Lecture 3:
TLS Part 1: AKE

CPSC 444/544, 2026 Spring

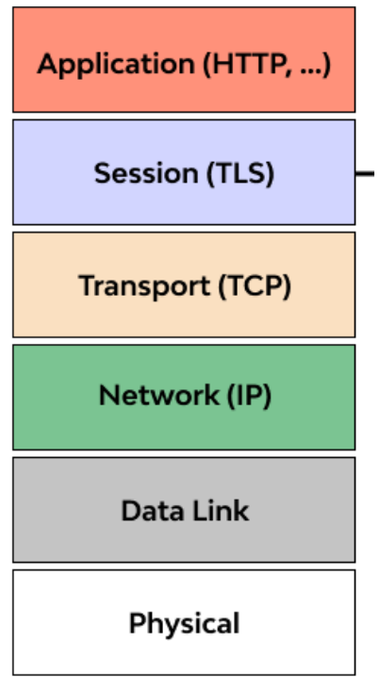
Instructor: Fan Zhang

Yale

Computer Science

What is Transport Layer Security?

- Designed by Netscape for Web Transaction (e.g., send CC# over internet)
- Support wide range of applications
  - HTTPS
  - VPN
  - E-mail
  - Voice/Audio
  - IoT



SSL 1.0	Terribly insecure; never released.
SSL 2.0	Released 1995; terribly insecure, deprecated in 2011
SSL 3.0	Released 1996; insecure, deprecated in 2015.
TLS 1.0	Released 1999; deprecated in 2020.
TLS 1.1	Released 2006; deprecated in 2020.
TLS 1.2	Released 2008. Ok.
TLS 1.3	Standardized in August 2018 and is being rolled out now; major change from TLS 1.2.

Impact of TLS

- Probably the most impactful piece of cryptography protocol
  - Run on pretty much every networked devices
- Scrutinized by the best attackers
  - We can learn from attacks

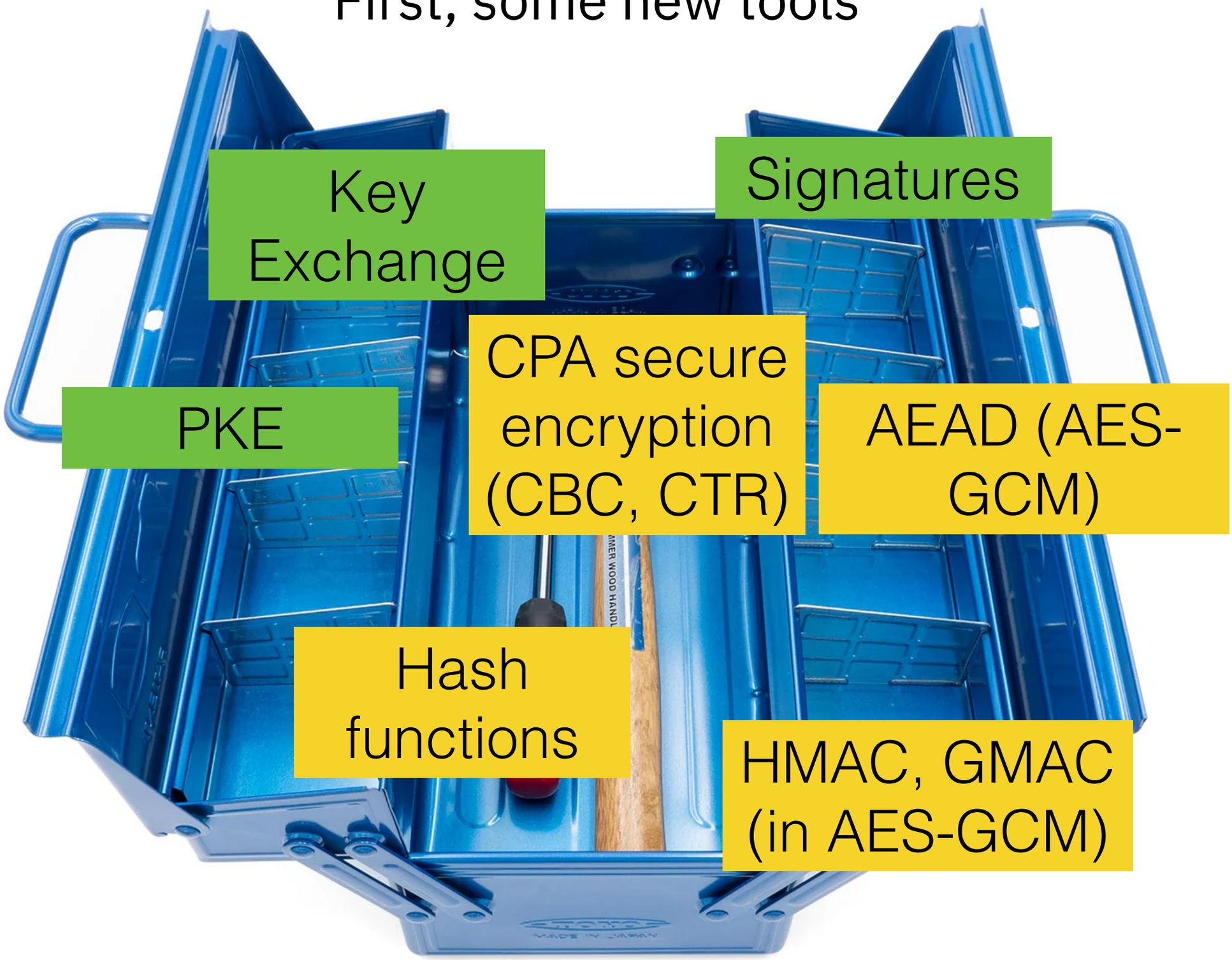
Motivation: We don’t own the network infrastructure

- The workflow of accessing a web service (e.g., online banking)
    1. Query DNS server to translate mybank.com to IP address
    2. build HTTP packets (e.g., including password & user instructions)
    3. Send packets through IP network, going through WiFi access point, ISP, data center administrators, etc.
    4. Bank server receives data & take actions
- Security concerns?

Motivation: We don’t own the network infrastructure

- Eyedroppers are listening to all traffic
  - E.g., the router in your office, Yale IT, AT&T
- Active adversary may inject, drop, tamper with package
- Malicious DNS may return bogus IP
  - E.g., sometimes WiFi access point sets the DNS
- Even if you got the right IP, a malicious (or hijacked) router may send packets to the wrong server
  - https://en.wikipedia.org/wiki/BGP_hijacking

First, some new tools



New tools

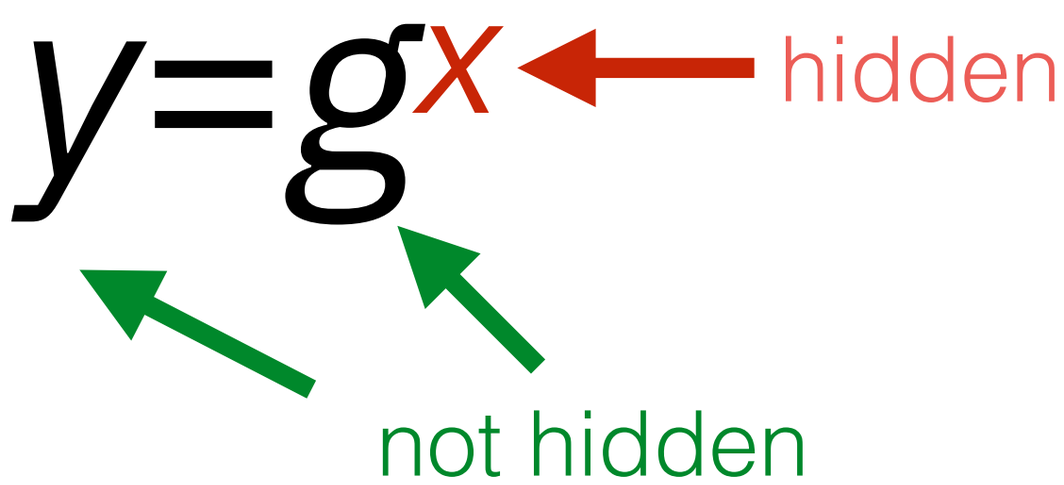
- Signatures → PK distribution, and authentication
- Key Exchange → secret establishment
- They require Public Key Cryptography.

Discrete Logs

(Foundation for a lot, not all, of PKC tools)

Intuition

Values in exponent space are “hidden,” so we can “compute secretly” in the exponent space.



Background on DL

- Taking logarithm $y = \log x$ is easy for real numbers $x, y \in \mathbb{R}$
- It is hard if x, y come from certain ***cyclic groups***.
- We now recall background [KL 9.1.3]

(Finite Abelian) Group

- A **finite set** $\mathbb{G}$ along with a **operation** $\circ$
  - $\mathbb{G}$ is closed under $\circ$
  - There is an "identity" $e \in \mathbb{G}$ s.t. $e \circ g = g$
  - Every element $g \in \mathbb{G}$ has an *inverse* (an element $h \in \mathbb{G}$ s.t. $g \circ h = e$), denoted g^{-1}
  - Associativity: $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$
  - Commutativity: $g_1 \circ g_2 = g_2 \circ g_1$
- We call $|\mathbb{G}|$ the ***order*** of the group

Notations

- Exponentiation (multiplicative notation)
  - $g^k \triangleq g \circ \cdots \circ g$ ($k \geq 1$ times);
  - $g^0 \triangleq e$
    - This is bad notation (IMO) because it suggests that $\circ$ is multiplication.
- Additive notion commonly for EC groups. Though we stick to the above.

