密码学进阶

第四课：可证明安全（2）

授课老师：张秉晟

bingsheng@zju.edu.cn

QUIZ: Is Exp one-way?

- What do you think?
  - <u>Depends on the group</u>
- **Easy:**
  - $(\mathbb{R}^*, \cdot)$: the inverse of exp is logarithm
  - $(\mathbb{Z}, +)$, $(\mathbb{Z}q, +)$: exp = multiplication, inverse = division
- In finite groups, inverse of exp is called **discrete logarithm**

Hard DL groups

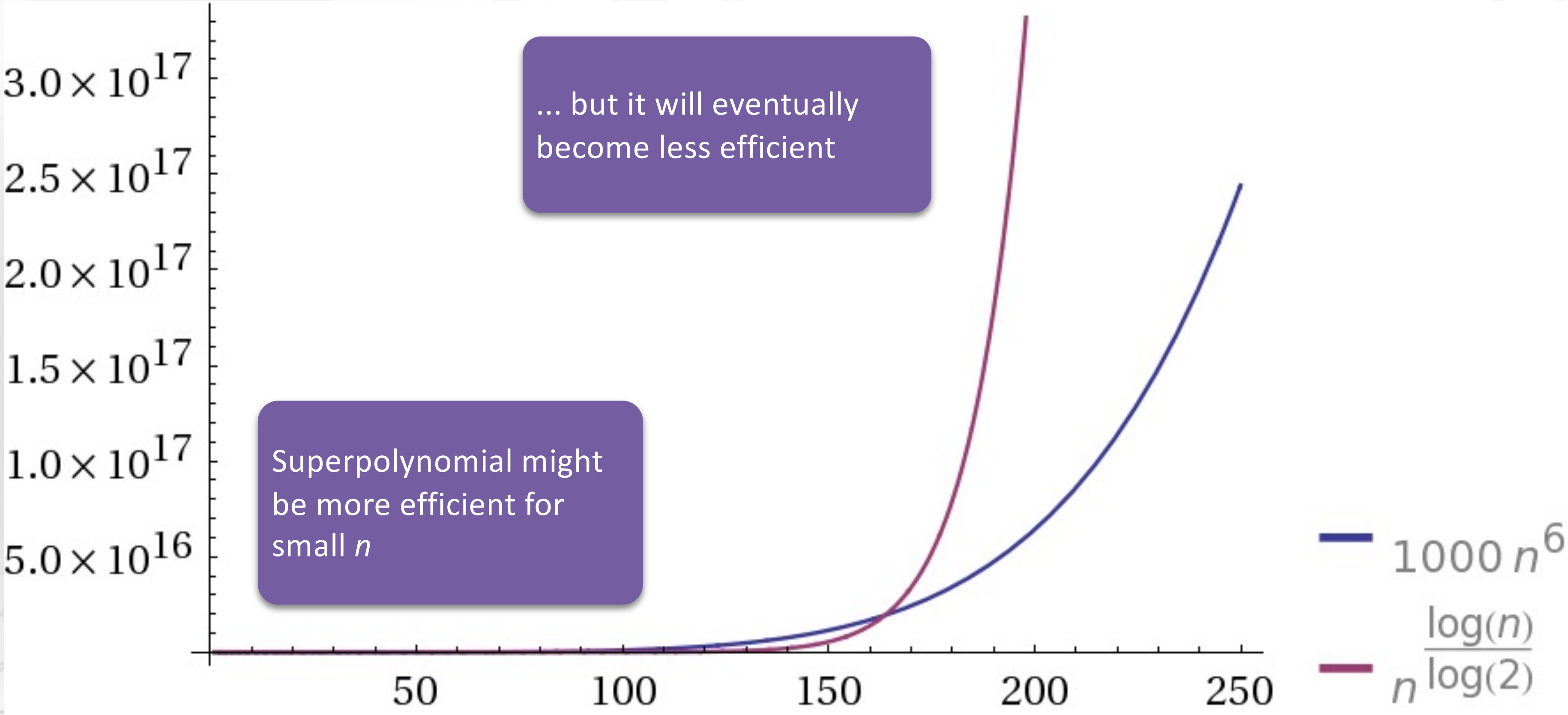
- **Instantiation 1**
  - Let p be a big prime (3000+ digits)
  - The order of $\mathbb{Z}_p^*=\{1, 2, \ldots, p-1\}$ is $p-1$
  - Let $q \mid (p-1)$ be a smaller prime (160+ digits)
  - By Cauchy/Sylow theorem, $\mathbb{Z}_p^*$ has a unique subgroup G of order q
- DL is assumed to be hard in G

