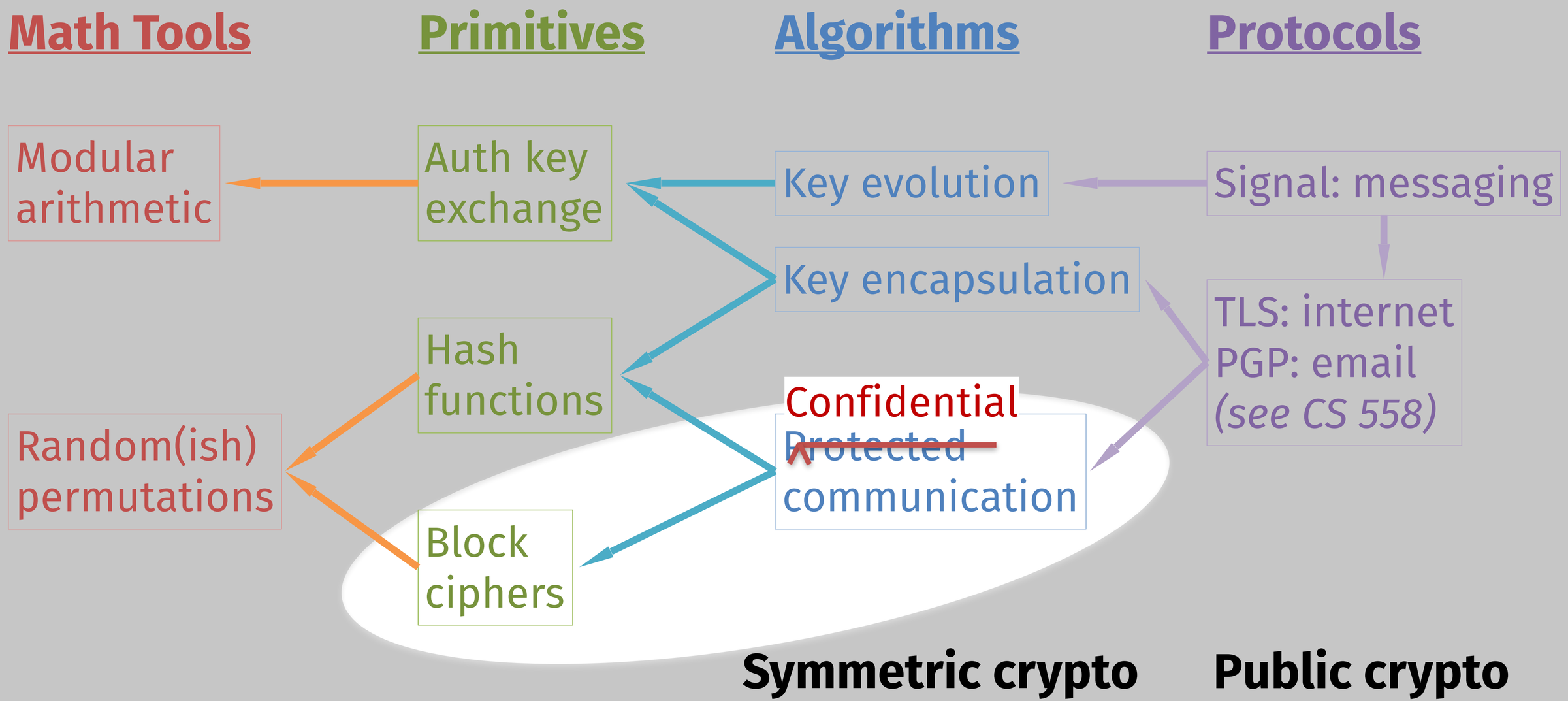
Lecture 7: Symmetric encryption

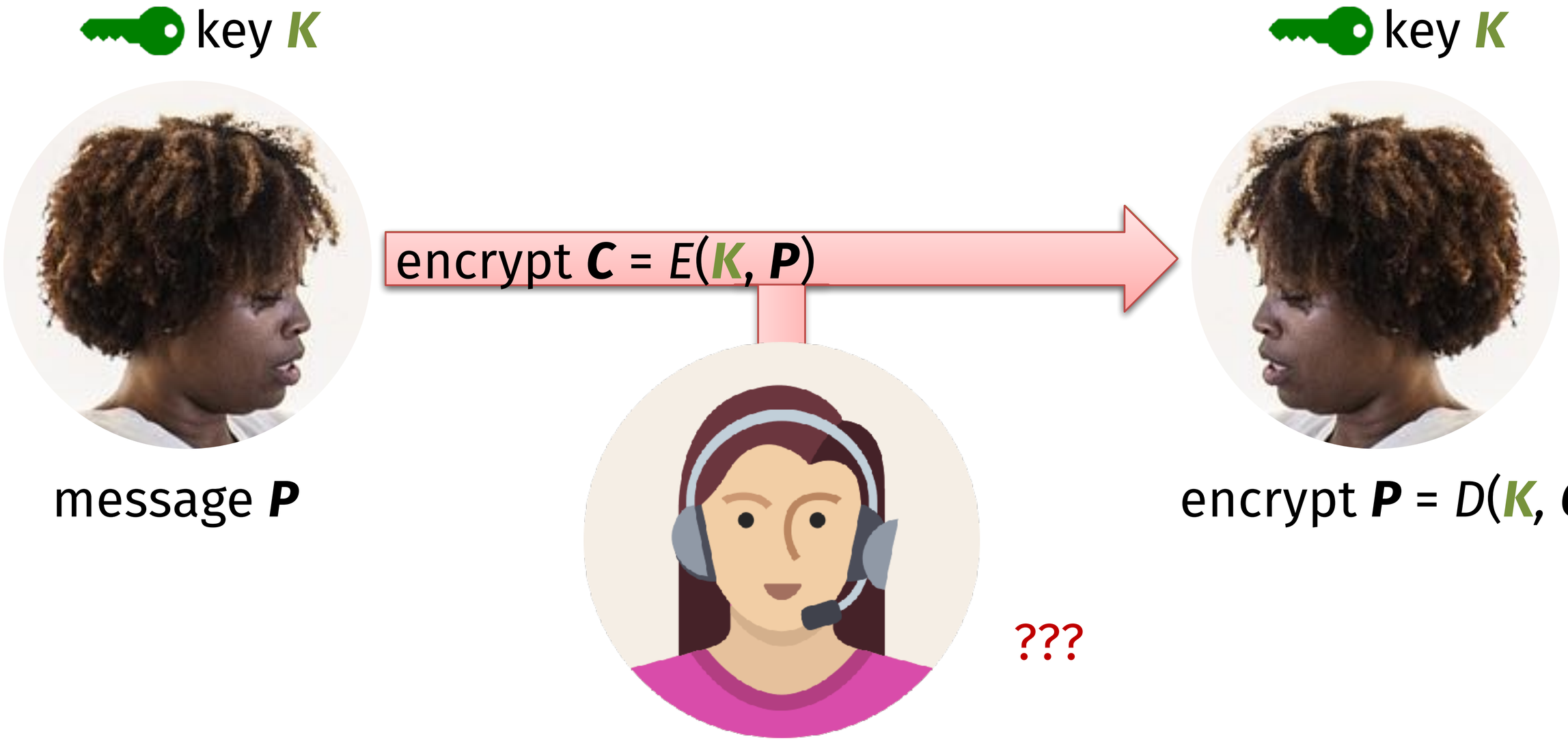
Posted on piazza.com

- Lab 4: due on Monday 2/18 at 11pm
- Office hours change this week: Thursday at 9-10am and 3-4pm

