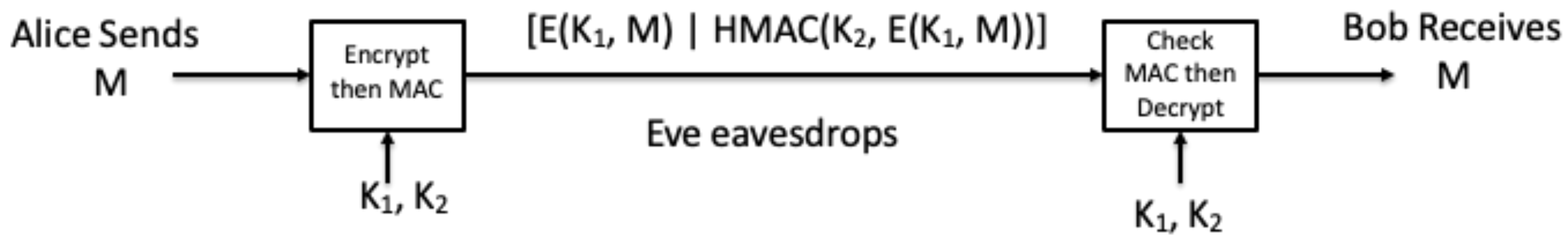
CSE 543: Computer Security
Module: Applied Cryptography

Prof. Syed Rafiul Hussain
Department of Computer Science and Engineering
Pennsylvania State University

Recap of Symmetric Key Cryptography

- Without knowing K1, Eve can't read M
- Without knowing K2, Eve can't compute a valid MAC
- Problem
  - How do Alice and Bob securely share their keys?



Public Key Cryptography

- Public Key cryptography
  - Each key pair consists of a public and private component: k^+ (public key), k^- (private key)

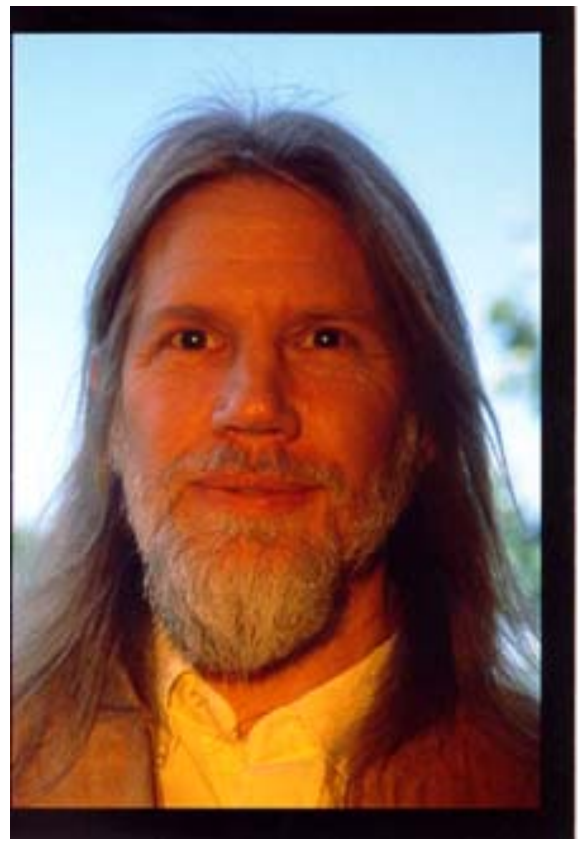
$$D(E(p, k^+), k^-) = p$$

$$D(E(p, k^-), k^+) = p$$

- Public keys are distributed (typically) through public key certificates
  - Anyone can communicate secretly with you if they have your certificate
  - E.g., SSL-based web commerce

Diffie-Hellman Key Agreement

- The DH paper really started the modern age of cryptography, and indirectly the security community
  - Negotiate a secret over an insecure media
  - E.g., "in the clear" (seems impossible)
  - Idea: participants exchange intractable puzzles that can be solved easily with additional information.



- Mathematics are very deep
  - Working in multiplicative group G
  - Use the hardness of computing discrete logarithms in finite field to make secure
  - Things like RSA are variants that exploit similar properties

Time for Revisiting Math

- **Divisibility:** an integer a divides b if b = ac for some integer c. This is denoted a | b
- **Prime:** an integer greater than 1 that is divisible by no positive integers other than 1 and itself
- **Greatest Common Divisor:** The GCD of two integers a and b is the largest integer n that divides both a and b
  - Denoted gcd(a, b) = n
  - Euclidean algorithm
- **Relatively prime:** Two integers a and b are relatively prime if gcd(a,b) = 1

Some Math for Cryptography

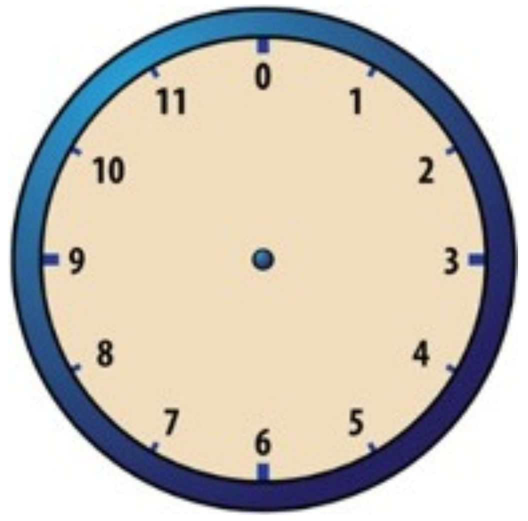
- $Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- $Z^+ = \{1, 2, 3, \ldots\}$
- prime vs. composite
  - prime divide by only itself and 1 (has to be positive)
  - 0, 1 are not prime numbers
- Prime factorization is unique
  - fundamental theorem of arithmetic
  - Any integer greater than 1 can be written as a product of primes
    - 12 = 2x2x3
- If GCD (a,b)=1, *a and b are relatively prime*

Modular Arithmetic

- Clock-face arithmetic
  - Modulo 12

- Remainder arithmetic
  - Think of this in the context of integer division
  - Anything with the same remainder after division by the modulus n is considered equivalent (mod n)

  - What is 7 + 11 (mod 12)?
  - 2 * 8 (mod 12)?
  - 52 (mod 12)?



Some Math for Cryptography

- Any integer can be written like n = d.q + r
  - d = divisor, q = quotient, r = remainder
- Two integers are congruent mod-N if their differences is divisible by N. In other words, a and b leave the same remainder when divided by N.
  - if a= p.N + r, and b = q.N+r, then (a-b) = (p-q) N
  - a ≡ b mod N (a is congruent to b w.r.t. modulo n)
  - a mod N = b mod N
- Modular Multiplicative inverse
  - a modular multiplicative inverse of an *integer a* is an *integer x* such that the product *ax* is congruent to 1 w.r.t modulus N.
  - ax ≡ 1 mod N
  - 3x ≡ 1 mod 10 ???

