Digital Signatures.

Secret key (Symmetric key)

A Symmetric key Encryption scheme is defined by a message space M and the following algorithms

Gen → Probabilistic algorithm that outputs a key k according to some distribution K.

Enc → takes input key k and message m, outputs a ciphertext c

$$c \leftarrow Enc_k(m)$$

Dec → Takes input key k and ciphertext c, outputs a message m

$$m \leftarrow Dec_k(c)$$

Public Key Enc

$(pk, sk) \leftarrow Gen$

$c \leftarrow Enc_{pk}(m)$

$m \leftarrow Dec_{sk}(c)$

Alice

k

Bob

k

Alice

pk_a, sk_a

Bob

pk_b, sk_b

$c \leftarrow Enc_{pk_b}(m)$ —c→

$m \leftarrow Dec_{sk_b}(c)$

$m' \leftarrow Dec_{sk_a}(c')$ ←c'— $c' \leftarrow Enc_{sk}(m')$

[Perfect Secrecy]

For every $m, m' \in M$, $c \in C$ (for any msgs m and ciphertexts c)

$$Pr[Enc_K(m) = c] = Pr[Enc_K(m') = c]$$

↑ the probability (over the choice of k) that c is an encryption of m is equivalent to the probability that c is an encryption of m'

Perfect secrecy can only be achieved when the key is at least as long as the message.

Chosen Plaintext Attack

- key k is generated by Gen
- A can utilize the encryption oracle $Enc_k(\cdot)$ to obtain for any message $m \in M$
- An attacker A chooses two messages $m_0, m_1 \in M$ to be encrypted.
- A random bit $b \in \{0,1\}$ is chosen uniformly, and computes $c \leftarrow Enc_k(m_b)$. c is sent to A.
- A outputs a guess, b', as to which message was encrypted.

CPA

Adversary A

$K \xleftarrow{\$} K$

$m_i \in M$

$c_i \leftarrow Enc_K(m_i)$

c_i

randomized

$(m_0, m_1) \neq m_i$

$b \xleftarrow{\$} \{0,1\}$

$c \leftarrow Enc_K(m_b)$

c

$\hat{b}$

Adv wins if $\hat{b} = b$

CPA-Security

No efficient (PPT) adversary A succeeds at the CPA-game with much greater than $\frac{1}{2}$

ie no efficient attacker can do better than just guessing.

Notice that since the attacker can get encryptions of any messages, a CPA-secure encryption scheme must be randomized

there must be multiple encryptions of a message m under key k.

Chosen Ciphertext Attack (CCA)

- key k is generated by Gen
- A outputs any two messages m_0, m_1 from $\mathcal{M}$ with access to encryption oracle $Enc_k(\cdot)$ and decryption oracles $Dec_k(\cdot)$
- A random bit $b \in \{0,1\}$ is chosen uniformly at random, and $c \leftarrow Enc_k(m_b)$ is given to A.
- A can request queries to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, but CANNOT request $Dec_k(c)$.
- A outputs a bit $\hat{b} \in \{0,1\}$

decryption schemes are always deterministic.

