Techniques in Cryptography and Cryptanalysis

Brice Minaud

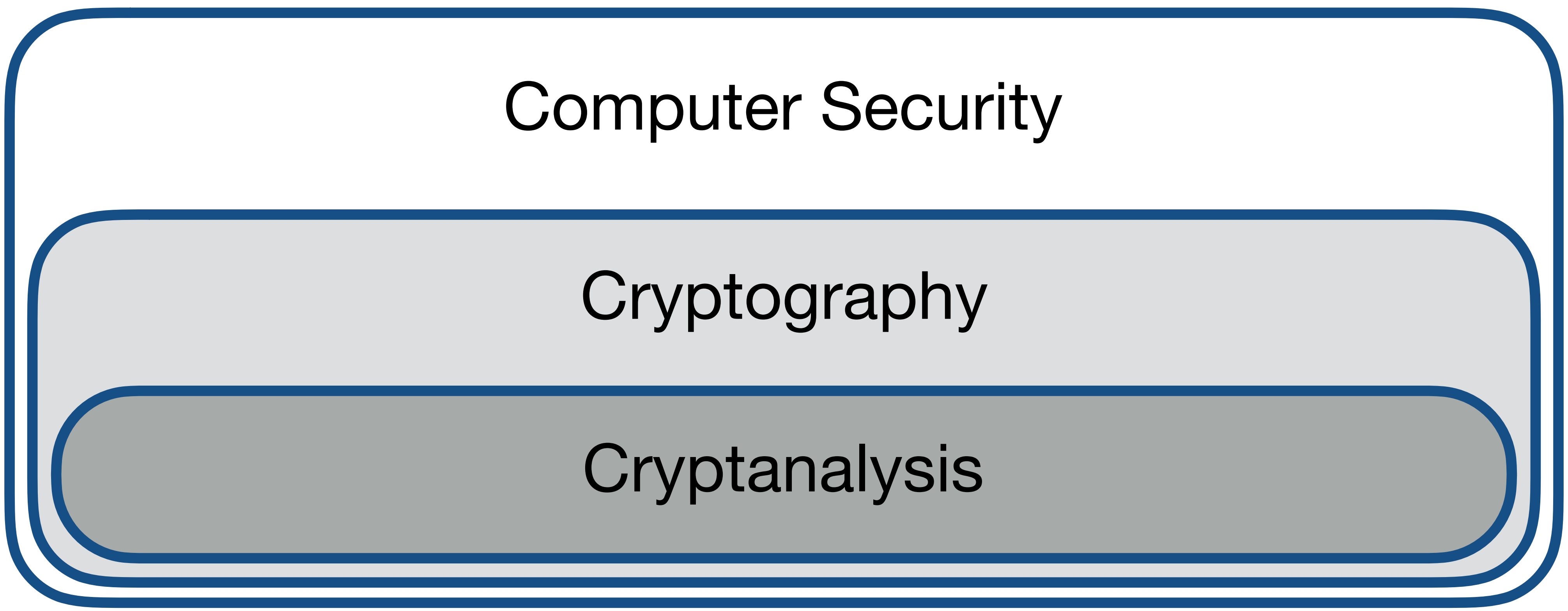
email: brice.minaud@inria.fr
website: www.di.ens.fr/brice.minaud/

MPRI, 2019

What is Cryptography?

Cryptography: design and analysis of secure communications.

Cryptanalysis: subfield that focuses on analyzing the security of those designs, including attacks.



Cryptography serves as a foundation of computer security.

What is *this course*?

"Techniques in Cryptography and Cryptanalysis": will cover (a choice of) important areas of cryptography.

Current plan:

Bases: public-key encryption, signatures, symmetric cryptography…

Some more advanced topics: lattices, zero-knowledge proofs, multi-party computation, identity-based encryption.

Teachers:

Brice Minaud

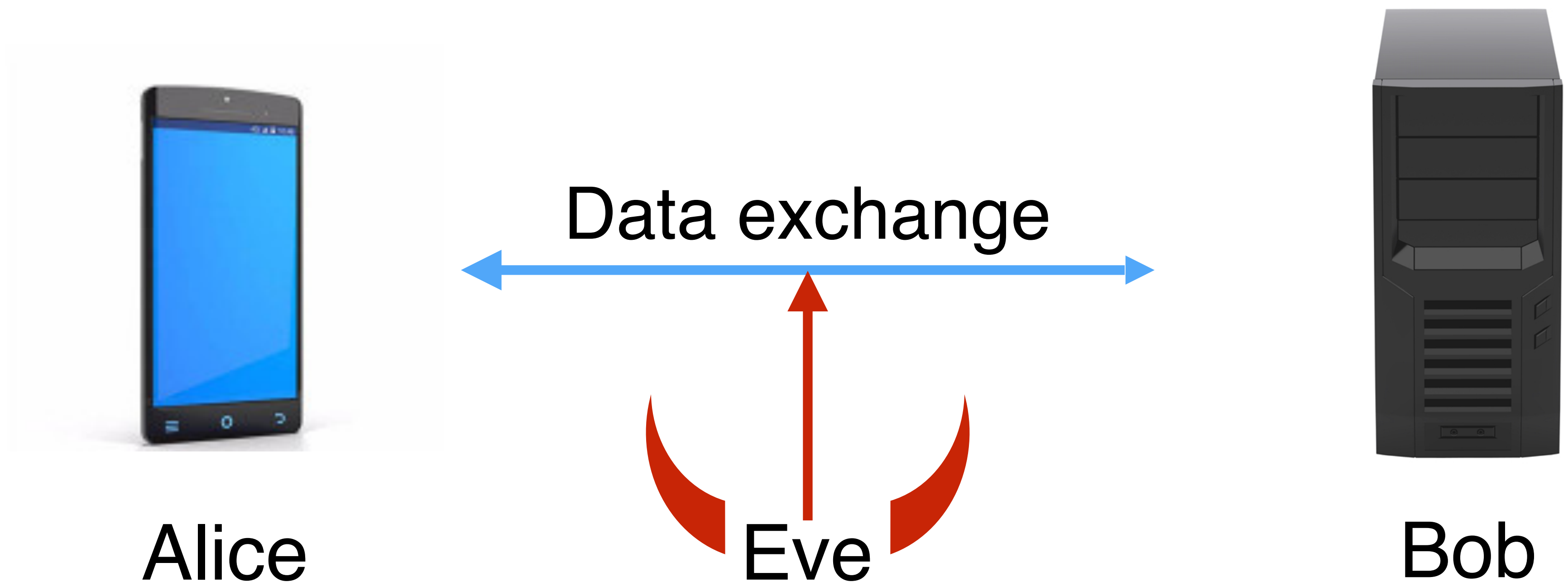
8 x 1.5h, 1st period

Michel Abdalla

8 x 1.5h, 2nd period

What is security?

Historically, most basic goal = protecting the confidentiality of data exchanges.