Euler's Totient Function

- Euler phi-function: for an integer n,
  $\phi(n)$ is defined as the number of positive integers that are:
  - Less than n
  - Relatively prime to n
- Multiplicative
  - For integers a and b such that gcd(a,b) = 1, $\phi(ab) = \phi(a)\,\phi(b)$
- For any prime p, $\phi(p) = p-1$
  - Example: Find $\phi(55)$

Diffie-Hellman Protocol

PennState

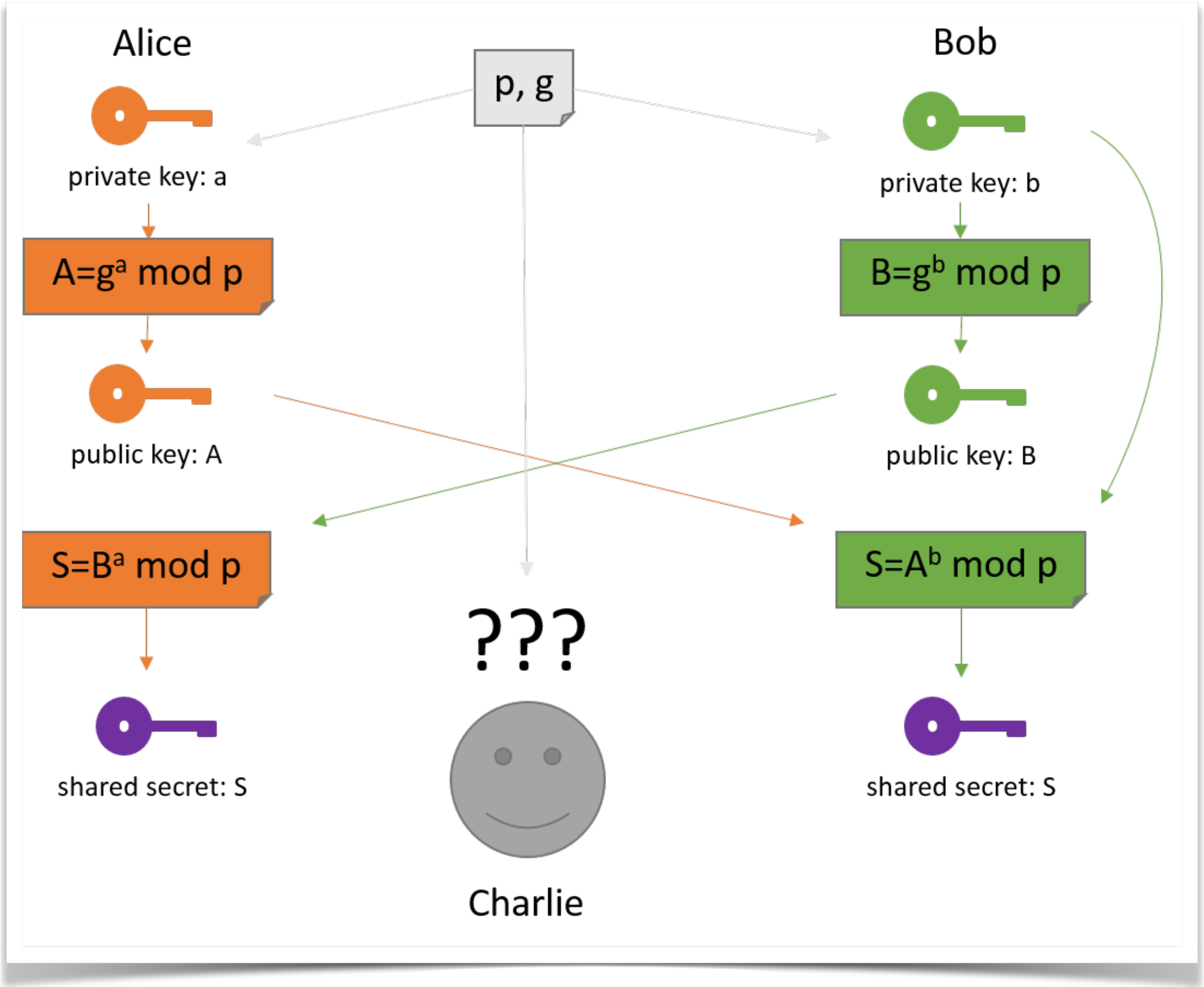
- For two participants p[1] and p[2]
- Setup: We pick a prime number p and a base g ($<p$)
  - This information is *public*
  - E.g., $p=13$, $g=4$
- Step 1: Each principal picks a *private* value a ($<p-1$) and b ($<p-1$), respectively
- Step 2: Each principal generates and communicates a new value A, B respectively

$A = g^a \bmod p$, $B = g^b \bmod p$

- Step 3: Each principal generates the secret shared key z

$s = A^b \bmod p = g^{ab} \bmod p$, $s = B^a \bmod p = g^{ab} \bmod p$

Perform a neighbor exchange.



A protocol run ...

p=17, g=6

Step 1)

Alice picks x=4

Bob picks x=5

Step 2)

Alice's y = 6^4 mod 17 = 1296 mod 17 = 4

Bob's y = 6^5 mod 17 = 7776 mod 17 = 7

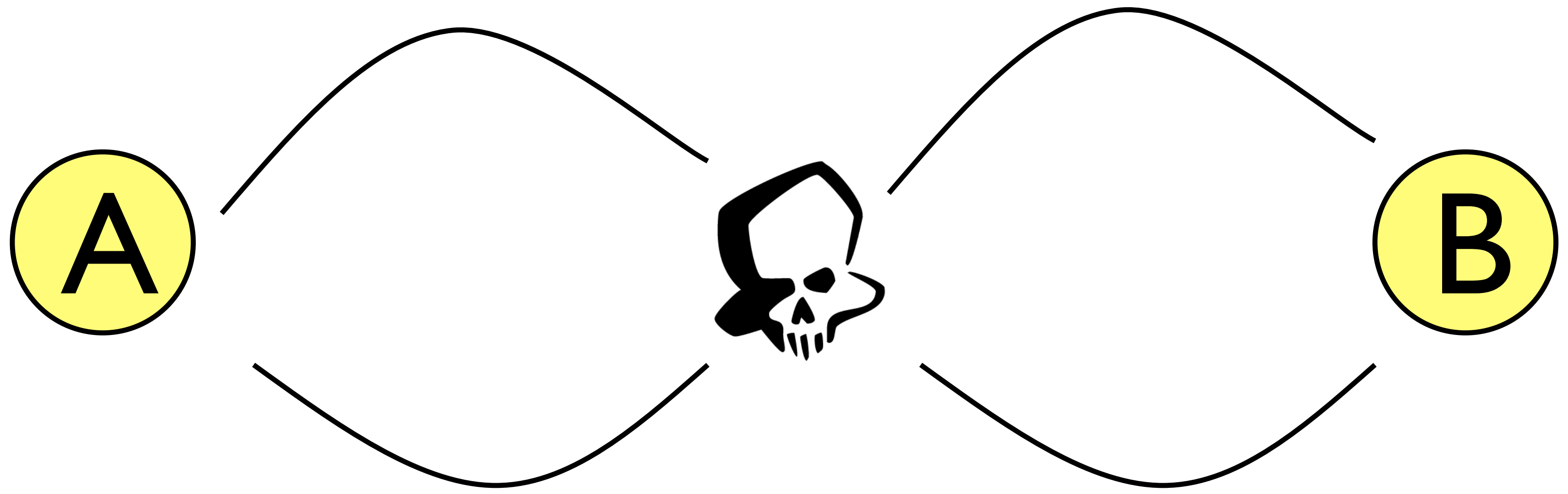
Step 3)

Alice's z = 7^4 mod 17 = 2401 mod 17 = *4*

Bob's z = 4^5 mod 17 = 1024 mod 17 = *4*

Attacks on Diffie-Hellman

- This is key agreement, not authentication.
  - You really don't know anything about who you have exchanged keys with
  - The man in the middle …



  - Alice and Bob think they are talking directly to each other, but Mallory is actually performing two separate exchanges
- You need to have an authenticated DH exchange
  - The parties sign the exchanges (more or less)
  - See Schneier for a intuitive description

