up to a negligible fraction of inputs

Now, a digression... appealing property of discrete log problem: either it is <u>hard everywhere</u> or <u>hard nowhere</u>.

- Suppose we have an efficient algorithm A

$$\Pr[x \xleftarrow{R} \mathbb{Z}_p : A(g, g^x) \to x] = \varepsilon \quad (\text{for non-negligible } \varepsilon)$$

- We can use A to build B that solves <u>any</u> discrete log instance arbitrarily close to 1:
  - On input $(g, h = g^x)$, B samples $y \xleftarrow{R} \mathbb{Z}_p$ and runs A on (g, h^y)
  - Since y is uniform, g^y is a uniform group element so

$$\Pr[A(g, h^y) \to xy] = \varepsilon$$

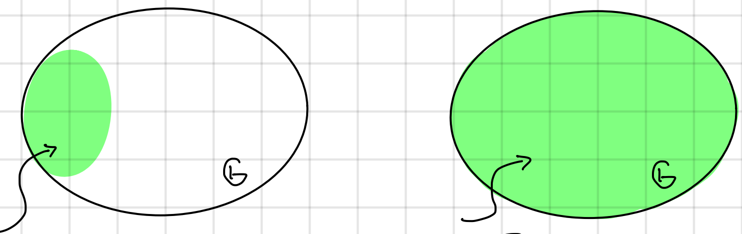
    If A succeeds (e.g., outputs $t = xy$ where $h^y = g^t$, then A outputs $x = y^{-1}t$
  - A can repeat this process n/ε times so the success probability becomes

$$1 - (1-\varepsilon)^{n/\varepsilon} \leq 1 - e^{-n} \qquad [\text{since } 1 + x \leq e^x \text{ for all } x]$$

"random self-reduction:" reduce problem to <u>random</u> instance of itself

- <u>Conclusion</u>: discrete log either easy everywhere or hard everywhere
  - ↳ easy on non-negligible fraction ⟹ easy everywhere

<u>Visually</u>:



Algorithm A works on an ε-fraction of $\mathbb{G}$

Algorithm B work <u>everywhere</u> in $\mathbb{G}$

In cryptography, we need problems that are hard in the <u>average</u> case (nearly all keys are "good")
- ↳ differs from worst-case hardness (e.g., NP-hardness – many NP-complete problems believed to be hard in worst case, but good algorithms exist for "typical" instances — <u>not</u> a good basis for crypto)
- ↳ when a problem has a random self-reduction, then worst-case hardness effectively implies average-case hardness; cannot have setting where problem is easy on ε-fraction of instances for any non-negligible ε)
  - ↳ appealing property for crypto!

Back to Diffie-Hellman key exchange...
- Everybody will use a fixed group $\mathbb{G}$ with generator g (e.g., P-256 or Curve25519) and prime order p

<u>Alice</u>		<u>Bob</u>
$x \xleftarrow{R} \mathbb{Z}_p$		$y \xleftarrow{R} \mathbb{Z}_p$
	$\xrightarrow{g^x}$	
	$\xleftarrow{g^y}$	

<u>shared key</u>: g^{xy} ← under the DDH assumption, (g, g^x, g^y, g^{xy}) looks indistinguishable from (g, g^x, g^y, g^r) for random r
- ↳ can be used as a key if the key is a <u>random</u> group element...

But usually, we want a random bit-string as the key, not random group element

↳ Element g^{xy} has $\log p$ bits of entropy, so should be able to obtain a random bit-string with $\ell < \log p$ bits

↳ Solution is to use a "randomness extractor"

  ↳ Information-theoretic constructions based on universal hashing / pairwise-independent hashing (loses some bits of entropy)

  ↳ Use a "random oracle" or an "ideal hash function" [Heuristic : SHA-256 (g, g^x, g^y, g^{xy})] (very efficient in practice)

  good practice to hash all components

  [binds the key to the entire transcript]