$n \in \mathbb{N}$ $c \in C$

CCA

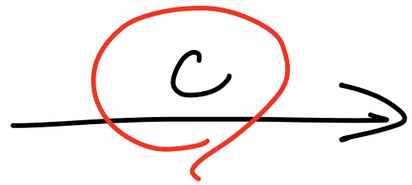
$k \xleftarrow{\$} K$

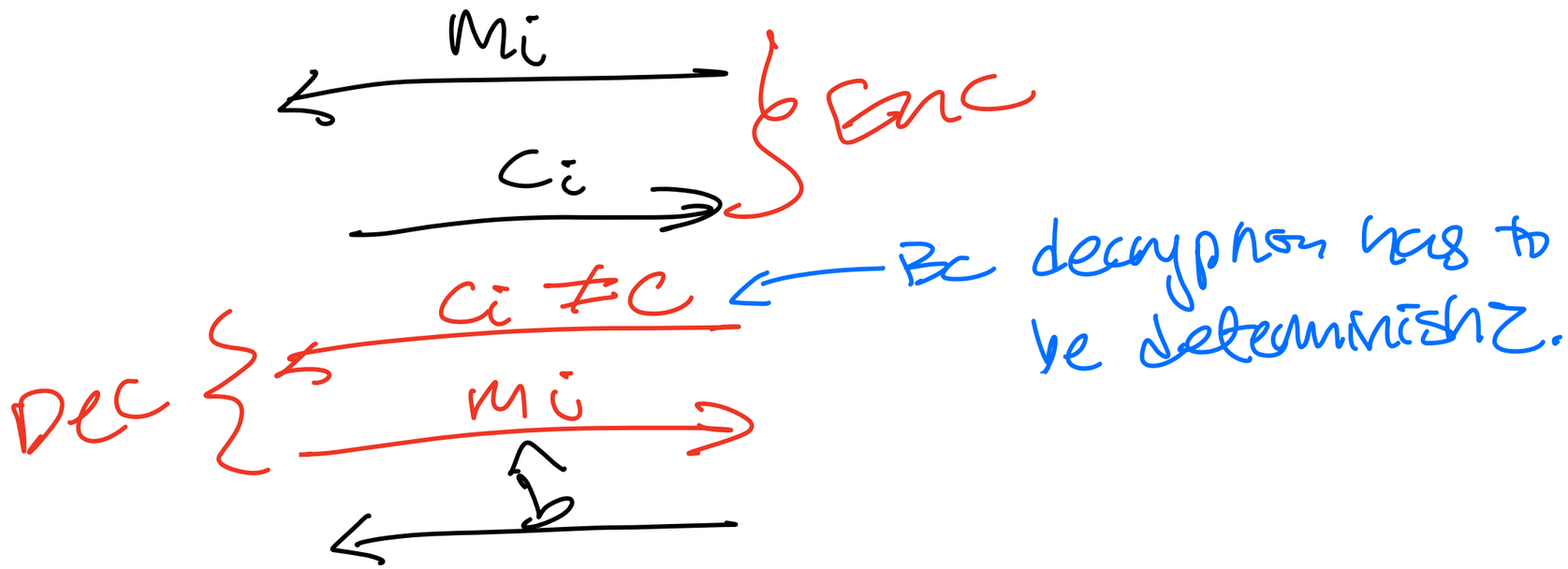
Adversary A

(m_0, m_1)

$b \xleftarrow{\$} \{0,1\}$

$c \leftarrow Enc_k(m_b)$





CCA Security

No efficient (PPT) attacker $\mathcal{A}$ succeeds at the CCA-game with greater probability than $\frac{1}{2}$.

└ can't do better than just guessing.

that was confidentiality.

PRFs

$$x \leftarrow \underbrace{F_k(m)}$$

deterministic.

Authenticity — Message Authentication Codes (MACs)

symmetric!

MACs are defined by the following algorithms:

1) Gen → probabilistic key gen algorithm.

2) Mac — takes input key k and message m, outputs tag t

$t \leftarrow Mac_k(m)$

3) Verify — takes input key k, tag t, and message m, outputs 1 if t is a valid tag, 0 otherwise.

$Mac_k(m) = t$

$t = t'$

$Verify_k(m, t) = 0/1$

↑valid

Correctness: $Verify_k(m, Mac_k(m)) = 1$

Secure MAC

- A gets access to a MAC oracle $MAC_k(\cdot)$. A can submit a query of any message m of its choice and get back a tag $t \leftarrow MAC_k(m)$

