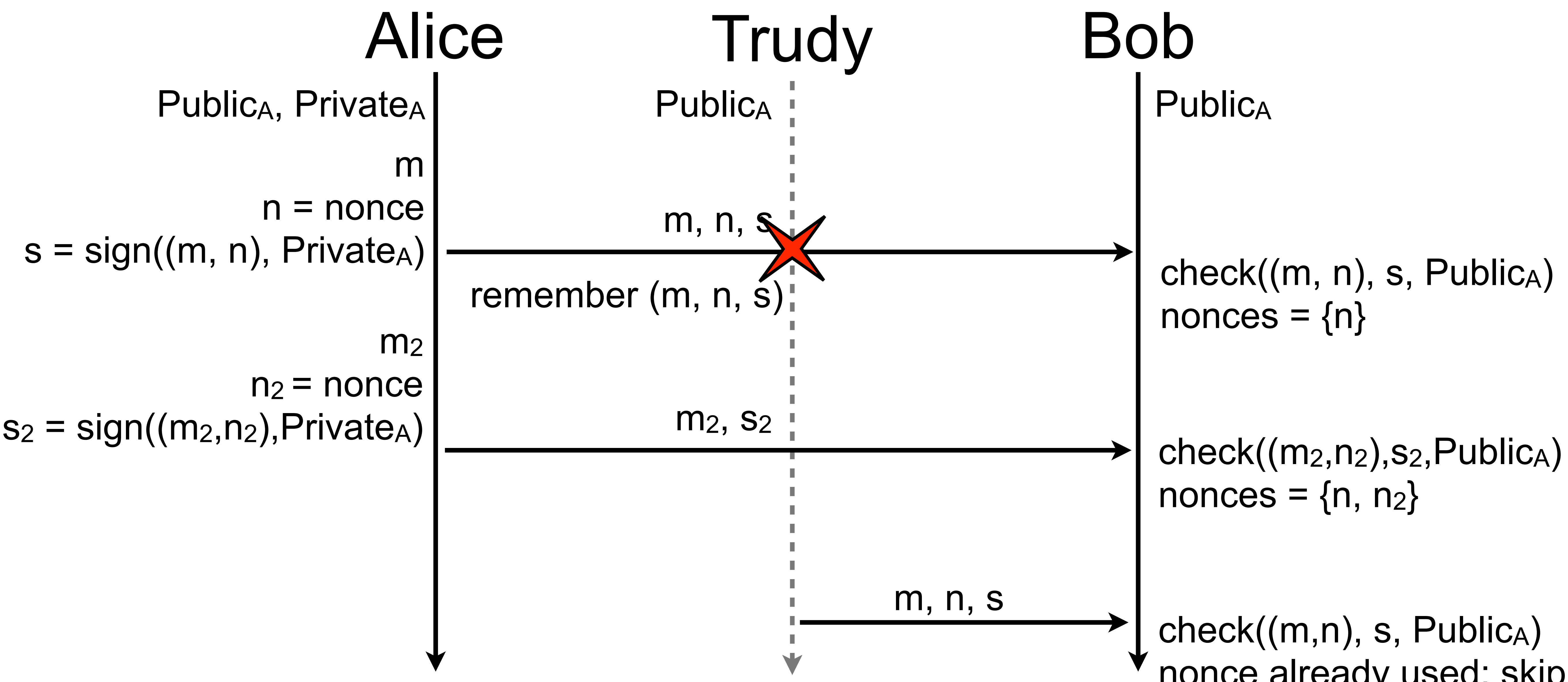
Nonce (contd.)

- Each message is make unique thanks to the nonce



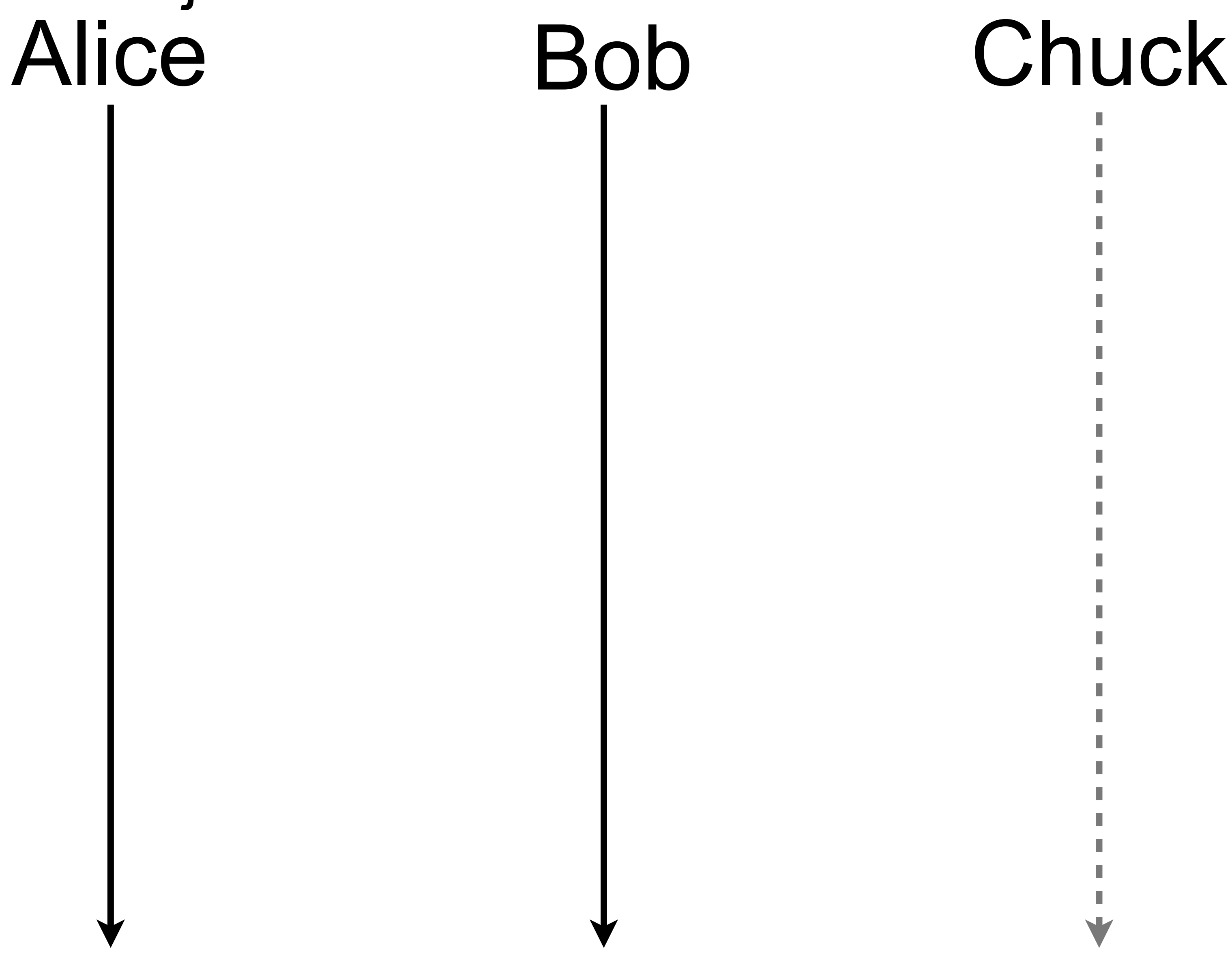
Nonce (contd.)

- Each message is make unique thanks to the nonce



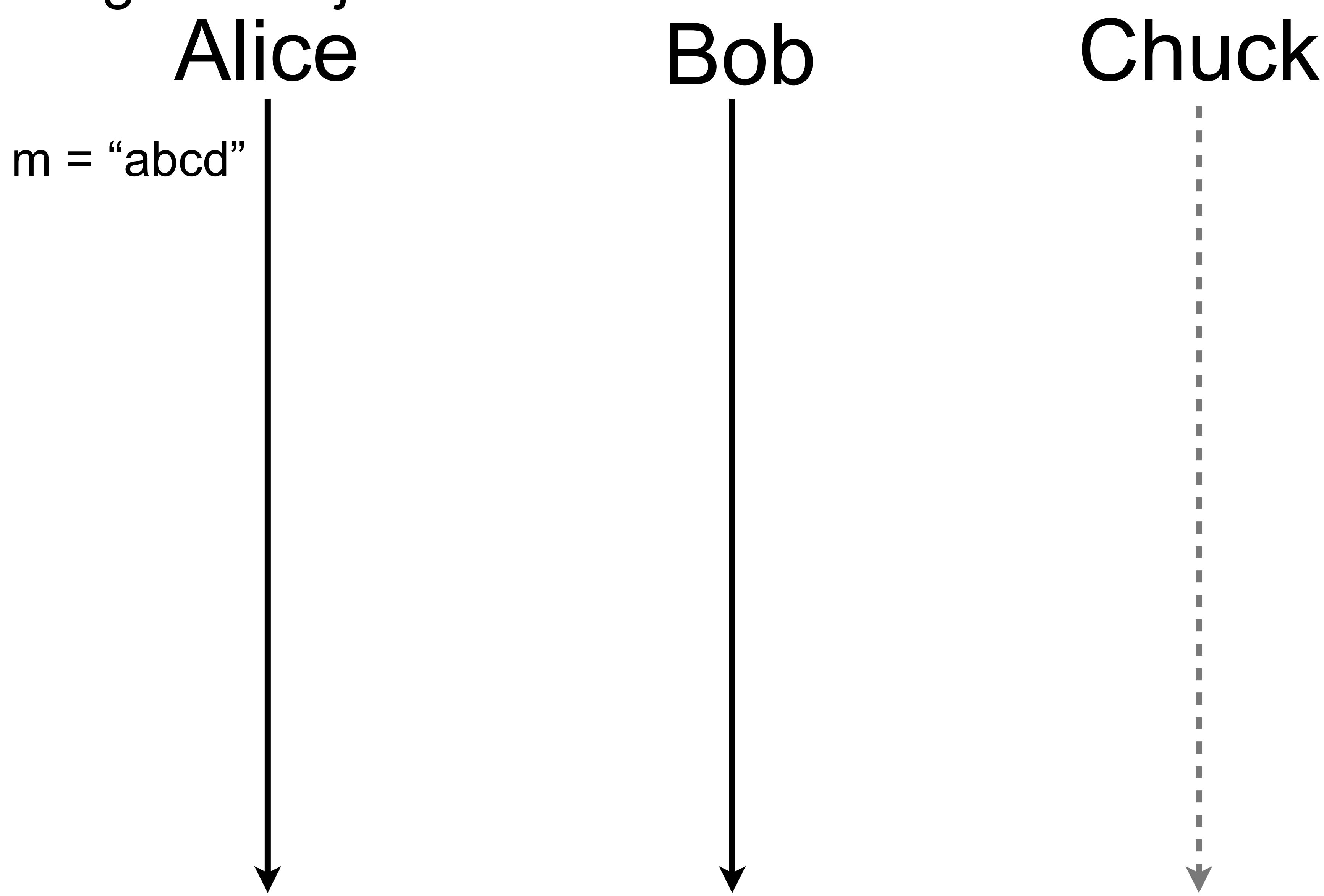
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



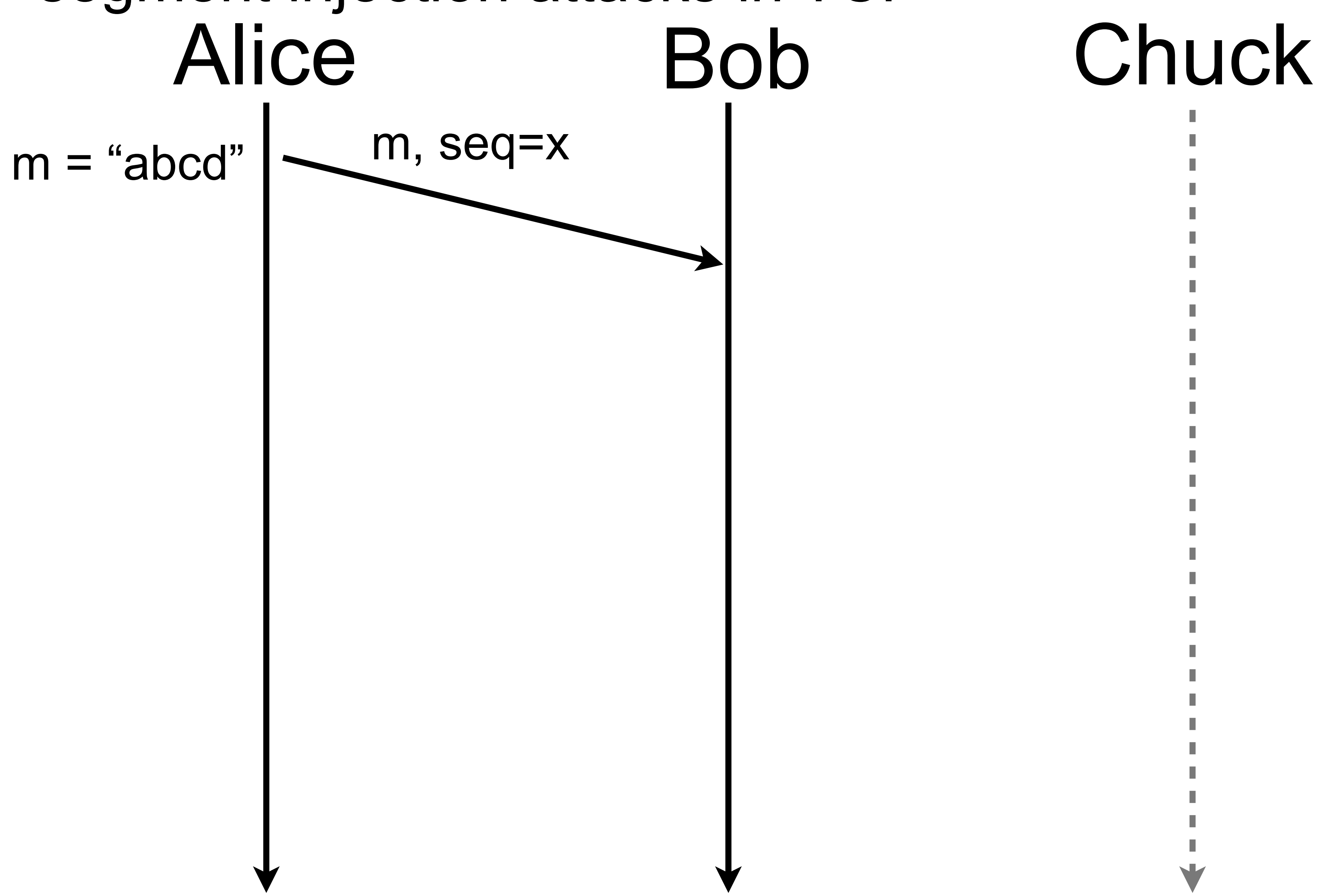
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