Best known algorithms to break DL in G have subexponential complexity in |p| and exponential complexity in |q|

Reminder: basic complexity theory

- **Running time** $T(n)$ of algorithm = function of input length n
  - **Example.** Running time of exponentiation is a function of the bitlength n of the group elements
- **Simplification:** in $\mathbb{Z}q$, $n = \log q$
- **Efficient algorithm:** $T(n)$ is polynomial in n
  - E.g.: $T(n) = 1000 \cdot n^6$
- **Inefficient algorithm:** $T(n)$ is not polynomial in n
  - E.g.: $T(n) = n^{(\log n)}$

Efficient vs inefficient



Complexity in cryptography

- When we encrypt, security should not depend on the message length but say on key size

- Instead of input length n, take security parameter κ

- Usually κ related to key length

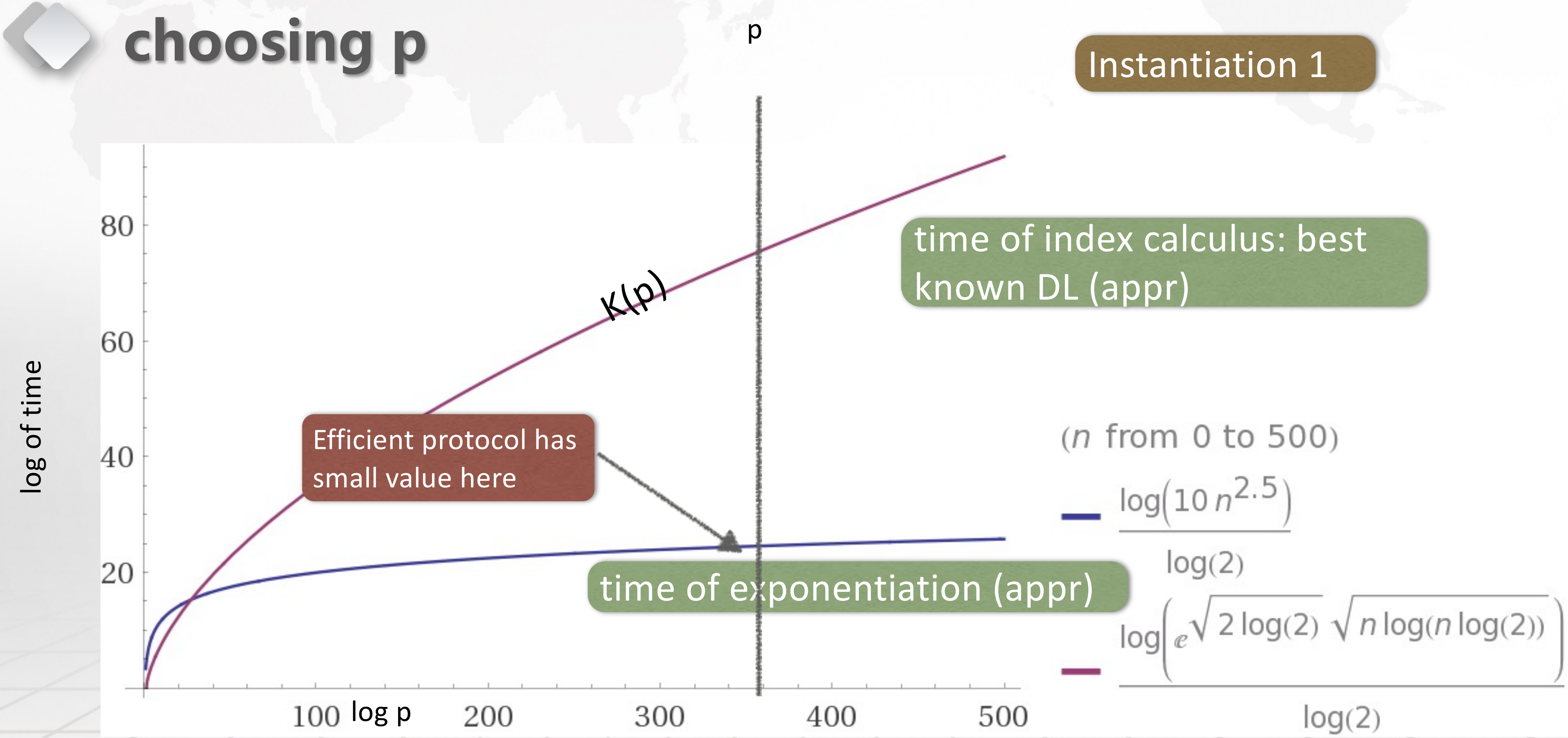
  - First, fix κ so that $T(\kappa)$ of attacks is big and of "honest" algorithms is small