Key Distribution/Agreement

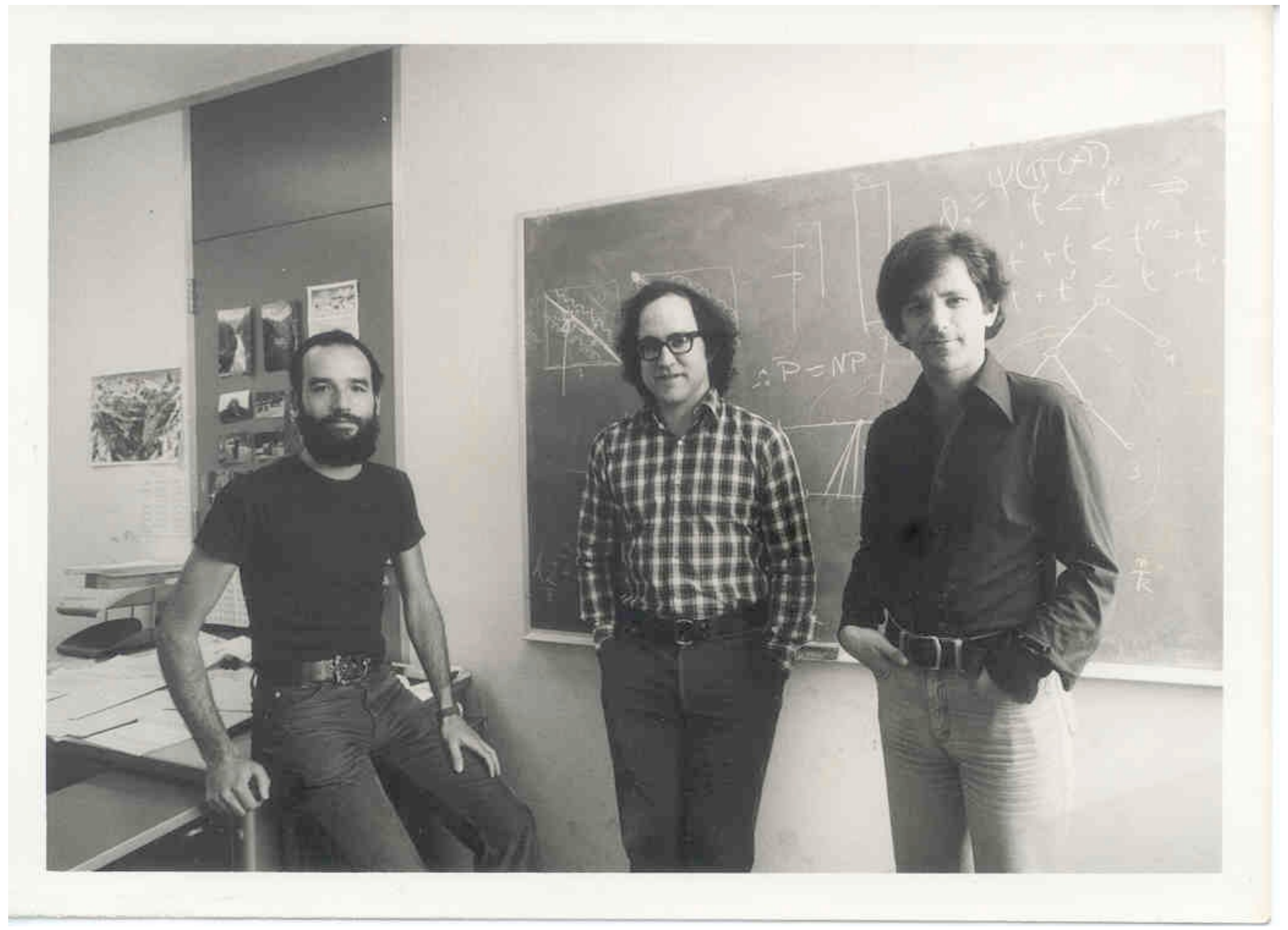
- Key Distribution is the process where we assign and transfer keys to a participant
  - Out of band (e.g., passwords, simple)
  - During authentication (e.g., Kerberos)
  - As part of communication (e.g., skip-encryption)
- Key Agreement is the process whereby two parties negotiate a key
  - 2 or more participants
- Typically, key distribution/agreement this occurs in conjunction with or after authentication.
  - However, many applications can pre-load keys

RSA (Rivest, Shamir, Adelman)

PennState

- A dominant public key algorithm
  - The algorithm itself is conceptually simple
  - Why it is secure is very deep (number theory)
  - Use properties of exponentiation modulo a product of large primes

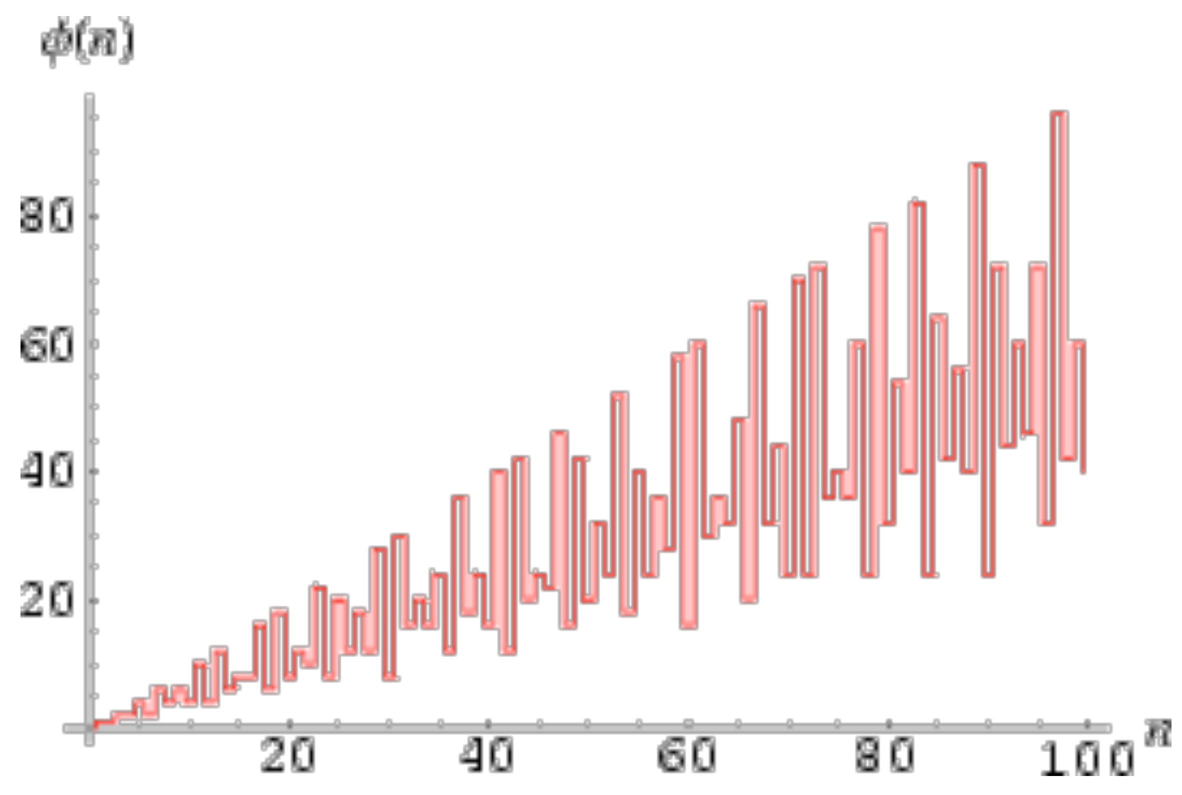
"A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, Feb., 1978, 21(2), pages 120-126.