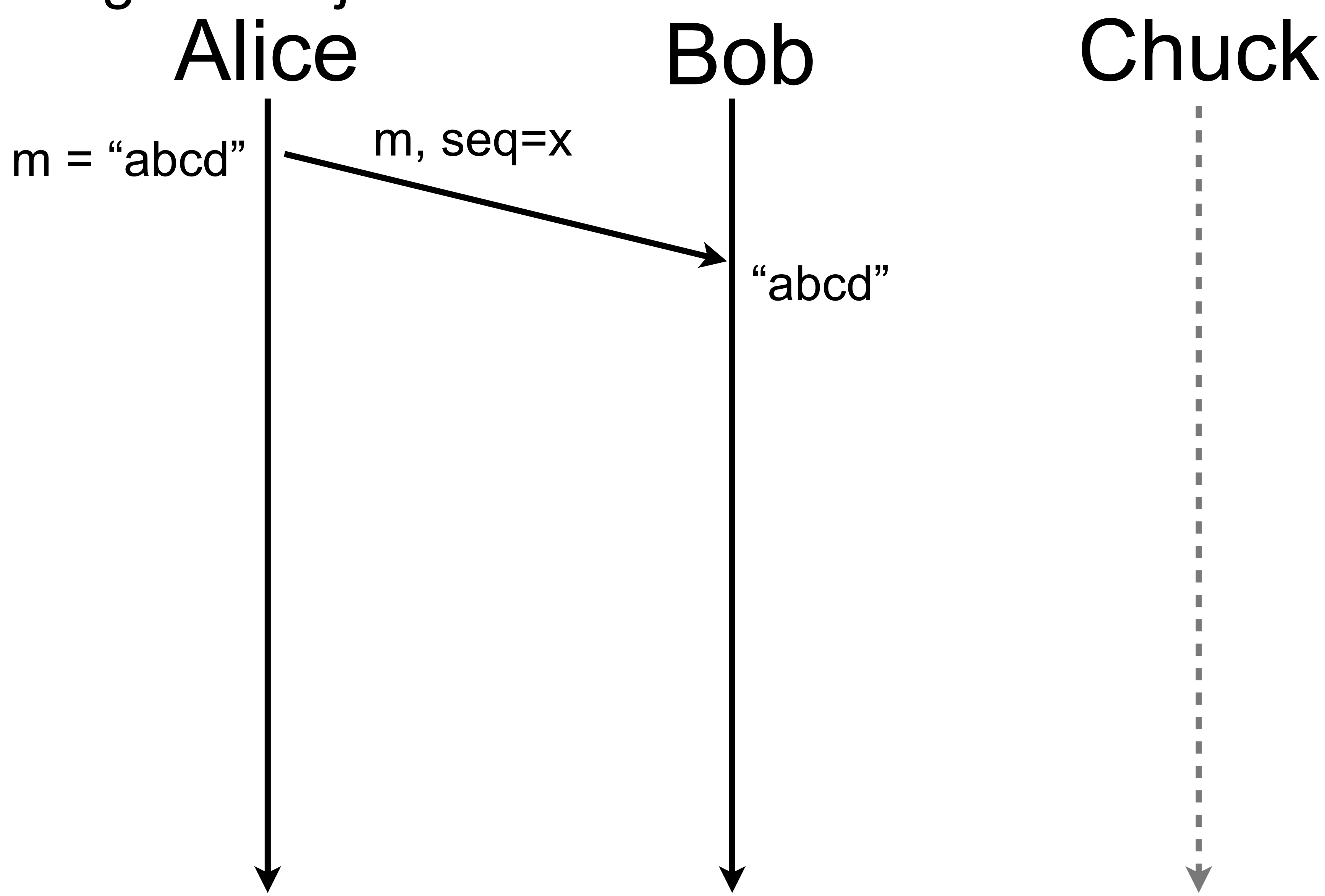
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



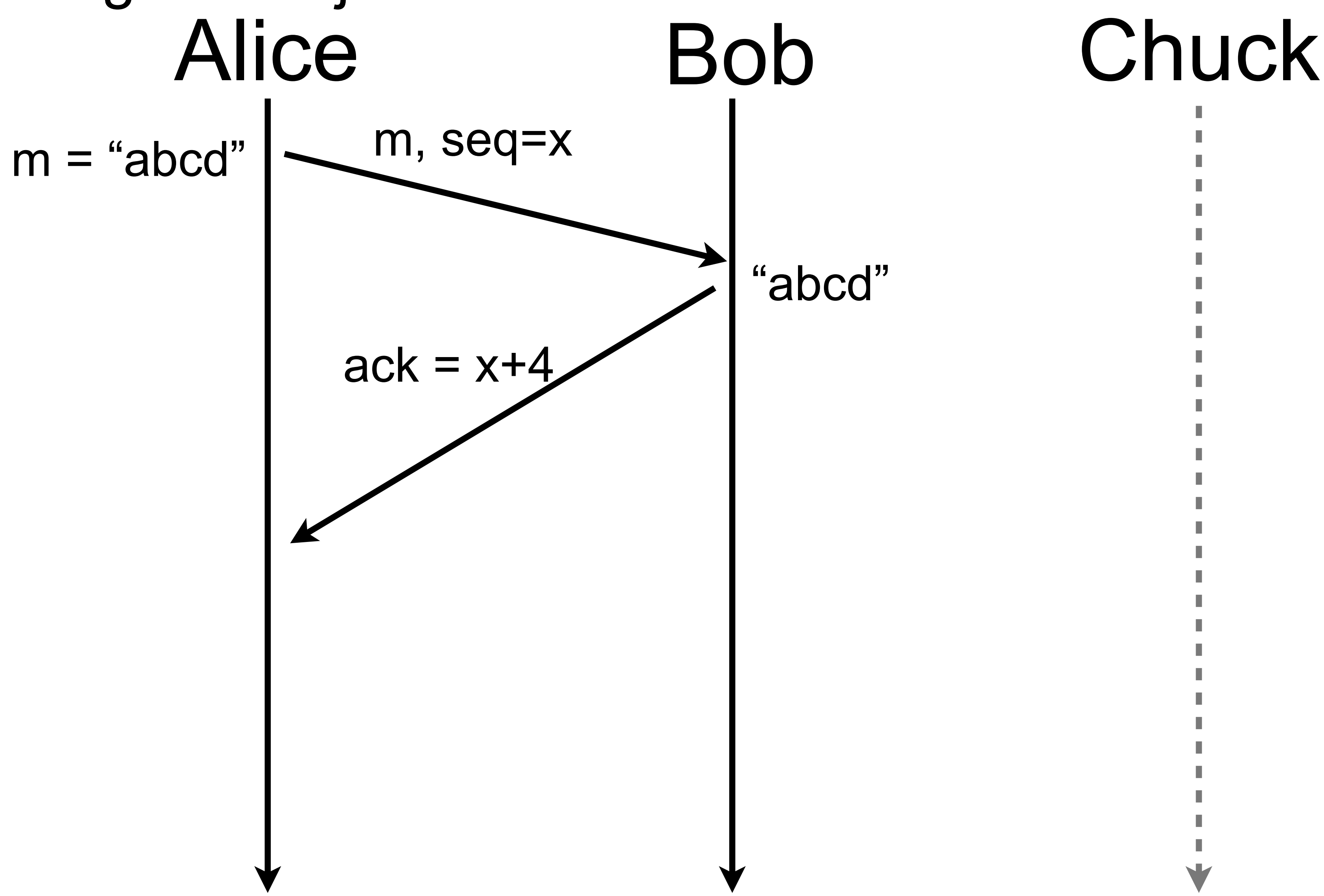
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



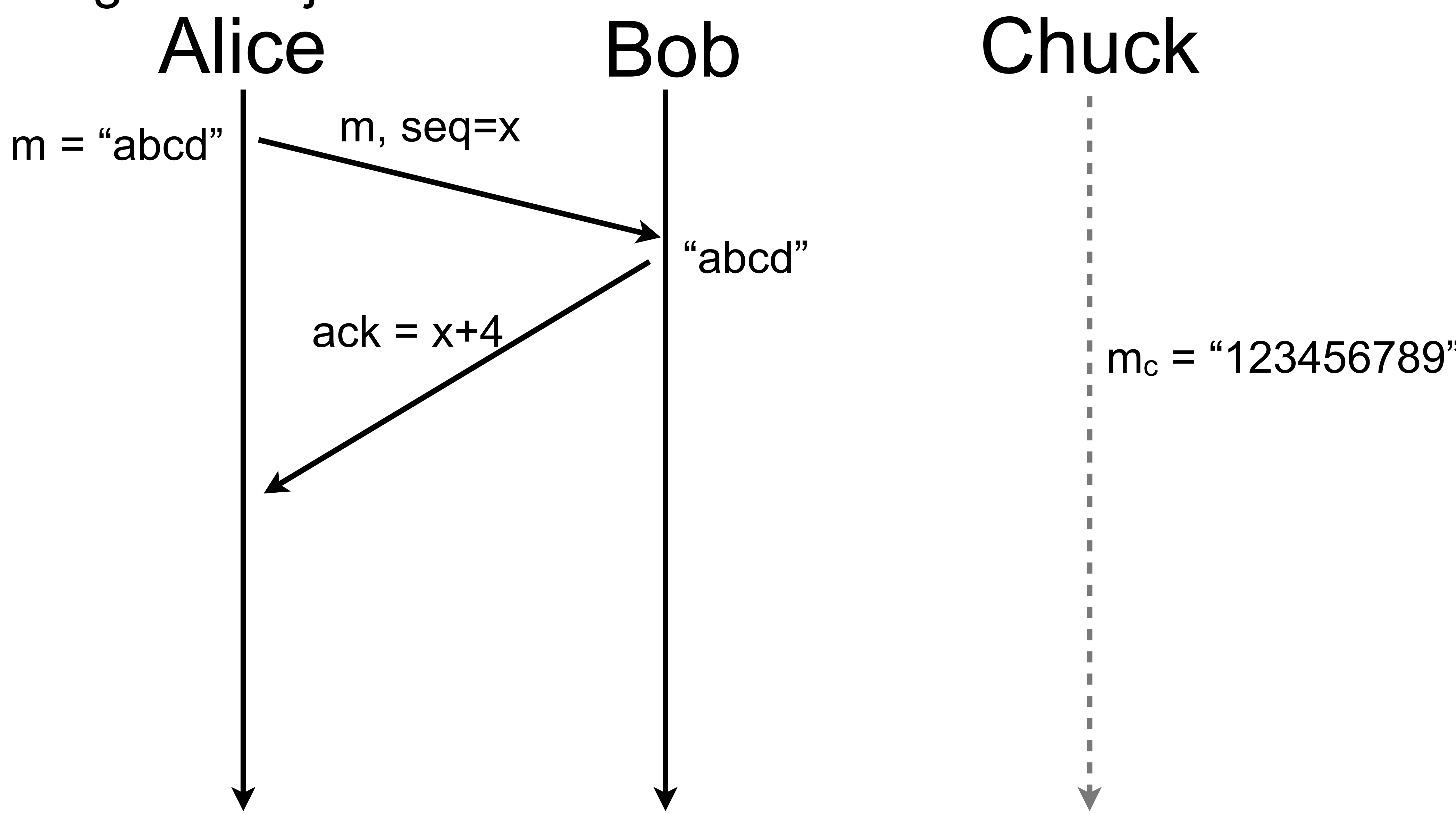
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



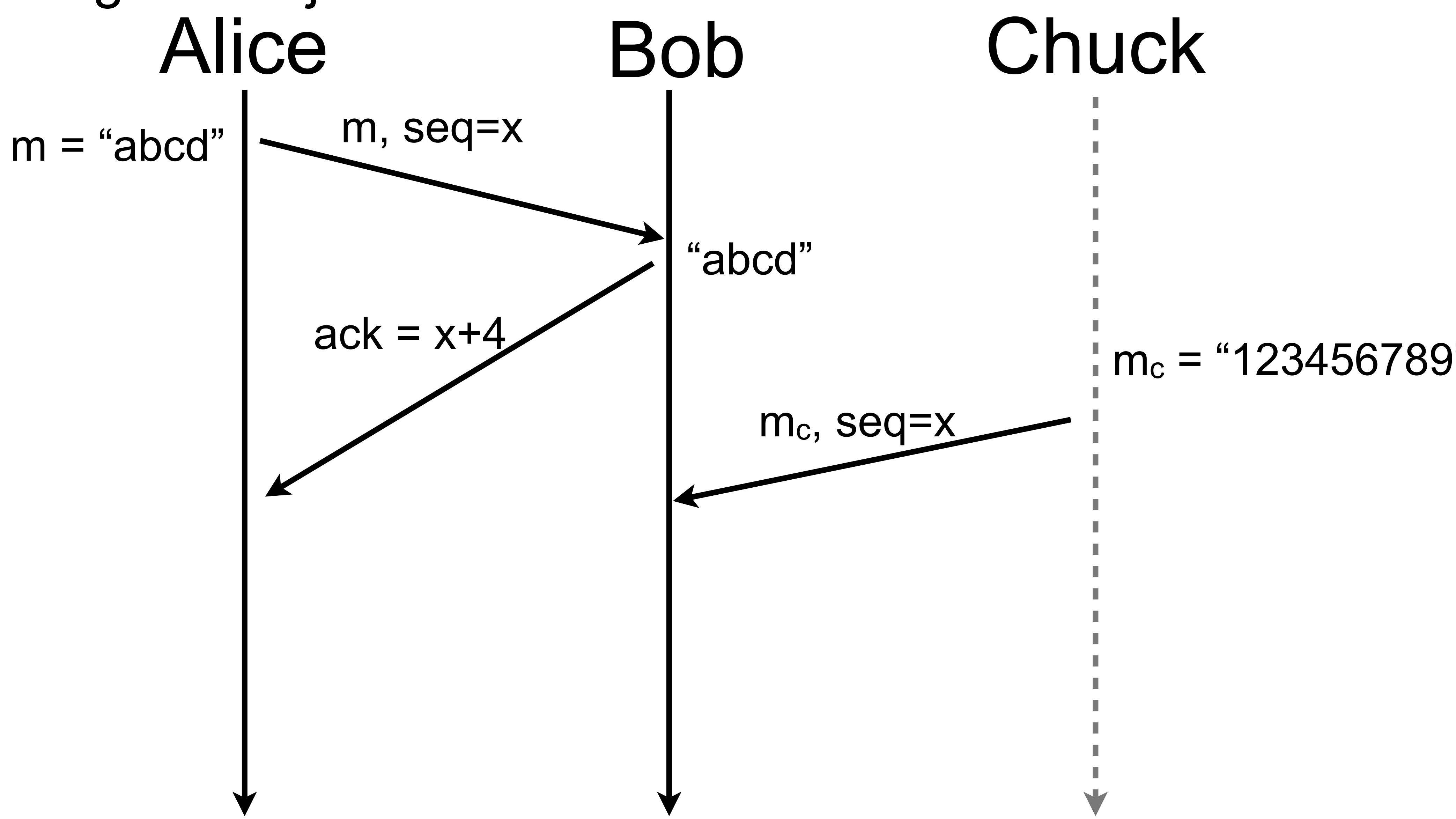
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