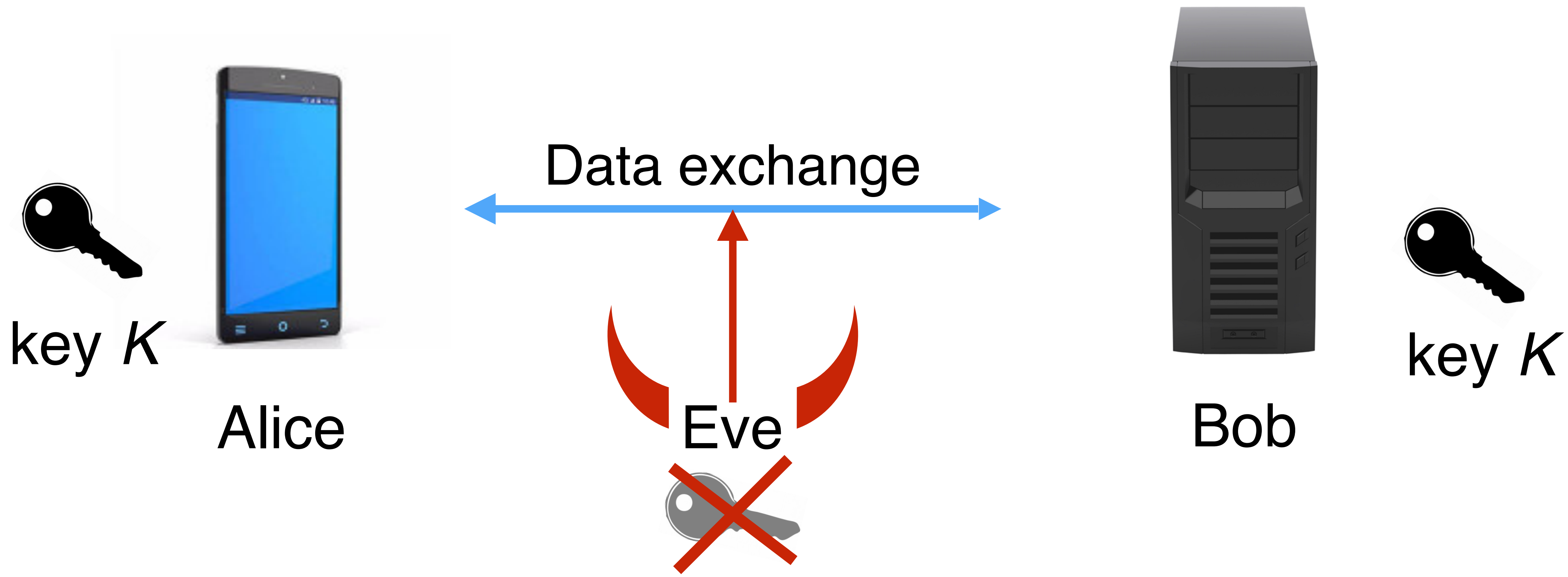


Kerckhoff's (first three) principles:

1. The system must be practically, if not mathematically, indecipherable.
2. It should not require secrecy, and it should not be a problem should it fall into enemy hands.
3. It must be possible to […] change or modify [the key] at will.

One-Time Pad

Modern version: the algorithms are public. They are parametrized by a (secret) key.



One-Time Pad.

Message space: $M \leftarrow \{0,1\}^n$ Key space: $K \leftarrow \{0,1\}^n$

Encryption(M): $C = M \oplus K$

Decryption(C): $M = C \oplus K$

Security of One-Time Pad

One-Time Pad.

Message space: $M \leftarrow \{0,1\}^n$ Key space: $K \leftarrow \{0,1\}^n$

Encryption(M): $C = M \oplus K$

Decryption(C): $M = C \oplus K$

Naive security: impossible for Eve to find M from C.

Not great. Encryption could leak last bit of M and still be secure by that definition.

We want to express that Eve learns *nothing* about M.

Perfect secrecy

Perfect secrecy, historical version, Shannon, 1949.

Prior distribution: distribution of M known a priori to Eve.

Posterior distribution: distribution of M known to Eve after seeing the encryption $\mathrm{Enc}_K(M)$ of M (for uniform K).

Perfect secrecy: posterior distribution = prior distribution.

Perfect secrecy, equivalent modern version, folklore, 20th century.

Let M_0 and M_1 be two arbitrary messages.

Perfect secrecy: $\mathrm{Enc}_K(M_0) = \mathrm{Enc}_K(M_1)$.

The equality is an equality of distributions. The randomness is over the uniform choice of K.

OTP and perfect secrecy

Proposition. The One-Time Pad achieves perfect secrecy.

Proof. $\mathrm{Enc}(M_0) = C$ iff $K = C \oplus M_0$.

So there is exactly one K that yields each possible C. Since K is uniform, so is C. Thus:

$$\mathrm{Enc}(M_0) = \mathrm{Unif}(\{0,1\}^n) = \mathrm{Enc}(M_1).$$

(Note: this would hold in any group.)

Theorem (Shannon '49). If perfect secrecy holds, it must be the case that the two parties share some prior information (a key) with:

$$\mathrm{length(key)} \geq \mathrm{length(message)}$$

where length denotes the bit length.

So OTP is essentially the only perfectly secure scheme.

Measuring Security

Advantage

- Previous solution is infeasible in most cases.

  → we must be content with *imperfect* security.
- The relevant notion to formally express that Eve cannot learn anything is often about the *indistinguishability* of two distributions.

  Roadmap of a security definition: the **adversary** is an algorithm attempting to infer secret information.

  Often, this will be expressed as the adversary trying to distinguish two distributions.

Advantage.

Let D_0 and D_1 be two probability distributions. The advantage of an adversary A (i.e. an algorithm, here with output in {0,1}) is:

$$\mathrm{Adv}^{D_0,D_1}(A) = |2\Pr_{b \leftarrow_\$ \{0,1\}}(A(D_b) = b) - 1|$$

Types of security

let M_0 and M_1 be two arbitrary messages…

Perfect security:

$\mathrm{Enc}_K(M_0) = \mathrm{Enc}_K(M_1)$ (as distributions, for uniform K).

Equivalently: $\mathrm{Adv}^{\mathrm{EncK}(M_0),\ \mathrm{EncK}(M_1)}(A) = 0$, for every A.

Statistical security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(A)$ is negligible, for every A.

Computational security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(A)$ is negligible, for every efficient adversary A.

Quantifying negligibility, efficiency

Security parameter, often denoted λ: used to quantify security.

- "Asymptotic" security: used in more theoretical results. λ remains a variable.
- "Concrete" security: used in more practical results. Typically λ = 80, 128, or 256. (e.g. "128-bit" security.)

	"Asymptotic" security	**"Concrete" security**
Negligible (probability)	$O(\lambda^{-c})$ for all c	usually $\leq 2^{-\lambda/2}$ or $2^{-\lambda}$
Efficient (adversary)	Poly(λ)	significantly less than 2^{λ} operations

Concreteness of security

Bits of security	Practical significance
32	Your phone can do it, instantly.
66	Bitcoin hashes *per second* worldwide.
80	Bitcoin hashes per year worldwide. *(Some state actors could do it?)*
128	Considered secure. Standard choice. *(Watch out for trade-offs, like time/data or multi-target)*
256	Arguments for impossibility based on physics. *(Relevant for very long-term or quantum security.)*

Bitcoin data from https://www.blockchain.com/en/charts/hash-rate in 2019.