RSA Key Generation

- Pick two large primes p and q
- Calculate n = pq
- Pick e such that it is relatively prime to phi(n) = (q-1)(p-1)
  - "Euler's Totient Function"
- d ~= e^{-1} mod phi(n)   or

  de mod phi(n) = 1



1. p=3, q=11
2. n = 3*11 = 33
3. phi(n) = (2*10) = 20
4. e = 7 | GCD(20,7) = 1
5. "Euclid's Algorithm"
   d = 7^{-1} mod 20
   d | d * 7 mod 20 = 1
   d = 3

RSA Encryption/Decryption

- Public key k^+ is {e,n} and private key k^- is {d,n}
- Encryption and Decryption

$$E(k+,P) : ciphertext = plaintext^e \bmod n$$

$$D(k-,C) : plaintext = ciphertext^d \bmod n$$

- Example
  - Public key (7,33), Private Key (3,33)
  - Data "4" (encoding of actual data)
  - E({7,33},4) = 4^7 mod 33 = 16384 mod 33 = 16
  - D({3,33},16) = 16^3 mod 33 = 4096 mod 33 = 4

Encryption using private key …

- Encryption and Decryption

$$E(k^-,P) : \text{ciphertext} = \text{plaintext}^d \bmod n$$

$$D(k^+,C) : \text{plaintext} = \text{ciphertext}^e \bmod n$$

- E.g.,
  - $E(\{3,33\},4) = 4^3 \bmod 33 = 64 \bmod 33 = 31$
  - $D(\{7,33\},31) = 31^7 \bmod 33 = 27{,}512{,}614{,}111 \bmod 33 = 4$
- Q: What is RSA's trapdoor function and trapdoor?
- Q: Why encrypt with private key?

Why does RSA work?

- Difficult to find $\phi(n)$ or d using only e and n
- Finding d equivalent difficulty to factoring p*q
  - Classical problem worked on for centuries; no known reliable fast method
  - Example: Took 18 months to factor a 200 digit number into its 2 prime factors
- It is feasible to encrypt and decrypt because
  - It is possible to find large primes
  - It is possible to find coprimes and their inverses
  - Modular exponentiation is feasible

{e,n} is public information
If you could factor n into p*q, then
Could compute $\phi(n) = (p-1)(q-1)$
Could compute $d = e^{-1} \bmod \phi(n)$
Would know the private key {d,n}!

“Textbook” RSA and Security

- What we’ve just seen is known as “textbook” RSA
- RSA must be used with proper padding to prevent certain attacks (including chosen plaintext attacks)
- As we’ve used it here, NO integrity!
- RSA keys can be of any length
  - The current recommendation is that important keys should be at least 2048-bits in length
  - 1024 bit keys are ok for most uses, but you should feel nervous about them

Cryptanalysis of RSA

- Survey by Dan Boneh
  - http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html
  - Real heavy math
- Results
  - Fascinating attacks have been developed
  - None devastating to RSA
- Cautions
  - Improper use
  - Secure implementation is non-trivial

Is RSA Secure?

- Premise: Breaking RSA == Factoring Large Integers
  - Factoring Large Integers is Hard
  - N=pq; if N is known, can we find p, q?
- Some Known (to cryptanalysts)
  - If (p-1)(q-1) is product of prime factors less than some number B
  - N can be factored in time less than B^3
- Best Known Approach: General Number Field Sieve
  - Significant early application by Arjen Lenstra

Is RSA Secure?

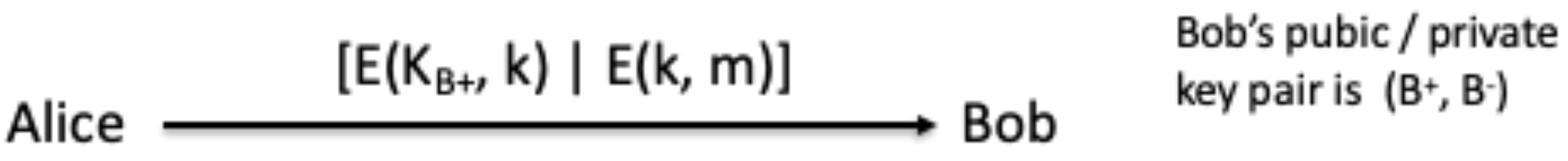
- Fundamental tenet of cryptography
  - Lots of smart people have tried but not (yet) figured out how to break RSA => RSA is secure
- RSA Laboratories challenge (Mar 1991)
  - Factor N into semiprimes (vary from 100 to 619 decimal digits).
  - Challenge ended in 2007
    - 16 of 54 listed numbers were factored
  - Current: up to 232 decimal digits factored
    - Using variations of "general number field sieve" algorithms

Misuse of RSA

- Common Modulus Misuse
  - Use the same N for all users
  - Since all have a private key for same N
    - Anyone can factor from their d and e
    - Exposing any d is same as factoring N
- Blinding Misuse
  - Suppose adversary wants you to
    - Sign an arbitrary message M
  - You don't sign
  - Adversary generates innocent M'
    - Where $M' = r^e M \bmod N$
    - Adversary can generate M signature from M' signature

Hybrid Cryptosystems

- In practice, public-key cryptography is used to secure and distribute session keys.
- These keys are used with symmetric algorithms for communication
- Sender generates a random session key, encrypts it using receiver's public key and sends it
- Receiver decrypts the message to recover the session key
- Both encrypt/decrypt their communications using the same key
- Key is destroyed in the end



k is the session key, sometimes called the ephemeral key

Digital Signatures

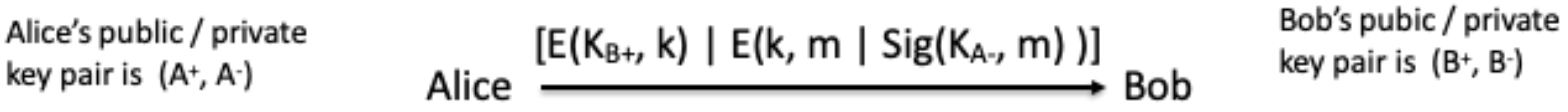
- Models physical signatures in digital world
  - Association between private key and document
  - … and indirectly identity and document.
  - Asserts that document is authentic and non-reputable
- To sign a document
  - Given document d, private key k-
  - Signature $S(k^-, d) = E(\ k^-, h(d)\ )$
- Validation
  - Given document d, signature $S(k^-, d)$, public key k+
  - Validate $D(k^+, S(k^-, d)) = h(d)$

Digital Signatures

- A digital signature serves the same purpose as a real signature
  - It is a mark that only the sender can make
  - Other people can easily recognize it belonging to the sender
- Digital signatures must be:
  - Unforgeable: If Alice signs message M with signature S, it is impossible for someone else to produce the pair (M, S).
  - Authentic: If Bob receives the pair (M, S) and knows Alice's public key, he can check ("verify") that the signature is really from Alice

How can Alice sign a digital document?

- Digital document: M
- Since RSA is slow, hash M to compute digest h(M)
- Signature: $Sig_{k-}(M) = E_{k-}(h(M)) = (h(M))^d \bmod n$
  - Since only Alice knows k-, only she can create the signature
- To verify: Verify $(M, Sig_{k-}(M))$
  - Bob computes h(M) and compares it with $D_{k+}(Sig_{k-}(M))$
  - Bob can compute D_{k+}(Sigk-(M)) since he knows k+ (Alice's public key)
  - If and only if they match, the signature is verified (otherwise, fails)



Birthday Attack and Signatures

- Since signatures depend on hash functions, they also depend on the hash function's collision resistance
- Don't use MD5 or SHA1
-

Properties of digital signature

- No forgery possible: No one can forge a message that is purportedly from Alice
- Authenticity check: If you get a signed message you should be able to verify that it's really from Alice
- No alteration/Integrity: No party can undetectably alter a signed message
- Provides authentication, integrity, and non-repudiation (cannot deny having signed a signed message)
-

Non-Repudiation

PennState

- Which offers non-repudiation, and why?
  - HMAC: [m | HMAC(k, m)]
  - Digital Signature: [m | Sigk-(m)]

Using Public Key Crypto

- Suppose you (Alice) want to send a document securely to another party (Bob)
  - You have each others' public keys
  - Obtained in some secure fashion (PKI, later)
- How do you send the document such that only Bob can read it?
- How do you send the document such that Bob knows it is from Alice?

Review: secret vs. public key crypto.