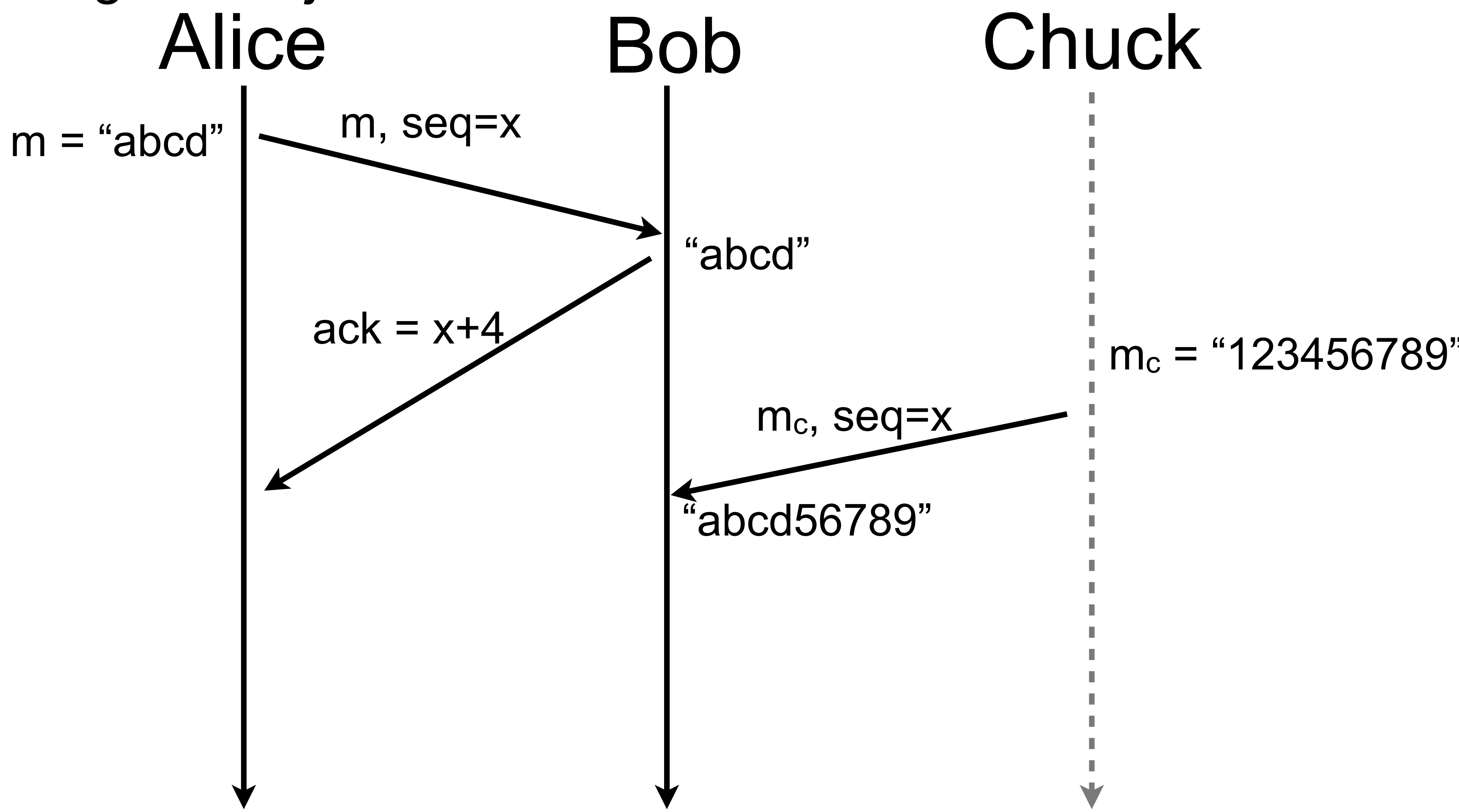
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



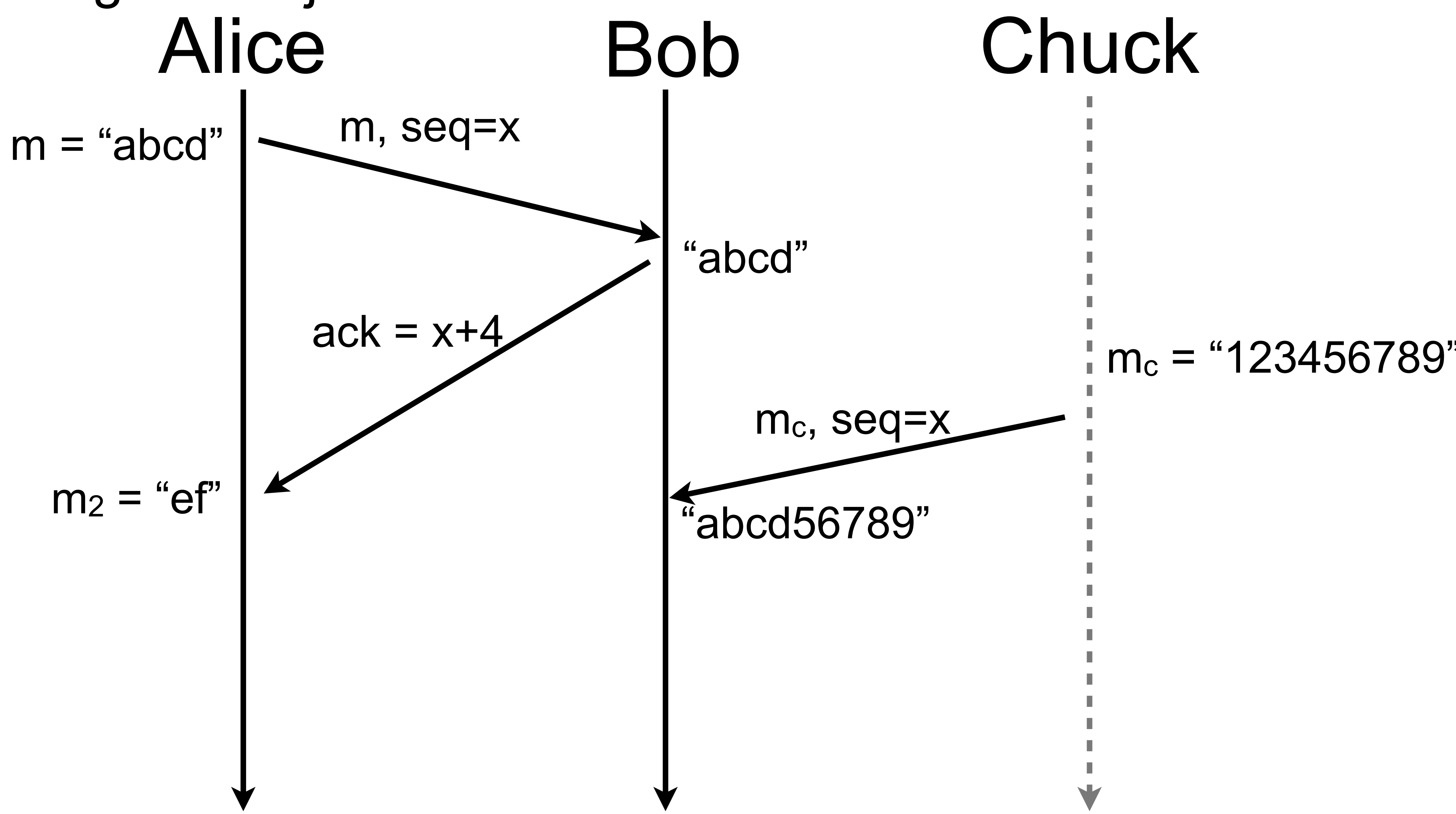
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



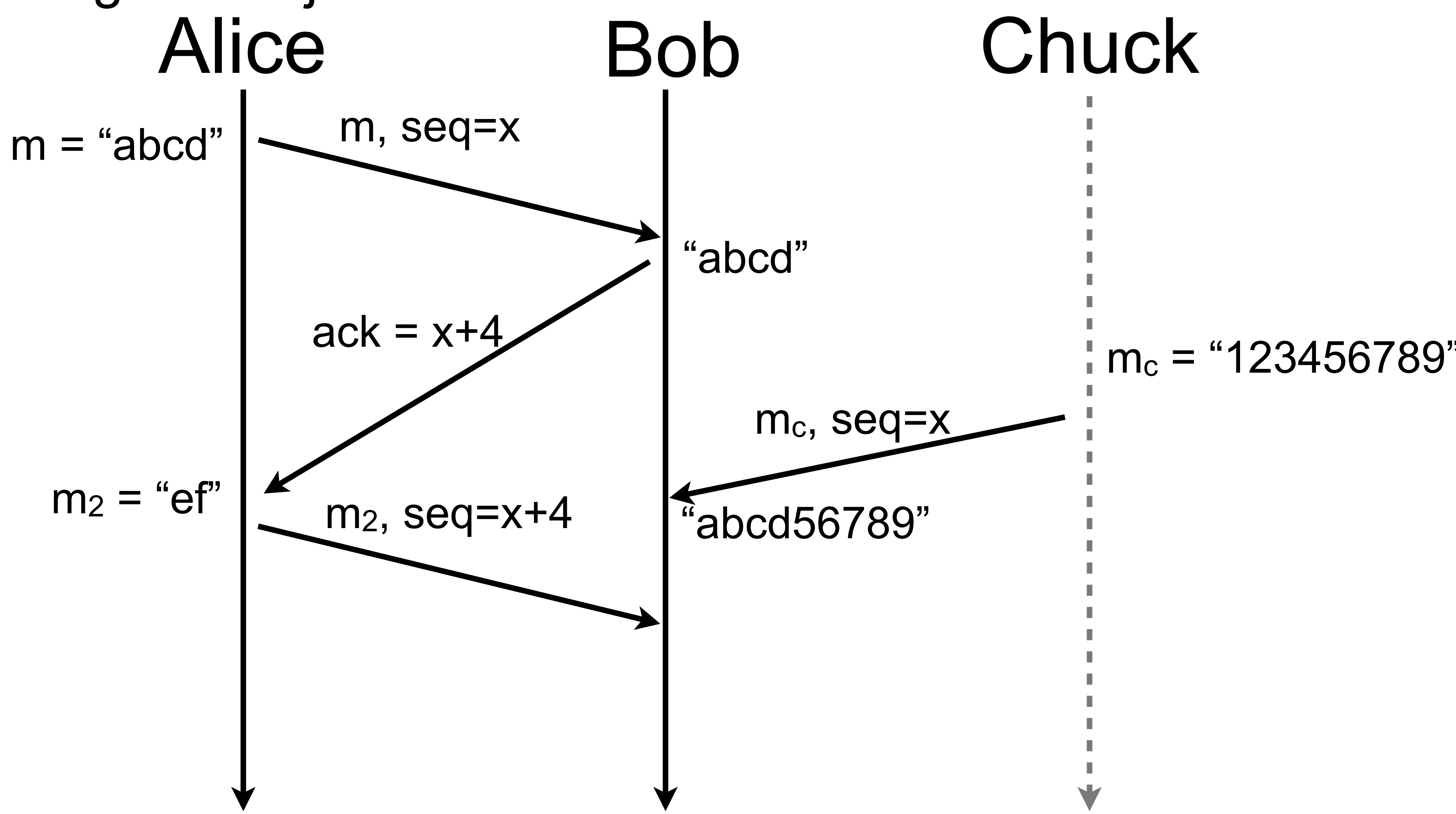
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



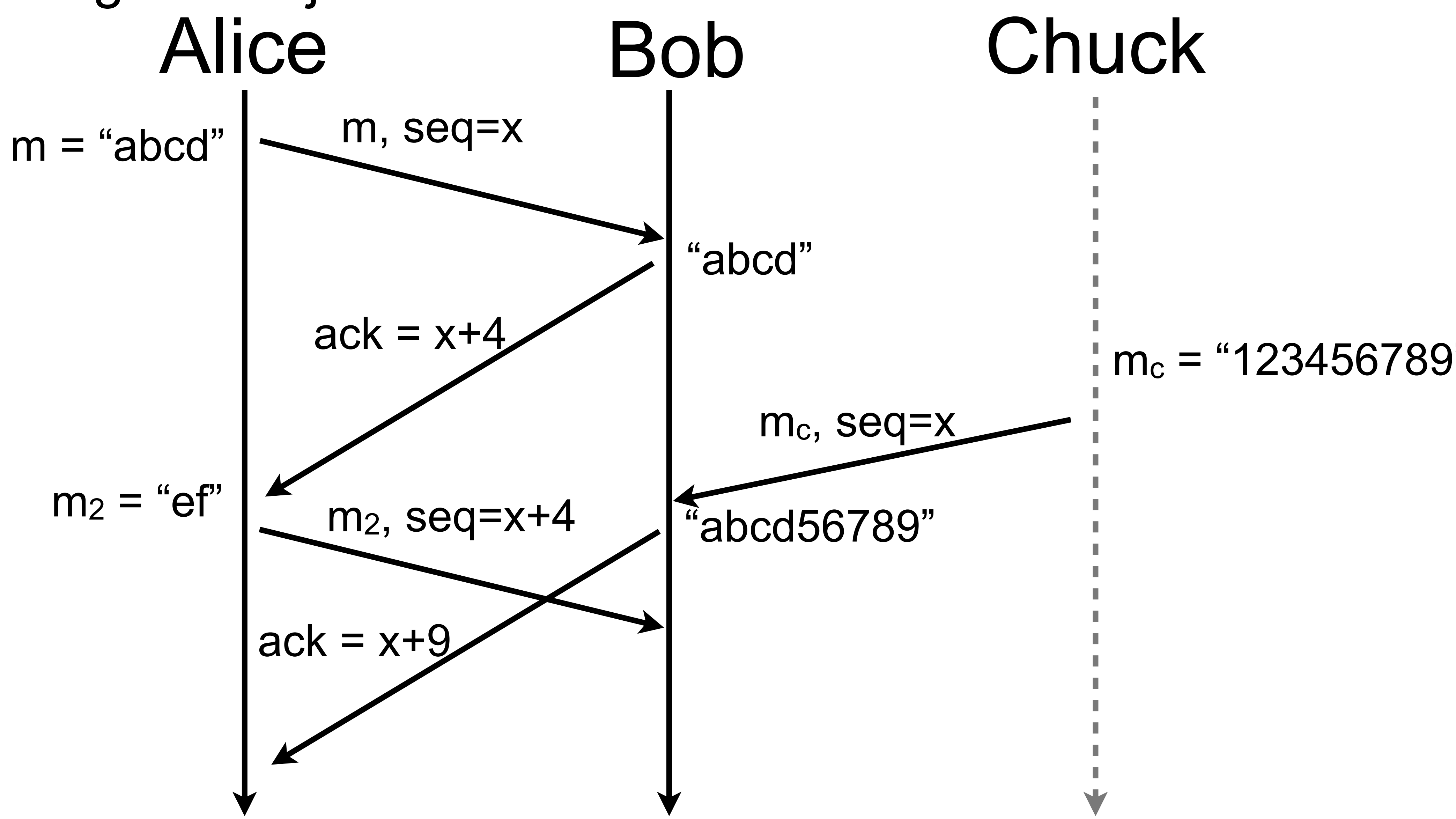
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