  - Finally, choose corresponding key

Corollaries of complexity

- Most algorithms work with undetermined κ
- In practical implementations fix κ so that protocol is fast but attacks are assumed to be hard
  - E.g., attacks take time 2^{80}
- If attacks are improved somewhat, increase κ accordingly

choosing p



- Too optimistic graph, in practice *p* is much larger

complexity notation

- $\Theta(f(n))$: asymptotically $c\,f(n)$ for some constant c
  - $100n^2 + 20n - 10 = \Theta(n^2)$
- $O(f(n))$: any func that does <u>not</u> grow faster than $\Theta(f(n))$
- $o(f(n))$: any function that grows slower than $\Theta(f(n))$
- $\Omega(f(n))$: any func that does <u>not</u> grow slower than $\Theta(f(n))$
- $\omega(f(n))$: any function that grows faster than $\Theta(f(n))$

quiz

$\Theta(n^8)$	$O(n^8)$	$\Omega(n^8)$	$o(n^8)$	$\omega(n^8)$
$\Theta(n^7)$	$O(n^7)$	$\Omega(n^7)$	$o(n^7)$	$\omega(n^7)$
$\Theta(n^6)$	$O(n^6)$	$\Omega(n^6)$	$o(n^6)$	$\omega(n^6)$
$\Theta(n^5)$	$O(n^5)$	$\Omega(n^5)$	$o(n^5)$	$\omega(n^5)$

- **Question:** What is $(n^8 + n + 1) / (n^2 + n + 1)$?

- **Answer:** it is n^6 + smaller terms
  - thus $\Theta(n^6)$

complexity notation

- **polynomial**: poly (n) = n^(O (1)) not faster than any polynomial
- **superpolynomial**: n^(ω (1)) faster than any polynomial
- **exponential**: 2^(Θ (n))
- **negligible**: negl (n) = n^(-ω (1)) slower than inverse of any polynomial
- **linear**: Θ (n) asymptotically c n for some constant c
- etc: logarithmic, superlogarithmic, sublinear

Best known dl algorithms

- **Any groups** of order q, $n := \log q$
  - **Baby-step-giant-step** and **Pohlig-Hellman** algorithms --- $O(\sqrt{q})$
- **Instantiation 1**, parameters p and q
  - **Index calculus**, $O(e^{\wedge}(\sqrt{(2 \ln p \ln \ln p)}))$
  - **BSGS/PH algorithms** $O(\sqrt{q})$
- Recent advances in groups of order p^m for midsize m
- DL in **any group** can be broken by using **quantum computer**

Generic algorithms: only use group operations

Hard DL groups

- **Instantiation 2**
  - Elliptic curve groups
  - Let q be a small prime (160+ digits)
  - Elliptic curve group G has order q
  - Definition complicated

Best known algorithms to break DL in G have exponential complexity in |q|

- DL is <u>assumed</u> to be hard in well-chosen G

comparison of instantiations

Exponent 1.58 due to Karatsuba algorithm

Asymptotically not optimal, but good for inputs of that size

	Parameters	Group element representation	Complexity of multiplication	Security
$\mathbb{Z}p^*$	p, log p≥3200 q, log q≥160	log p	O((log p)^1.58)	2^{80}
E.C.G.	q, log q≥160	log q	O((log q)^1.58)	2^{80}

q is much smaller than *p*, though constant in *O*( ) is larger

DL assumption: Formal

- Informally, we need that inverting exponentiation is hard
- Complications:
  - when exponent is smaller than L , one can compute DL in $\Theta(\sqrt{L})$ steps
  - inverting is impossible when $g = 1$
  - inverting is always possible with probability $1 / q$ (guessing answer randomly)