- Secret key cryptography
  - Symmetric keys, where A single key (k) is used is used for E and D
  - D(E(p, k), k) = p
- All (intended) receivers have access to key
- Note: Management of keys determines who has access to encrypted data
  - E.g., password encrypted email
- Also known as symmetric key cryptography

- Public key cryptography

Each key pair consists of a public and private component:

k+ (public key), k- (private key)

D(E(p, k+), k-) = p

D(E(p, k-), k+) = p

- Public keys are distributed (typically) through public key certificates

– Anyone can communicate secretly with you if they have your certificate

– E.g., SSL-based web commerce

The symmetric/asymmetric key tradeoff

- Symmetric (shared) key systems
  - Efficient (Many MB/sec throughput)
  - Difficult key management
    - Kerberos
    - Key agreement protocols
- Asymmetric (public) key systems
  - Slow algorithms (so far …)
  - Easy (easier) key management
    - PKI - public key infrastructures
    - Webs of trust (PGP)

Meet Alice and Bob ….

- *Alice* and *Bob* are the canonical players in the cryptographic world.
  - They represent the end points of some interaction
  - Used to illustrate/define a security protocol

- Other players occasionally join …
  - Syed - trusted third party
  - Mallory - malicious entity
  - Eve - eavesdropper
  - Ivan - an issuer (of some object)

Some notation …

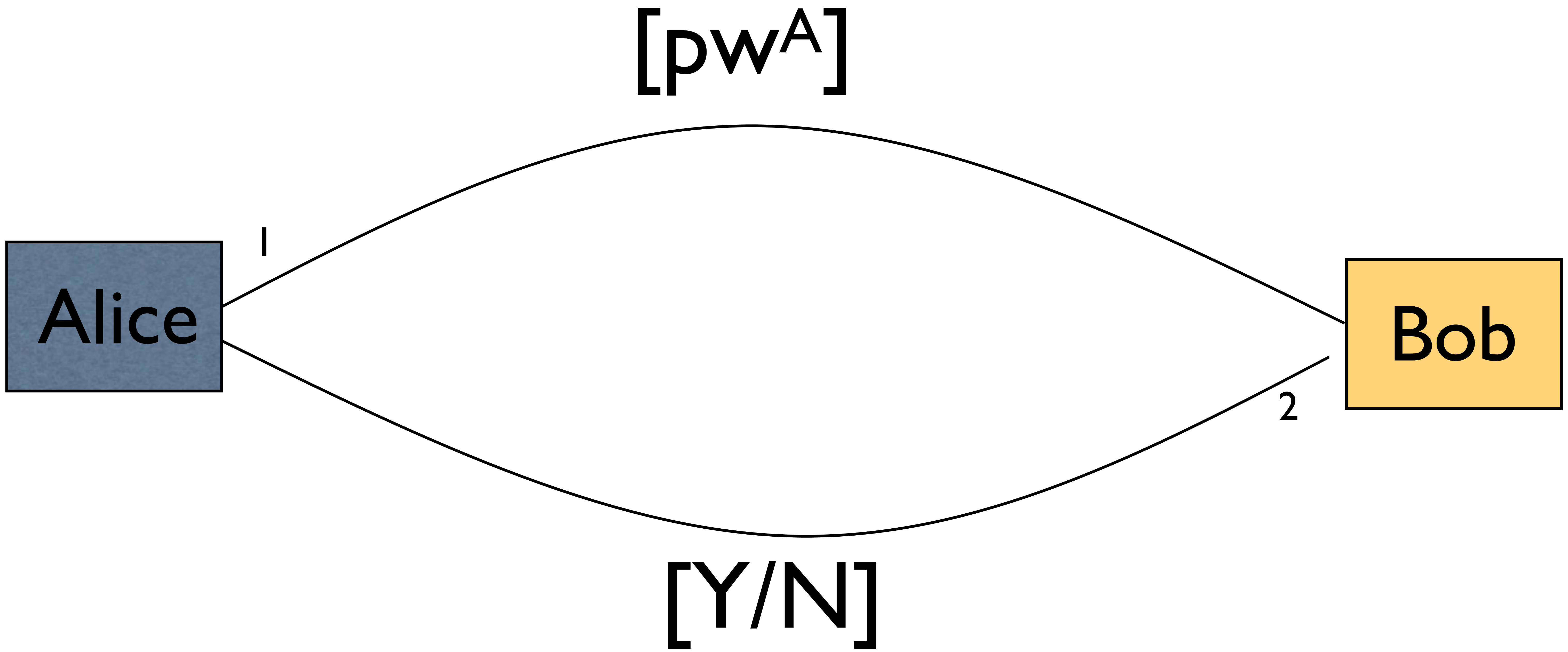
- You will generally see protocols defined in terms of exchanges containing some notation like
  - All players are identified by their first initial
    - E.g., Alice=A, Bob=B
  - *d* is some data
  - pw^A is the password for A
  - k_{AB} is a symmetric key known to A and B
  - K_A^+,K_A^- is a public/private key pair for entity A
  - E(*k*,*d*) is encryption of data *d* with key *k*
  - H(*d*) is the hash of data *d*
  - S(K_A^-,*d*) is the signature (using A's private key) of data *d*
  - "+" is used to refer to concatenation

Some interesting things you want to do …

- … when communicating.
  - Ensure the *authenticity* of a user
  - Ensure the integrity of the data
    - Also called *data authenticity*
  - Keep data *confidential*
  - Guarantee *non-repudiation*

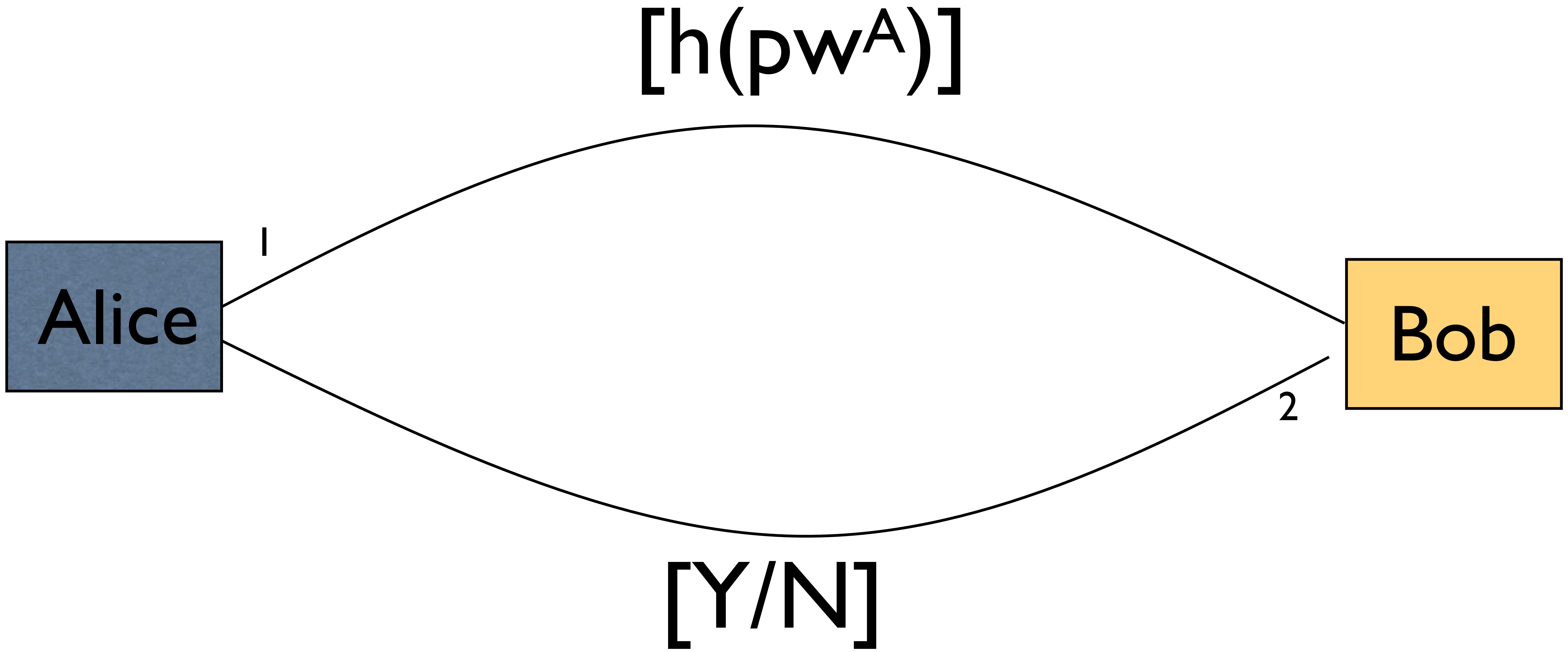
Basic (User) Authentication

- Bob wants to authenticate Alice's identity
  - (is who she says she is)