Cyclic Groups

- A group $\mathbb{G}$ of order q is **cyclic** if there exists a special "generator" $g \in \mathbb{G}$ s.t. $\{g^0, g^1, \cdots, g^{q-1}\} = \mathbb{G}$.
- In a cyclic group $\mathbb{G}$, for any $h \in \mathbb{G}$, there is a **unique** $x \in \mathbb{Z}_q$ such at $g^x = h$
  - x is the ***Discrete Log*** of h w.r.t. g, denoted $\log_g h$
  - Note: any $x' \equiv x \mod q$ also satisfies $g^{x'} = y$
- Most commonly in crypto: **a cyclic group $\mathbb{G}$ with prime order q and a generator g**

Discrete Log assumption

- We know of groups in which computing the discrete log of a ***random*** element is believed to be computationally hard.

- Examples of DL groups
  - Prime-order subgroup of $\mathbb{Z}_p^{\times}$ (e.g., RFC 3526)
  - Eliptic curves
  - (You will implement them in Lab 1.)

Typical choices of *G*

- Prime-order subgroup of $\mathbb{Z}_p^{\times}$ (e.g., RFC 3526)
  - Find two primes p and q such that $p = rq + 1$ for some r (e.g., 2)
  - $\mathbb{G} = \{h^r \mod p \,|\, h \in \mathbb{Z}_p^{\times}\}$ is a subgroup of order prime q [KL Thm 9.66]
    - Long keys: |p|=3072-bit, |q|= 256-bit (for example)
- Another good choice is points on an **elliptic curve**
  - Yields very *compact* public keys, e.g., 256-bit, and efficient computation.
- **You will implement both in Lab 1.**

Effective Key Length	RSA	Discrete Logarithm	
	Modulus N	Order-q Subgroup of $\mathbb{Z}_p^*$	Elliptic-Curve Group Order q
112	2048	p: 2048, q: 224	224
128	3072	p: 3072, q: 256	256
192	7680	p: 7680, q: 384	384
256	15360	p: 15360, q: 512	512

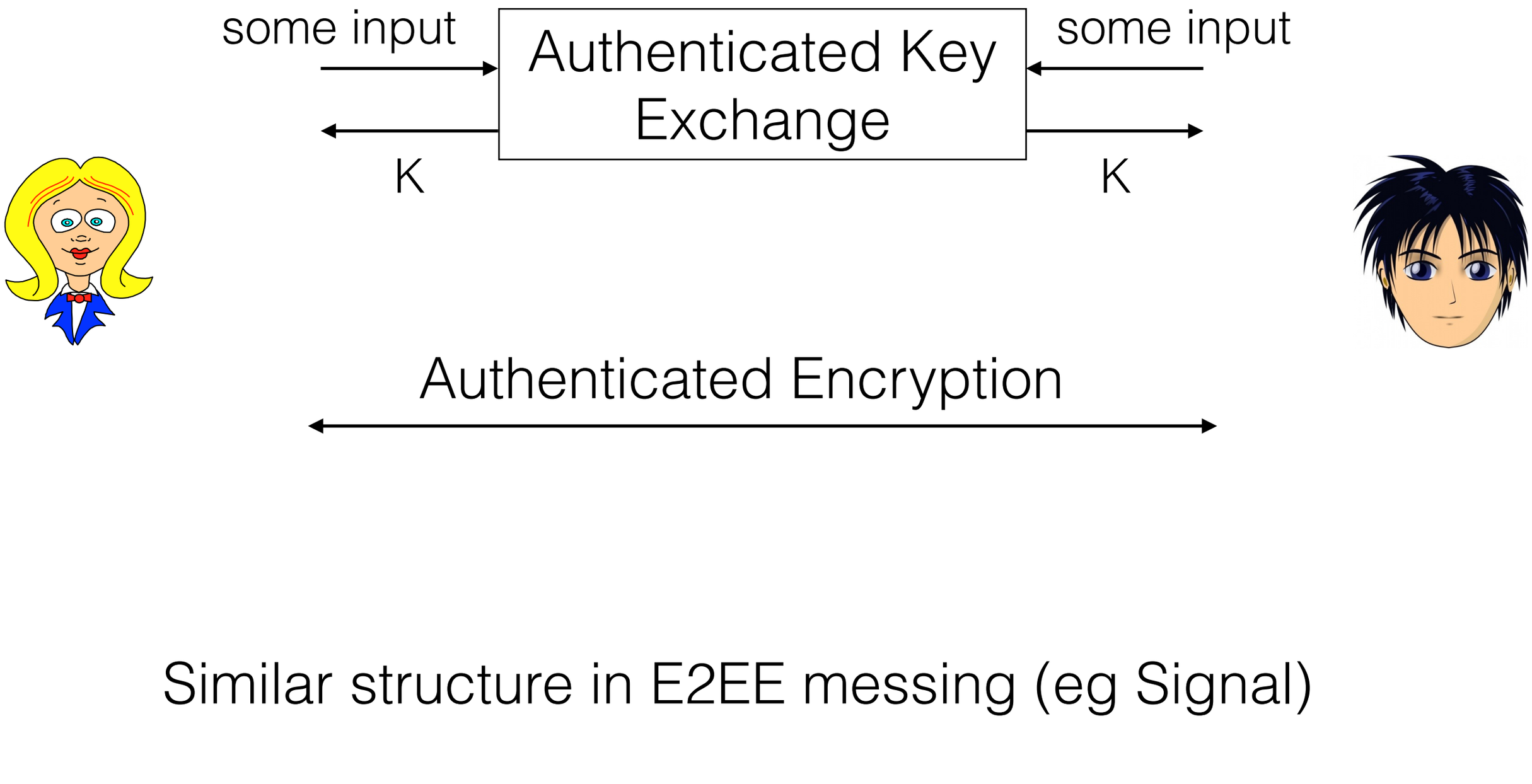
FIGURE 10.1: All values are in bits, e.g., for a 112-bit effective key length in the RSA setting, a 2048-bit modulus N should be used.

KL Ch 10.4

- End of DL

Assumption-

General structure of TLS

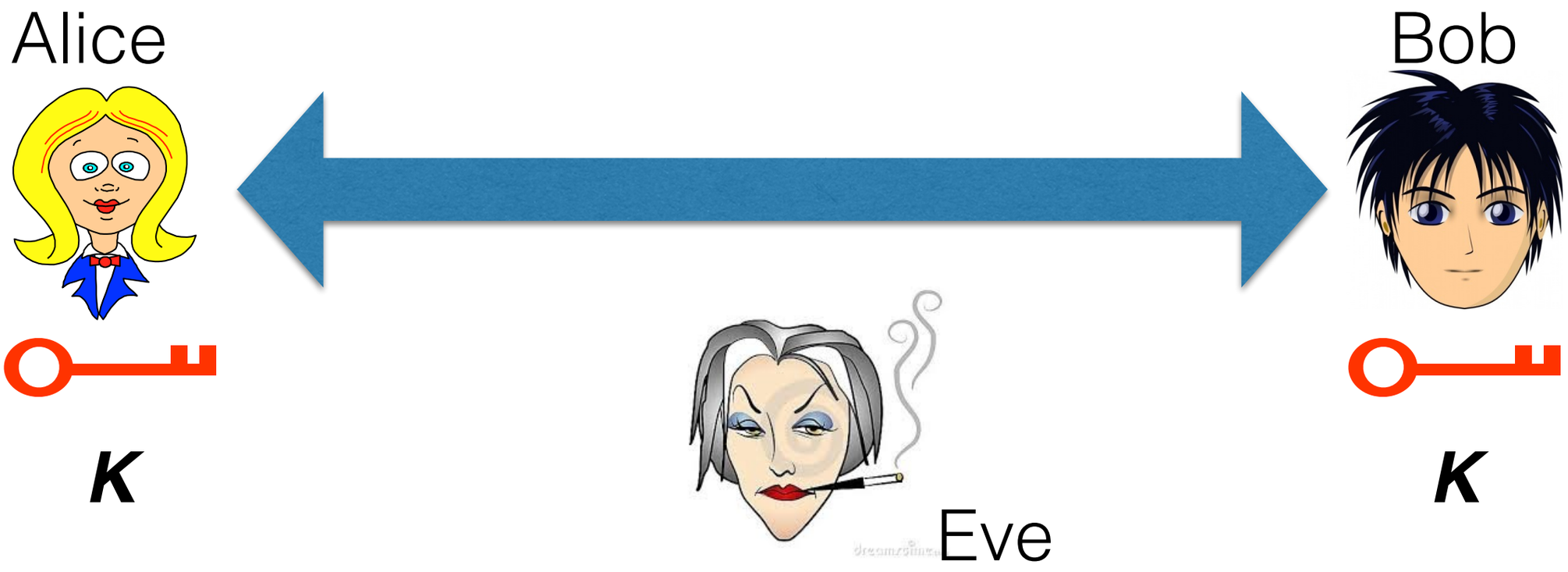


Similar structure in E2EE messing (eg Signal)

Plan for this week

- Today: Four AKEs, last one close to TLS 1.3
- Next (**Friday**): instantiating AKEs with El Gamal, AKE to Handshake