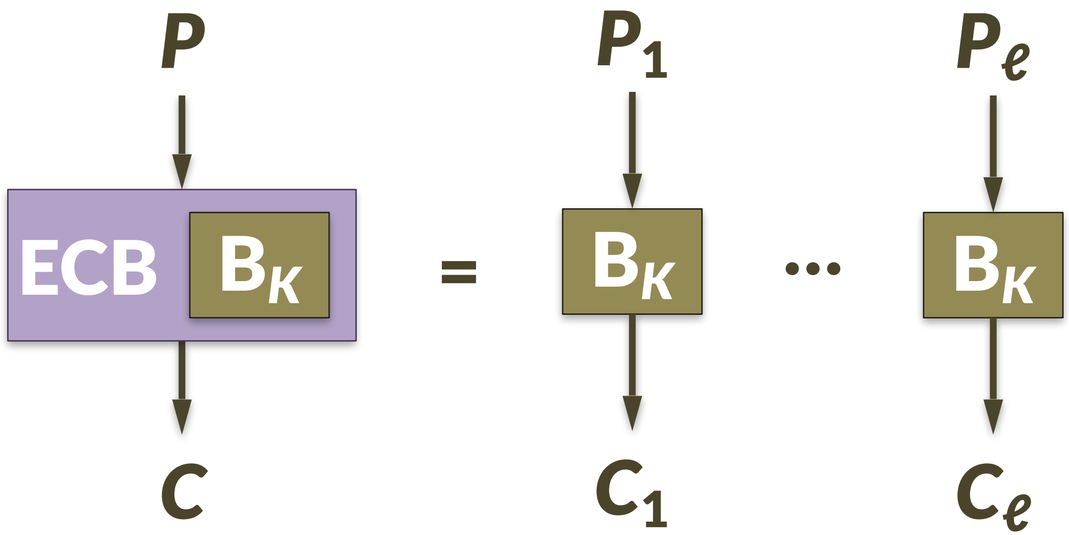
Encryption via enciphering



Protecting *privacy* of data at rest



Bad attempt: Electronic Codebook (ECB) mode



Raw image of
Linux penguin

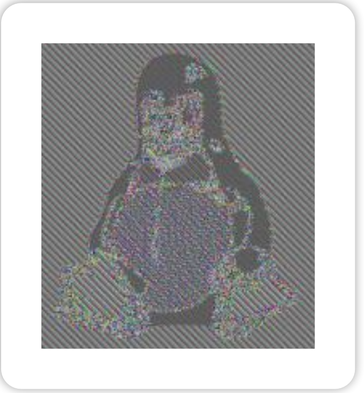


Image after
ECB mode

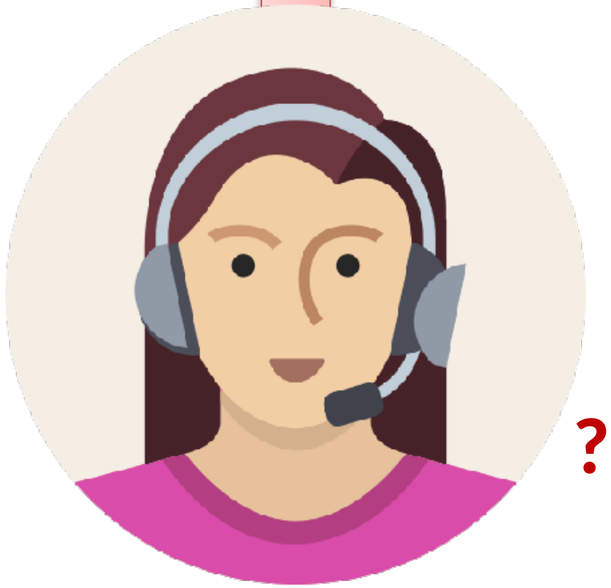
What we want
from encryption

What if message blocks *don't* repeat?

key K

private data
$P_1, P_2, \ldots P_\ell$

encode $C_i = B_K(P_i)$



???

key K

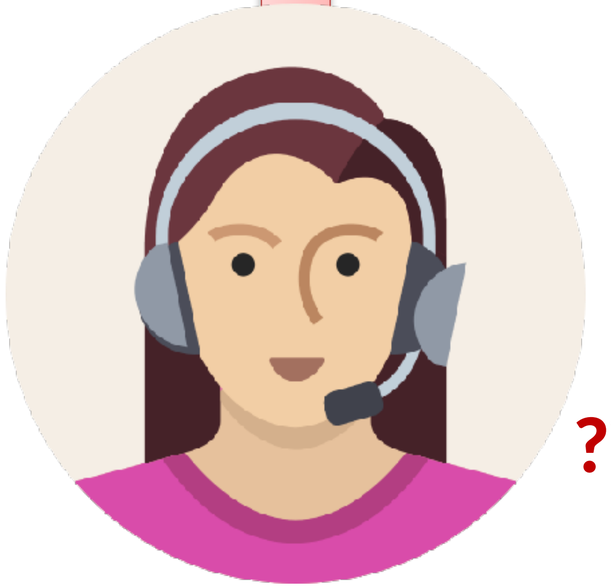
decode $P_i = B_K^{-1}(C_i)$

What if message blocks *don't* repeat?

key K

private data
$P_1, P_2, \ldots P_\ell$

encode $C_i = \Pi(P_i)$



???