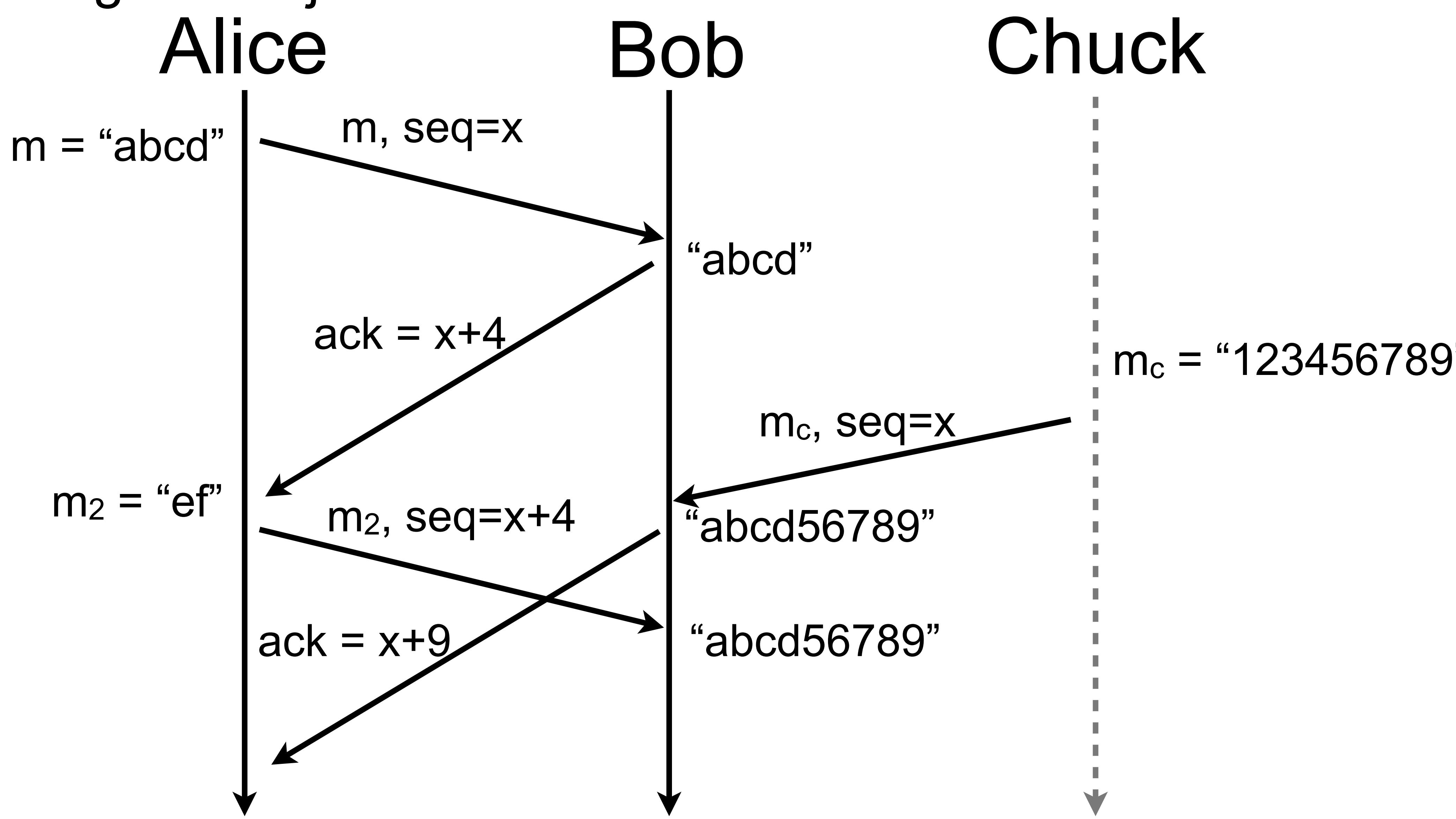
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



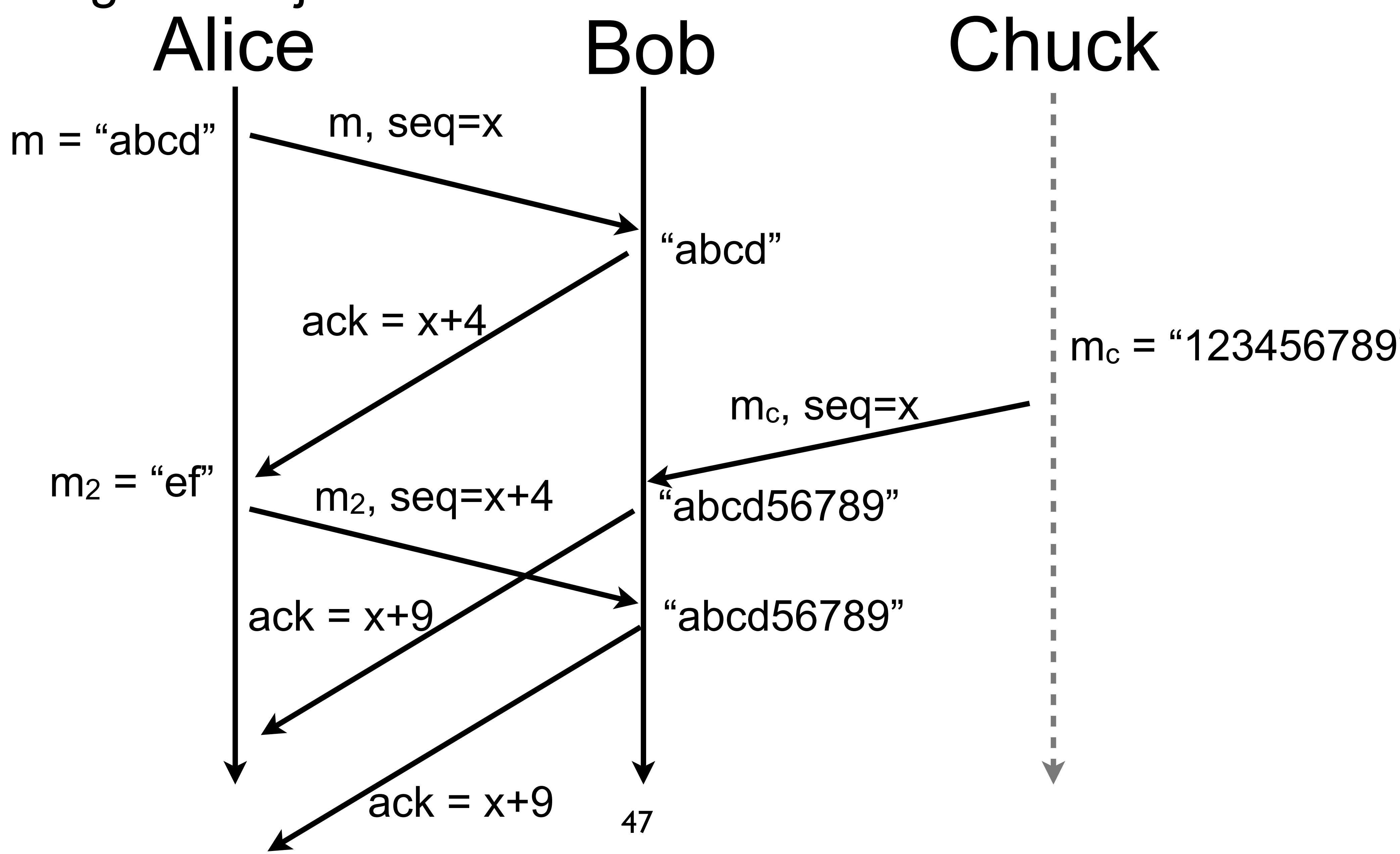
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



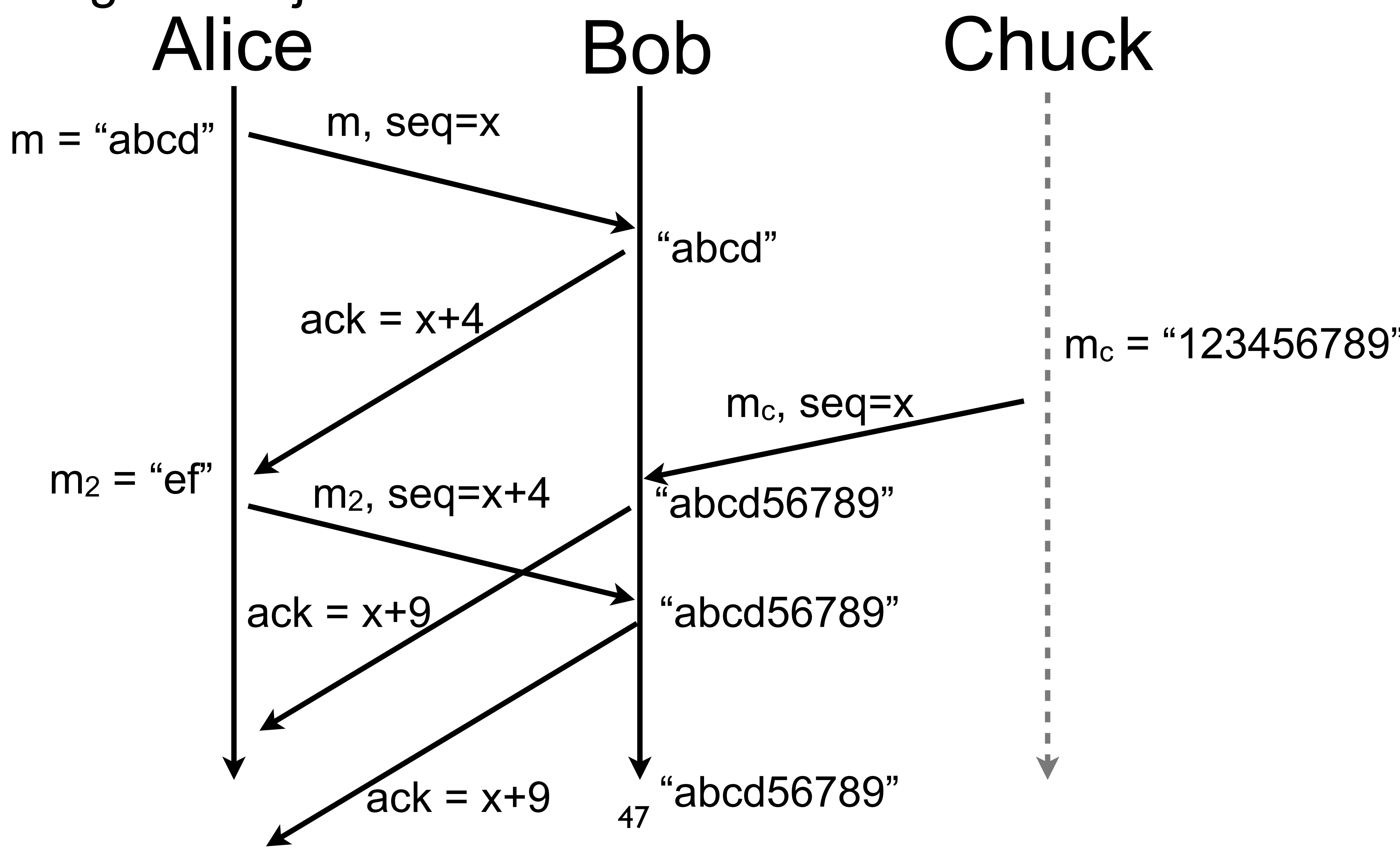
Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



Nonce (contd.)

- TCP sequence number does not protect against segment injection attacks in TCP



Nonce (contd.)

- TCP segment injection attack can be mitigated for short connections when there is not eavesdropping by
  - setting the initial sequence number with a good nonce, but sequence number is short (32 bits)
  - only allowing reception of segments that fit in the window
  - keeping small enough window (attackers can try a lot of sequence numbers on 1Gbps links!)
- In case of eavesdropping or long connections, segments should be authenticated
  - TCP MD5 option [RFC2385] tags every segment with its MD5 hash (without options and checksum) and a secret shared between Alice and Bob

Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

DoS attacks are still possible!

Denial of Services

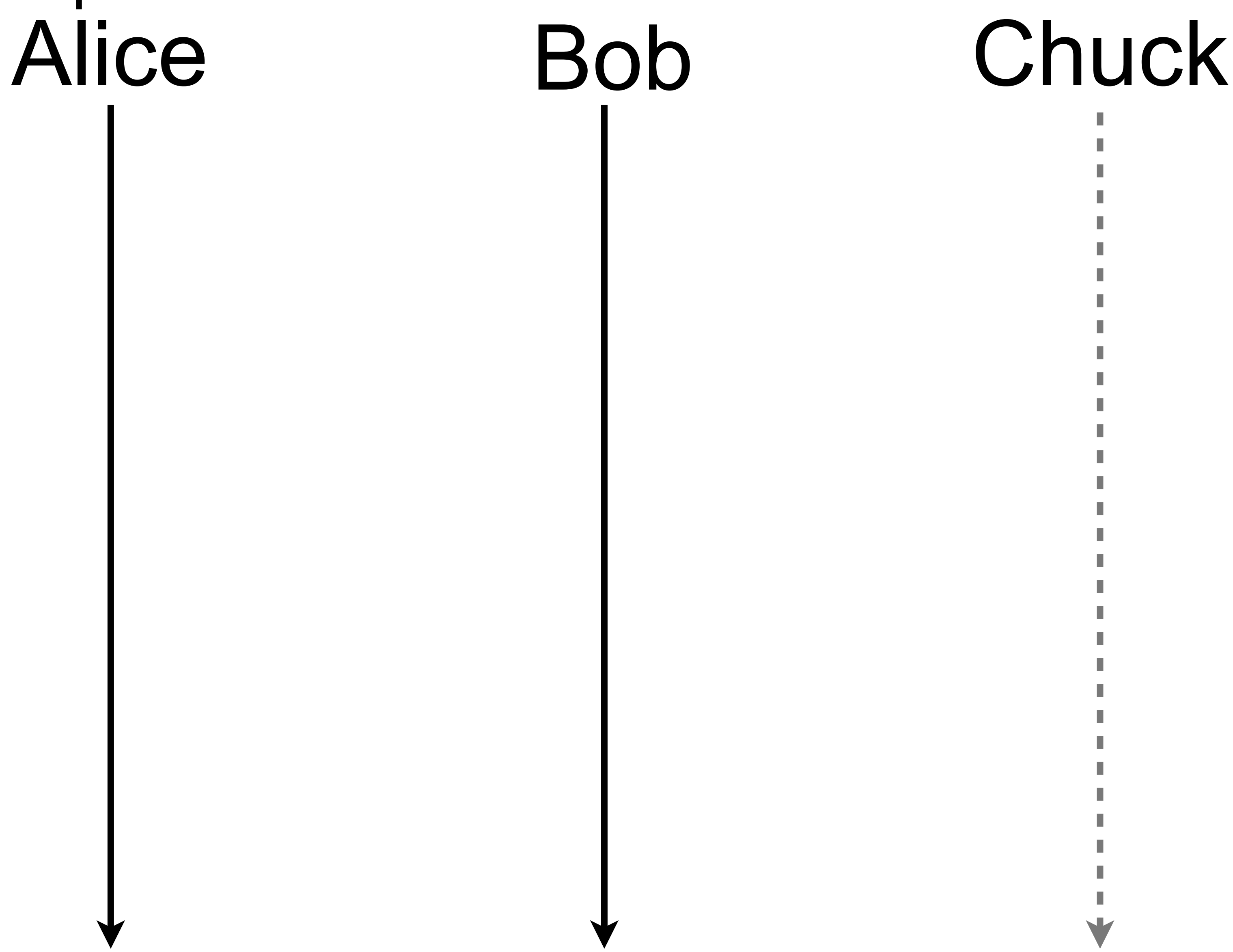
- Resources are always limited
  - e.g., processor, memory, link capacity
- The easiest way of leading a DoS is to overwhelm CPUs, memory, or links of the target
- A more complicated way is to manage an intrusion and neutralize the target
  - imagine you gain administrative access to border router of your network!