Adversary model



Alice and Bob want to share a secret key *K*, but:

- Eavesdropper listening all the time.
- There may be **imposters**

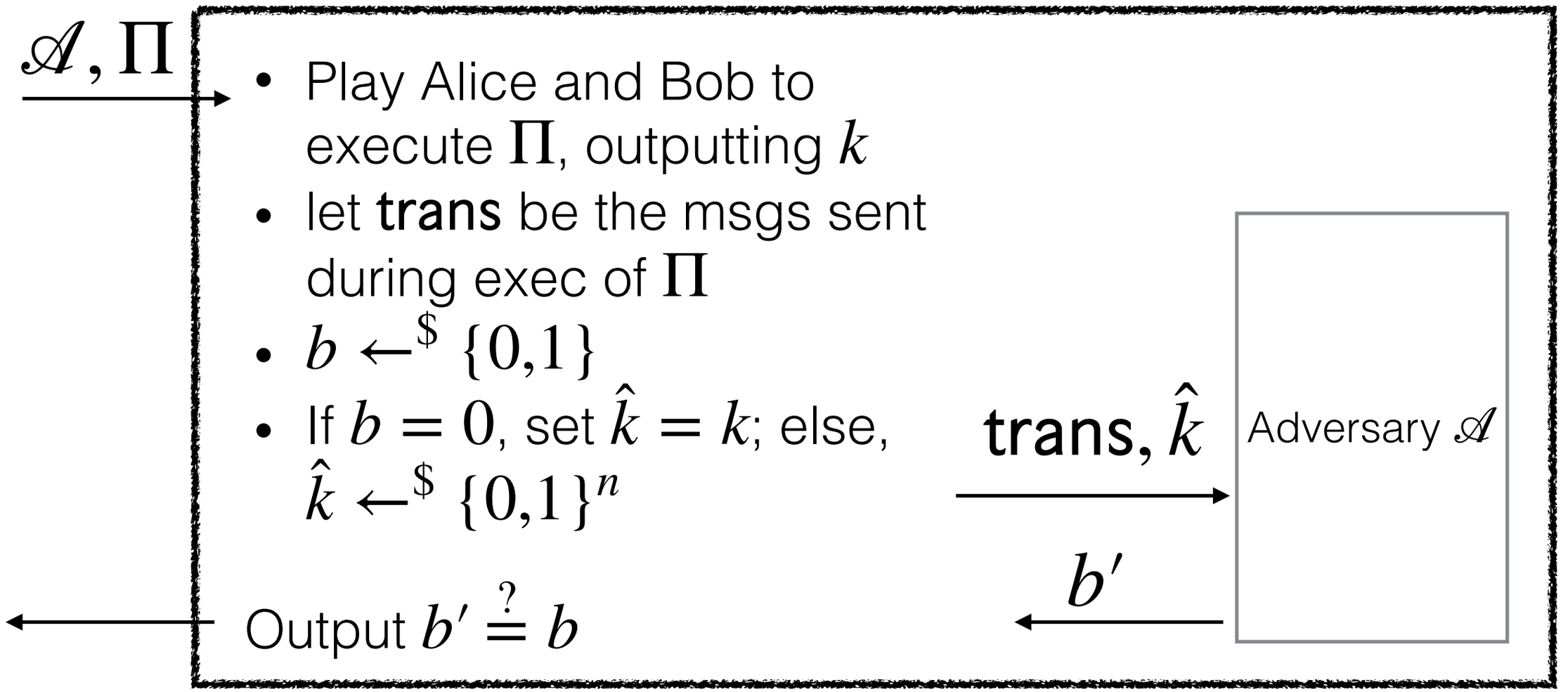
Security goals

- Correctness
- Security

How to define security?

- Attempt: Pr[Adversary guess the key] = negl
- Reason #1: Too lenient:
  - A protocol that leaks $n/2$ bits will be considered secure.
- Reason #2: Subsequent use of KE outputs often requires indistinguishability from random keys
  - E.g., encryption is secure only if random keys were used
- Formally: Attacker cannot distinguish session key from random group elements.

Game $\mathrm{KE}^{\mathrm{eav}}_{\mathscr{A},\Pi}$



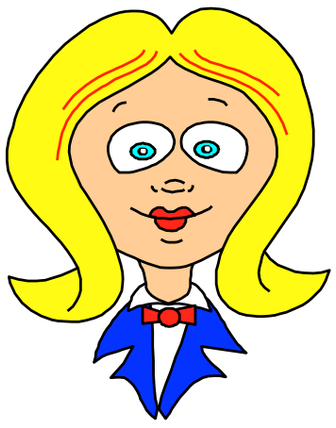
I.e., $\mathscr{A}$ wins if she can **distinguish** the key generated by Π from a uniform key.


Diffie-Hellman key agreement

First practical public-key cryptosystem… and the simplest.

DH Key Agreement

Eavesdropper
No imposters*



Alice

$x \leftarrow^{\$} \mathbb{Z}_q$

$h_A = g^x$

$h_B = g^y$

Bob

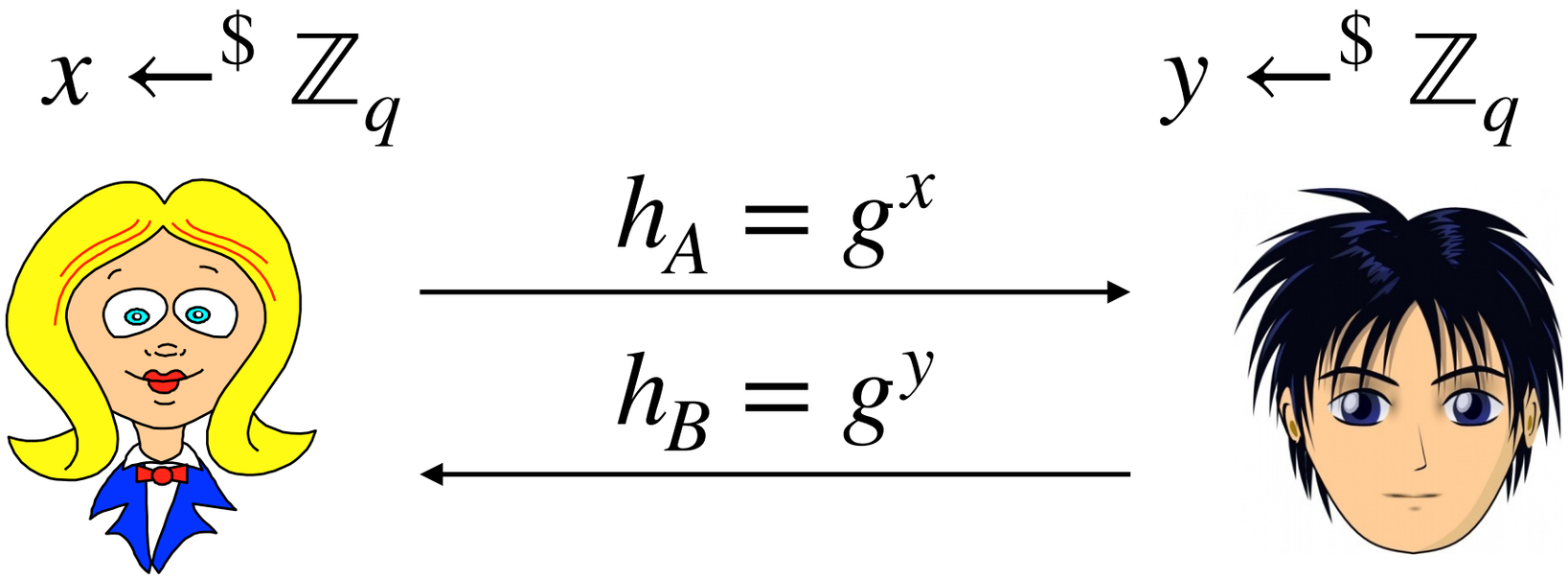
$y \leftarrow^{\$} \mathbb{Z}_q$

Output h_B^x

Output h_A^y

Correctness: $h_B^x = g^{yx} = g^{xy} = h_A^y$

Is DH KE secure against Eav?



- Is DL sufficient; is it necessary?
- Maybe Eav can distinguish g^{xy} from a uniform group element without computing it?