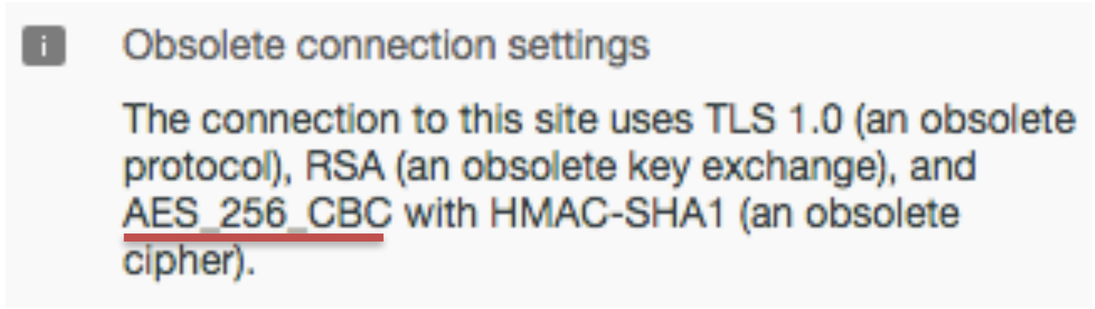
key K

decode $P_i = \Pi^{-1}(C_i)$

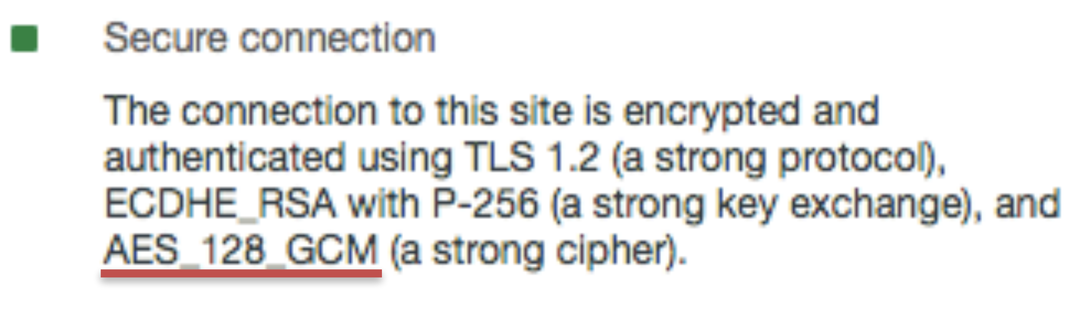
Question: *How do we guarantee that message blocks don't repeat?*

Encryption in practice

bu.edu homepage (2017)



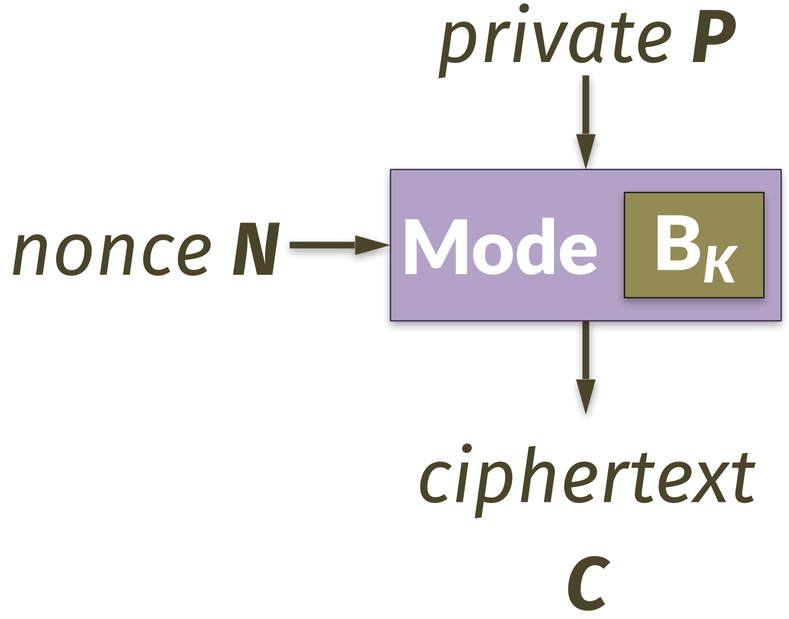
www.amazon.com



Lessons learned

- **Randomness matters:** We can confuse Eve! Just need to design a mode of operation that guarantees each enciphered block is unique.

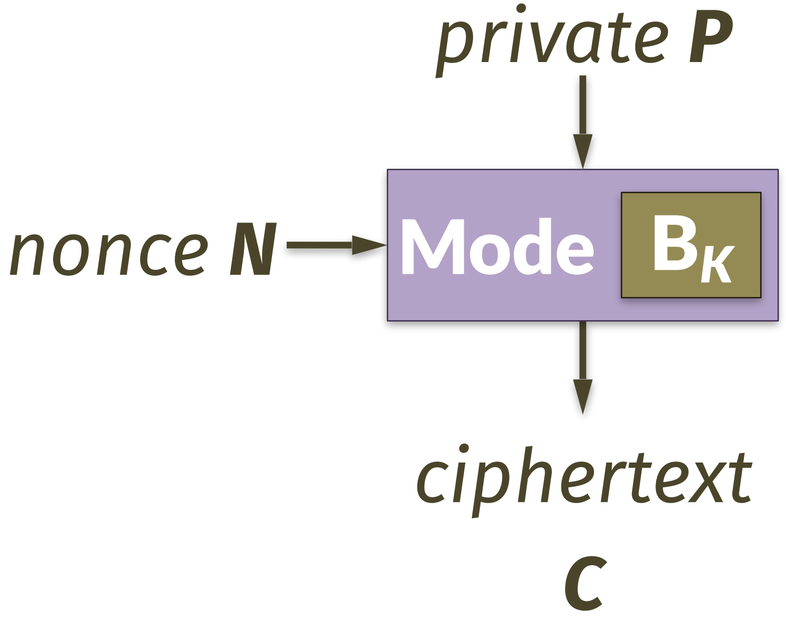
- **Definitions matter:** Our argument leveraged the concept that a block cipher “looks like” a random permutation from Eve’s point of view.