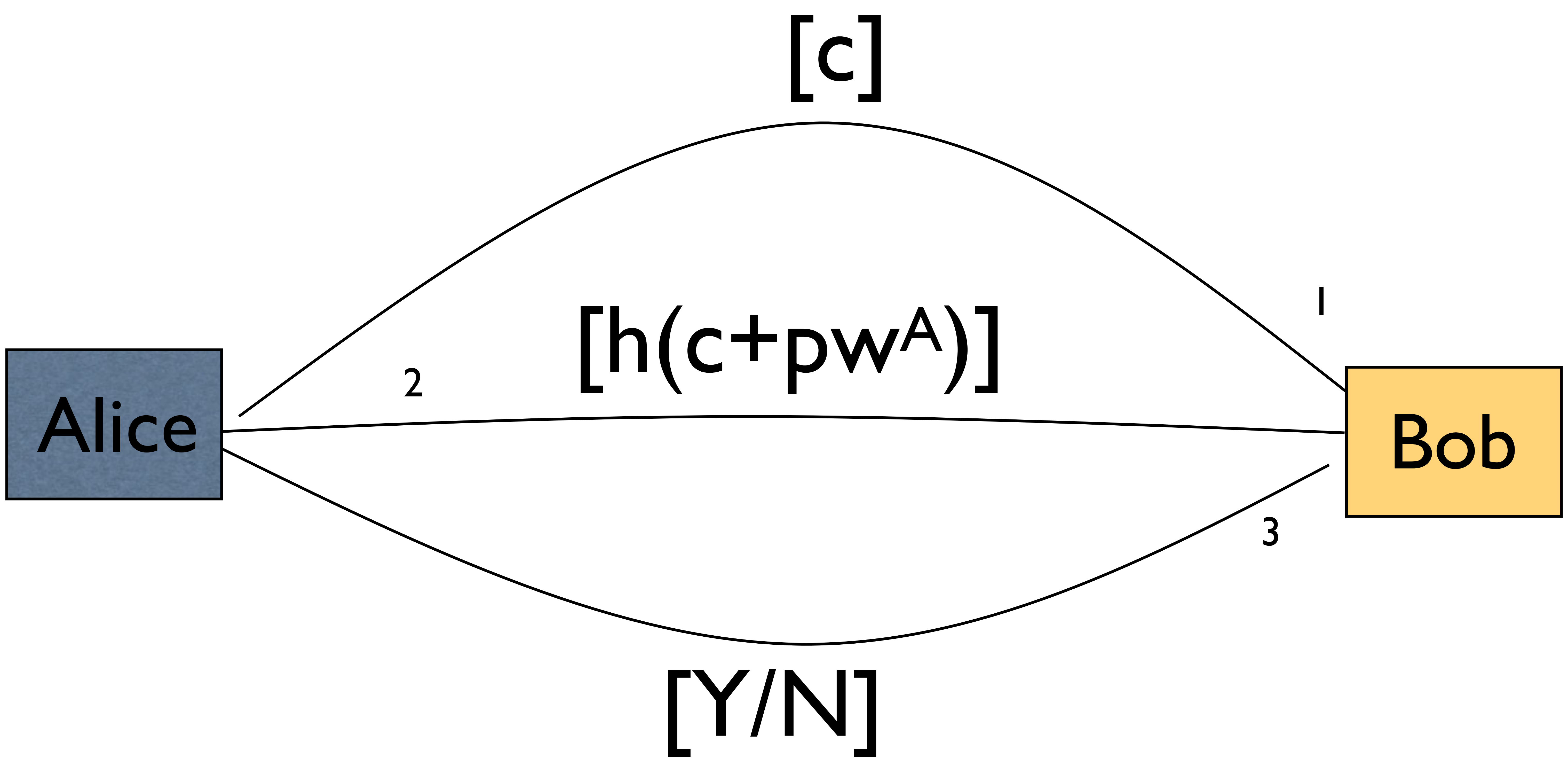
Hash User Authentication

- Bob wants to authenticate Alice's identity
  - (is who she says she is)



Challenge/Response User Authentication

- Bob wants to authenticate Alice's identity
  - (is who she says she is)



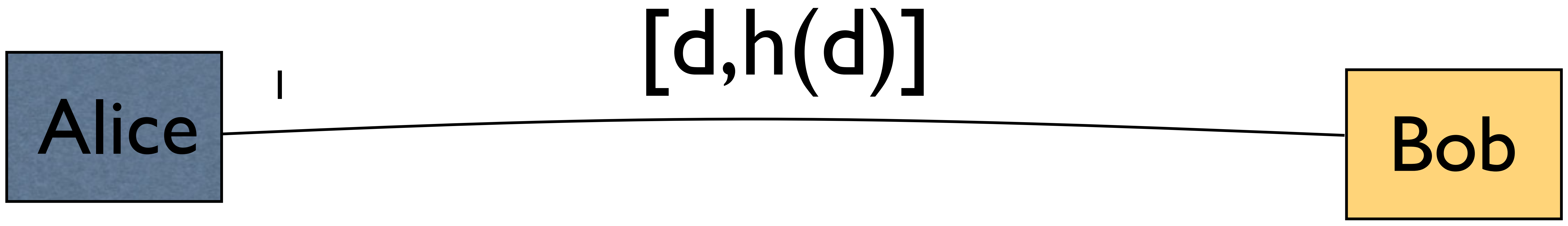
User Authentication vs. Data Integrity

- User authentication proves a property about the communicating parties
  - E.g., I know a password
- Data integrity ensures that the data transmitted...
  - Can be verified to be from an authenticated user
  - Can be verified to determine whether it has been modified

- Now, lets talk about the latter, *data integrity*

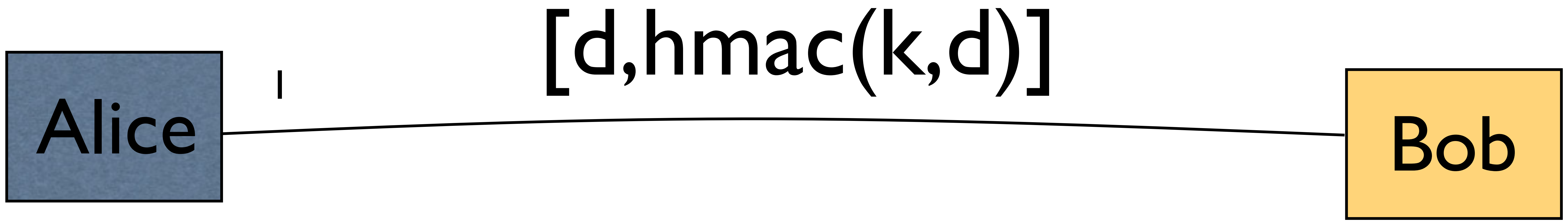
Simple Data Integrity?