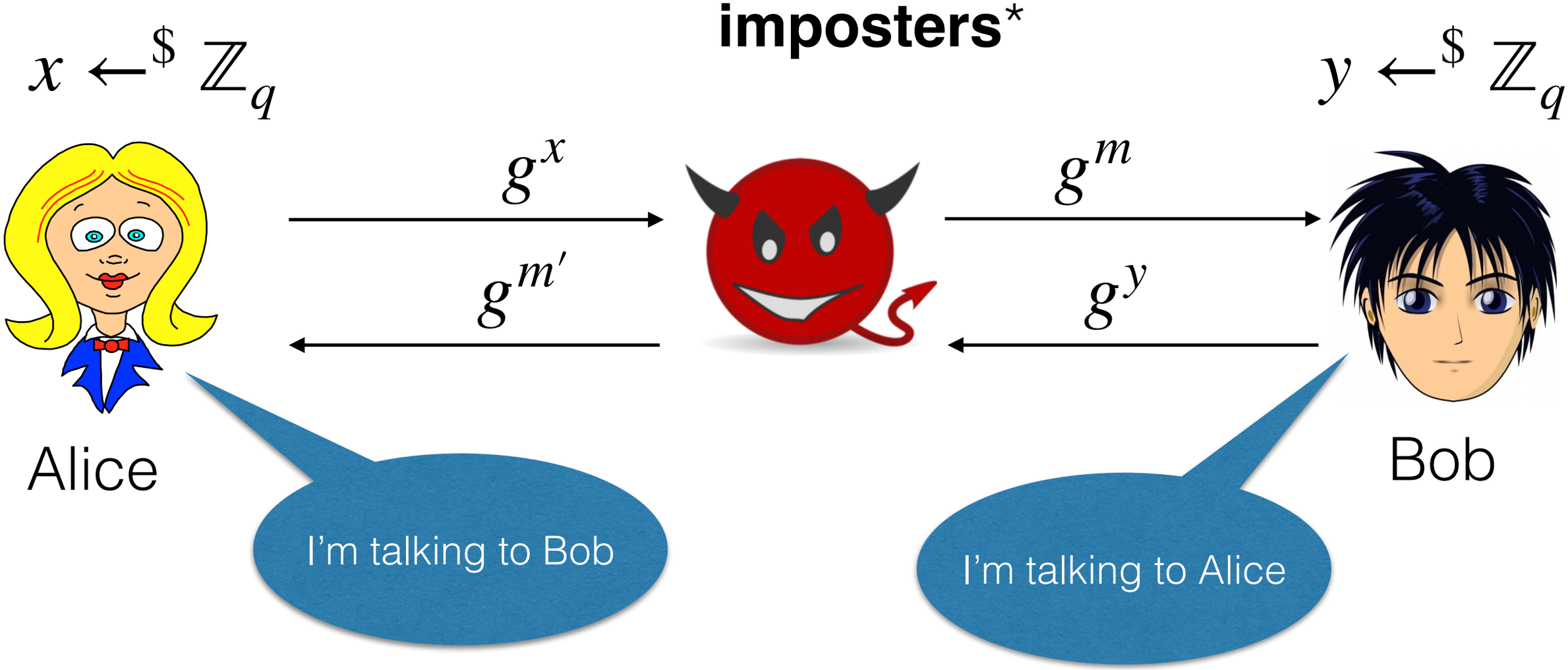
New assumption

- DL not enough; a new assumption was introduced to prove the security of DH KE.
- **Decision DH (DDH) assumption**: given random $g^x, g^y \leftarrow^{\$} \mathbb{G}$, it's computationally hard to distinguish g^{xy} from a uniform element in $\mathbb{G}$.
  - Computational version (CDH): hard to compute g^{xy}.
- Isn't this cheating (security by assumption)?
  - We can study the assumptions separately from protocols.
  - DDH assumption are very useful in many others protocols.

How to get keys?

- KDF: Key Derivation Function
  - converts high-entropy secrets to uniform n-bit string (which encryption requires)
- To use with DH KE:
  - Serialize the group element output by DH to a binary string
  - Use the binary string as input to KDF to get keys
- Examples: HKDF or SHA-256 (in ROM)

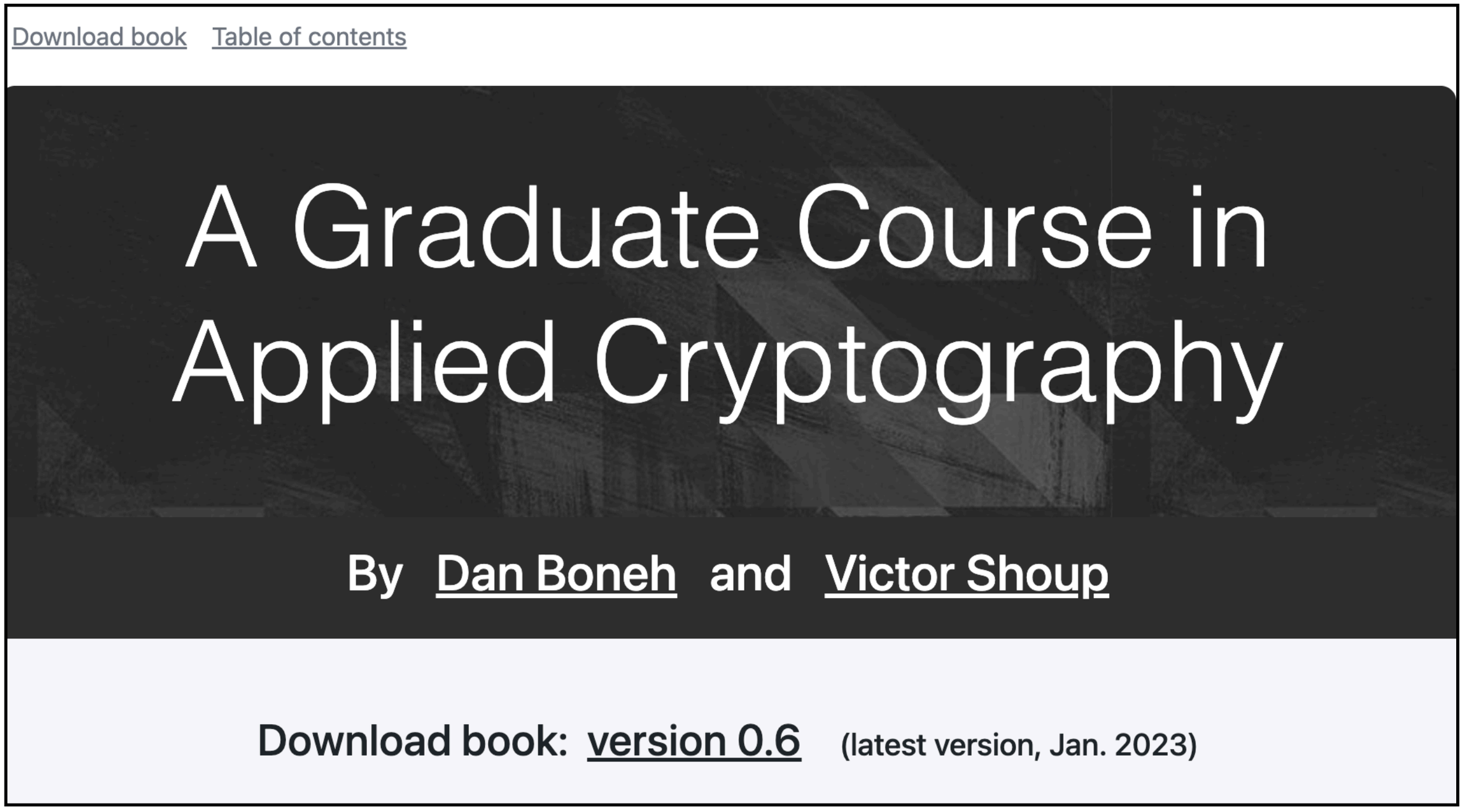
Imposters: MITM attack



Attacker can relay messages (decrypt then encrypt) without being detected!

Authenticated KE

Reference to BV

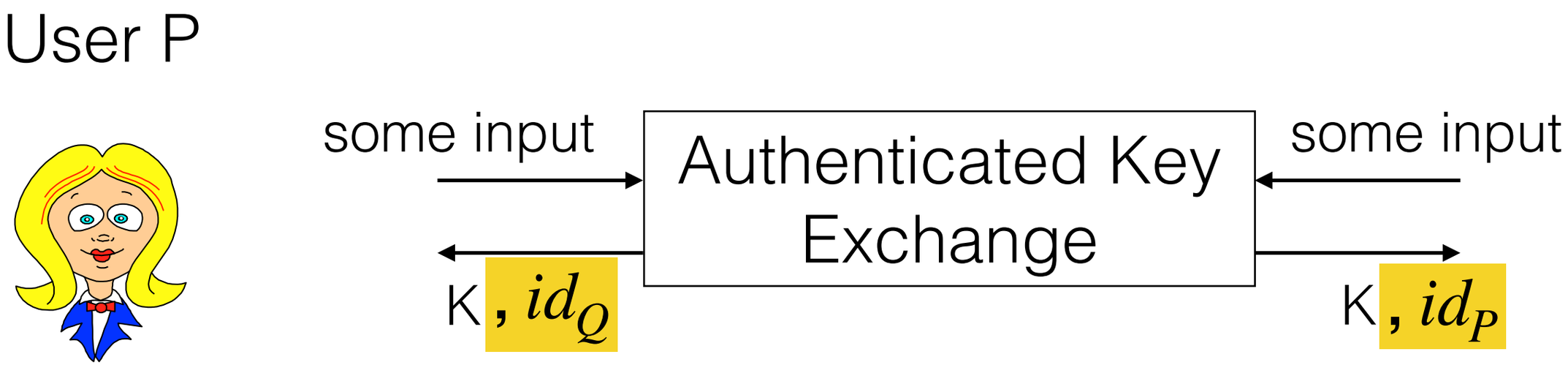


- Freely available from https://toc.cryptobook.us/

AKE: Identity and PK

- Each player identity **id** and **pk**
  - E.g., **id** can be domain names (TLS), email addresses (S/MIME), real name (e-signatures)
- PKs and identities must be connected. We assume a trustworthy **certificate authority**
  - P presents identity documents to CA (e.g., legal docs, DNS challenges) and a PK_P of their choice
  - CA verifies identity documents
  - CA issues a *certificate* $\mathsf{Cert}_P = (\mathsf{id}_P, \mathsf{pk}_P, \sigma_{\mathsf{ca}})$
- Certificates are public (known to all)