Types of security, again

let M_0 and M_1 be two arbitrary messages…

Perfect security:

$\mathrm{Enc}_K(M_0) = \mathrm{Enc}_K(M_1)$ (as distributions, for uniform K).

Equivalently: $\mathrm{Adv}^{\mathrm{EncK}(M_0),\ \mathrm{EncK}(M_1)}(A) = 0$, for every A.

Statistical security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(A)$ is negligible, for every A.

Computational security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(A)$ is negligible, for every efficient adversary A.

Statistical distance

Good tool to bound or analyze advantage.

Statistical distance.

Let D_0 and D_1 be two probability distributions over some set X.

$$\mathrm{Dist}(D_0, D_1) = \frac{1}{2} \sum_{x \in X} |D_0(x) - D_1(x)|$$

Proposition 1. This is, in fact, a distance.

Proof. x, y ↦ |y - x| is a distance. So Dist(·,·) is a sum of distances. (Can also write it out.)

Statistical distance, cont'd

Proposition 2. The statistical distance Dist(D_0,D_1) is equal to the advantage of the best adversary trying to distinguish D_0 from D_1.

Proof. Let A be the adversary such that, given $x \leftarrow D_b$, A outputs 0 iff $D_0(x) \geq D_1(x)$. A is clearly best possible.

$$\begin{aligned}
\mathrm{Adv}^{D_0,D_1}(A) &= 2\Pr_{x \leftarrow D_b(x), b \leftarrow_\$ \{0,1\}}(A(x) = b) - 1 \\
&= 2\sum_{x'}\sum_{b'} \Pr(A(x) = b | x = x', b = b') \\
&\qquad\qquad \cdot \Pr_{x \leftarrow D_b}(x = x' | b = b') \Pr_{b \leftarrow_\$ \{0,1\}}(b = b') - 1 \\
&= \sum_{x'}\sum_{b'} \mathbb{1}_{A(x')=b'} D_b(x') - 1 \\
&= \sum_{x'} \max(D_0(x'), D_1(x')) - 1 \\
&= \mathrm{Dist}(D_0, D_1) \qquad \text{using: } \max(a,b) = \frac{1}{2}(a + b + |b - a|).
\end{aligned}$$

Statistical distance, cont'd

Corollary. Let A be any algorithm. Then:

$$\mathrm{Dist}(A(D_0),A(D_1)) \leq \mathrm{Dist}(D_0,D_1)$$

Proof. Let B be the best adversary distinguishing D_0 from D_1, and C be the best adversary distinguishing $A(D_0)$ from $A(D_1)$.

$$\mathrm{Dist}(A(D_0),A(D_1)) = \mathrm{Adv}^{A(D_0),A(D_1)}(C) = \mathrm{Adv}^{D_0,D_1}(C \circ A)$$

$$\leq \mathrm{Adv}^{D_0,D_1}(B) = \mathrm{Dist}(D_0,D_1).$$

Proposition 3. For all n, $\mathrm{Dist}(D_0^n,D_1^n) \leq n\mathrm{Dist}(D_0,D_1)$.

Proof.

$\mathrm{Dist}(A^n,B^n) \leq \mathrm{Dist}(A^n,A^{n-1}B) + \mathrm{Dist}(A^{n-1}B,A^{n-2}B^2) + \ldots + \mathrm{Dist}(AB^{n-1},B^n)$.

Sometimes called the “hybrid” argument, although the same term is also used in more general settings.

Computational version

Advantage of the best adversary = statistical distance.

By extension:

Advantage of a class of adversaries.

Let D_0 and D_1 be two probability distributions, and $\mathbf{A}$ a set of adversaries. Define:

$$\mathrm{Adv}^{D_0,D_1}(\mathbf{A}) = \sup\{\mathrm{Adv}^{D_0,D_1}(A) : A \in \mathbf{A}\}$$

Define $\mathbf{A}(t)$ the set of adversaries that terminate in time t. Let:

$$\mathrm{Adv}^{D_0,D_1}(t) = \mathrm{Adv}^{D_0,D_1}(\mathbf{A}(t))$$

This is still a distance! (exercise)

NB For asymptotic security, what matters usually is to distinguish two *families* of distributions. We want (abuse of notation):

$$\mathrm{Adv}^{D_0,D_1}(\mathrm{Poly}(\lambda)) = \mathrm{Negl}(\lambda)$$

with D_0, D_1 (implicitly) parametrized by λ.

Types of security, revisited

let M_0 and M_1 be two arbitrary messages…

Perfect security:

$\mathrm{Enc}_K(M_0) = \mathrm{Enc}_K(M_1)$ (as distributions, for uniform K).

Equivalently: $\mathrm{Dist}(\mathrm{Enc}_K(M_1), \mathrm{Enc}_K(M_2)) = 0$.

Equivalently: $\mathrm{Adv}^{\mathrm{EncK}(M_0),\ \mathrm{EncK}(M_1)}(\{\text{all } A\}) = 0$.

Statistical security:

$\mathrm{Dist}(\mathrm{Enc}_K(M_1), \mathrm{Enc}_K(M_2))$ is negligible.

Equivalently: $\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(\{\text{all } A\})$ is negligible.

Computational security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_0),\ \mathrm{Enc}_K(M_1)}(\{\text{efficient } A\})$ is negligible.

A simple example

Consider a Bernoulli (coinflip) distribution B with $B(0) = 1/2 - \varepsilon$ and $B(1) = 1/2 + \varepsilon$. Let U be the uniform distribution on $\{0,1\}$. Observe:

$$\mathrm{Dist}(B,U) = \varepsilon.$$

Assume we are doing a one-time pad with an imperfect randomness source, where the key bits are drawn according to B:

$$K \leftarrow B^n \text{ (instead of } U^n\text{)}$$

Say ε is negligible (asymptotic sense).

Is this still secure?

Perfect security? Statistical? Computational?

A simple example, cont'd

Let's encrypt a message $M \in \{0,1\}^n$.

$$\mathrm{Dist}(\mathrm{Enc}_K(M),U^n) = \mathrm{Dist}(K \oplus M, U^n)$$

$$\leq \Sigma_{i<n} \mathrm{Dist}(\,(K \oplus M)_i\,,U)$$

$$= n\varepsilon$$

i-th bit of $K \oplus M$

For M_0, $M_1 \in \{0,1\}^n$.

$$\mathrm{Dist}(\mathrm{Enc}_K(M_0),\mathrm{Enc}_K(M_1)) \leq \mathrm{Dist}(\mathrm{Enc}_K(M_0),U^n) + \mathrm{Dist}(\mathrm{Enc}_K(M_1),U^n)$$