Exponent must be random (e.g.,exponent is secret key)

G must be a fenerator

security must hold against probabilistic algorithms that can use random numbers

break is only successful when adversary's advantage is >> 1 / q

Security game

A challenger generates values from some fixed "valid" distributions and sends them to the adversary $\mathcal{A}$

After some computation, $\mathcal{A}$ returns some value to the challenger

Depending on the input and the output, the challenger declares $\mathcal{A}$ to be either successful or not

- $\mathcal{A}$ breaks the assumption if her advantage is big compared to random guessing

Def: DL groups

- Let G be a finite cyclic group of order q, let g be its fixed generator
  - One can take any g, or a random g
  - Assume desc(G) contains a description of G, incl. g
- Adv[DL(G, $\mathcal{A}$)] := | Pr[DL(G,$\mathcal{A}$) = 1] - 1 / q |
- $\mathcal{A}$ **ε-breaks DL in G** iff Adv[DL(G, $\mathcal{A}$)] $\geq \varepsilon$
- G is a **(τ,ε)-DL group** iff Adv[DL(G, $\mathcal{A}$)] $\leq \varepsilon$ for all probabilistic polynomial time adversaries $\mathcal{A}$ that take time $\leq \tau$
- G is a **DL group** iff it is a (poly(κ),negl(κ))-DL group

Game DL(G, $\mathcal{A}$)

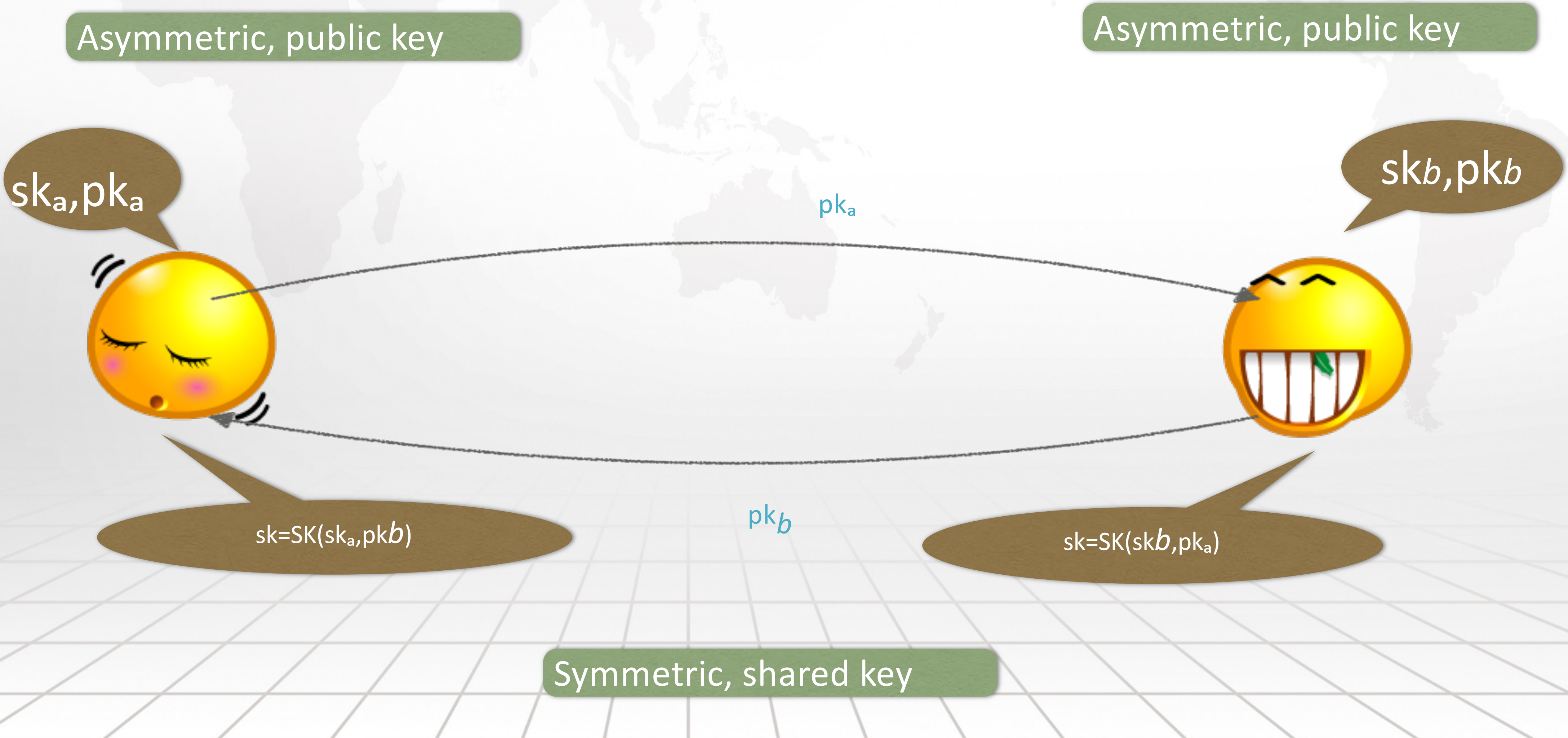
gk $\leftarrow$ desc(G)
$m \leftarrow \mathbb{Z}q$
$h \leftarrow g^m$
$m^* \leftarrow \mathcal{A}$(gk, h)
If $m = m^*$
  return 1
else
  return 0