    ↳ Arguing security : 1. Rely on HashDH assumption $(g, g^x, g^y, H(g, g^x, g^y, g^{xy})) \overset{c}{\approx} (g, g^x, g^y, r)$ where $H : \mathbb{G}^4 \to \{0,1\}^n$ and $r \xleftarrow{R} \{0,1\}^n$

    2. Model H as ideal hash function $H : \mathbb{G}^4 \to \{0,1\}^n$ (i.e., random oracle) and rely on CDH in $\mathbb{G}$ [inability to evaluate H on g^{xy} $\Rightarrow$ output is random string]

Public-key encryption : Encryption scheme where encryption is public (does not require shared secrets)

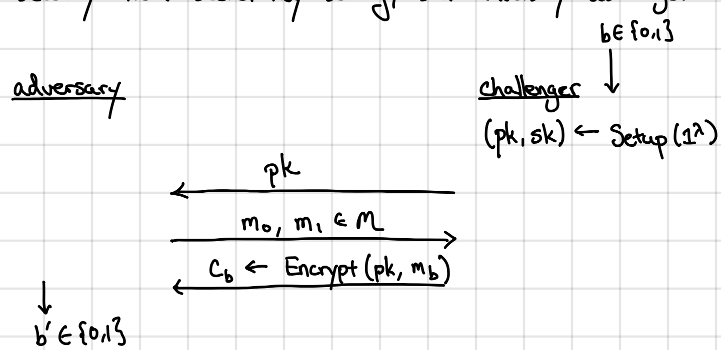
- $\mathrm{Setup}(1^\lambda) \to (pk, sk)$ [generates a public/private key-pair — also called KeyGen]
- $\mathrm{Encrypt}(pk, m) \to c$
- $\mathrm{Decrypt}(sk, c) \to m$

Everyone can publish a public key (in a directory)

↳ Can encrypt to anyone without exchanging keys (recipient can be offline)

Correctness : $\forall m \in \mathcal{M} : \Pr[(pk, sk) \leftarrow \mathrm{Setup}(1^\lambda) : \mathrm{Decrypt}(sk, \mathrm{Encrypt}(pk, m)) = m] = 1$

Security : semantic security from secret-key setting, but adversary also gets public key

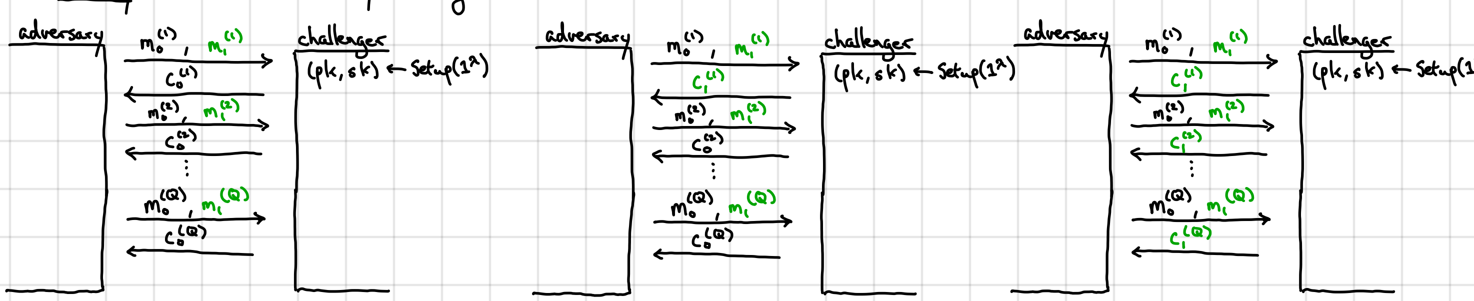


$$\mathrm{SSAdv}[\mathcal{A}, \Pi_{PKE}] = \left|\Pr[\mathcal{A} \text{ outputs } 1 \mid b = 0] - \Pr[\mathcal{A} \text{ outputs } 1 \mid b = 1]\right|$$

In the secret-key setting, we distinguished between semantic security and CPA-security. Here, this is unnecessary since semantic security $\Rightarrow$ CPA security [means that public-key encryption must be randomized!]