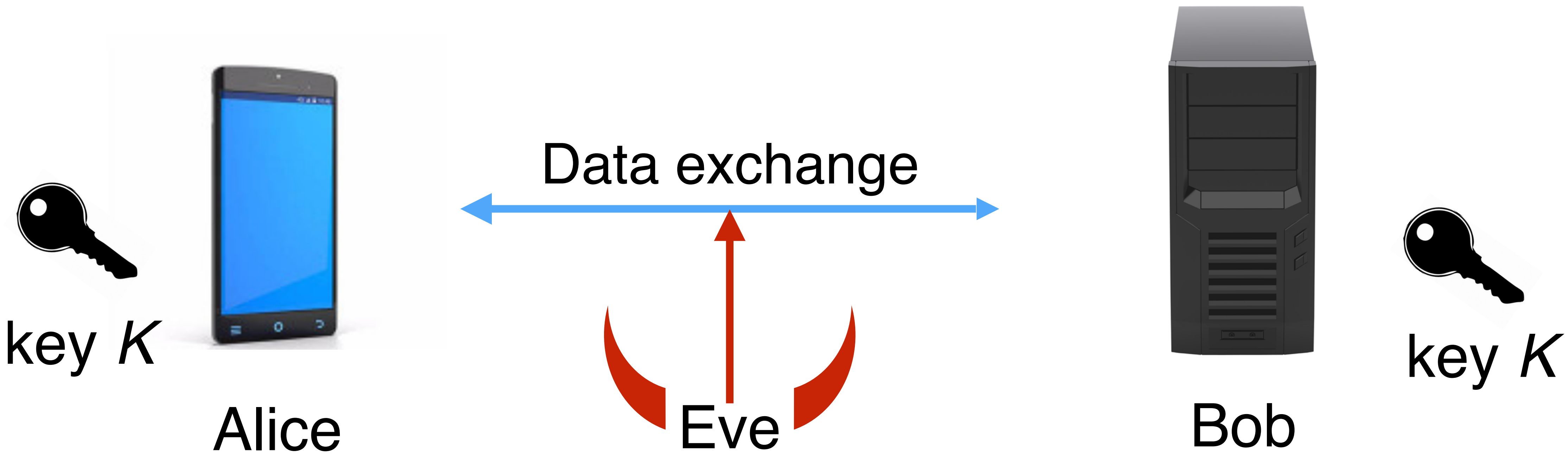
$$\leq 2n\varepsilon$$

Note that $n \cdot \mathrm{Negl}(n) = \mathrm{Negl}(n)$ so this is (statistically) secure!

(A more refined analysis shows this grows in $\sqrt{n\epsilon}$. The hybrid argument is a little crude here.)

Defining Security

Symmetric encryption: definition



Symmetric Encryption.

Message space **M**, ciphertext space **C**, key space **K**.

Setup: Pick key $K \leftarrow_{\$} \mathbf{K}$.

Encryption: encryption of $M \in \mathbf{M}$ is $C = \mathrm{Enc}_K(M) \in \mathbf{C}$.

Decryption: decryption of C is $M = \mathrm{Dec}_K(C)$.

These three algorithms/protocols are assumed to be efficient.

Symmetric encryption: confidentiality

Symmetric Encryption.

Message space **M**, ciphertext space **C**, key space **K**.

Setup: Pick key $K \leftarrow_{\$} \mathbf{K}$.

Encryption: encryption of $M \in \mathbf{M}$ is $C = \mathrm{Enc}_K(M) \in \mathbf{C}$.

Decryption: decryption of C is $M = \mathrm{Dec}_K(C)$.

Cryptographic definitions usually require two properties:

Correctness: scheme fulfills desired functionality.

Security: scheme is secure. (Usually the hard one.)

Here, for symmetric encryption:

Correctness: for all $M \in$ M,

$$\mathrm{Dec}_K(\mathrm{Enc}_K(M)) = M.$$

Summary of the previous episodes

Proposition. The One-Time Pad achieves perfect secrecy.

…as long as a fresh key of the same length of the message is used for each message. Impractical. We have seen Shannon's theorem: this is essentially the only perfectly secret scheme.

Hence we are content with:

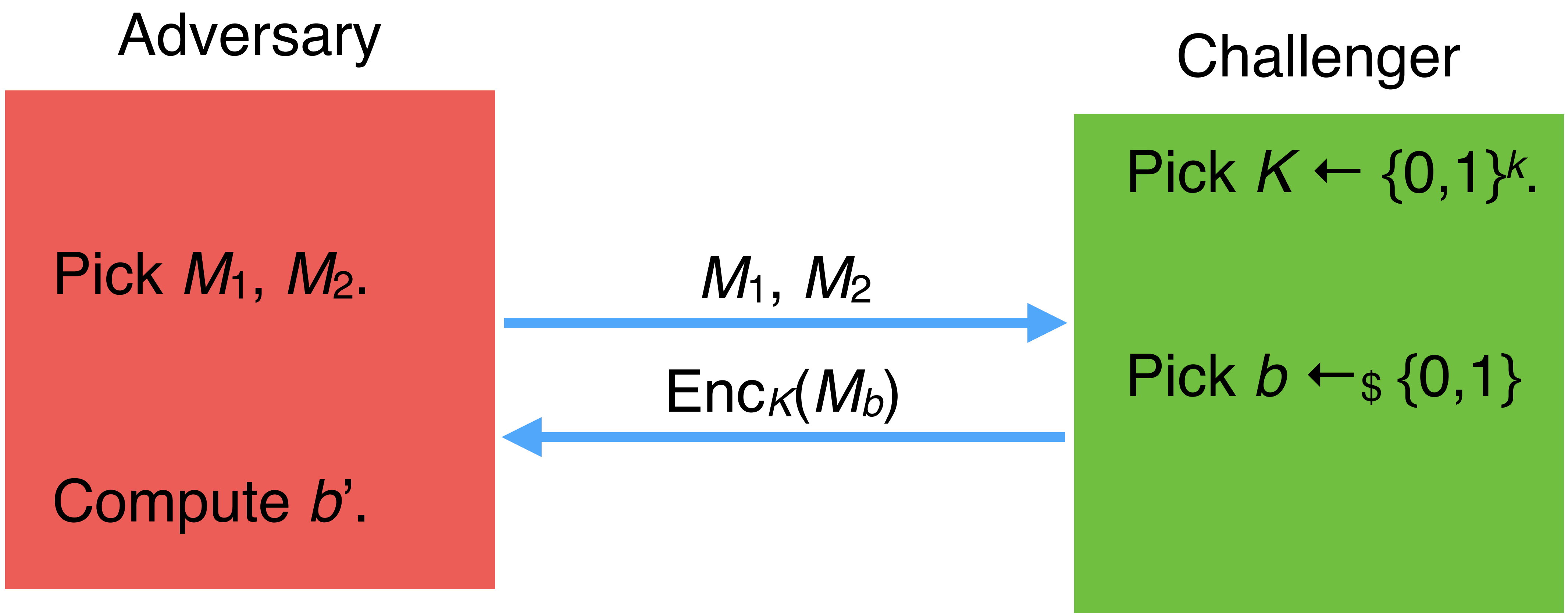
Computational security:

$\mathrm{Adv}^{\mathrm{Enc}_K(M_1),\, \mathrm{Enc}_K(M_2)}(A)$ is negligible, for every efficient adversary A.

Is this enough?

No: we want security even if adversary knows encryption of **known**, or **chosen** plaintexts.