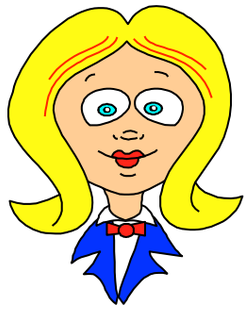
AKE: Goals



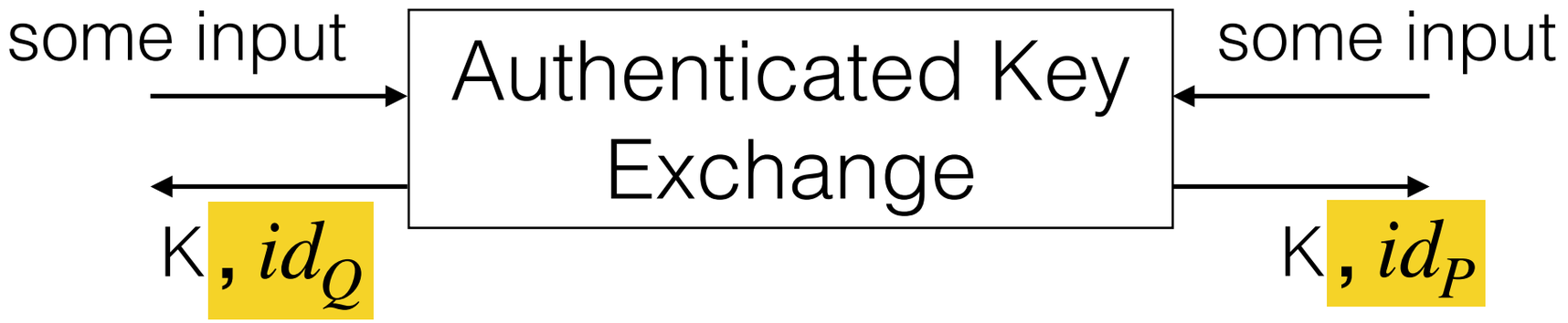
- If P successfully terminates, she outputs
- 1) a session key K
- 2) the identity of party that she believes she is talking to
  - If P expects to talk to Q, she aborts when she realizes that she is not talking to Q.
- (The same holds for Q)

AKE: Basic Goals

User P



User Q

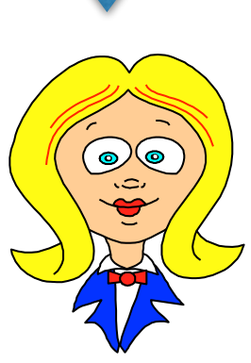


- **Authenticity**: If a party outputs $(K, \mathbf{id})$, then K is shared with only **id** — or nobody.
- **Secrecy**: adversary can't indistinguishable K from a random string by observing the communication
  - e.g., ***Key re-use attack***: It breaks secrecy if the adversary can force re-use of an old key from a previous run.

Tool: Signatures

message ***m***

Alice's private key: ***SK***



(m, S)

Alice's public key: ***PK***

$s = \text{Sign}(SK, m)$

$\text{Verify}(PK, m, S)$ ✔

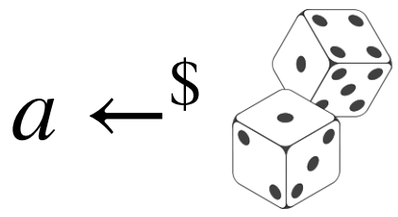
- Security goals (**unforgeability**): Infeasible to forge signatures after observing many valid (m, S) pairs

Tool: Public Key Encryption

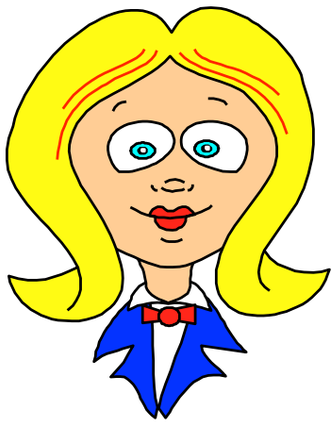
- $\mathsf{sk}, \mathsf{pk} \leftarrow^{\$} \mathsf{Gen}()$
- $c = \mathsf{Enc}(\mathsf{pk}, m)$
- $m = \mathsf{Dec}(\mathsf{sk}, c)$
- Examples of PKE
  - RSA — the first PKE
  - El Gamal (based on DDH) — to be introduced later.

Warmup: DHE with Sigs

- **D**iffie–**H**ellman with **e**phemeral key
  - $\mathbb{G}$ is cyclic group of prime order q with generator g

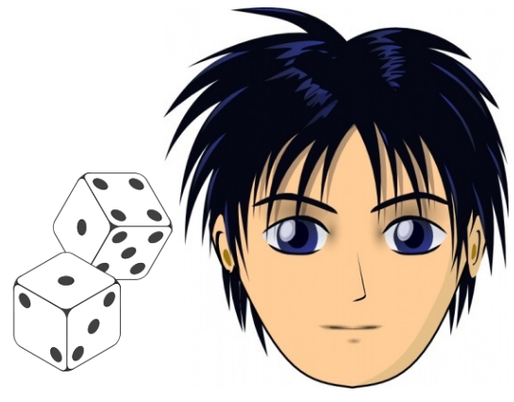


$a \leftarrow^{\$}$



$u := g^a, \mathsf{Sig}_P(u), \mathsf{Cert}_P$

$v := g^b, \mathsf{Sig}_Q(u, v), \mathsf{Cert}_Q$



$b \leftarrow^{\$}$

P

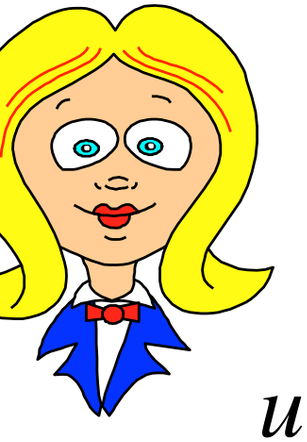
Q

Derive keys from g^{ab}

Derive keys from g^{ab}

Is this protocol a secure AKE?

Identity misbinding attack



P

$u := g^a, \mathsf{Sig}_P(u), \mathsf{Cert}_P$

E

$u, \mathsf{Sig}_E(u), \mathsf{Cert}_E$

Q

$v := g^b, \mathsf{Sig}_Q(u, v), \mathsf{Cert}_Q$

$v := g^b, \mathsf{Sig}_Q(u, v), \mathsf{Cert}_Q$

$k = H(g^{ab})$

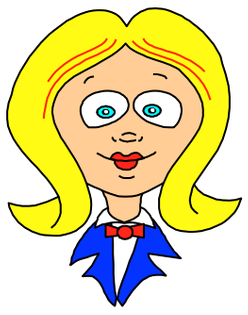
$k = H(g^{ab})$

Thinking: I'm talking with Q

Thinking: I'm talking with E

N*ot* a secure AKE: no authenticity.

Identity misbinding attack



P think K is associated with Q

Q think K is associated with E

P

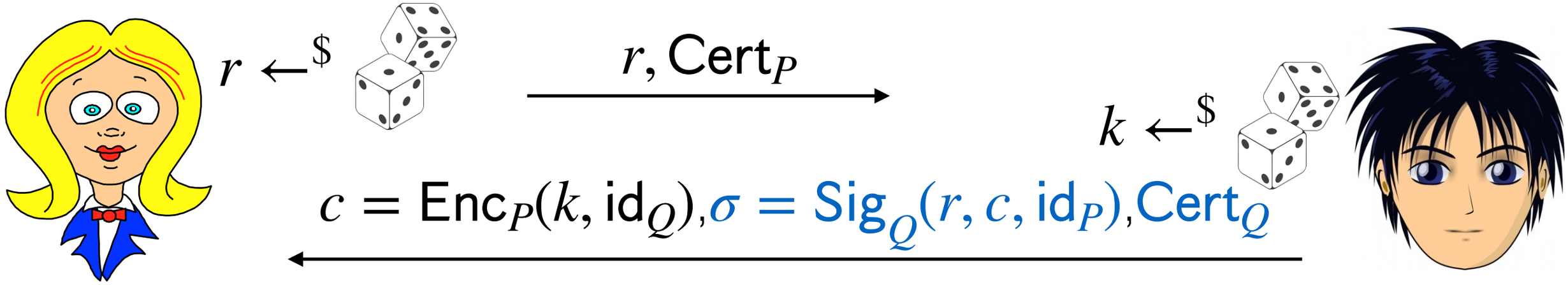
E

Q

- E does NOT learn the session key
- The damage is confusion (or authenticity)
  - E.g. P = CFO says “move all funds to my account”. E = a regular employee.
- Point: AKE requires careful combination of **identification** and **key exchange**.