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)



HMAC Integrity

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)



Signature Integrity

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)

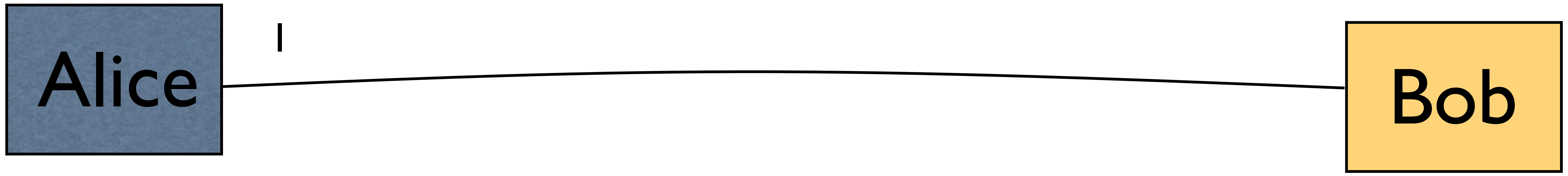
Data Integrity vs. Non-repudiation

- If the integrity of the data is preserved, is it provably from that source?
  - HMAC integrity says what about non-repudiation?
  - Signature integrity says what about non-repudiation?

Confidentiality

Alice wants to ensure that the data is not exposed to anyone except the intended recipient (confidentiality)

$$[E(k_{AB}, d), \ hmac(k_{AB}, d)]$$



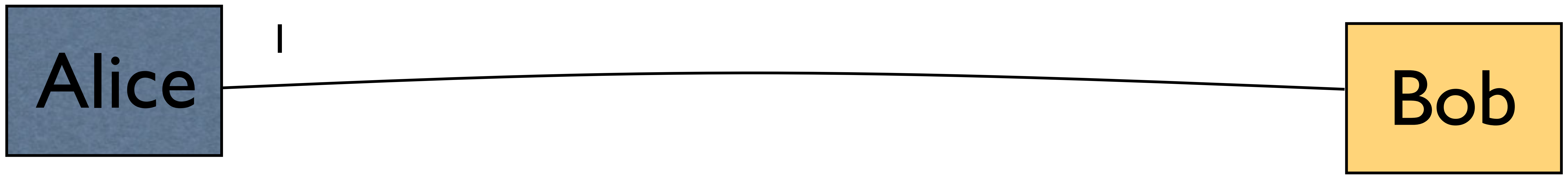
Question

- If I already have an authenticated channel (e.g., the remote party's public key), why don't I simply make up a key and send it to them?

Confidentiality

- Alice wants to ensure that the data is not exposed to anyone except the intended recipient (confidentiality)
- But, Alice and Bob have ***never met***!!!!

$$[E(k_x,d),\ hmac(k_x, d), E(K_B^+,k_x)]$$



- Alice randomly selects key k_x to encrypt with

Key Distribution Revisited

- How do we distribute a key in an untrusted network?
  - Diffie-Hellman
    - Beware of Man-in-the-Middle Attacks
  - Public key
    - Offline and via certificates (more later)
    - What about without certs
  - Symmetric key
    - Offline
    - How about online?

Needham-Schroeder

- Goal
  - Two parties want to communicate securely
- Threat Model
  - Network is untrusted
  - Other nodes may be untrusted
- Requirements
  - Mutual Authentication
  - Prove that only the appropriate parties hold secrets
- Assumptions
  - Trusted Third Party

N-S Protocol Detail

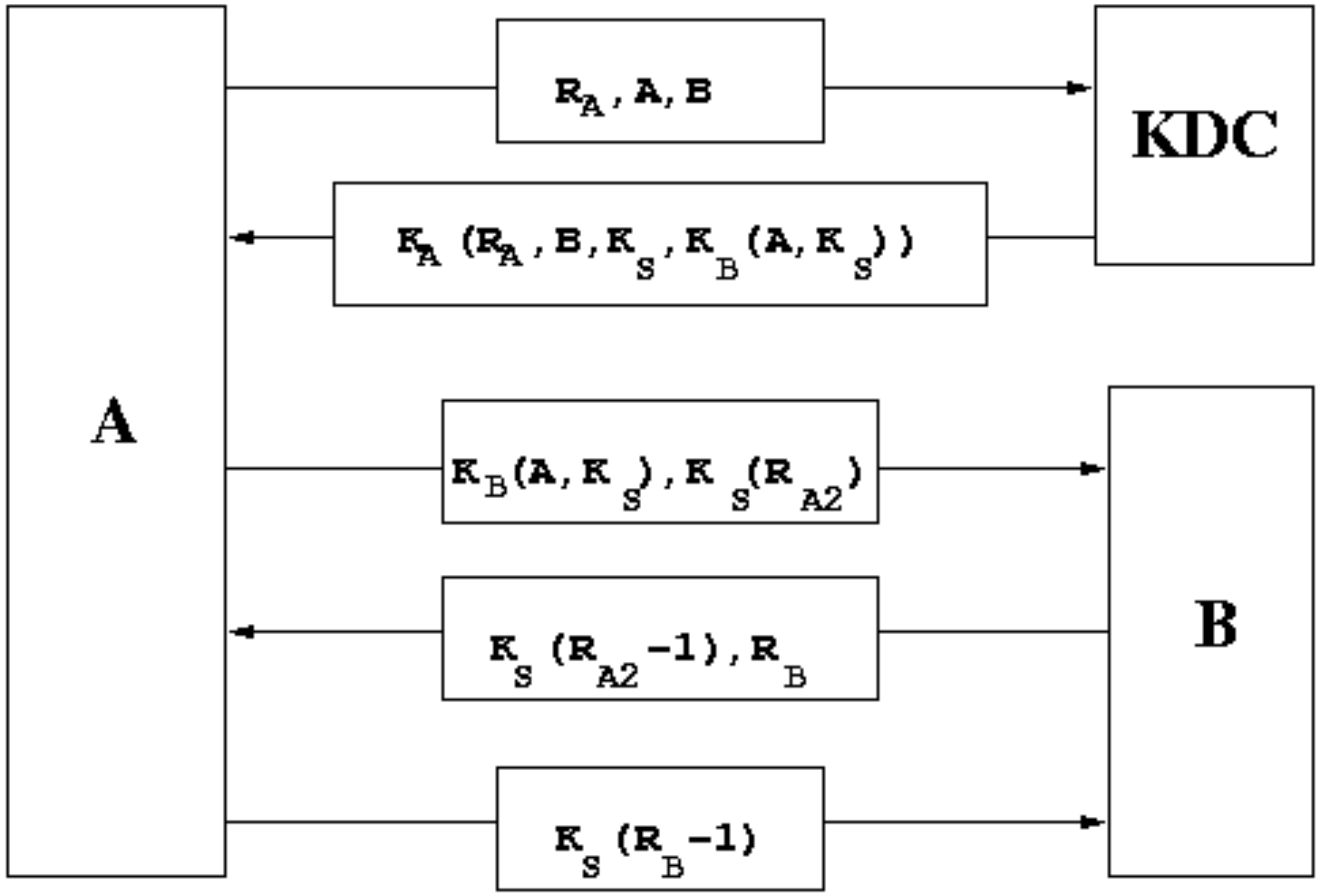
- Message 1: A --> S : A, B, N_A [N=R=nonce-random value]
  - A asks TTP S for a session key for A and B to use
- Message 2: S --> A : $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{BS}\}_{AS}$
  - S returns messages for A that includes the session key
  - And a message for A to give to B
- Message 3: A --> B : $\{K_{AB}, A\}_{BS}$
  - A passes “ticket” on to B
- Message 4: B --> A : $\{N_B\}_{AB}$
  - B asks A to demonstrate knowledge of K_{AB} through N_B
- Message 5: A --> B : $\{N_B - 1\}_{AB}$
  - A does!

N-S Protocol Detail

- Message 1: A --> S : A, B, N_A [N=R=nonce-random value]
  - A asks TTP S for a session key for A and B to use
- Message 2: S --> A : $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{BS}\}_{AS}$
  - S returns messages for A that includes the session key
  - And a message for A to give to B
- Message 3: A --> B : $\{K_{AB}, A\}_{BS}$,$\{N_{A2}\}_{AB}$
  - A passes "ticket" on to B
- Message 4: B --> A : $\{N_B, N_{A2}-1\}_{AB}$
  - B asks A to demonstrate knowledge of K_{AB} through N_B
- Message 5: A --> B : $\{N_B-1\}_{AB}$
  - A does!