IND: indistinguishability game

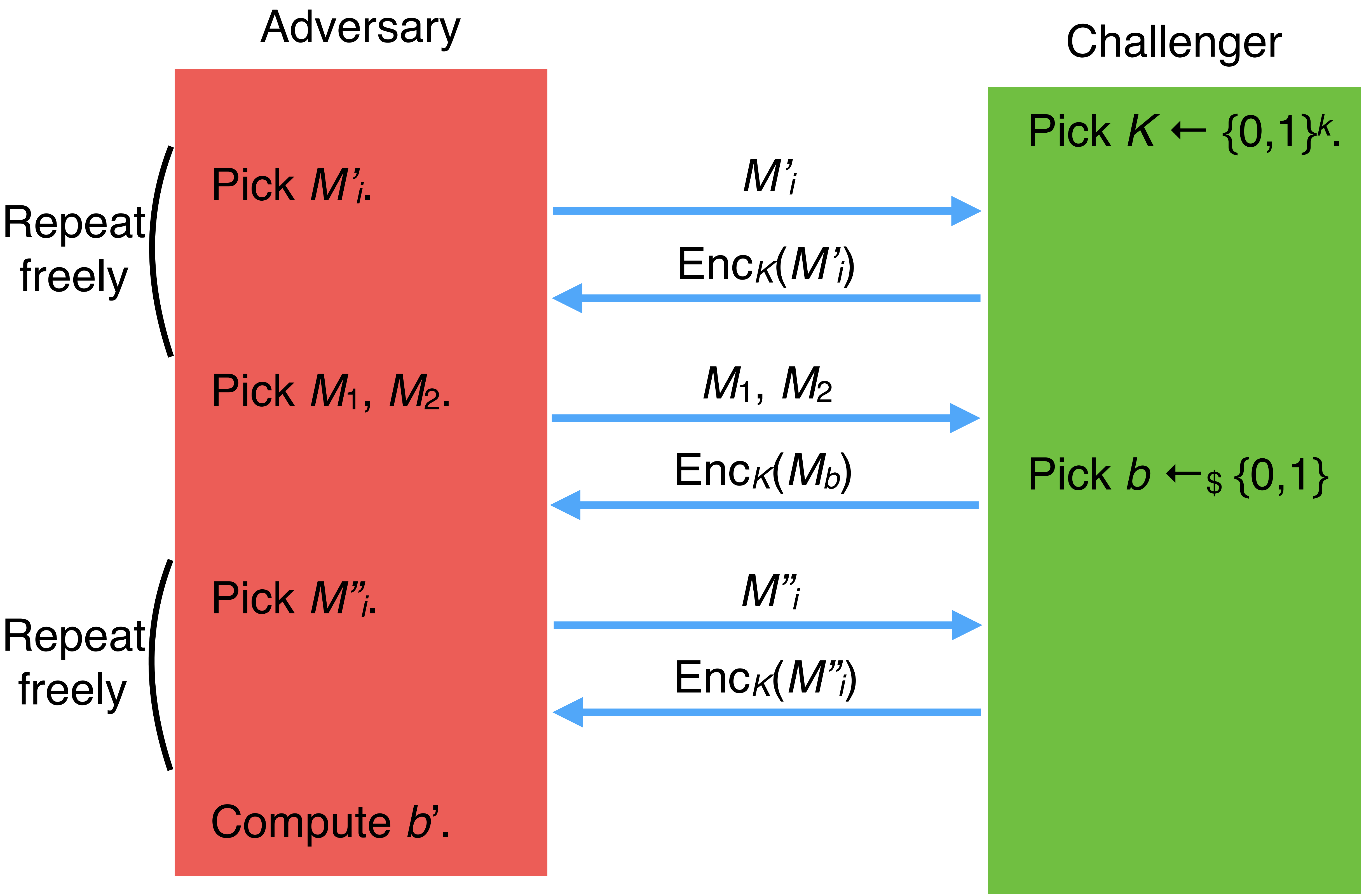


Adversary wins iff $b' = b$.

Computational security: the advantage of an efficient adversary in this game is negligible.

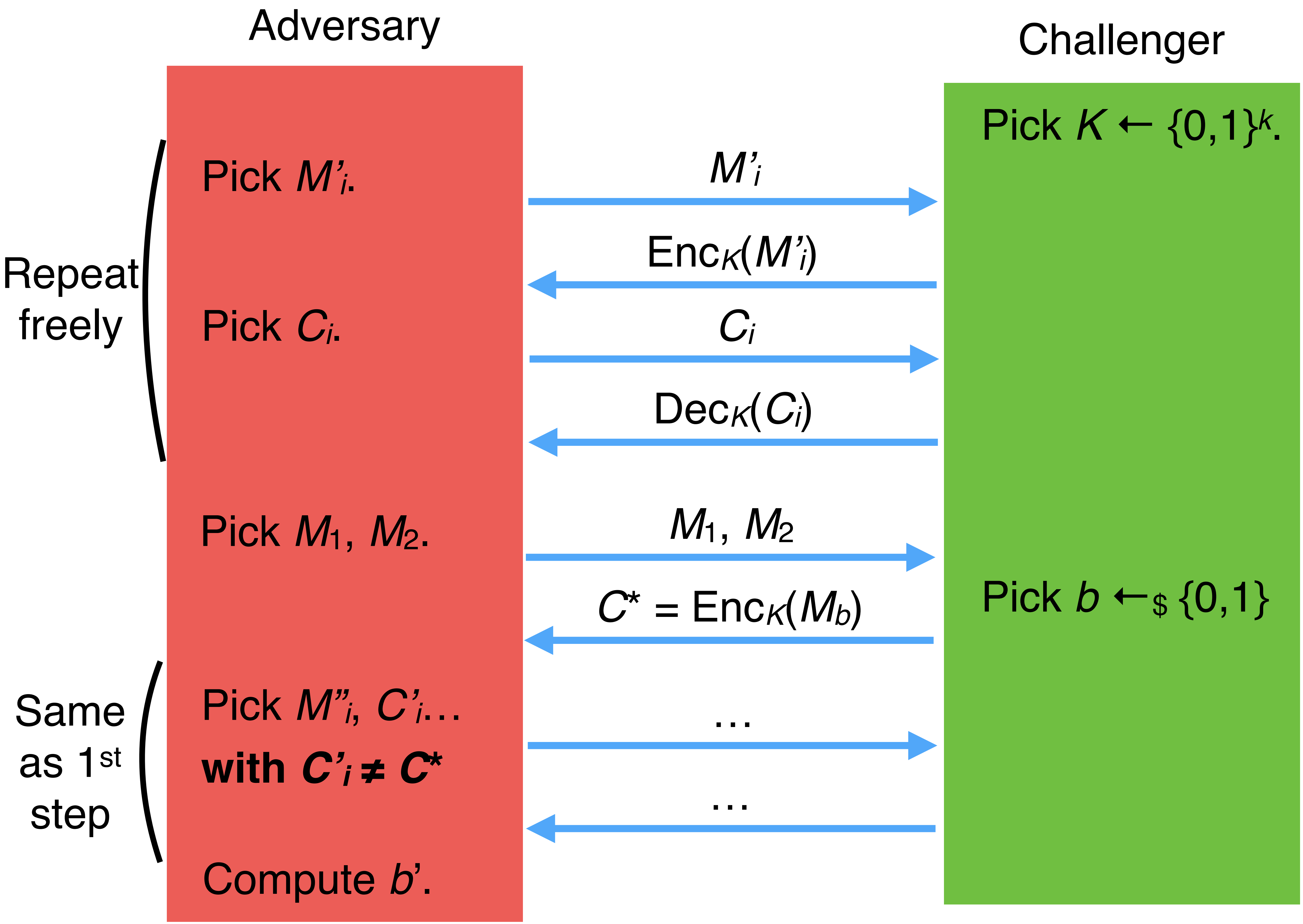
(This is just an equivalent statement of what was before, using games.)