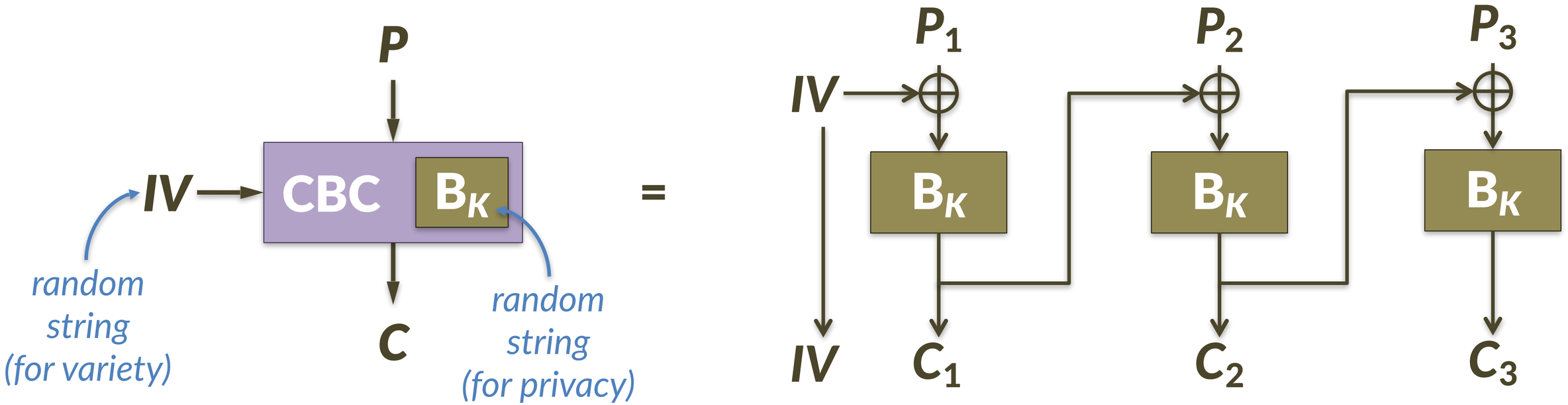
Lessons learned

- **Randomness matters:** We can confuse Eve! Just need to design a mode of operation that guarantees each enciphered block is unique.

- **Definitions matter:** Our argument leveraged the concept that a block cipher “looks like” a random permutation from Eve’s point of view.



Cipher block chaining (CBC) mode



Two differences from CBC-MAC:

1. All blocks are output
2. First block is protected by a public, randomly chosen *initialization vector*

Apple's Common Crypto Library Defaults to a Zero IV if One is not Provided

Today I was writing some guidelines about generating keys for mobile applications at work. While providing code examples in Java and Obj-C for AES encryption I happened to look at Apple's Common Crypto library . While going through the source code for CommonCryptor.c, I noticed that IV is commented as `/* optional initialization vector */`. This makes sense because not all ciphers use IV and not all AES modes of operation (e.g. ECB mode). However; if an IV is not provided, the library will default to a zero IV.