Danger of state

- Establishment and maintenance of session requires state
  - often maintained in “tables” with predefined capacity
- An attacker can saturate state tables by initiating multiple sessions
- Principle
  - require attacker to maintain state before maintaining state yourself
  - in general it is too costly for an attacker to maintain state

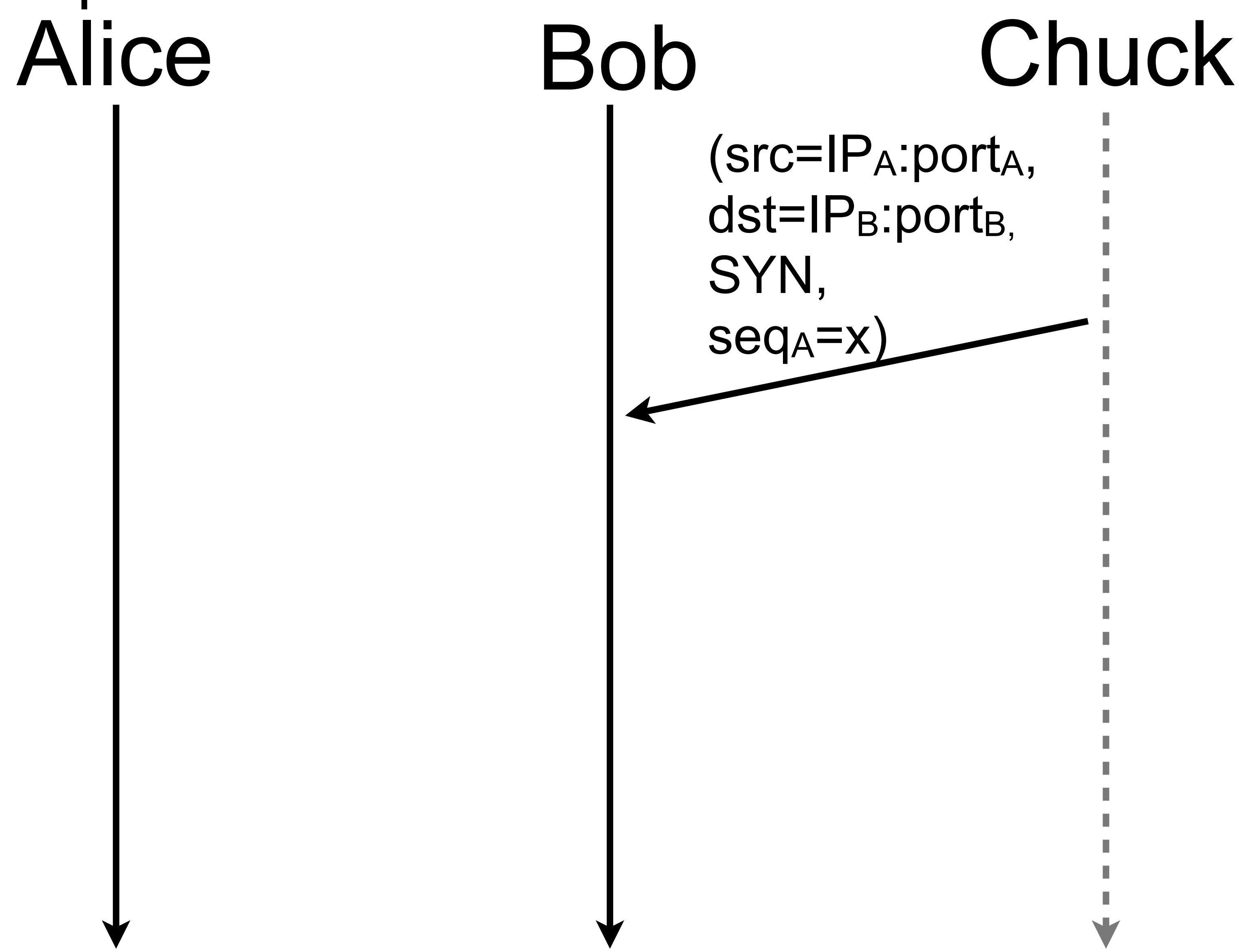
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



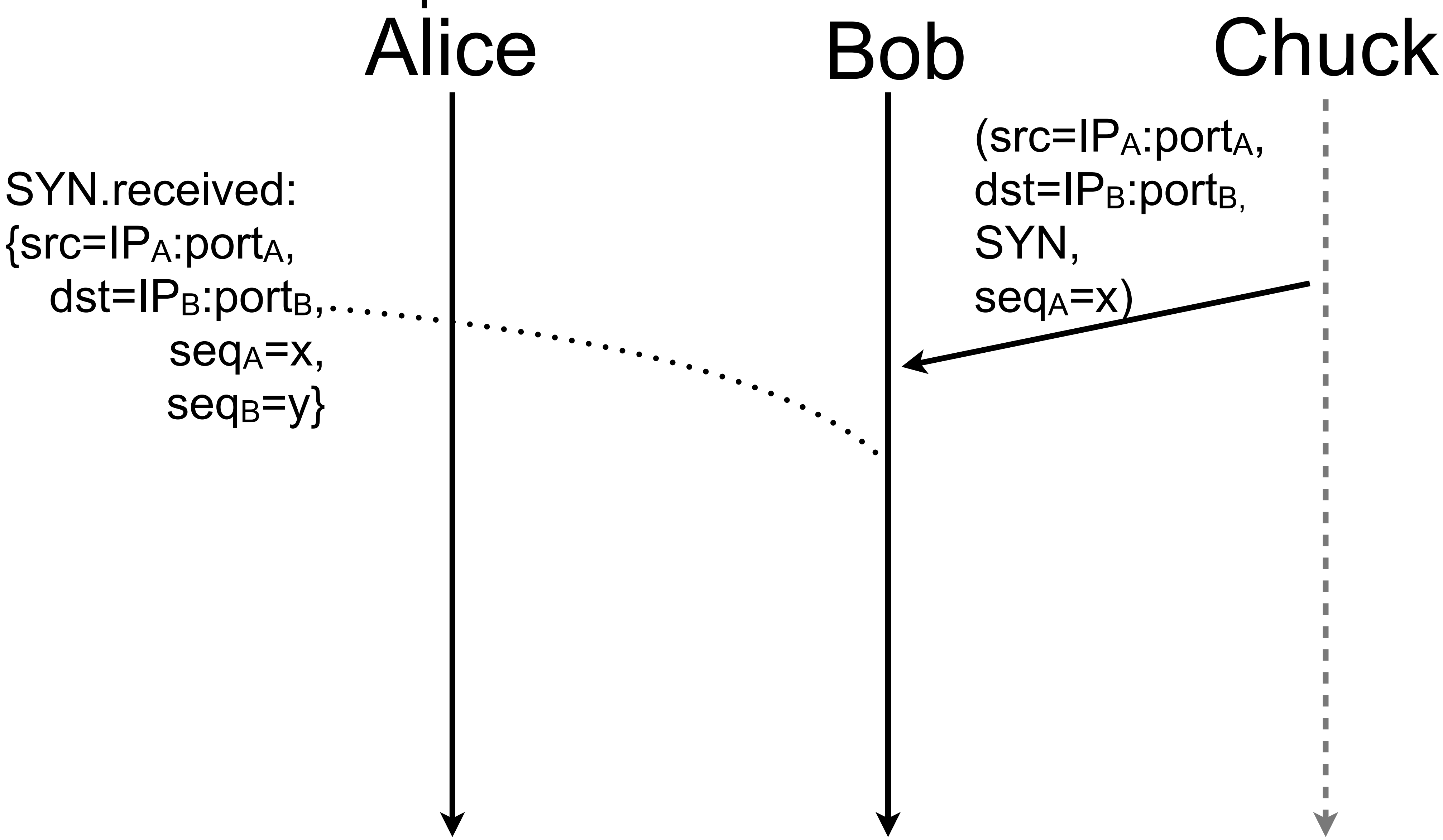
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



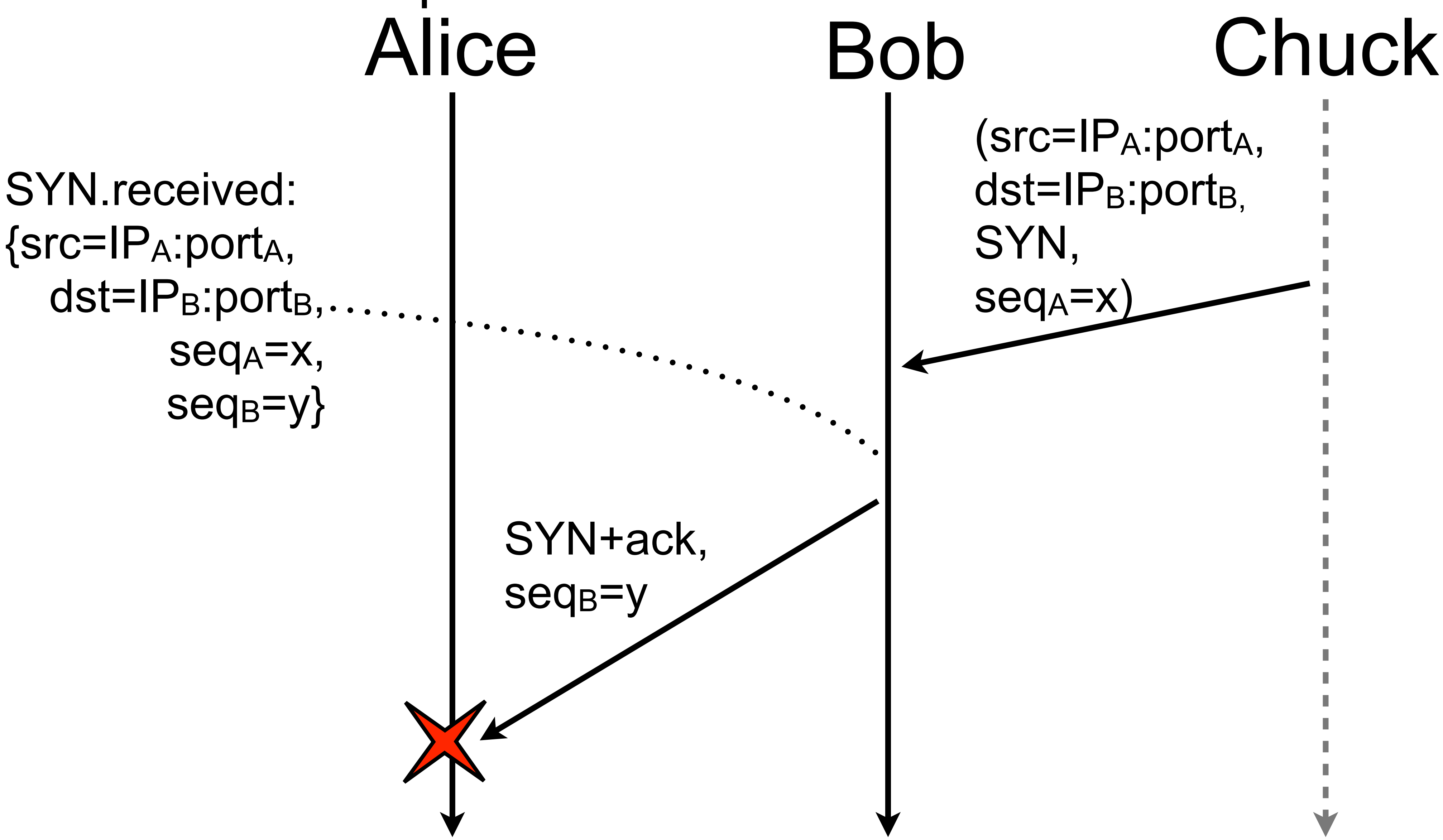
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



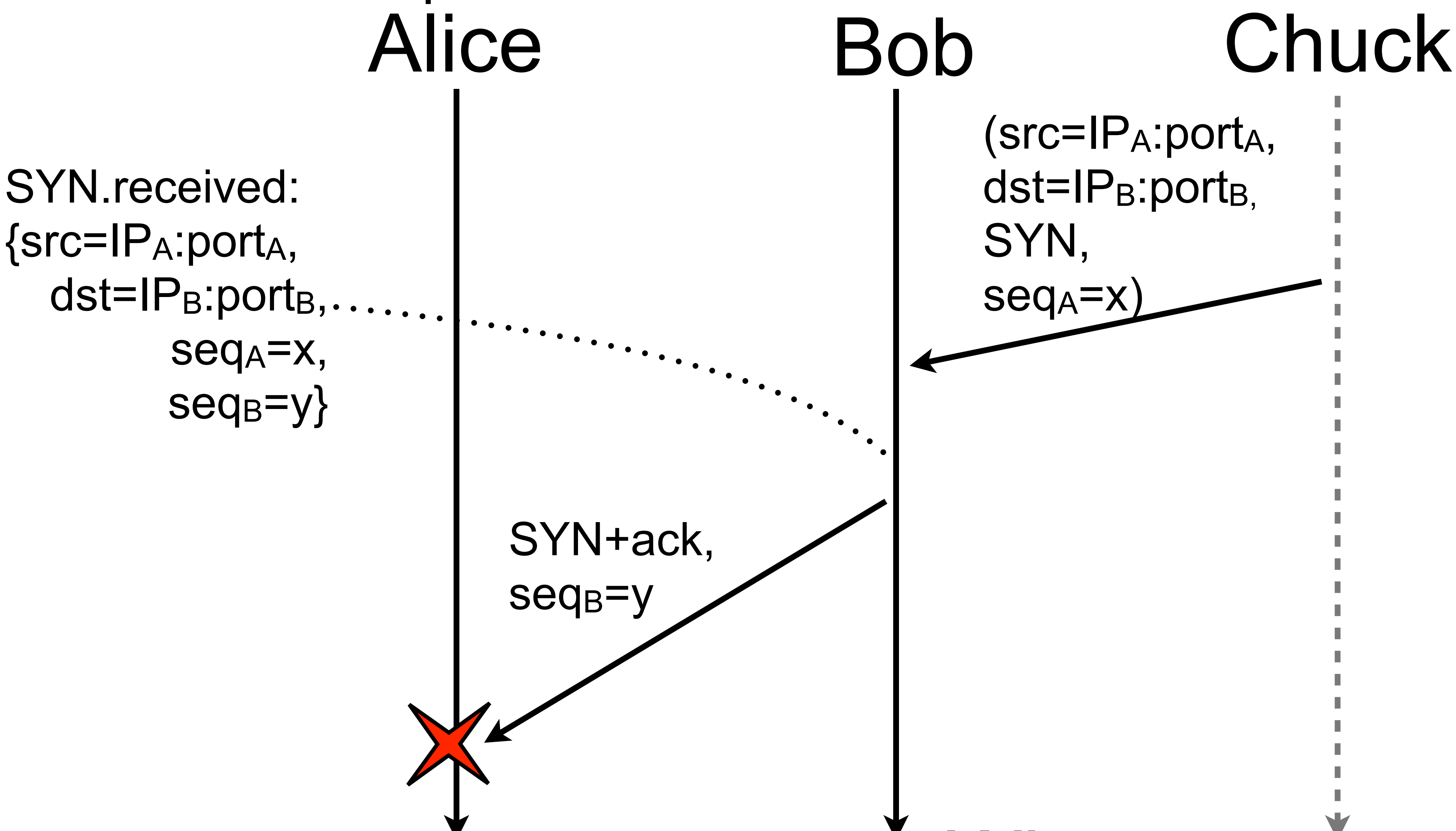
Danger of state (contd.)