IND-CPA: indist. under Chosen-Plaintext Attacks



Adversary wins iff $b' = b$.

IND-**CCA**: indist. under Chosen-**Ciphertext** Attacks



Symmetric encryption: complete definition

Symmetric Encryption.

Message space **M**, ciphertext space **C**, key space **K**.

Setup: Pick key $K \leftarrow_{\$}$ **K**.

Encryption: encryption of $M \in$ **M** is $C = \mathrm{Enc}_K(M) \in$ **C**.

Decryption: decryption of C is $M = \mathrm{Dec}_K(C)$.

Correctness: for all $M \in$ M,

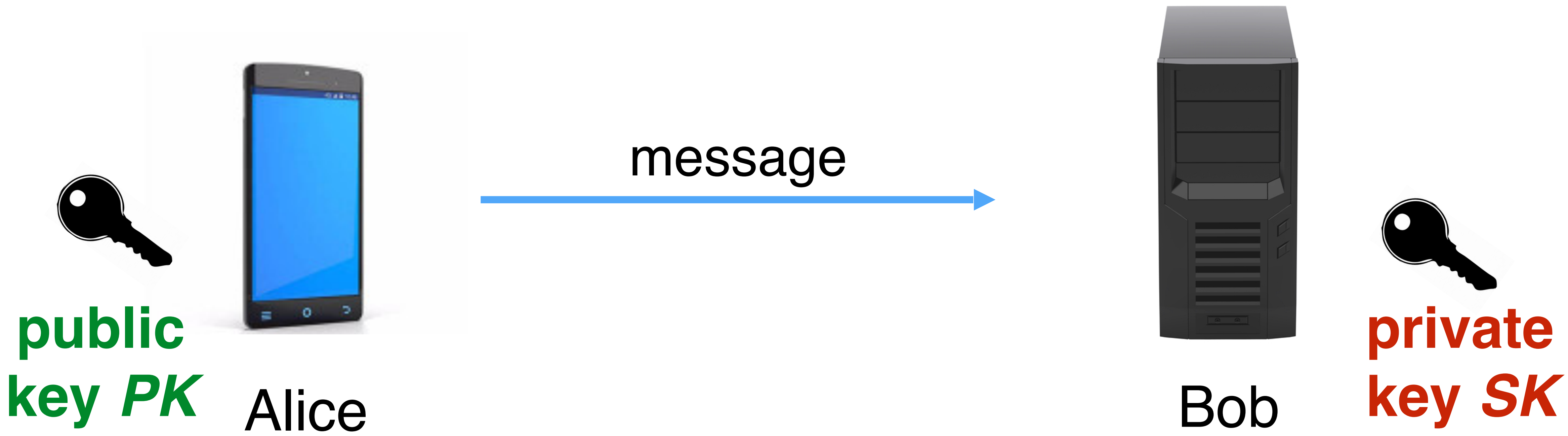
$$\mathrm{Dec}_K(\mathrm{Enc}_K(M)) = M.$$

Security: typically IND-CPA, or IND-CCA…

Caveats:

- Deterministic scheme cannot be IND-CPA or IND-CCA. Need randomness, or nonces.
- "Security" above only covers **confidentiality**, not **integrity**.

Publick-key encryption: definition



Publick-Key Encryption.

Message space **M**, ciphertext space **C**,

and secret (a.k.a. private) key space **SK**, public key space **PK**.

Setup: output private/public key pair (SK,PK) from (**SK**, **PK**).

Encryption: encryption of $M \in \mathbf{M}$ is $C = \mathrm{Enc}_{PK}(M) \in \mathbf{C}$.

Decryption: decryption of C is $M = \mathrm{Dec}_{SK}(C)$.

Public-key encryption

Publick-Key Encryption.

Message space **M**, ciphertext space **C**,

and secret (a.k.a. private) key space **SK**, public key space **PK**.

Setup: output private/public key pair (SK, PK) from (**SK**, **PK**).

Encryption: encryption of $M \in \mathbf{M}$ is $C = \mathrm{Enc}_{PK}(M) \in \mathbf{C}$.

Decryption: decryption of C is $M = \mathrm{Dec}_{SK}(C)$.

You **encrypt** with the **public key**, **decrypt** with the **private key**.

Correctness: for all $M \in$ M, if (SK, PK) is output by **Setup**, then

$$\mathrm{Dec}_{SK}(\mathrm{Enc}_{PK}(M)) = M.$$

Security: typically IND-CPA, or IND-CCA… (note: IND = IND-CPA.)

So how do you prove security?

Short answer: we cannot. We don't even know $P \neq NP$.

Illustration: say that a PKE scheme exists iff the **Setup**, **Enc**, **Dec** algorithms are polynomial, and the scheme is correct and IND-CPA secure (against polynomial adversaries).

Question: *does a public-key encryption scheme exist?*

Answer: *we don't know.*

Workaround: rely on problems that are *assumed* intractable for polynomial-time adversaries, like integer factorization.

General paradigm: **hard problem** + **trapdoor**.

The trapdoor is typically the private key. Without it, decryption is hard; with it, it is easy. (e.g. easy = polynomial-time.)

Security reduction

To prove the security of a construction, we **reduce** it to the hardness of a standard problem.

That means a security proof proves something of the form :

> **If** there exists an efficient adversary ***A*** achieving a non-negligible advantage against the cryptographic scheme,
>
> **Then** there exists an efficient adversary ***B*** achieving a non-negligible advantage against the hard problem.

→ if the hard problem *is* in fact hard, the scheme is secure.

Typically, the proof builds ***B*** from ***A***.

RSA

Rivest, Shamir, Adleman ’77.

- Select a pair of random primes p, q. Set $N = pq$.
- Select integers d, e such that $de = 1 \bmod (p-1)(q-1)$.
  - The public key is $pk = (e,N)$.
  - The secret key is $sk = d$.

Encryption: for a message $m \in \mathbb{Z}^*_N$, the ciphertext is:

$$c = m^e \bmod N.$$

Decryption: for a ciphertext c, the message is:

$$m = c^d \bmod N.$$

You can think of $e = 3$.

Hard problem: computing third root modulo N.

Trapdoor: knowledge of prime decomposition $N = p \cdot q$.