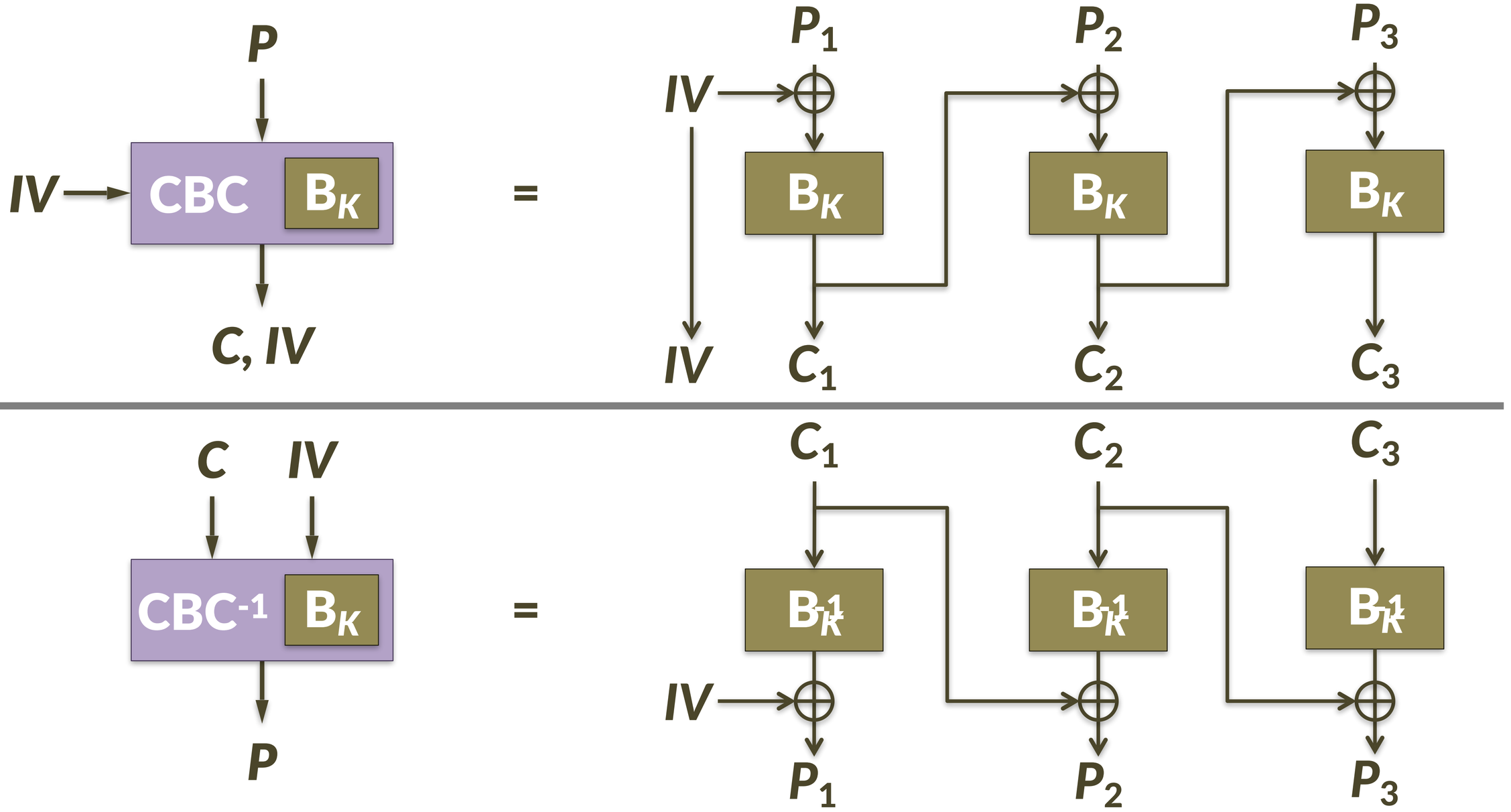
You can see the code here inside the function `ccInitCryptor` (search for defaultIV) source. `CC_XZEROMEM` resets all bytes of IV to zero (that is 0x00):

```
static inline CCCryptorStatus ccInitCryptor
(CCCryptor *ref, const void *key, unsigned long key_len, const void *tweak_key, const void *iv

    size_t blocksize = ccGetCipherBlockSize(ref);
    uint8_t defaultIV[blocksize];


    if(iv == NULL) {
        CC_XZEROMEM(defaultIV, blocksize);
        iv = defaultIV;
    }

    ...

    return kCCSuccess;
}
```

While I am told this is probably common behavior in crypto libraries, I think it's dangerous. I ended up putting a comment in code examples warning developers about this behavior. So, heads up ;)

Source: parsiya.net/blog/2014-07-03-apples-common-crypto-library-defaults-to-a-zero-iv-if-one-is-not-provided/

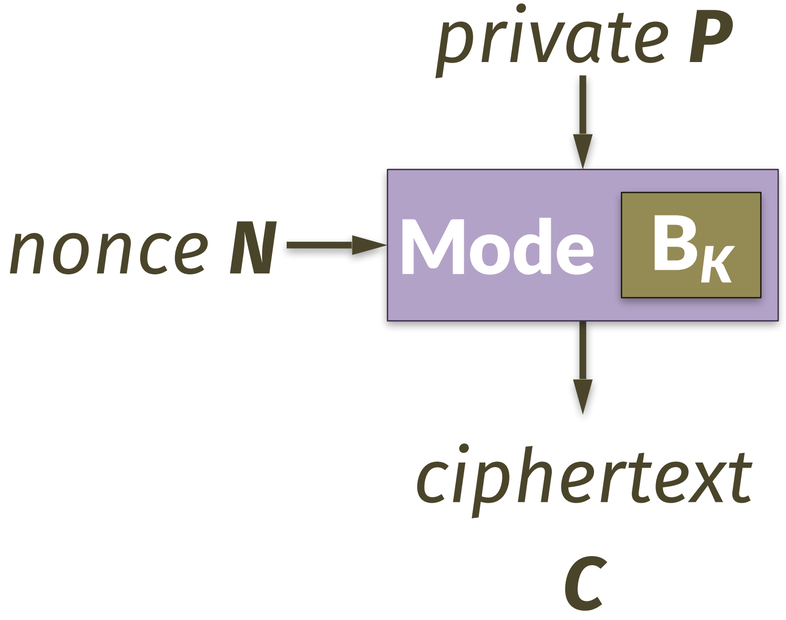
CBC decryption



Lessons learned

- **Randomness matters:** We can confuse Eve! Just need to design a mode of operation that guarantees each enciphered block is unique.