- A tries to create a message tag pair $(\hat{m}, \hat{t})$ s.t

$$Verify_k(\hat{m}, \hat{t}) = 1$$

The attacker breaks the scheme if

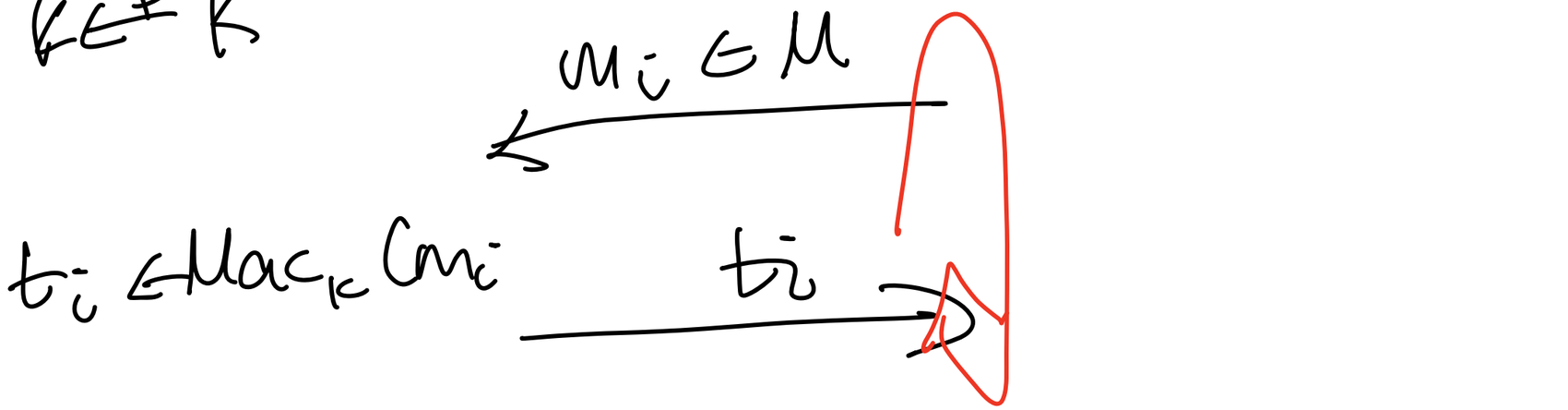
1) The pair $(\hat{m}, \hat{t})$ is valid. $Verify_k(m, t) = 1$

2) A has not previously requested a tag on the message m.

MAC

$k \xleftarrow{\$} K$

Adversary A



Strong MAC

- A gets access to a MAC oracle $MAC_k(\cdot)$. A can submit a query of any message m of its choice and get back a tag $t \leftarrow MAC_k(m)$

- A tries to create a message tag pair $(\hat{m}, \hat{t})$ s.t

$$Verify_k(m, t) = 1$$

The attacker breaks the scheme if

1) The pair (m, t) is valid. $Verify_k(m, t) = 1$
2) A has not previously ~~requested a tag on the~~ received the tag t for message m.

Digital Signatures vs MACs

ensures the integrity of the msgs.

Public key and secret key pair	1 key.
verified by anyone.	can only be verified by key holders.

Digital Signatures.

A digital signature scheme is defined by the following algorithms:

1) Gen — probabilistic algorithm that outputs a pair of keys (sk, pk)

2) Sign — takes as input a secret key sk and a message m and outputs a signature σ

$\sigma \leftarrow Sign_{sk}(m)$

3) Verify — takes as input a public key pk, signature σ, and message m, outputs 1 if signature is valid, 0 otherwise.