↳ Intuitively : adversary can encrypt messages on its own (using the public key)

Formally : Follows from a "hybrid" argument



$b = 0$ [always encrypt m_0]   Intermediate $b = 1$ [always encrypt m_1]

Total of $Q-1$ intermediate distributions

↳ i^{th} distribution and $(i+1)^{st}$ distribution identical except on $(m_0^{(i)}, m_1^{(i)})$, challenger encrypts $m_0^{(i)}$ in distribution i and $m_1^{(i)}$ in distribution $i+1$

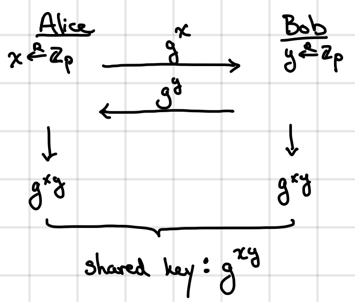
↳ these two distributions are indistinguishable by semantic security [in the reduction, the encryptions of the other messages (index $\neq i$) can be constructed using the public key (and do not depend on the challenger's choice bit)]

↳ if an adversary can distinguish endpoints $(b=0, b=1)$, then it must be able to distinguish a pair of intermediate distributions [by triangle inequality]

$\therefore$ semantic security $\Rightarrow$ every pair of distributions is computationally indistinguishable
$\Rightarrow$ CPA-security

PKE from DDH (ElGamal): Let $\mathbb{G}$ be a group with generator g and prime order p

Recall Diffie-Hellman key exchange:



Idea: Alice will publish $h = g^x$ as her public key

Bob encrypts by choosing fresh share g^y and uses g^{xy} to encrypt the message

security parameter dictates what group is used (eg, P-256, P-384, P-512)

Setup(1^λ): $x \xleftarrow{R} \mathbb{Z}_p$ pk: h $\mathcal{M} = \mathbb{G}$
$h \leftarrow g^x$ sk: x $\mathcal{C} = \mathbb{G}^2$

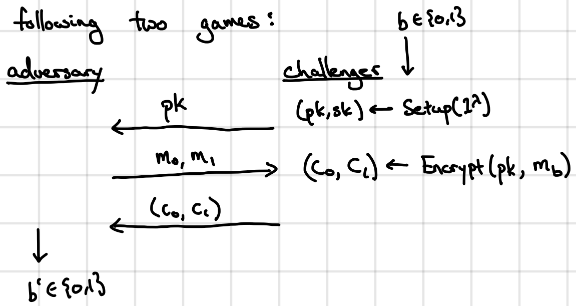
Encrypt$(\overset{=h}{pk}, m)$: $y \xleftarrow{R} \mathbb{Z}_p$
$c \leftarrow (g^y, m \cdot h^y)$

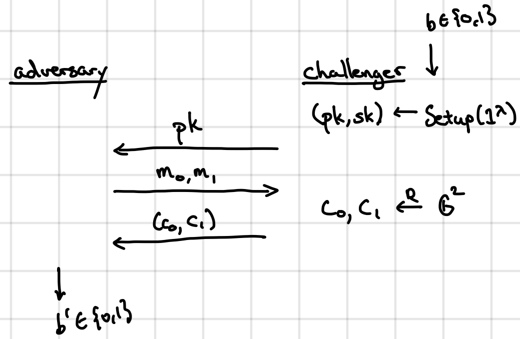
Decrypt$(\overset{=x}{sk}, c)$: $m \leftarrow c_1 / c_0^x$

Correctness:
$$\frac{c_1}{c_0^x} = \frac{m \cdot h^y}{(g^y)^x} = \frac{m \cdot (g^x)^y}{(g^y)^x} = \frac{m \cdot g^{xy}}{g^{xy}} = m$$

Security: If DDH holds in $\mathbb{G}$, then ElGamal is semantically secure.