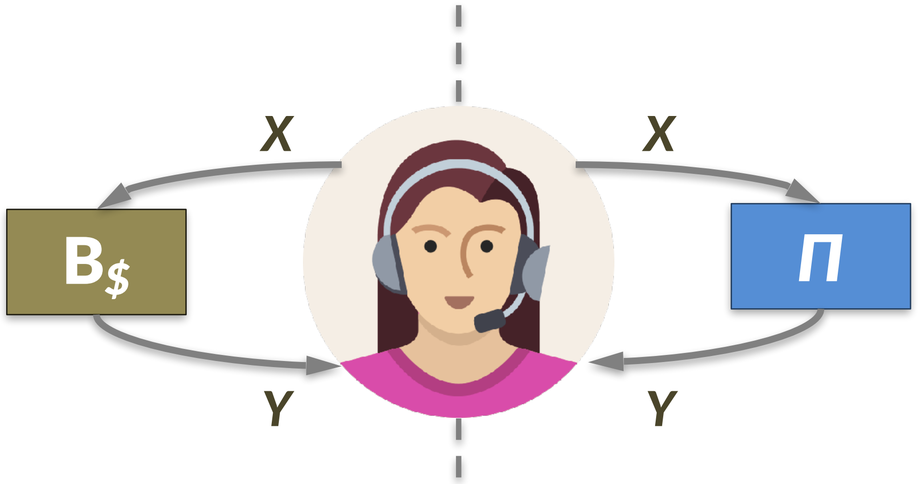
- **Definitions matter:** Our argument leveraged the concept that a block cipher “looks like” a random permutation from Eve’s point of view.



A new type of pseudorandomness

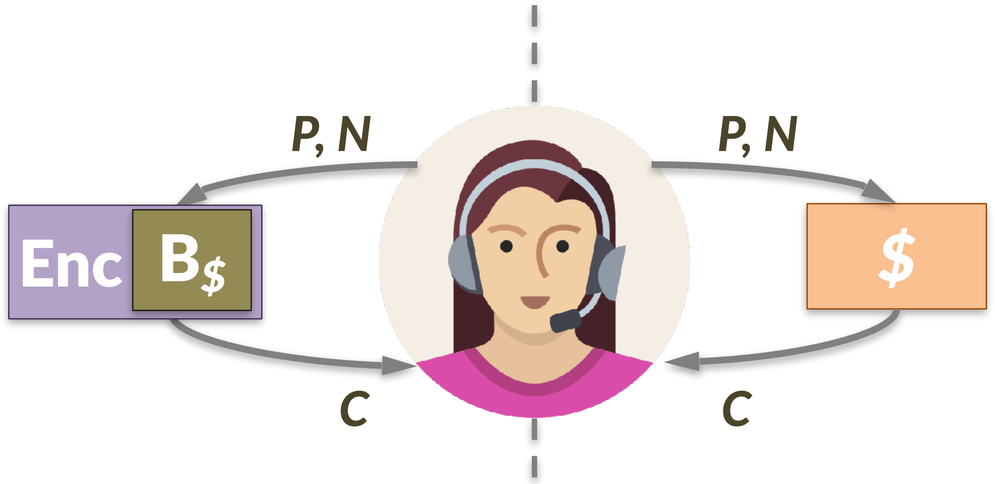
Block cipher

B_K looks like a truly random function, meaning nobody can tell them apart



Encryption scheme

Similar, except even making the same request twice yields different answers



Defining symmetric encryption

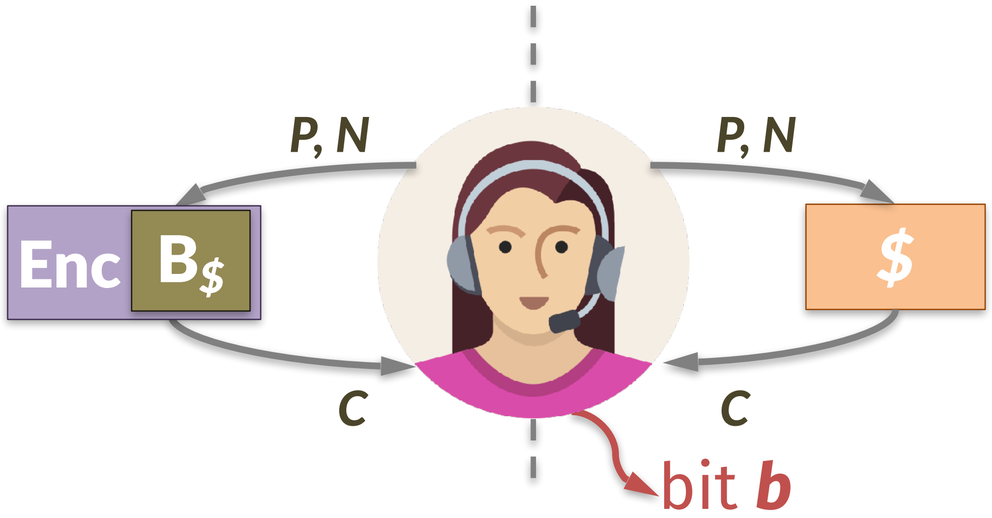
Algorithms

- **KeyGen:** choose key $K \leftarrow \{0,1\}^{\lambda}$
- **Encrypt**$_K(P \in \{0,1\}^{\rho}, N) \rightarrow$ ct $C \in \{0,1\}^{\gamma}$
  - Must be randomized with $\gamma \geq \rho$
- **Decrypt**$_K(C \in \{0,1\}^{\tau}, N) \rightarrow P$