What can be done with DL?

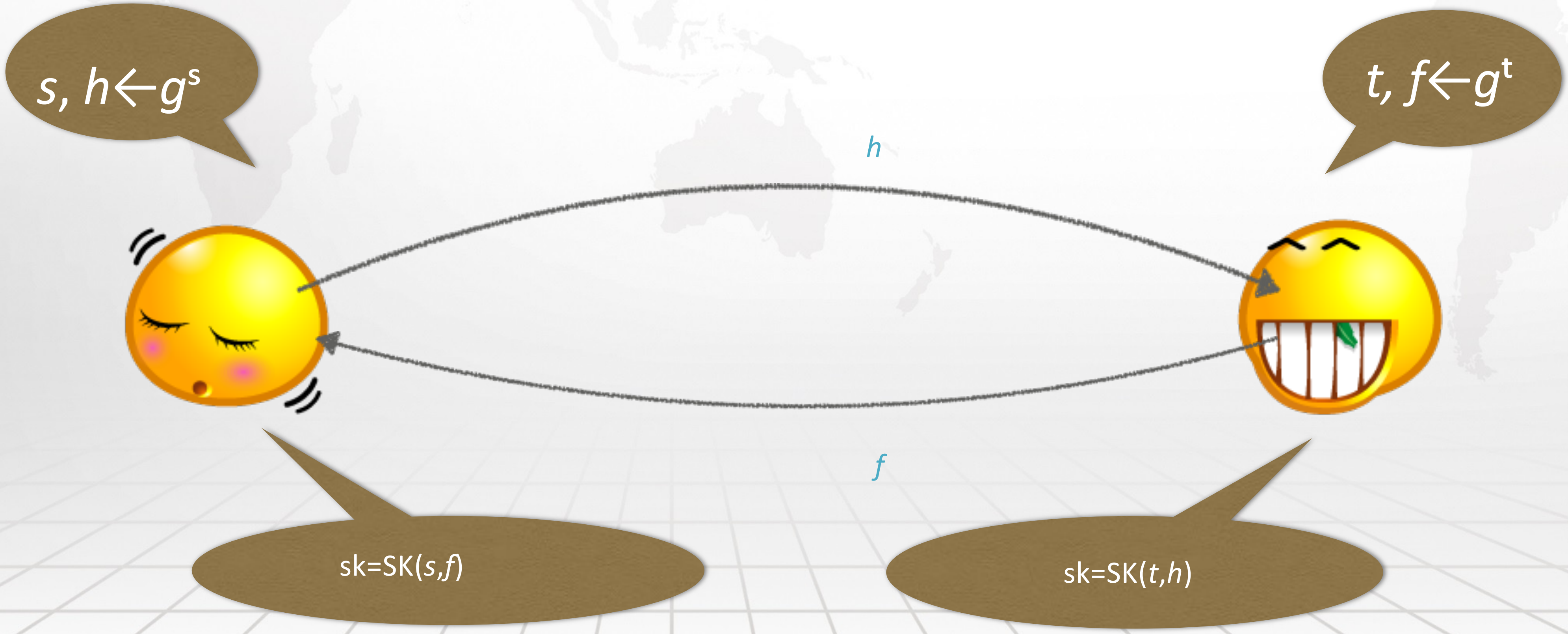
- **First idea:**
  - let $s \leftarrow \mathbb{Z}q$ be secret key and $h=g^s$ be public key
  - computation of s from h is infeasible
- Use the keys to "encrypt", "sign", etc
- **This lecture:** more details