RSA: basic facts

Caveats:

- This was "textbook" RSA. It **is not** IND-CPA or IND-CCA. Why?
- Basic RSA is **malleable**. It is **multiplicatively homomorphic**:

$$\mathrm{Enc}(a)\mathrm{Enc}(b) = a^e b^e = (ab)^e = \mathrm{Enc}(ab)$$

- If $e = 3$ and $m < N^{1/3}$, $\mathrm{Enc}(m) = m^3$ over the integers!
- many other issues…

This is because RSA is not actually a PKE scheme. It is a trapdoor permutation.

In order to use it as PKE, it must be combined with a mode of operation such as OAEP. (Often implicit when people say "RSA".)

Hardness of RSA

If you can factorize $N = pq$, you recover the secret key.

The converse is not true: the security of RSA does not reduce to integer factorization (well-known hard problem).

Security of RSA.

Given $N = pq$, e, and $x^e \bmod N$ for $x \leftarrow \mathbb{Z}^*_N$, find $x \bmod N$.

Essentially ad-hoc, but the best known attack is integer factorization. Much better than brute-force (sub-exponential).

See course by Morain/Blanchet to learn (much) more!

Hardness of integer factorization

Check out https://www.keylength.com/en/3/

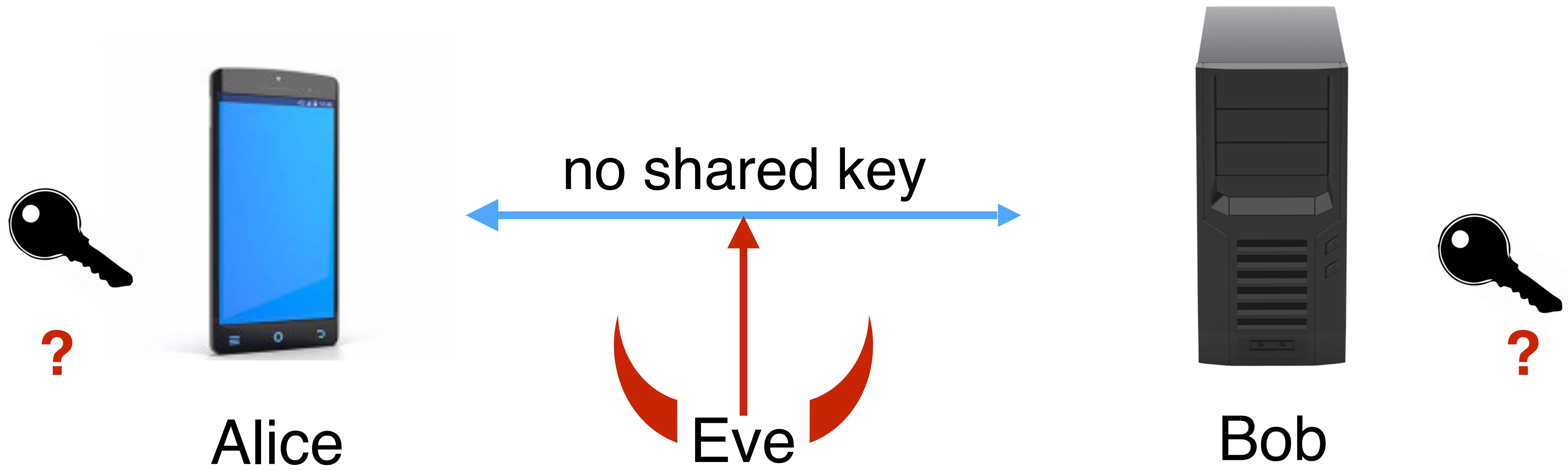
ECRYPT recommendations:

Protection	Symmetric	Factoring Modulus
Legacy standard level *Should not be used in new systems*	80	1024
Near term protection *Security for at least ten years (2019-2028)*	128	3072
Long-term protection *Security for thirty to fifty years (2019-2068)*	256	15360

ANSSI recommendations:

Date	Symmetric	Factoring Modulus
2014 - 2020	100	2048
2021 - 2030	128	2048
> 2030	128	3072

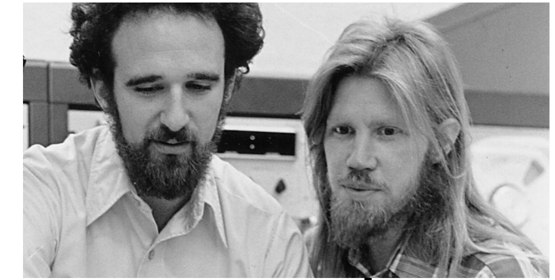
Key exchange



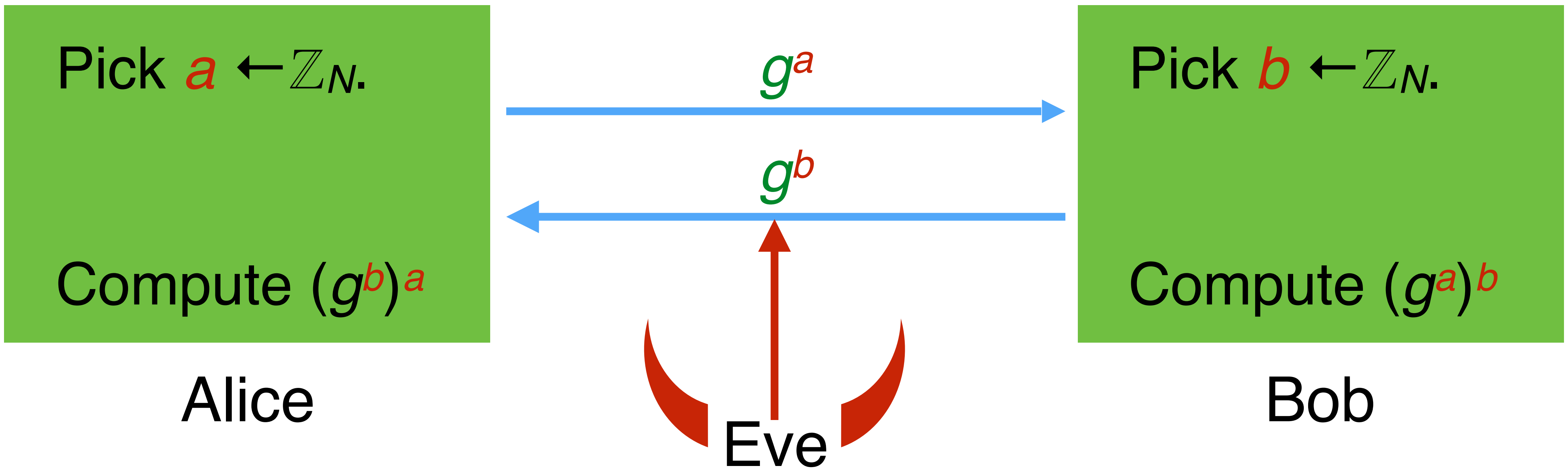
Problem: Alice and Bob do not share a key. Assume a secure channel with an eavesdropper.

Goal: Alice and Bob will generate a shared key. Eve learns nothing.

Diffie-Hellman key exchange



Fix a cyclic group **G** of order N, generated by g. These parameters are public and can be reused. (There is no "trapdoor".)