N-S Protocol

- For Symmetric Key Cryptosystems



The Needham-Schroeder Authentication Protocol

Needham-Schroeder Public Key

Nonce

- Message a.1: A --> B : A, B, $\{N_A, A\}_{K_B^+}$
  - A initiates protocol with fresh value for B
- Message a.2: B --> A : B, A, $\{N_A, N_B\}_{K_A^+}$
  - B demonstrates knowledge of N_A and challenges A
- Message a.3: A --> B : A,B, $\{N_B\}_{K_B^+}$
  - A demonstrates knowledge of N_B

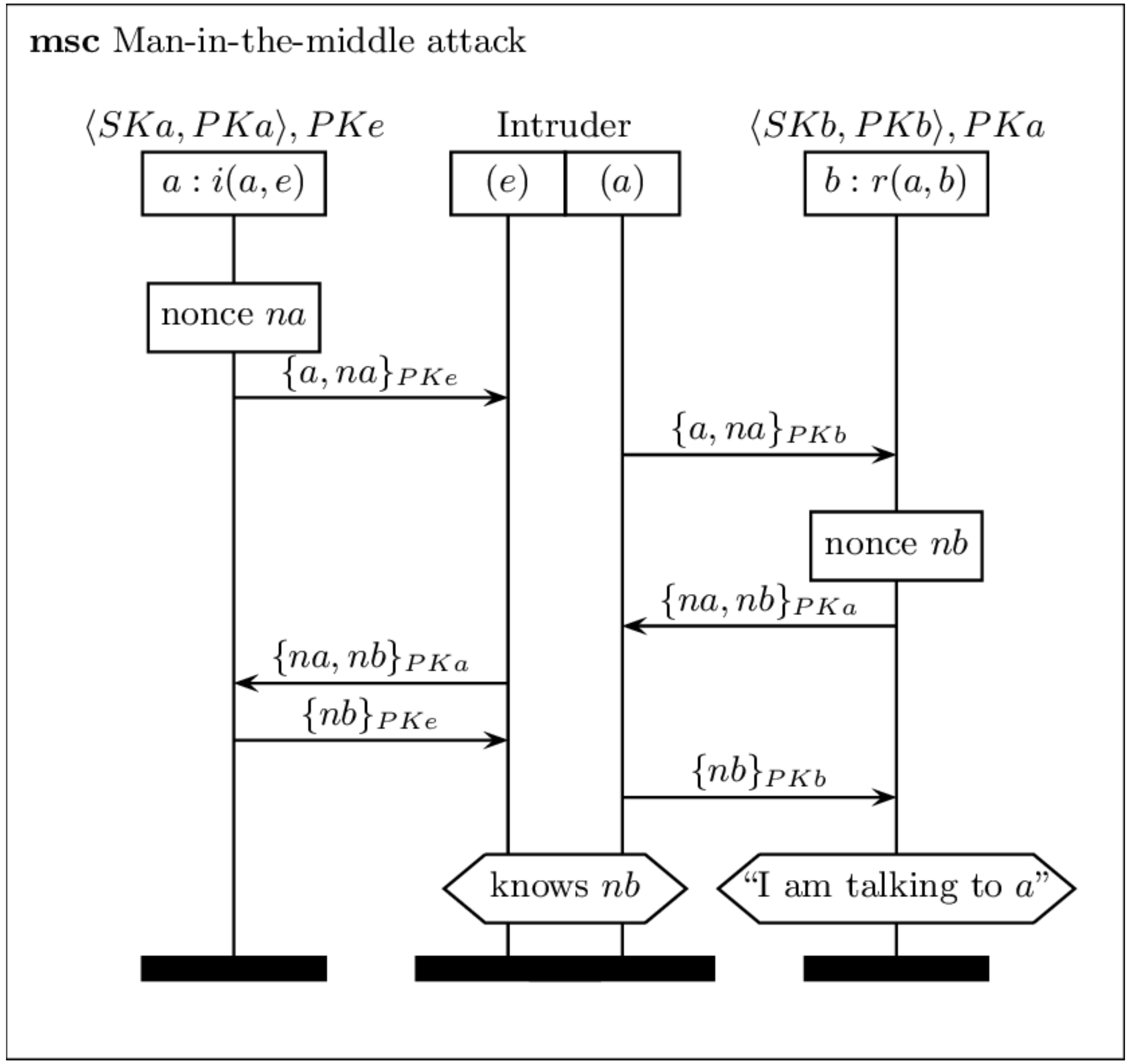
- A and B are the only ones who can read N_A and N_B

A Protocol Story

- Needham-Schroeder Public Key Protocol
  - Defined in 1978
- Assumed Correct
  - Many years without a flaw being discovered
- Proven Correct
  - BAN Logic (early 1990s)
- So, It's Correct, Right?

Gavin Lowe Attack

- An active intruder X participates…
- Message a.1: A --> X : A,X, $\{N_A, A\}_{PKX}$
- Message b.1: X(A) --> B : A,B, $\{N_A, A\}_{PKB}$
  - X as A initiates protocol with fresh value for B
- Message b.2: B --> X(A) : B,A, $\{N_A, N_B\}_{PKA}$
- Message a.2: X --> A : X,A, $\{N_A, N_B\}_{PKA}$
  - X asks A to demonstrates knowledge of N_B
- Message a.3: A --> X : A,X, $\{N_B\}_{PKX}$
  - A tells X N_B; thanks A!
- Message b.3: X(A) --> B : A,B, $\{N_B\}_{PKB}$
  - X completes the protocol as A



Operational Semantics of Security Protocols, Cremers et al, 2003

What Happened?

- What is the cause of this attack?

What Happened?

- X can get A to act as an "oracle" for nonces
  - Hey A, what's the N_B in this message from any B?
- A assumes that any message encrypted for it is legit
  - Bad idea
- X can enable multiple protocol executions to be interleaved
  - Should be part of the threat model

The Fix

- It’s Trivial (find it)

- Message 3: A --> B : A,B, $\{N_A, A\}_{PKB}$
  - A initiates protocol with fresh value for B
- Message 4: B --> A : B,A, $\{N_A, N_B, B\}_{PKA}$
  - B demonstrates knowledge of N_A and challenges A
- Message 5: A --> B : A,B, $\{N_B\}_{PKB}$
  - A demonstrates knowledge of N_B

Impact on Protocol Analysis

- Protocol Analysis Took a Black Eye
  - BAN Logic Is Insufficient
  - BAN Logic Is Misleading
- Protocol Analysis Became a Hot Topic
  - Lowe's FDR
  - Meadow's NRL Analyzer
  - Millen's Interrogator
  - Rubin's Non-monotonic protocols
  - ....
- In the end, could find known flaws, but...
  - Attacker model is too complex

Dolev-Yao Result

- Strong attacker model
  - Attacker intercepts every message
  - Attacker can cause operators to be applied at any time
    - Operators for modifying, generating any kind of message
  - Attacker can apply any operator except other's decryption
- Theoretical Results
  - Polynomial Time for One Session
  - Undecidable for Multiple Sessions
  - *Moral: Protocol Validation is Difficult Because Attacker Can Exploit Interactions of Multiple Sessions*

Real Systems Security

- The reality of the security is that 90% of the frequently used protocols use some variant of these constructs.
    - So, get to know them … they are your friends
    - We will see them (and a few more) over the semester

- They also apply to systems construction
    - Protocols need not necessarily be online
    - Think about how you would use these constructs to secure files on a disk drive (integrity, authenticity, confidentiality)
    - We will add some other tools, but these are the basics