- TCP relied on a state machine started upon reception of a SYN packet



Danger of state (contd.)

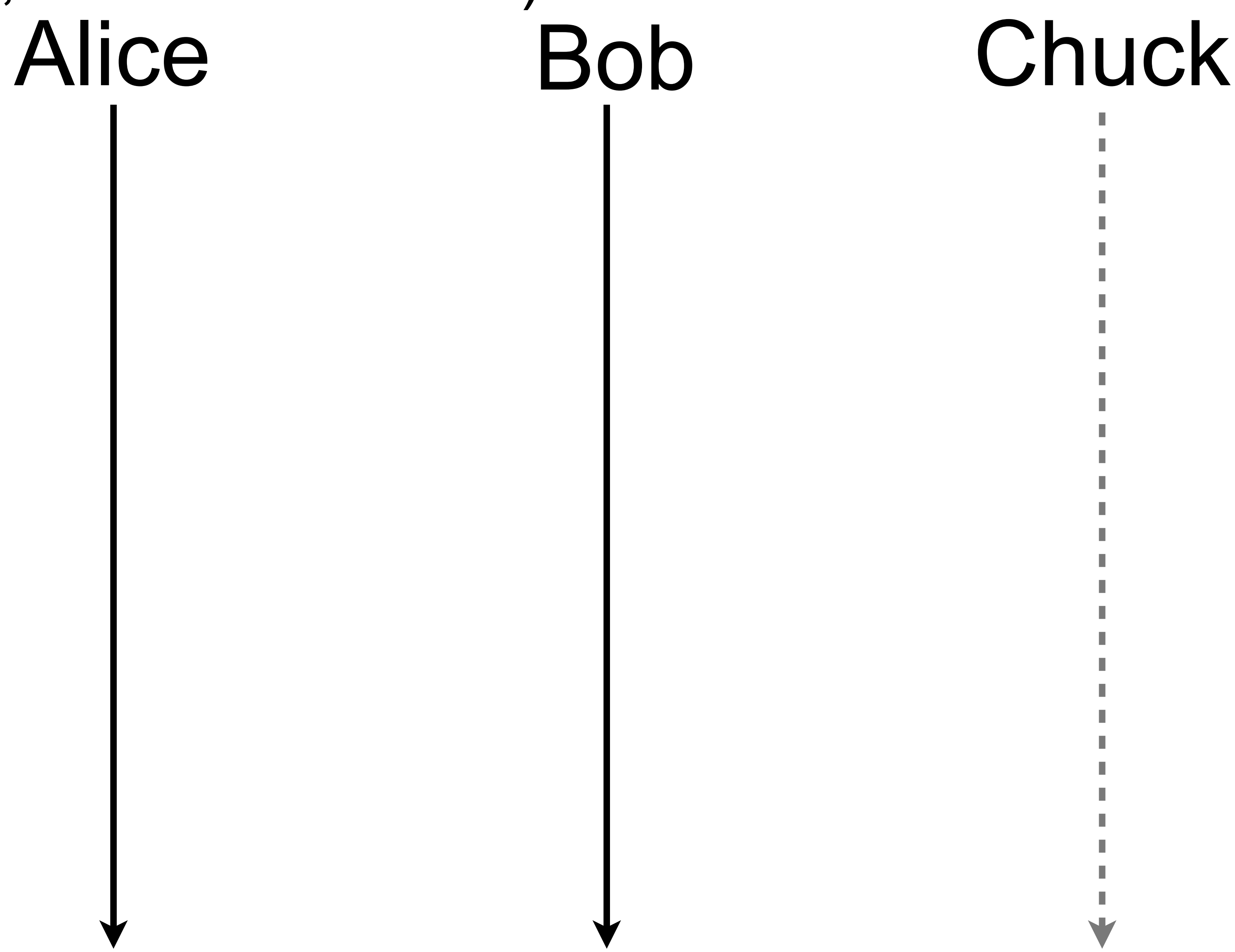
- TCP relied on a state machine started upon reception of a SYN packet



When to remove state?

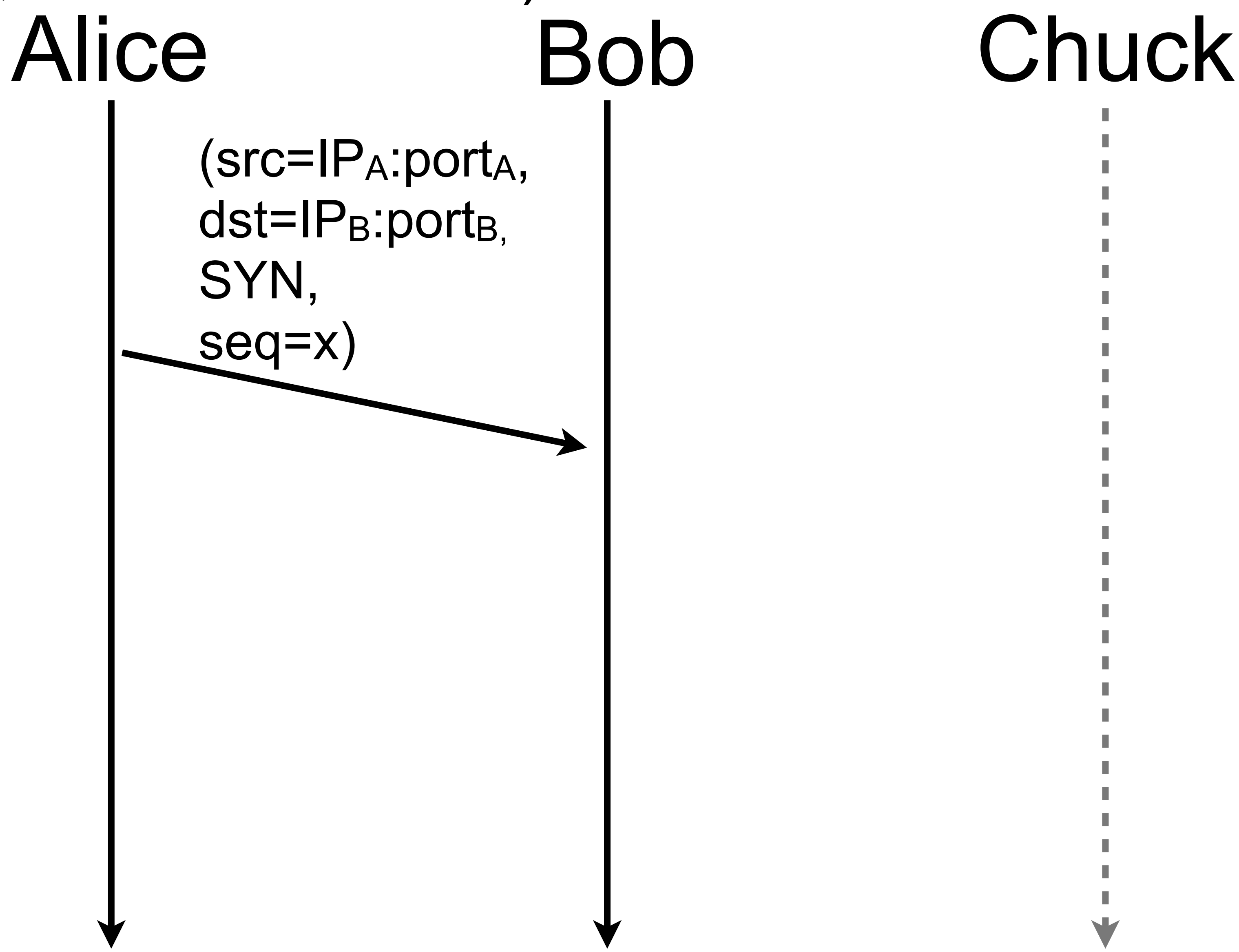
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



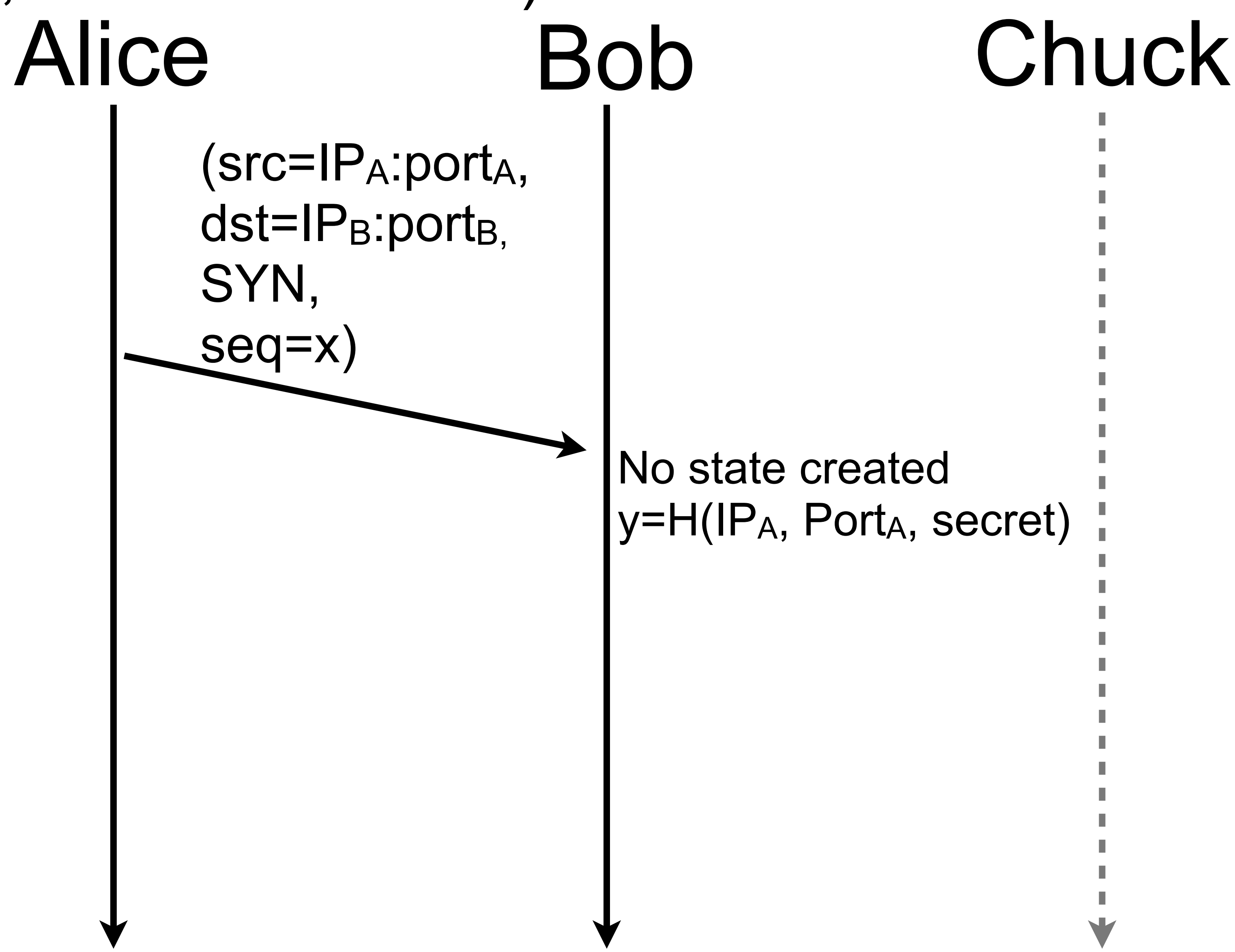
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



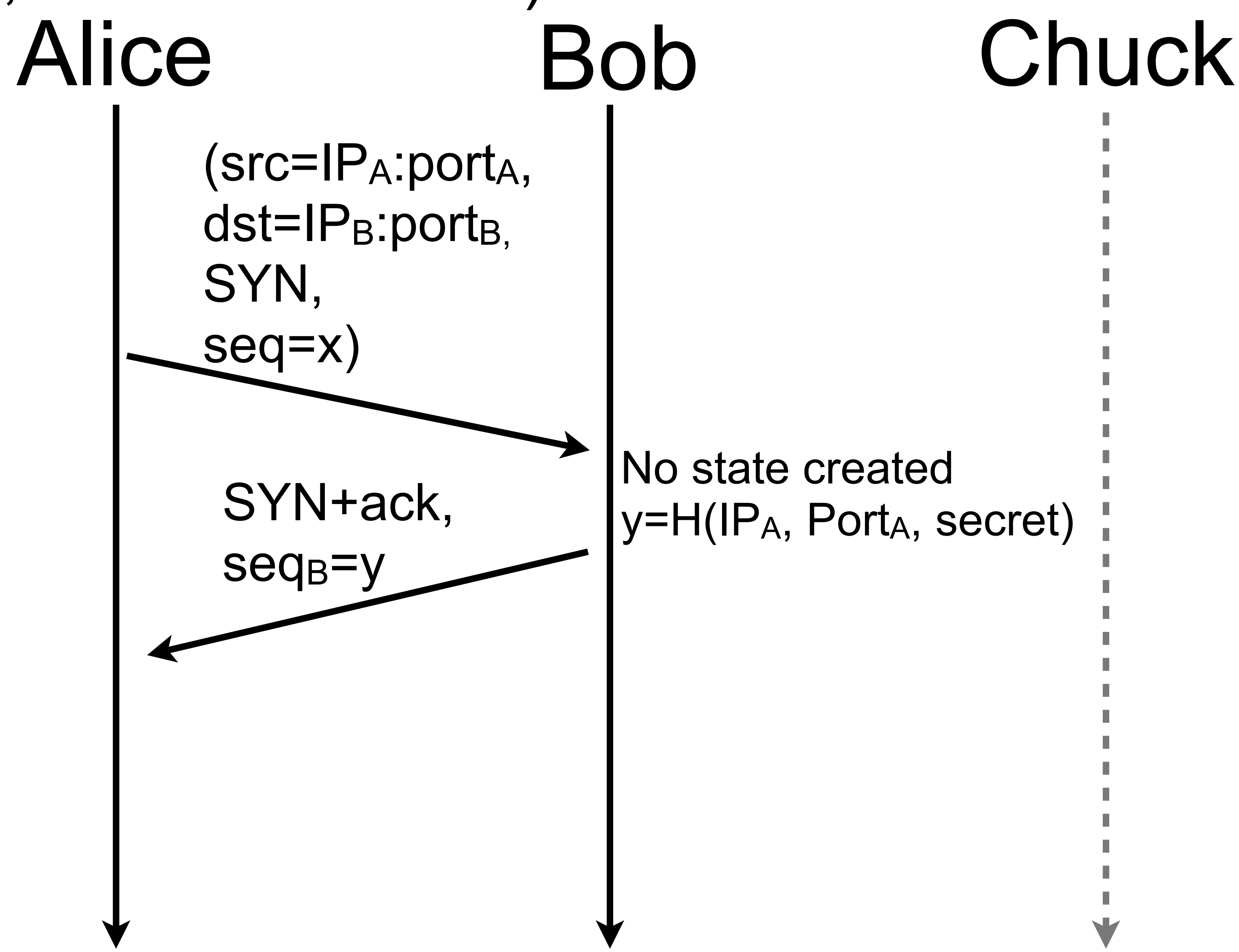
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