Key Exchange

Key Exchange



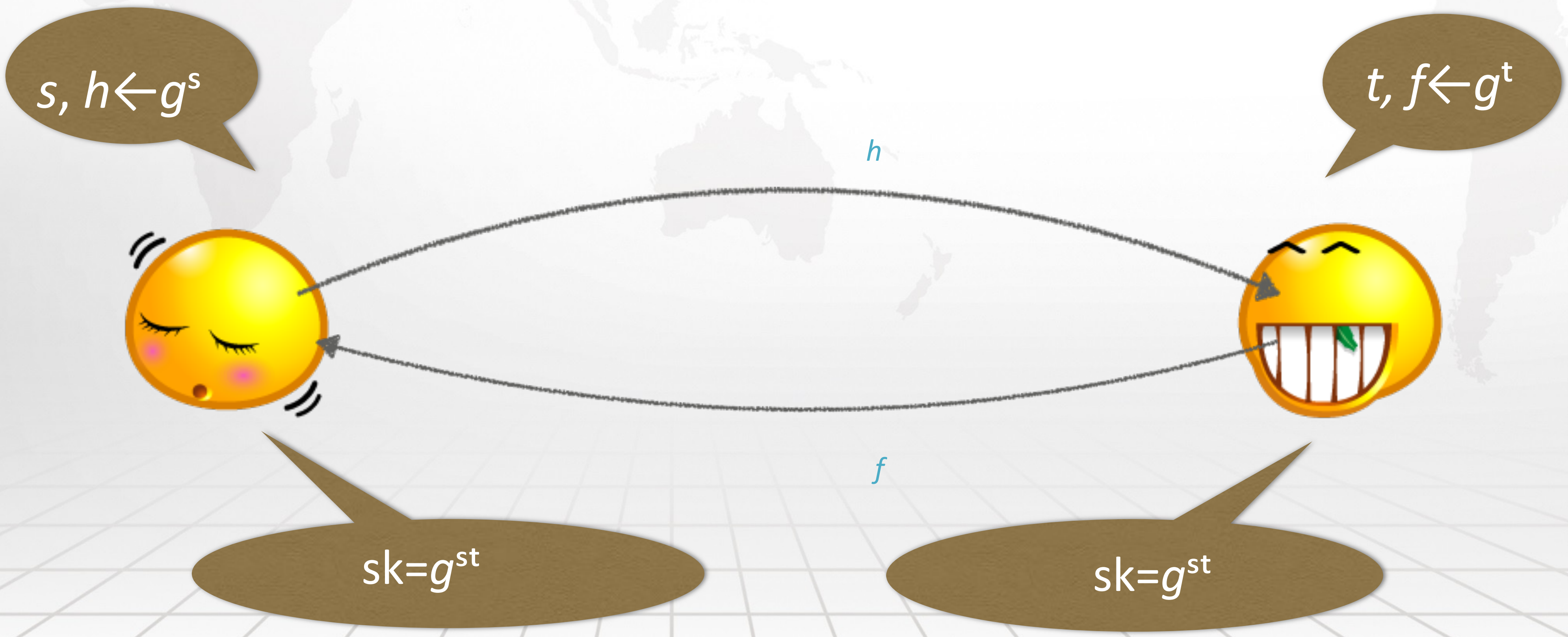
Key Exchange with dl



QUIZ

- $SK(t, g^s) = sk = SK(s, g^t)$
- What could SK be?
- **Hint:** we are working in a group
  - Use commutativity + efficient operations
- **Answer:** $SK(s, h) = h^s$
  - $SK(t, g^s) = g^{st} = g^{ts} = SK(s, g^t)$