AKE1

- Notations:
  - $\mathsf{id}_P, \mathsf{pk}_P$ are contained in $Cert_P$,
  - pk_P consists of two keys: one for encryption and one for signature
  - sk_P consists of two keys too (they are long-term)
  - Shorthands: $\mathsf{Enc}_P(m) := \mathsf{Enc}(\mathsf{pk}_P, m)$; $\mathsf{Sig}_P(m) := \mathsf{Sig}(\mathsf{sk}_P, m)$



P

Output k

- Encrypting k
- Bind k to id_Q

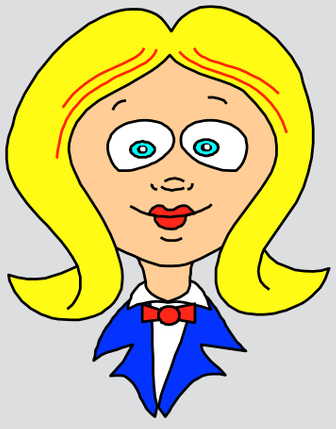
- Sig proves integrity and origin of c
- r ensures Q's response is **fresh**
- "echo" Alice's id so there is no IMB

Q

Output k

Can we simply?

Insecure variant #1: not signing c

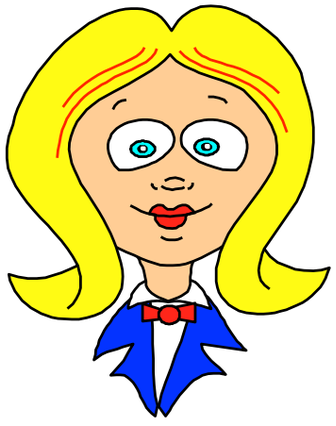


P

r, Cert_P →

← $c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(r, c, \mathsf{id}_P), \mathsf{Cert}_Q$

Q



P

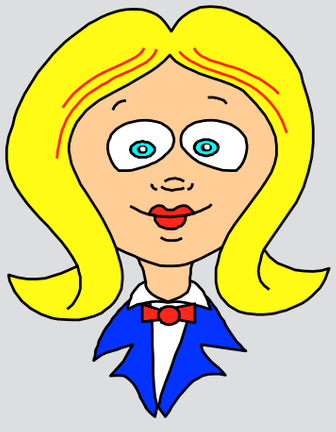
r, Cert_P →

← $c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(r, \mathsf{id}_P), \mathsf{Cert}_Q$

Q

Key exposure attack: Attacker can replace c with encryption of her own key. Violates secrecy.

Insecure variant #2: not including r

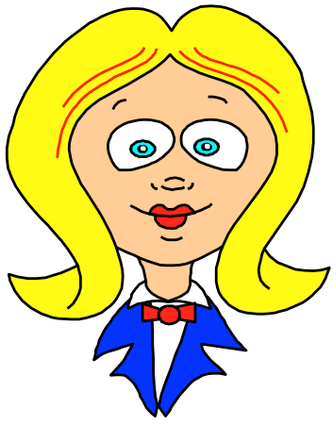


P

r, Cert_P

$c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(r, c, \mathsf{id}_P), \mathsf{Cert}_Q$

Q



P

Cert_P

$c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(c, \mathsf{id}_P), \mathsf{Cert}_Q$

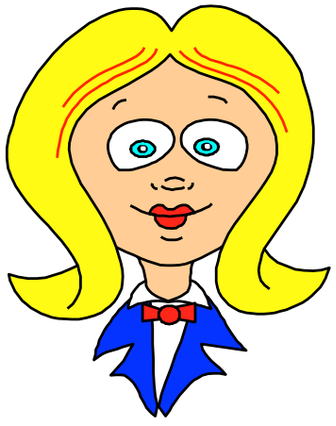
Q

Replay attack: causing P to re-use old sessions key. Violates secrecy.

Insecure variant #2: not including r

$$\text{Cert}_P \longrightarrow$$

$$\longleftarrow c = \text{Enc}_P(k, \text{id}_Q), \sigma = \text{Sig}_Q(c, \text{id}_P), \text{Cert}_Q$$



P

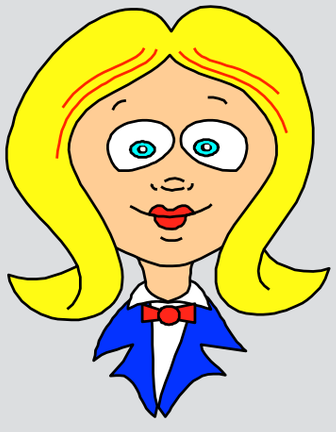
Q

$$\text{Cert}_P \longrightarrow$$

$$\longleftarrow c, \sigma, \text{Cert}_Q$$

Why re-using old keys bad?: 1) may break security of encryption (eg CTR mode); 2) enables message replay.

Insecure variant #3: not signing id

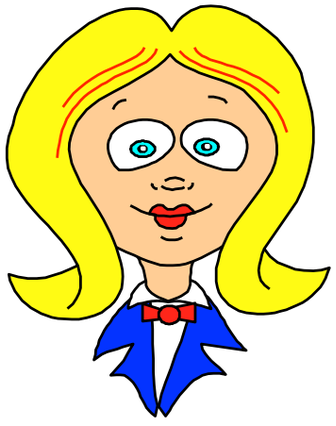


P

$$r, \mathsf{Cert}_P \longrightarrow$$

$$\longleftarrow c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(r, c, \mathsf{id}_P), \mathsf{Cert}_Q$$

Q



P

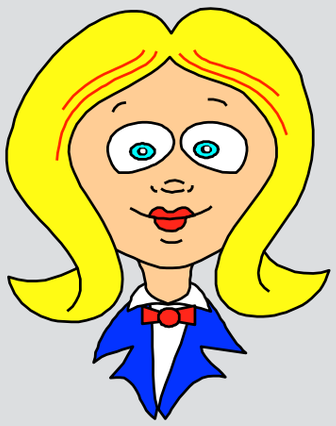
$$r, Cert_P \longrightarrow$$

$$\longleftarrow c = Enc_P(k, id_Q), \sigma = Sig_Q(r, c), Cert_Q$$

Q

Hint: attacker can stitch messages from different runs.

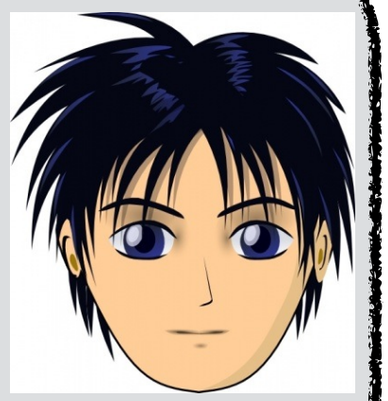
Insecure variant #3: not signing id



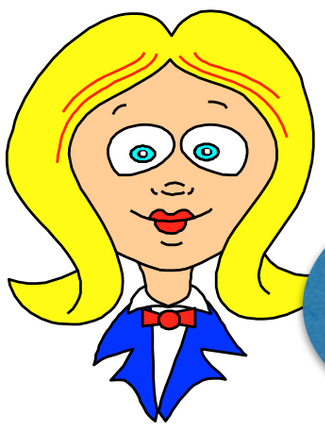
P

$$r, \mathsf{Cert}_P \longrightarrow$$

$$\longleftarrow c = \mathsf{Enc}_P(k, \mathsf{id}_Q), \sigma = \mathsf{Sig}_Q(r, c, \mathsf{id}_P), \mathsf{Cert}_Q$$



Q



P

I'm talking to Q

$$r, Cert_P \longrightarrow$$

I'm talking to A

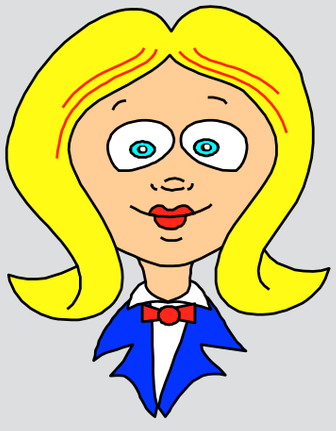
$$r, Cert_A(pk_P) \longrightarrow$$

Q

$$\longleftarrow c = \mathsf{Enc}_P(k, id_Q), \sigma = Sig_Q(r, c), Cert_Q$$

Identity Misbinding Attack (similar to DHE)