Proof. Consider following two games:

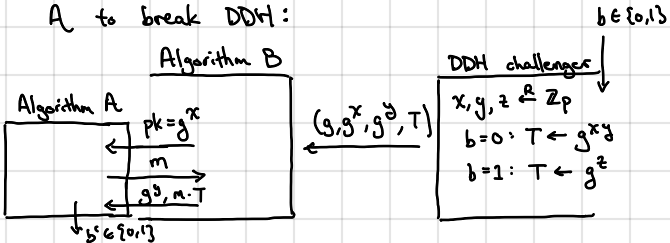




adversary's advantage in guessing b is 0 here since (c_0, c_1) is independent of (m_0, m_1)!

Claim: these two games are indistinguishable under DDH

Proof. Suppose there exists efficient $\mathcal{A}$ that can distinguish $(c_0, c_1) \leftarrow$ Encrypt(pk, m) from $(c_0, c_1) \xleftarrow{R} \mathbb{G}^2$. We use $\mathcal{A}$ to break DDH:



Observe: x is uniform over $\mathbb{Z}_p$ so g^x is a properly-generated public key (for ElGamal)

if $T = g^{xy}$, then $(g^y, T \cdot m) = (g^y, g^{xy} \cdot m)$ which is the output of Encrypt(pk, m) with randomness y — this is exactly the distribution where A sees Encrypt(pk, m)

if $T = g^z$, then $(g^y, g^z \cdot m)$ is uniform over $\mathbb{G}^2$ (since y, z are sampled independently of each other and of m) — this is exactly the distribution where A sees $(c_0, c_1) \xleftarrow{R} \mathbb{G}^2$

distinguishing advantage of B = distinguishing advantage of A

Equivalent view: Under DDH, g^{xy} looks uniform even given g, g^x, g^y, so an ElGamal ciphertext looks indistinguishable (to an efficient adversary) from a OTP encryption

What if we want to encrypt longer messages? [or messages that is not a group element]

- Hybrid encryption (key encapsulation [KEM]):
  - Use PKE scheme to encrypt a secret key
  - Encrypt payload using secret key + authenticated encryption

called key encapsulation

PKE. Encrypt(pk, k) "header" [slow]

AE. Encrypt(k, m) "payload" [fast]

secret-key operations much much faster than public-key operations!

- How to derive key from group element?

  Same as in key-exchange: hash the group element to a bit-string (symmetric key)

  e.g., Hash-ElGamal: Encrypt(pk, m): $y \xleftarrow{R} \mathbb{Z}_p$

  $c = (g^y, m \oplus H(g, h, g^y, h^y))$

  as before, can also rely on CDH + ideal hash function (random oracle)

  $H : \mathbb{G}^4 \to \{0,1\}^n$

Vanilla ElGamal described above is not CCA-secure!

Ciphertexts are malleable: given $ct = (g^y, h^y \cdot m)$, can construct ciphertext $(g^y, h^y \cdot m \cdot g)$ which decrypts to message $m \cdot g$

↳ directly implies a CCA attack

Several approaches to get CCA security from DH assumptions:

- Cramer-Shoup (CCA-security from DDH) - based on hash-proof systems
- Fujisaki-Okamoto transformation (using an ideal hash function + CDH)
- Make stronger assumption ("interactive" CDH + use ideal hash function):

We do not know of any groups where CDH believed to be hard, but interactive CDH is easy.

"CDH is hard even given access to a DDH oracle"

also called strong DH assumption

- Setup(1^λ): $x \xleftarrow{R} \mathbb{Z}_p$ pk : h

  $h \leftarrow g^x$ sk : x

- Encrypt(pk, m): $y \xleftarrow{R} \mathbb{Z}_p$ $k \leftarrow H(g^y, h^y)$ $ct' \leftarrow \mathrm{Enc}_{AE}(k, m)$

  symmetric authenticated encryption scheme

  $c \leftarrow (g^y, ct')$

- Decrypt(sk, c): $k \leftarrow H(c_0, c_0^x)$

  $m \leftarrow \mathrm{Dec}_{AE}(k, c_1)$

Essentially ElGamal where key derived from hash function