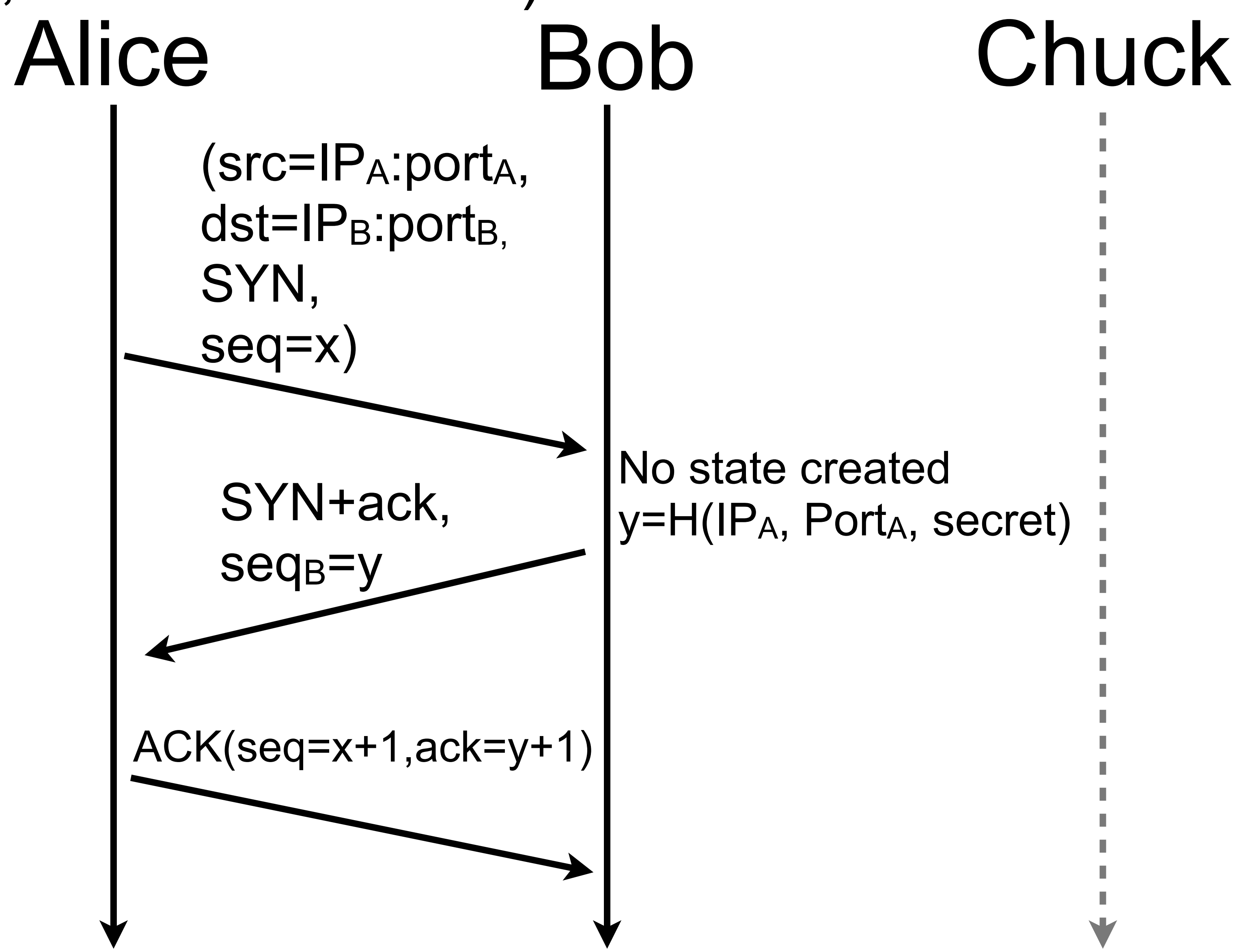
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



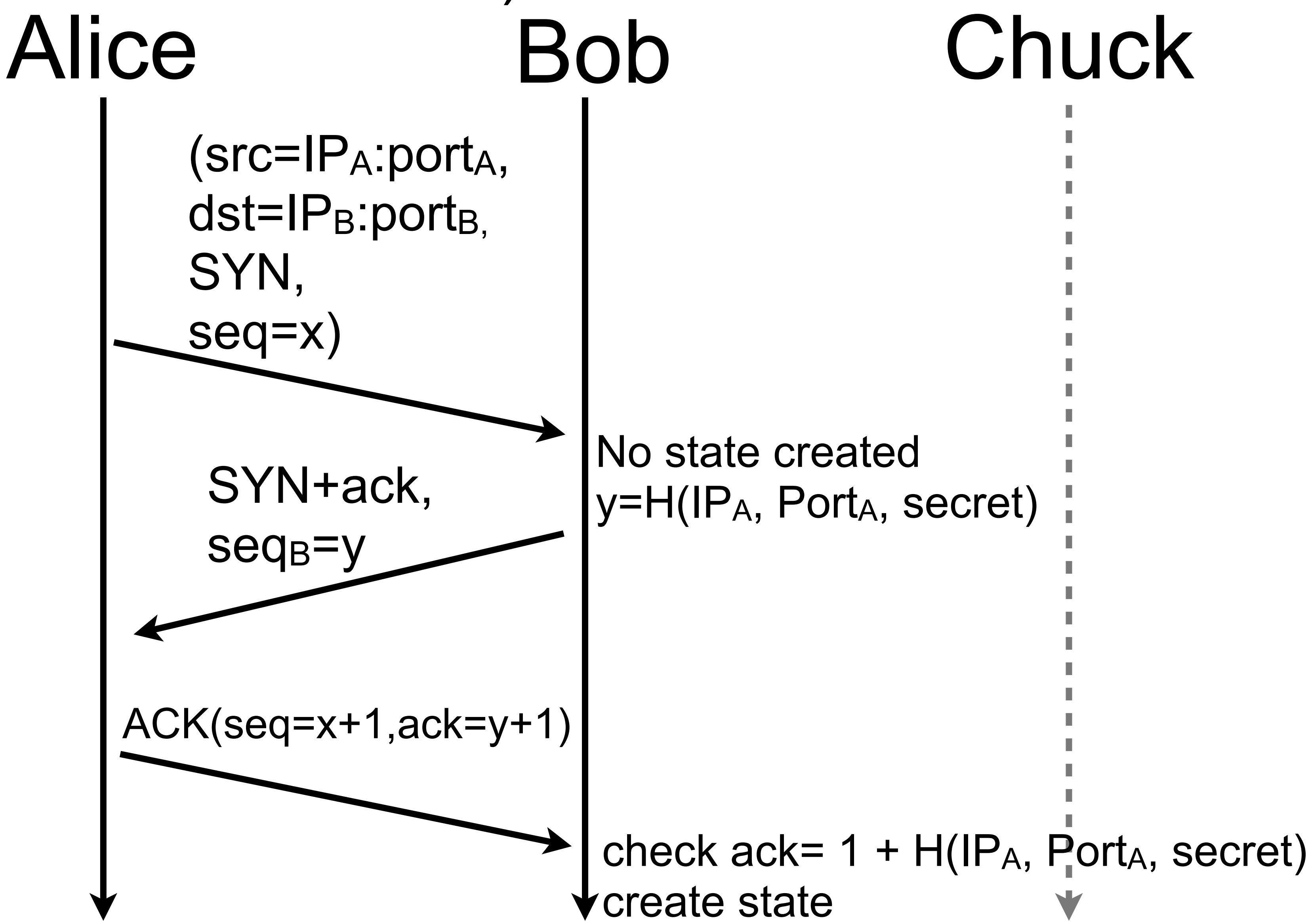
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



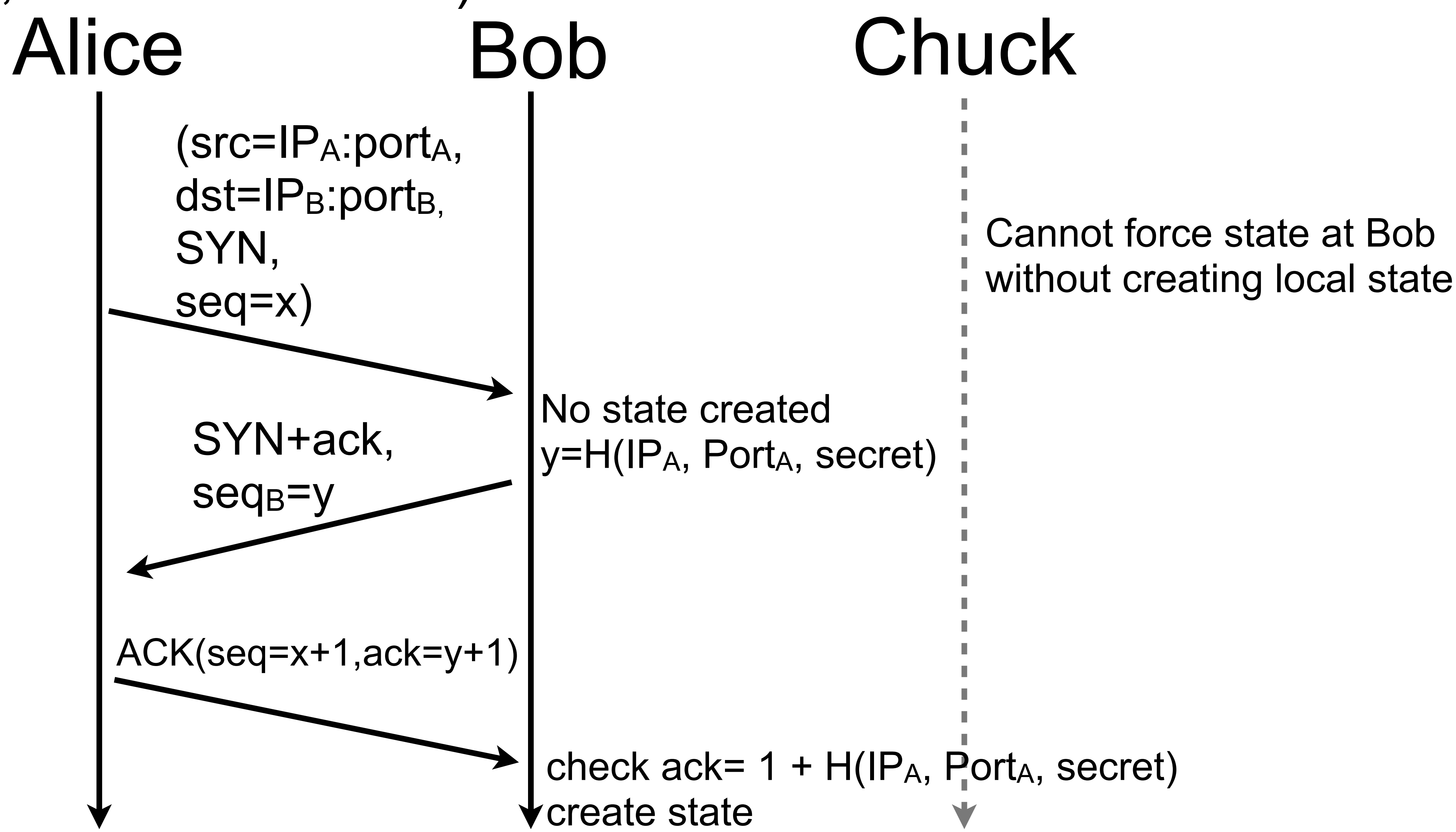
Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Danger of state (contd.)

- Always create state at the end of session establishment (e.g., TCP SYN cookie)



Danger of complexity

- Protection mechanism can be complex and can require important processing power
- An attacker can overwhelm her target CPU by triggering protection mechanisms
- Principle
  - require attacker to perform more processing than yourself
  - in general an attacker does not want to have to do heavy computation

Danger of complexity (contd.)

- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
    - e.g., Bob asks Alice to find a J such that the K lowest order bits of H((N,J)) are zeros. N is a nonce and K sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don't want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA

Danger of complexity (contd.)