Insecure variant #4: not encrypting id

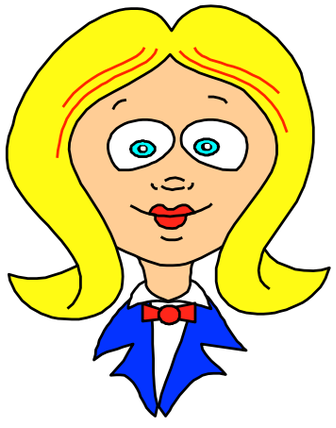


P

$$r, Cert_P \longrightarrow$$

$$\longleftarrow c = Enc_P(k, id_Q), \sigma = Sig_Q(r, c, id_P), Cert_Q$$

Q



P

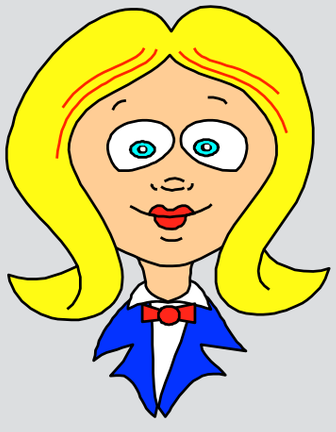
$$r, Cert_P \longrightarrow$$

$$\longleftarrow c = Enc_P(k), \sigma = Sig_Q(r, c, id_P), Cert_Q$$

Q

Hint: stitch

Insecure variant #4: not encrypting id

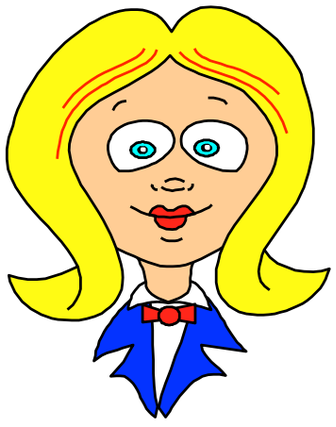


P

$$r, Cert_P \longrightarrow$$

$$\longleftarrow c = Enc_P(k, id_Q), \sigma = Sig_Q(r, c, id_P), Cert_Q$$

Q



$$\longleftarrow c = Enc_P(k), \sigma = Sig_Q(r, c, id_P), Cert_Q$$

$$r, Cert_P \longrightarrow$$

$$c = Enc_P(k), \sigma = Sig_A(r, c, id_P), Cert_A \longleftarrow$$

P

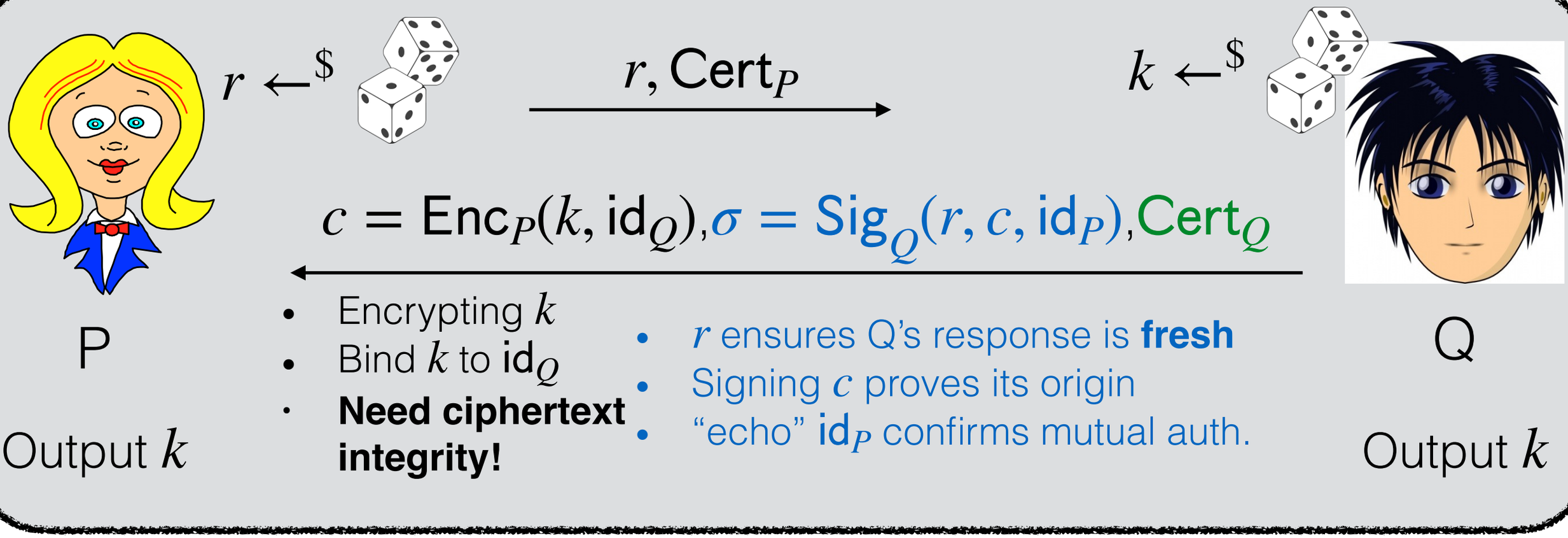
Q

Combination of **replay & Id misbiding attack**:

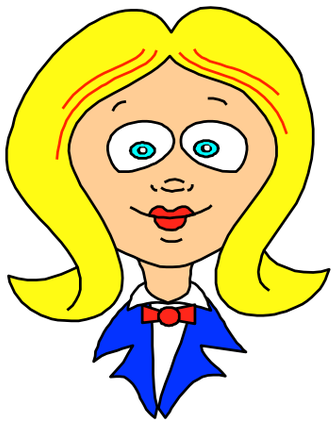
- causing P to re-use old sessions key. Violates secrecy.
- Plus P thinks she is talking to A instead of Q.

AKE1

- Indeed every bit of it is essential
- See [BV 21.9] for a (rather involved) proof
- Quirk: Q can't know if P participated in the protocol live.



Need ciphertext integrity



P

$r, Cert_P$

$c = Enc_P(k, id_Q), \sigma = Sig_Q(r, c, id_P), Cert_Q$

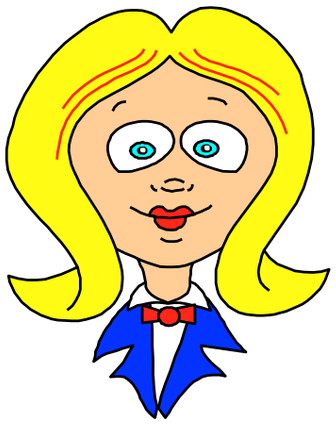
Q

If Enc is only CPA secure, this is possible.

$c = Enc_P(k, id_A)$

How to use this to attack?

Need ciphertext integrity (CCA)



P

$r, Cert_P$

$c = Enc_P(k, id_Q), \sigma = Sig_Q(r, c, id_P), Cert_Q$

Q

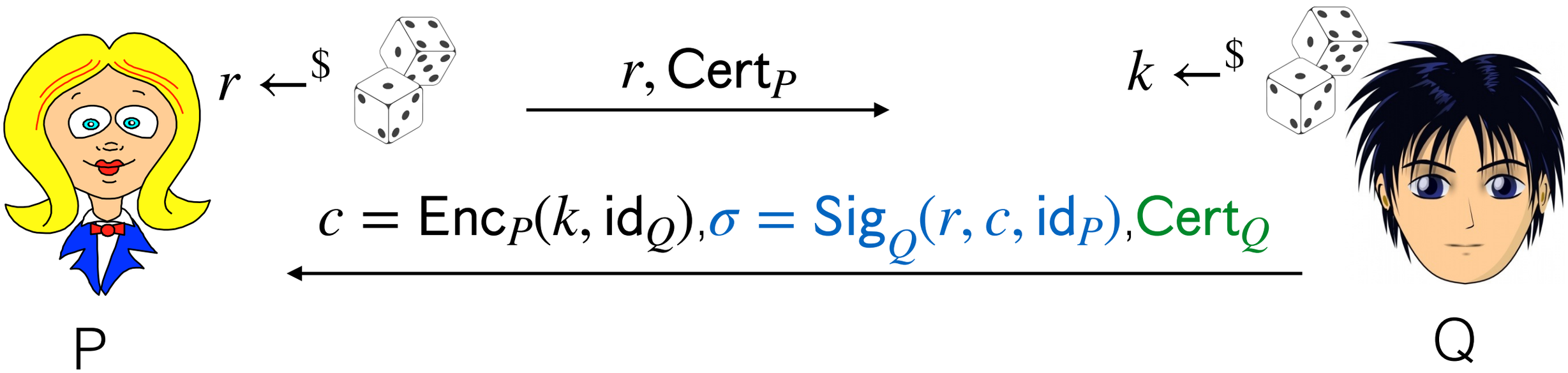
$c = Enc_P(k, id_A), \sigma = Sig_A(r, c, id_P), Cert_A$

Replay attack: causing P to re-use old sessions key. Violates secrecy.

Solution: Use CCA secure PKE (next lecture)

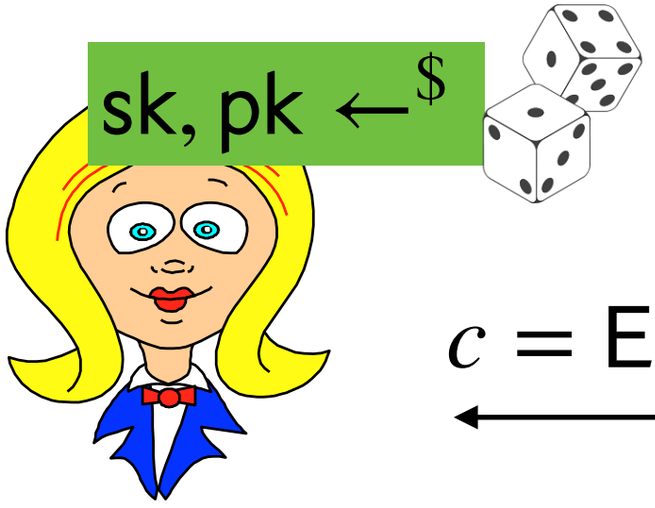
Limitations of AKE1

- What happens if secret keys are compromised?
  - sk_Q: can impersonate Q going forward
  - sk_P: can decrypt all session keys (future and past)
- Given that key compromise may be inevitable, it would be nice if session keys generated *before the compromise* remain secret.
- Known as **Perfect Forward Secrecy (PFS)**
  - Old keys remain secure going *forward*