In the end, Alice and Bob can both compute g^{ab} = the shared key.

Security of computational Diffie-Hellman.

Given **G**, g, g^a, g^b, for a,b $\leftarrow \mathbb{Z}_N$, find g^{ab}.

Like RSA, essentially ad-hoc. But best known attack is to compute the **discrete logarithm**.

Diffie-Hellman: security

Security of computational Diffie-Hellman (CDH).

Given $\mathbf{G}$, g, g^a, g^b, for a,b $\leftarrow \mathbb{Z}_N$, find g^{ab}.

That is: Eve cannot compute the shared key.

Security of decisional Diffie-Hellman (DDH).

Given $\mathbf{G}$, g, distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^c) for a,b,c $\leftarrow \mathbb{Z}_N$.

That is: Eve knows *nothing* about the shared key.

Discrete logarithm.

Given $\mathbf{G}$, g, and g^a, for a $\leftarrow \mathbb{Z}_N$, find a.

If you can solve the discrete logarithm, you can solve DDH. No converse, but it is the best known attack. Note: N is usually known.

$$\text{DDH} \leq \text{CDH} \leq \text{DL}$$

Security of the discrete logarithm

Typical groups for Diffie-Hellman (hence, DL is hard):

- subgroup of prime order of $\mathbb{Z}_p^*$. Note: not $\mathbb{Z}_p^*$ itself.
- elliptic curves.

To learn more, see the Morain/Blanchet course.

For secure size of N, in $\mathbb{Z}_p^*$, same recommendations as RSA! For both ECRYPT and ANSSI.

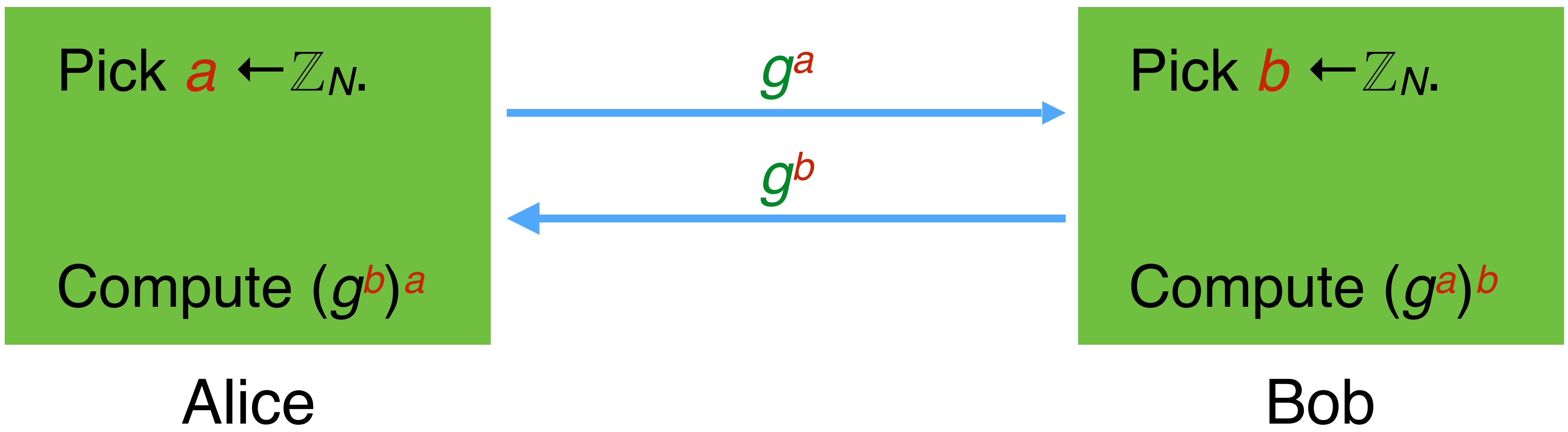
Deep connections between integer facorization & DL algorithms.

In elliptic curves, for security parameter λ, 2λ bits is enough. Elliptic curves behave roughly like a **generic group**: best attacks are generic "square root" attacks that work in any group.

NB: like RSA, all this is broken by quantum computers…

From Diffie-Hellman to ElGamal

Diffie-Hellman:



If Alice wants to send a message $m \in \mathbf{G}$ to Bob, she can send:

$$c = m \cdot g^{ab}$$

Indeed, DDH says g^{ab} is indistinguishable from random, so $m \cdot g^{ab}$ is essentially a one-time pad.

ElGamal encryption

- Assume a group **G** of order N, generator g, where DDH is hard.
- Pick $k \leftarrow_{\$} \mathbb{Z}_N$.
  - The public key is $pk = (g^k)$.
  - The secret key is $sk = k$.

Encryption: to encrypt $m \in \mathbf{G}$, pick $r \leftarrow_{\$} \mathbb{Z}_N$. The ciphertext is:

$$c = (g^r, m \cdot g^{kr}).$$

Decryption: for a ciphertext $c = (c_1, c_2)$, the message is:

$$m = c_2 / c_1^k.$$

Hard problem: DDH in **G**.

Trapdoor: knowledge of discrete logarithm k of g^k.

ElGamal security

Proposition. If DDH is hard, then ElGamal is IND-CPA secure.

Reduction: Assume adversary ***A*** against ElGamal.

Build ***B*** against DDH:

- ***B*** receives sample $(x,y,z) \leftarrow D_b$ from DDH challenge, $b \in \{0,1\}$.
- ***B*** calls ***A***, who provides m_0,m_1.
- ***B*** picks $c \leftarrow_\$ \{0,1\}$, sends ciphertext:

$$(y, z \cdot m_c)$$

- ***B*** receives c' from ***A***.
- ***B*** outputs $(c = c')$.

Note: ElGamal is not IND-CCA secure. It is **malleable.**

ElGamal security

Proof of reduction:

- If $b = 0$, (x,y,z) is a real DDH instance, and $(y, z \cdot m_c)$ is a valid ElGamal encryption of m_c. So $\mathrm{Adv}^{\mathrm{DDH}}(B) = \mathrm{Adv}^{\mathrm{ElGamal}}(A)$.
- If $b = 1$, (x,y,z) is uniform, and so is $(y, z \cdot m_c)$. So $\mathrm{Adv}^{\mathrm{DDH}}(B) = 0$.

Hence:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{DDH}}(B) \\
&= 2\Pr(B(x,y,z) = b) - 1 \\
&= \Pr(B(x,y,z) = b \mid b = 0) + \Pr(B(x,y,z) = b \mid b = 1) - 1 \\
&= \frac{1}{2}(\mathrm{Adv}^{\mathrm{ElGamal}}(A) + 1) + \frac{1}{2} - 1 \\
&= \frac{1}{2}\mathrm{Adv}^{\mathrm{ElGamal}}(A)
\end{aligned}$$

Hybrid Encryption

Symmetric crypto: very fast, limited functionality. Used to encrypt the bulk of data and communications.

Publick-key crypto: slow, rich functionality. Used sparely, for critical security properties.

Example.

Hybrid encryption: use PKE to send a symmetric key, then use that key to encrypt the rest of the data.