Diffie-Hellman Key Exchange



DHKE: Formally

DHKE.Setup (κ):

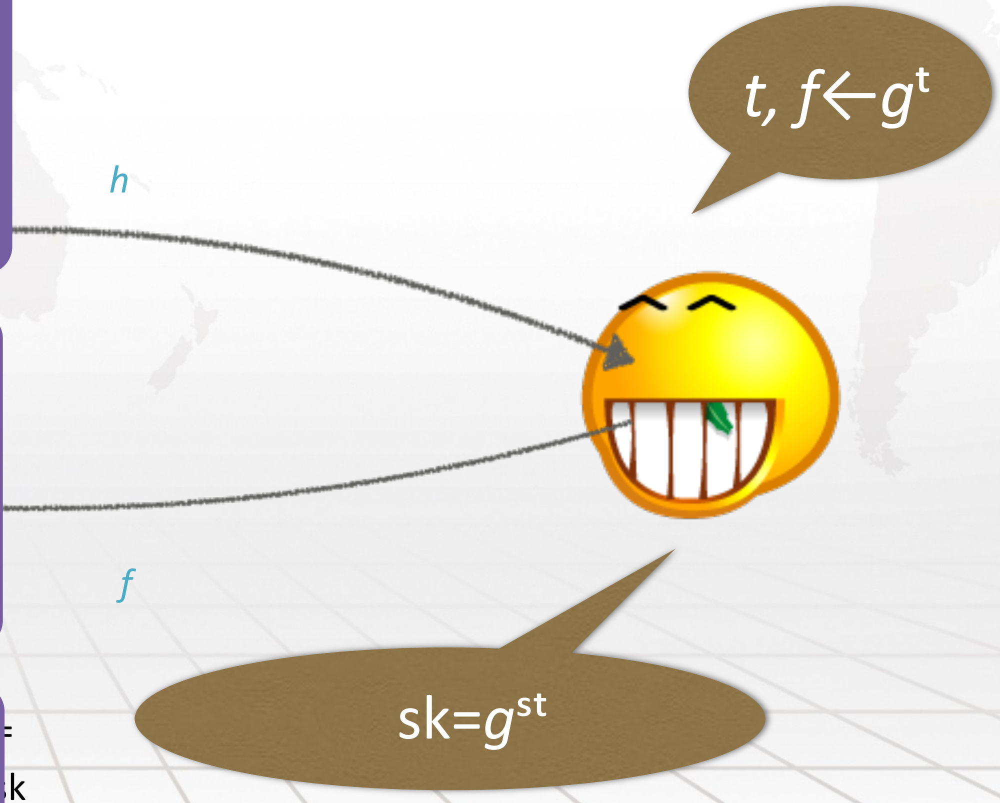
1. Choose a group G of order q where breaking DL has complexity 2^κ
2. Choose a generator g of G
3. Return gk ← desc (G) = (..., q, g)

DHKE.Keygen (gk):

1. sk = $s \leftarrow \mathbb{Z}q$
2. pk ← g^s
3. Return (sk, pk)

DHKE.SK (gk, s, h):