$Verify_{pk}(m, \sigma) = 0/1$

Public Key Enc

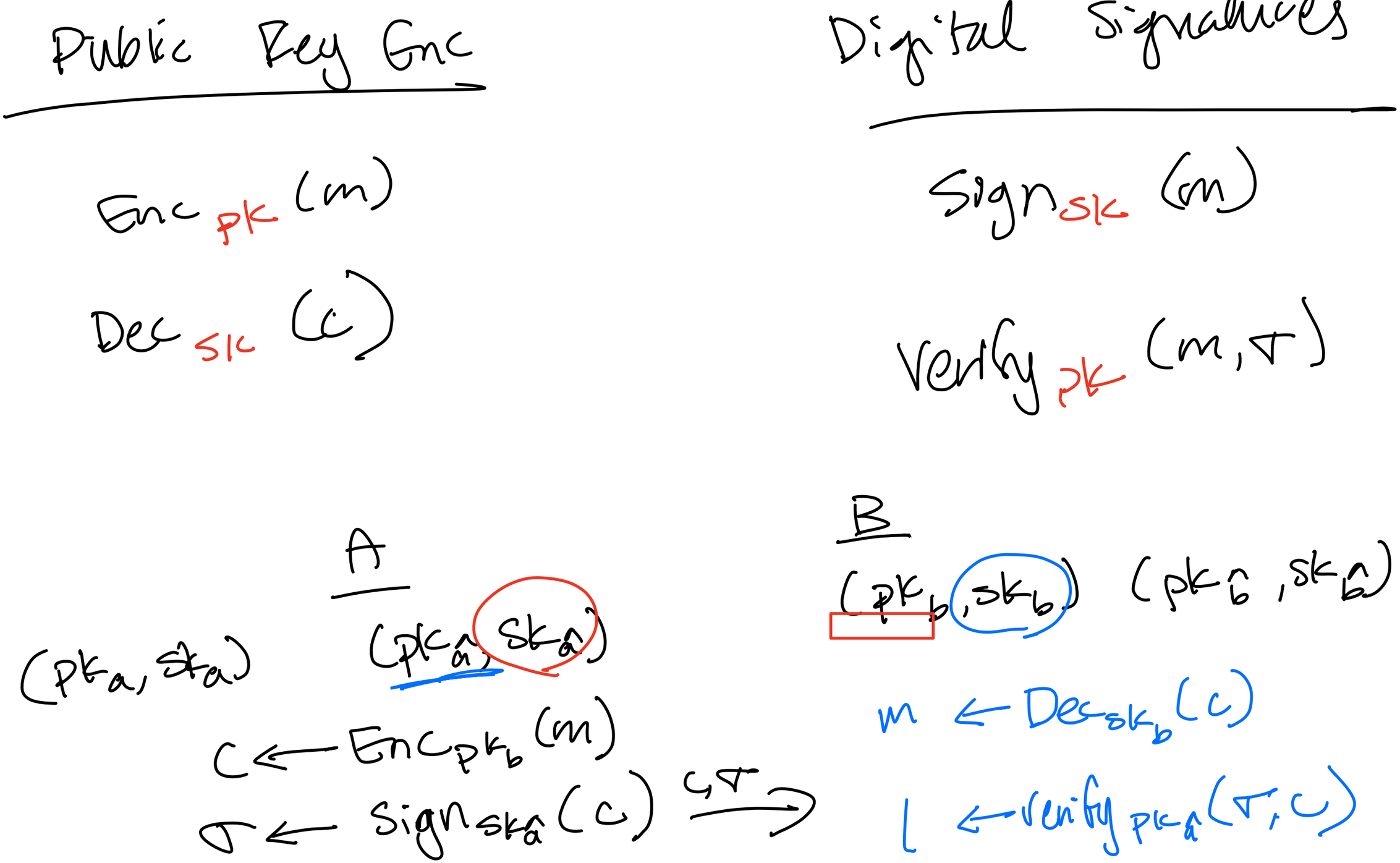
$Enc_{pk}(m)$

$Dec_{sk}(c)$

Digital signatures

$Sign_{sk}(m)$

$Verify_{pk}(m, \sigma)$



Digital Signatures

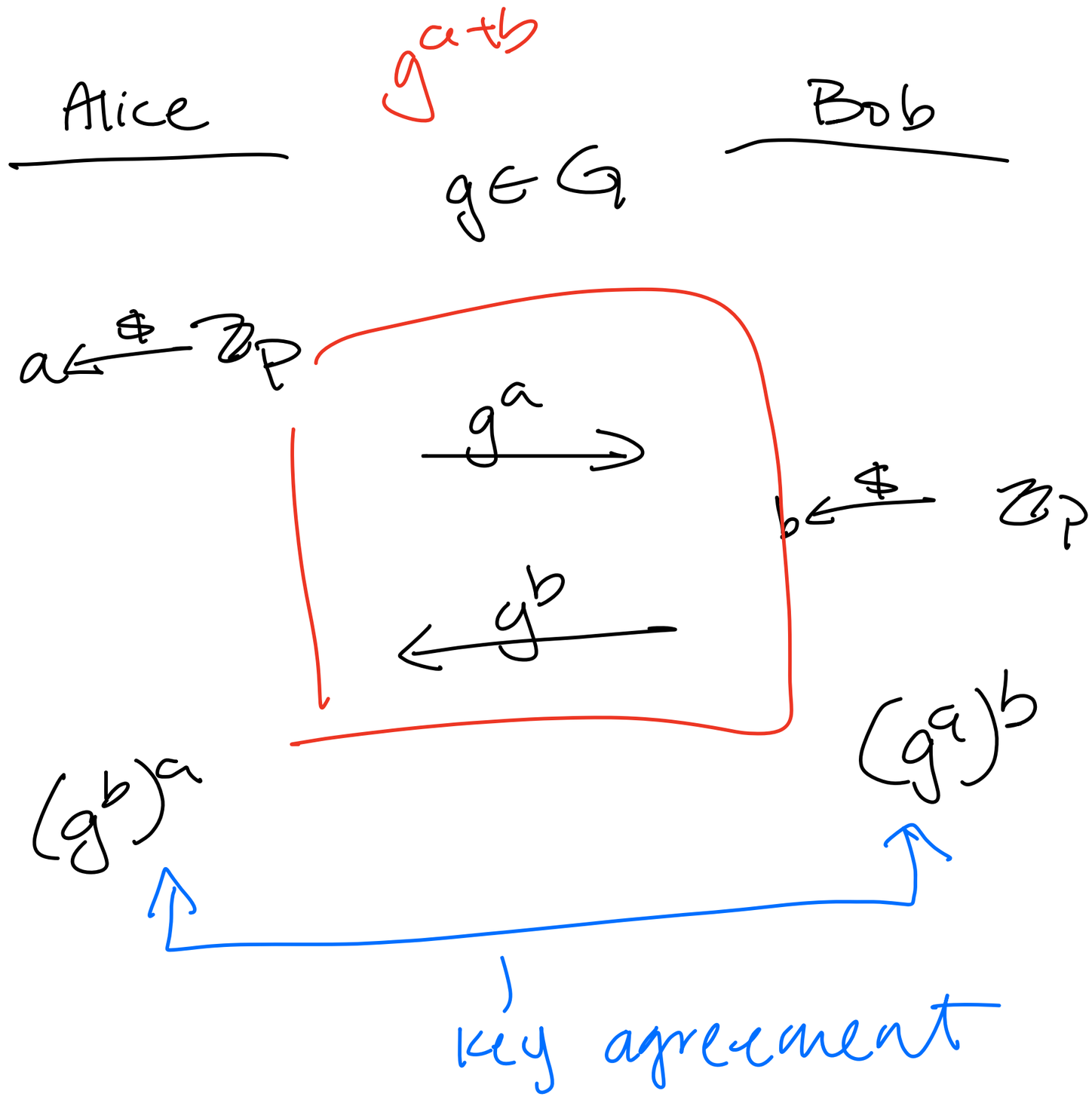
Adversary A

Key Exchange

↳ Secure if no efficient attacker can tell the real key from one chosen randomly, even with access to the transcript of messages exchanged between Alice and Bob.

Diffie Hellman Key Exchange.

- Let G be a DDH group, and g be a generator in G.
- Alice chooses $a \xleftarrow{\$} \mathbb{Z}_p$ and sends g^a to Bob.
- Bob chooses $b \xleftarrow{\$} \mathbb{Z}_p$ and sends g^b to Alice.
- Alice computes $(g^b)^a$ using exponent a.
- Bob computes $(g^a)^b$ using exponent b


Alice
Bob
g^{a+b}
$g \in G$
$a \xleftarrow{\$} \mathbb{Z}_p$
g^a
$b \xleftarrow{\$} \mathbb{Z}_p$
g^b
$(g^b)^a$
$(g^a)^b$
key agreement