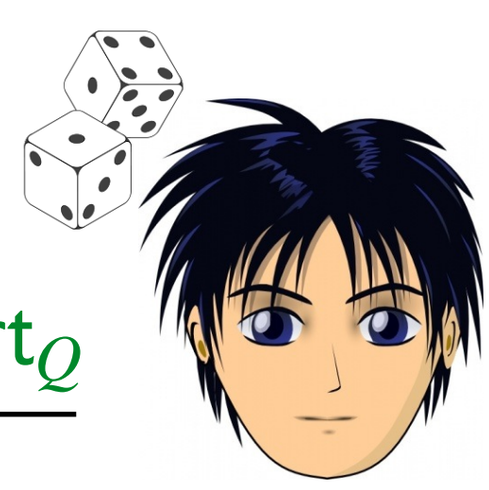
AKE2

- P samples a fresh **pk** every time to encrypt session key
- AKE2 = AKE1 with r replaced by **pk**
- Same reasoning of security as AKE1; Insecure simplifications to AKE1 are also insecure for AKE2.



$$\text{pk}, \sigma_1 = \text{Sig}_P(pk), \text{Cert}_P \longrightarrow$$

$$k \leftarrow^{\$}$$



$$\longleftarrow c = \text{E}(\text{pk}, (k, \text{id}_Q)), \sigma = \text{Sig}_Q(\text{pk}, c, \text{id}_P), \text{Cert}_Q$$

P

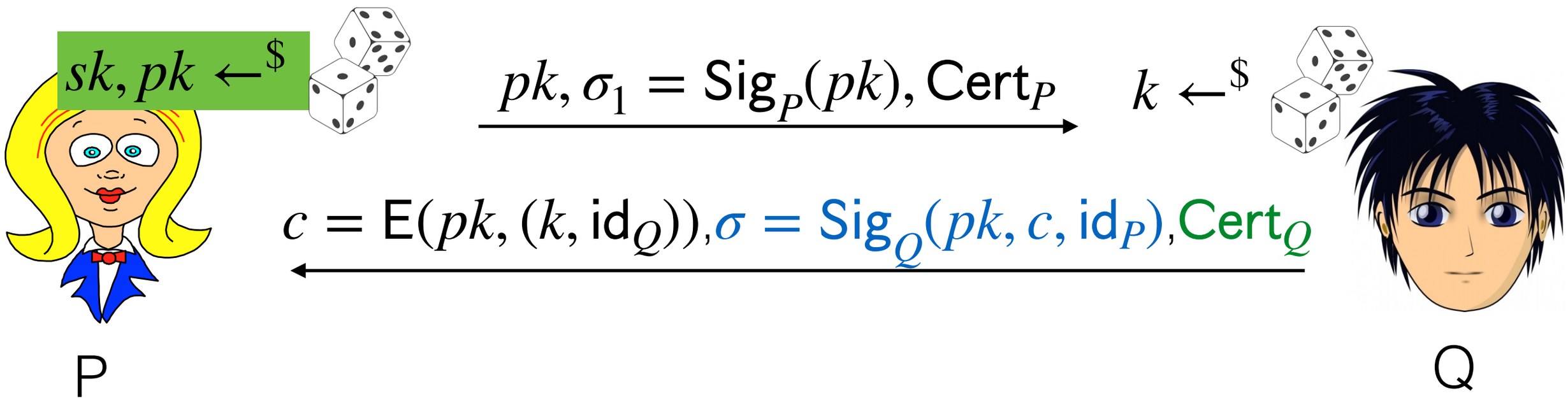
- Encrypting k
- Bind k to id_Q
- Requires CCA

- pk provides PFS

Q

Brittleness of AKE2

- AKE2 is quite brittle: If an ephemeral sk is leaked, then attacker can impersonate P **at any time, as often as he likes, to any user**
  - E.g., temporary infection with malware
  - Good design: leaking ephemeral secrets should not cause long-last damage (e.g., until removal of malware).

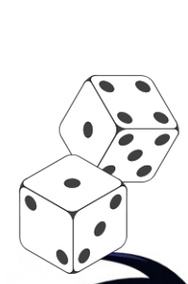


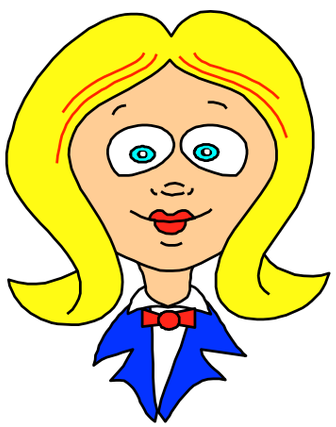
AKE3

- Idea: P needs to sign **pk** with fresh randomness from Q
- c is used as fresh randomness from Q

Need signing key to run protocol

$\mathsf{pk}, \mathsf{Cert}_P$





P

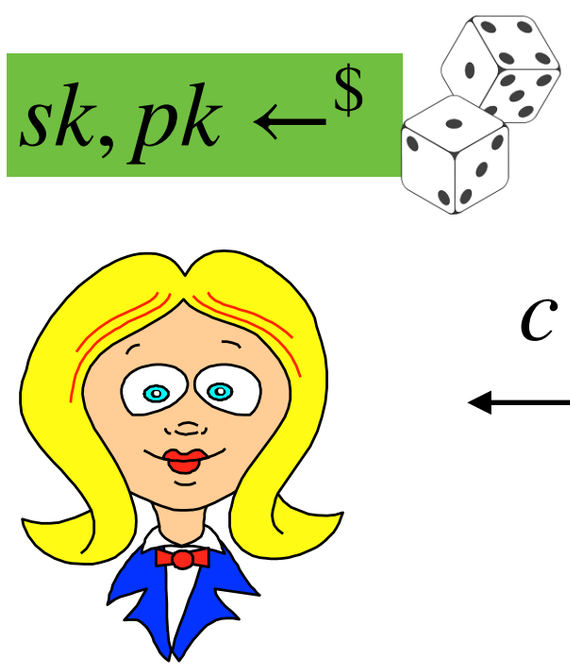
$c = \mathsf{E}(\mathsf{pk}, k), \sigma_1 = \mathsf{Sig}_Q(1, \mathsf{pk}, c, \mathsf{id}_P), \mathsf{Cert}_Q$

Q

$\sigma_2 = \mathsf{Sig}_P(2, \mathsf{pk}, c, \mathsf{id}_Q)$

1 and 2 are **domain separator**

Insecure variant: not signing $\mathbf{id}_Q$ in σ_2



P

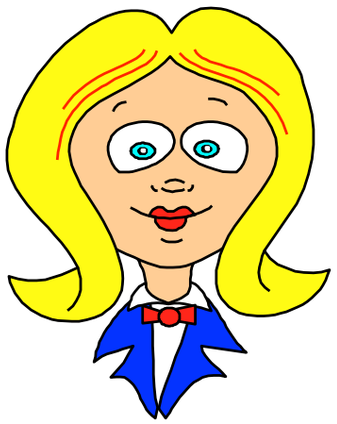
Q

Another requirement: Identity Protection

- AKE1-3 leak who is talking to whom
  - Certs are send in the clear
- **Privacy from Eavesdropper**: P and Q's certs better be encrypted
- **Full identity protection for P**: only reveals identity to Q after P is sure who she is talking to.
  - At least one party can't enjoy full identity protection

AKE4 (Identity Protection)

AKE3



$\mathsf{pk}, \mathsf{Cert}_P$

$c = \mathsf{E}(\mathsf{pk}, k), \sigma_1 = \mathsf{Sig}_Q(1, \mathsf{pk}, c, \mathsf{id}_P), \mathsf{Cert}_Q$

$\sigma_2 = \mathsf{Sig}_P(2, \mathsf{pk}, c, \mathsf{id}_Q)$

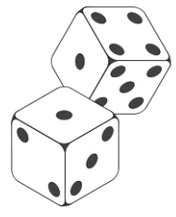
P

$sk, pk \leftarrow^{\$}$

pk

Q

$k, k_1, k_2 \leftarrow^{\$}$

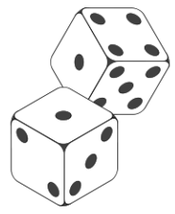


$c = \mathsf{E}(\mathsf{pk}, (k, k_1, k_2)), c_1 = \mathsf{E}(k_1, \mathsf{Sig}_Q(1, \mathsf{pk}, c), \mathsf{Cert}_Q)$

$c_2 = \mathsf{E}(k_2, \mathsf{Sig}_P(2, \mathsf{pk}, c), \mathsf{Cert}_P)$

$sk, pk \leftarrow^{\$}$ 

pk

$k, k_1, k_2 \leftarrow^{\$}$ 

$c = \mathsf{E}(\mathsf{pk}, (k, k_1, k_2)), c_1 = \mathsf{E}(k_1, \mathsf{Sig}_Q(1, \mathsf{pk}, c), \mathsf{Cert}_Q)$

- Decrypt c (abort if any step below fails)
- Decrypt c_1
- Verifies σ_1 against Cert_Q and $1, pk, c$

$c_2 = \mathsf{E}(k_2, \mathsf{Sig}_P(2, \mathsf{pk}, c), \mathsf{Cert}_P)$

- Outputs k, and id_Q

id_Q is not in second sig (c.f. AKE3). Why is it okay?

B/c of ciphertext integrity.

- Decrypt c_2
- Verifies sig against Cert_P and $2, pk, c$
- Output k, and id_P

Pretty close to TLS 1.3 handshake!