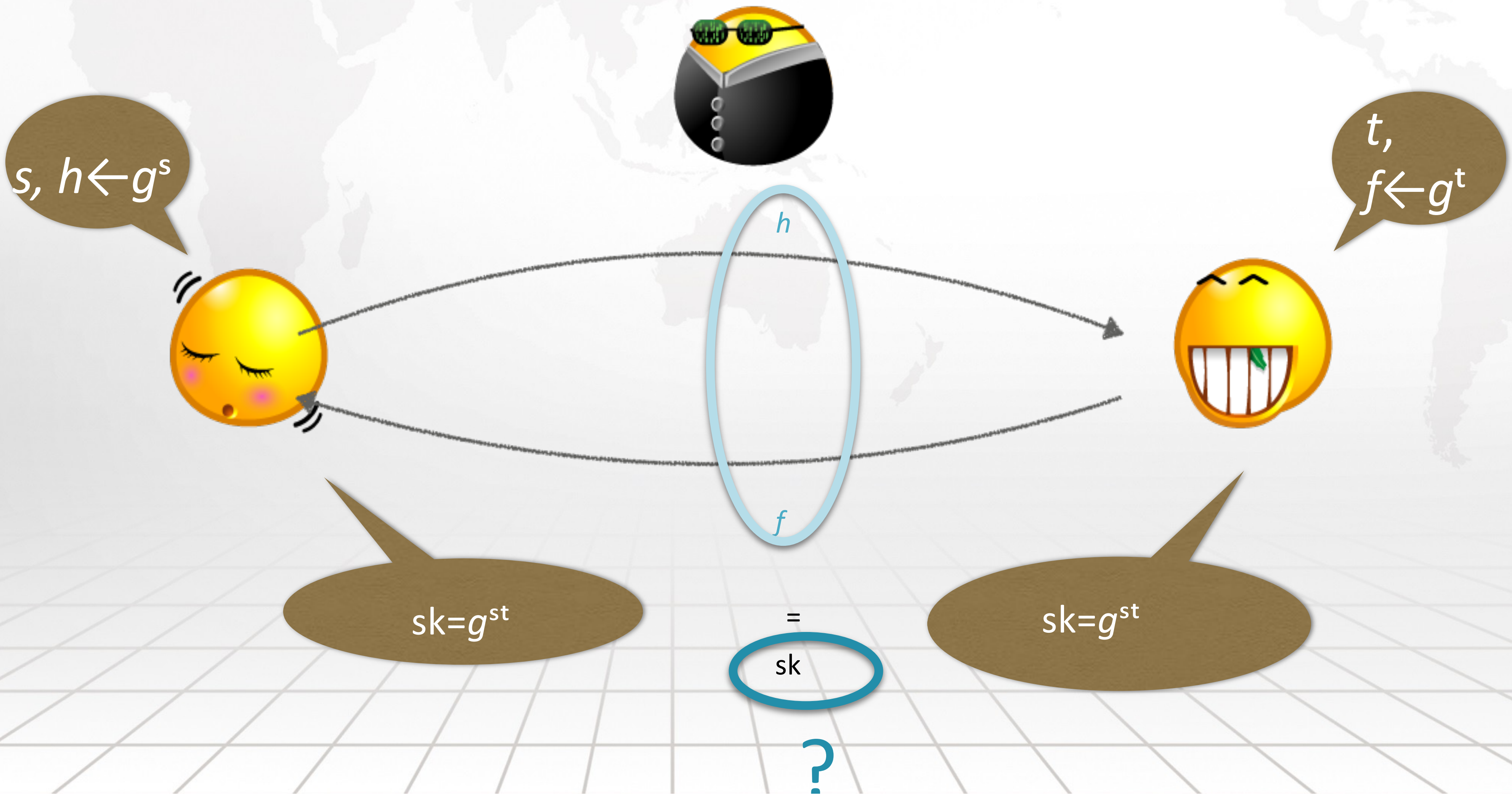
1. Return h^s



QUIZ: is dhke secure?

- **Correct question:**
  - is DHKE what-secure under X assumption
- Three tasks:
- Formalize security of KE
- Decide on X
- Provide a proof by reduction (X holds => DHKE what-secure)

DHKE: intuitive security



key recovery security

- Three algorithms KE = (Setup, Keygen, SK)
- Adv[KR] := | Pr[KR = 1] - 1 / q |
- $\mathcal{A}$ **ε-breaks KR (key recovery) security of** KE iff Adv[KR] $\geq \varepsilon$
- KE is **(τ,ε)-KR secure** iff no adversary ε-breaks KR security of KE in time $\leq \tau$
- KE is **KR secure** iff it is (poly(κ),negl(κ))-KR secure

Game KR(κ,KE,$\mathcal{A}$)

gk ← Setup(κ)
(sk_a,pk_a)←Keygen (gk)
(sk_b,pk_b)←Keygen (gk)
sk* ← $\mathcal{A}$ (gk, pk_a, pk_b)
If sk* = SK (gk, sk_a, pk_b)
  return 1
else
  return 0

Email: bingsheng@zju.edu.cn