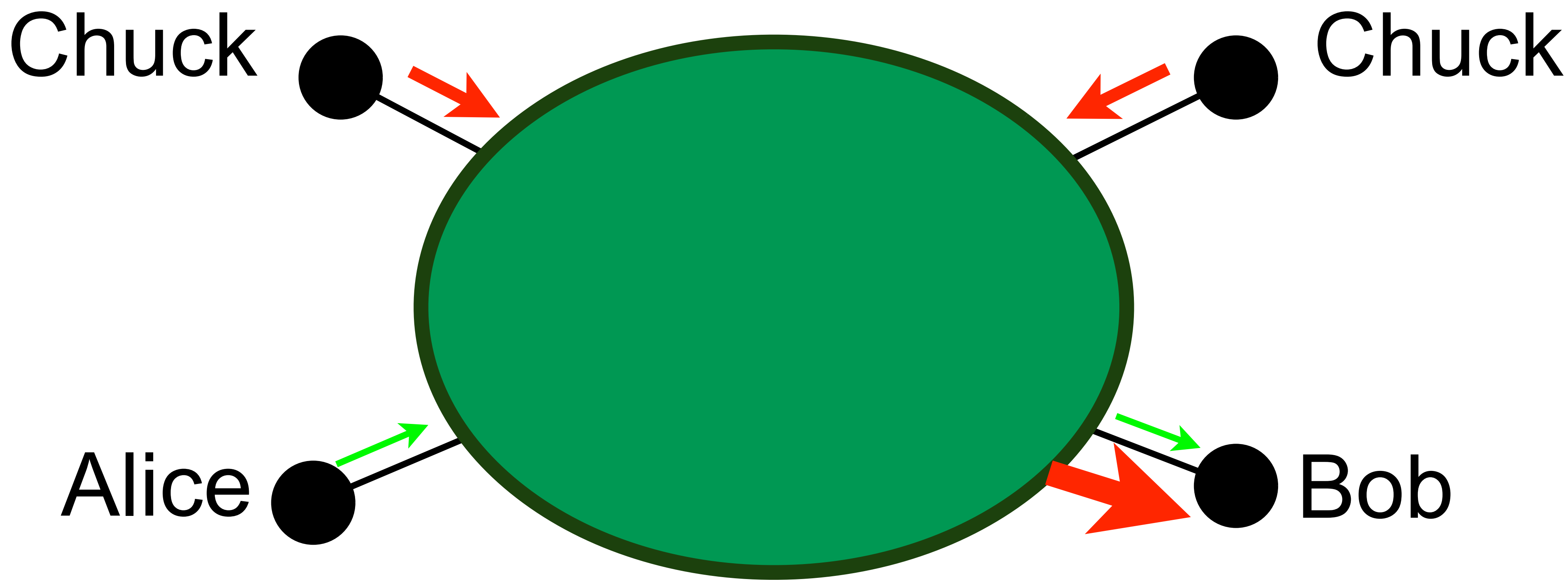
- Hard, if not impossible, to remove processing requirements but still possible to force the attacker to succeed some challenges to get access. This technique is usually called challenge-response
  - time challenges
    - when an attack is suspected, force the attacker to wait or slow down but the DoS protection can lead to a DoS
      - e.g., rate limiting
  - mathematical challenges
    - ask the initiator to solve a mathematical challenge that is hard to compute but easy to check, this might negatively impact legitimate clients
    - e.g., Bob asks Alice to find a J such that the K lowest order bits of H((N,J)) are zeros. N is a nonce and K sets the complexity of the puzzle, both parameters are decided by Bob [RFC5201]
  - human processing challenge
    - some services are reserved for users and don't want to be accessed by bots
    - ask Alice to succeed a challenge that is simple for a human but hard for a computer
      - e.g., CAPTCHA

Link overloading

- Messages are sent to Bob by traversing links
- If an attacker can send packets at a high enough rate, she can saturate links toward Bob and make him unavailable
- Unfortunately, Bob cannot make anything to block packet before they reach him
- Principle
  - tweak the network to not suffer too much of such attacks

Link overloading (contd.)

- Example of Distributed Denial of Service (DDoS) attack



Link overloading (contd.)

- A first parade is to filter illicit traffic before it can harm the target
  - e.g., firewall, access lists
- A set of rules is specified a priori, if the traffic does not match the rules, it is discarded
  - always block everything but what is acceptable

Link overloading (contd.)

- Filtering based on origin
  - useful to avoid spoofing
    - e.g., block any packet which source address does not belong to the customer cone of a BGP neighbor
  - does not work so well as it depends on every network between the origin and the target
- Filtering based on traffic pattern
  - analyze the traffic and if it deviates from what is normal, drop it
    - e.g., drop malformed packets, rate limit a source if it sends too much SYN packets, ignore mails from well known SPAM servers, block any flow initiated by the outside if there is no server in the network

Network Intrusion Detection System (NIDS)

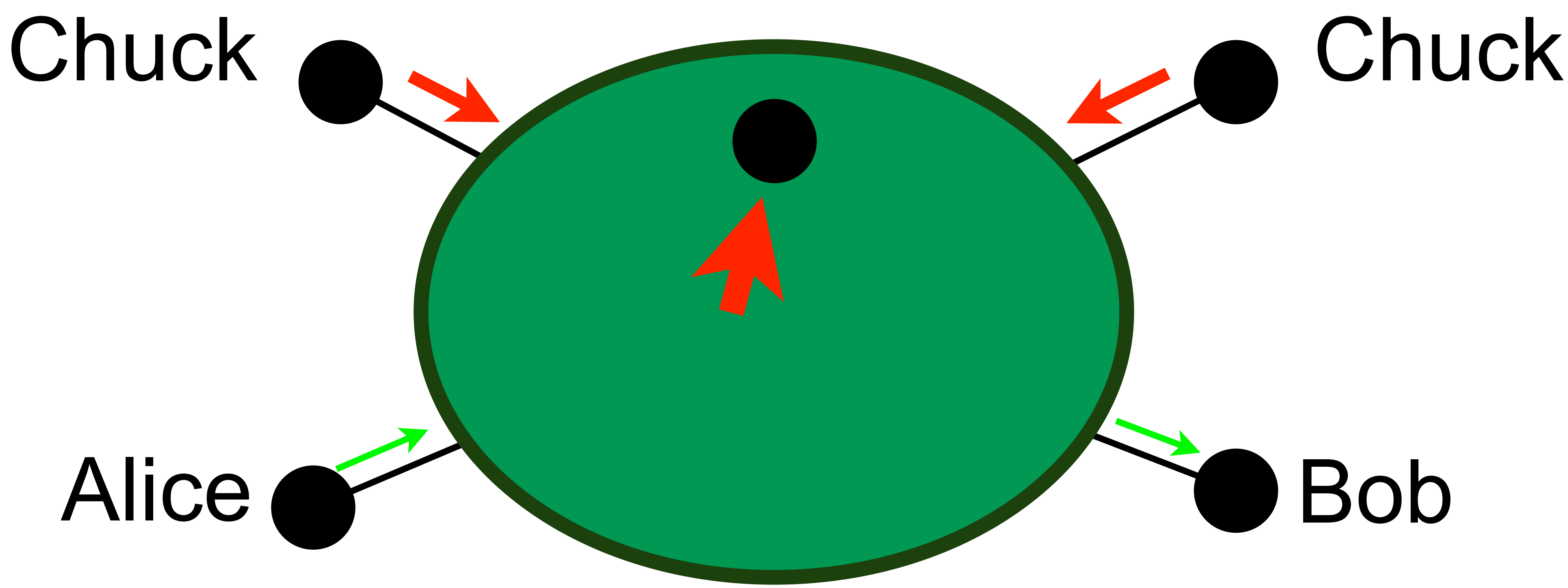
- An NIDS aims at discovering non-legitimate operations
- The NIDS analyses the traffic to detect abnormal patters
- Upon anomaly detection, the NIDS triggers an alert with a report on the anomaly
- NOC follows procedures upon detection

Network Intrusion Detection System (contd.)

- Signature based detection
  - a database of abnormal behavior is maintained to construct a signature for each attack
  - if the traffic corresponds to a signature in the database, trigger an alarm
  - risk of false negative (0-day attack)
  - e.g., Snort, Bro, antivirus
- Outlier detection
  - the anomaly detector learns what is the normal behavior of the network
  - went an outlier is detected, an alarm is triggered
  - risk of false positive and false negative
  - e.g., cluster analysis, time series analysis, spectral analysis

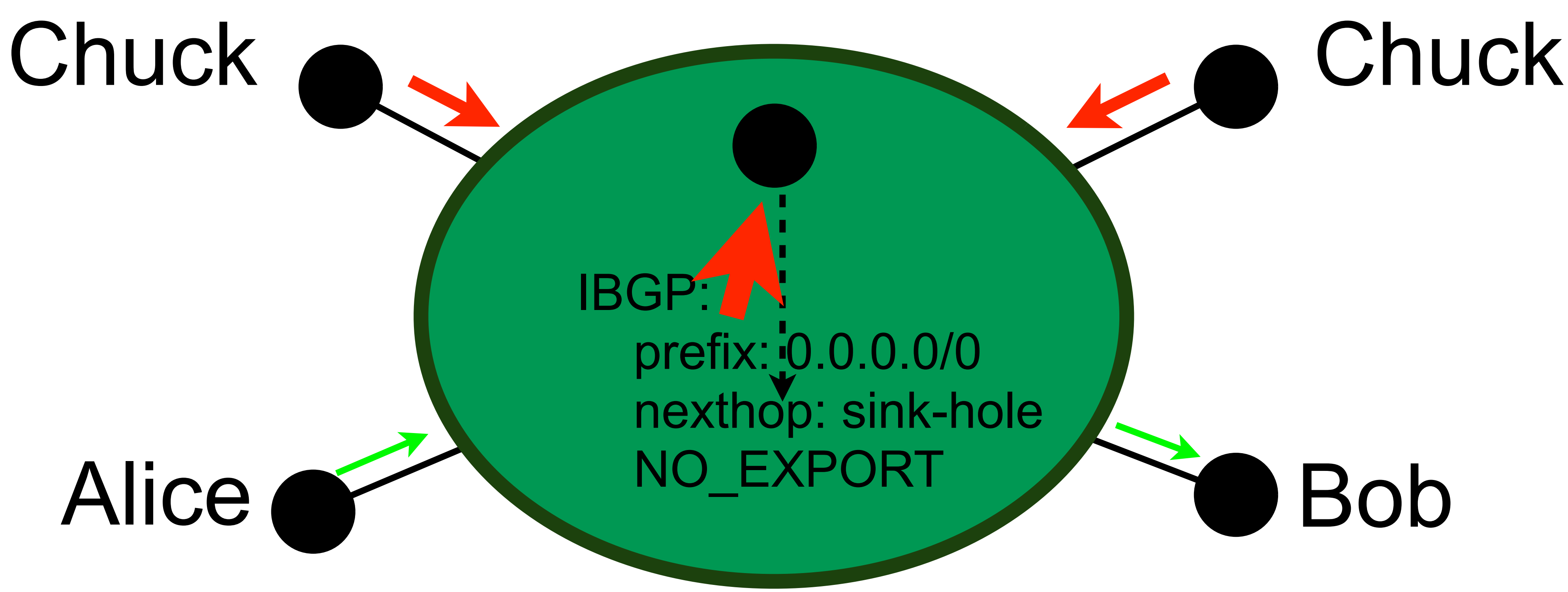
Link overloading (contd.)

- Attacks are often to random destinations or with random sources
  - backscatter traffic to a sink-hole that can receive a lot of traffic attack without impacting the network



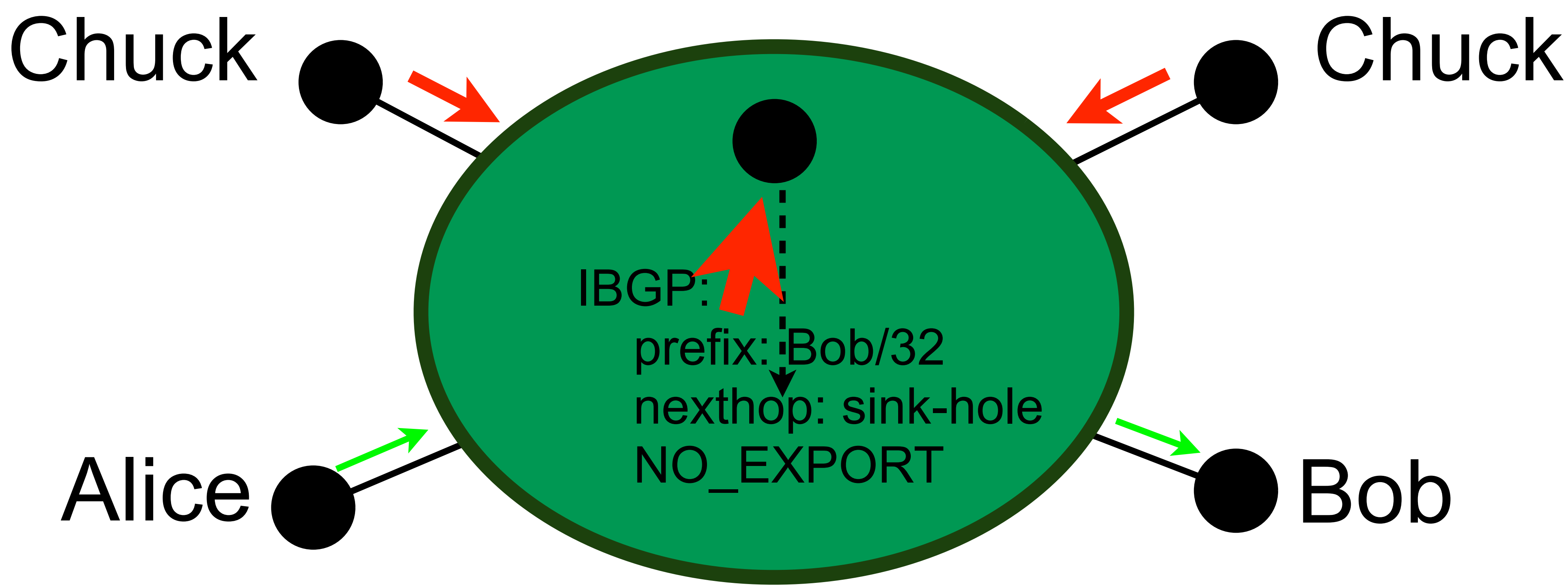
Link overloading (contd.)

- Use the sink-hole to attract bizarre packets



Link overloading (contd.)

- Use the sink-hole to protect the target



Problem solved?

- fill me
- fill me
- fill me

Problem solved?

- fill me
- fill me
- fill me

Relay attacks are still possible!

Relay attack

- In a relay attack, Chuck does not contact Alice directly but goes via Bob
- If the traffic from Bob to Alice is bigger than the traffic from Chuck to Bob, the attack is called amplification attack
- As for DoS, hard to protect correctly against relay attacks
  - use filters (e.g., deactivate ICMP)
  - authentication of the source
    - but correct spoofing protection that doesn’t open a relay attack door is very hard to deploy in practice as it requires messages in both directions between parties

What did we miss?

What did we miss?

- To terminate the session!
  - with the same care as the opening of the session
  - this is often neglected

Perfect Forward Secrecy

- With perfect forward secrecy (PFS), Eve cannot decrypt messages sent between Alice and Bob
  - even if she captures every message
  - even if she breaks into Alice and Bob after the communication to steal their secrets (e.g., private keys)

Perfect Forward Secrecy (contd.)

- PFS is provided using ephemeral keys
  - the ephemeral key is generated and used only during the session
  - the session key is not stored after the communication
  - the session key is independent of stored information (e.g., good PRNG)
  - for long sessions, change the session key regularly

Perfect Forward Secrecy (contd.)

1. Initiate the communication between Alice and Bob
   - authenticity proven with public/private key pairs
2. Alice and Bob agree on a secret K
   - use Diffie-Hellman
     - authenticate DH messages with public/private key pairs
3. Encrypt/Decrypt messages with symmetric cryptography using K as the key
   - no need to sign as it is encrypted
   - be sure a nonce is used to avoid replay
4. If session is too long, back to 2.
5. Close the session correctly and be sure K is not stored anywhere

Operational corner

Timing cryptanalysis

- Public-key cryptography is complex
  - processing time depends on data
- An attacker that can frequently measure the time necessary to decrypt (or sign) some data, she can determine the private key that is used
  - the public key is obtained by analyzing crypt (or check) but not really useful as it is already public!
- Countermeasures
  - randomize operation time is not effective
  - ensure that any operation using the private key takes a fixed amount of time

Research corner

How would you protect BGP against prefix hijacking?

- fill me
- fill me
- fill me

Homework

due date 02/15/2013

Exam

- Everything is part of the exam, even homework, operational corners, and research corners
- No book, no computer, no (smart)phone