Constraints

- **Performance:** All algorithms are efficiently computable
- **Correctness:** For every K, Enc$_K$ and Dec$_K$ are inverses
- **Security:** ???

Pseudorandomness under chosen plaintext attack (IND$-CPA)



For every adv *A* with runtime ≤ t and queries totaling ≤ q blocks,

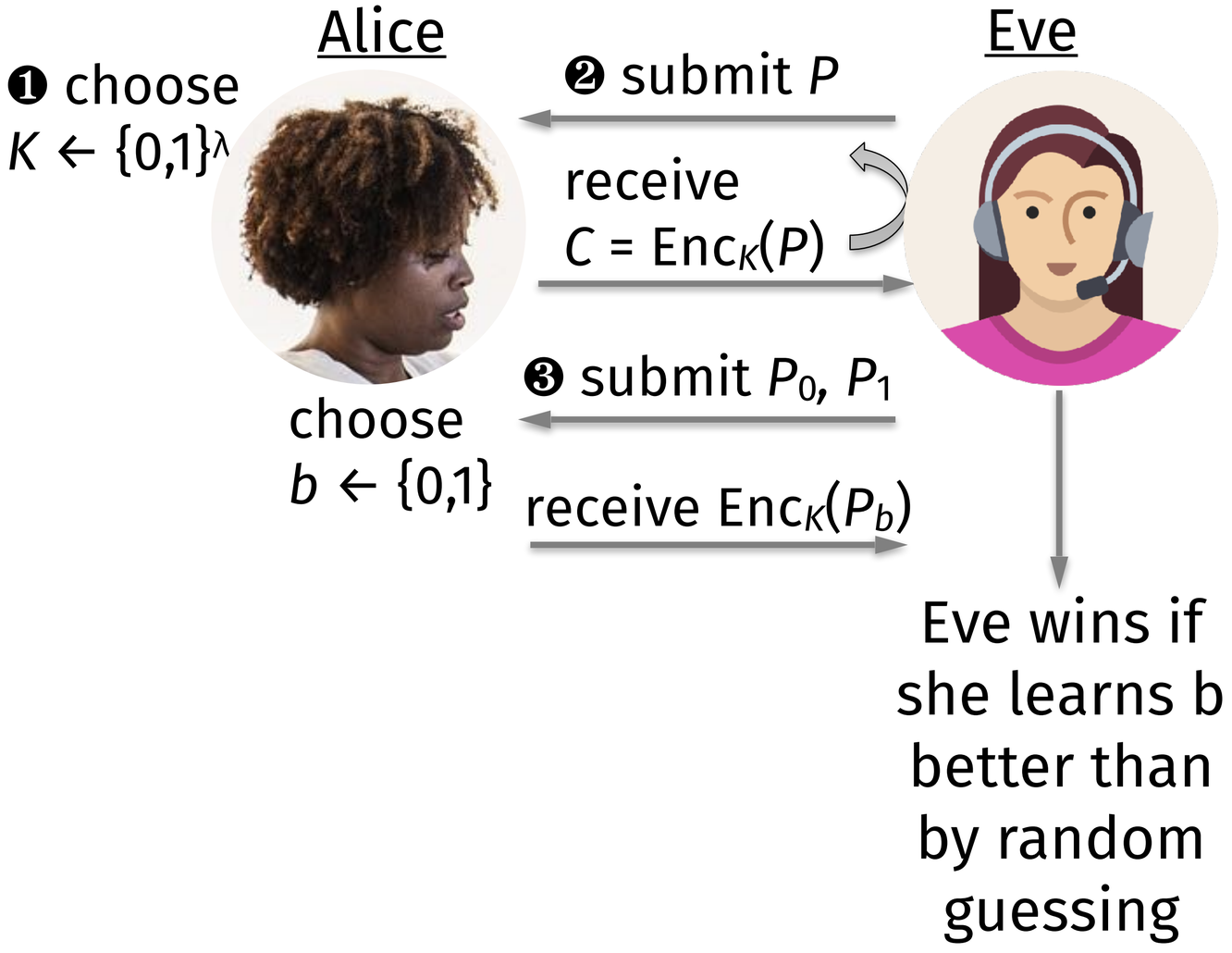
$$A^{Enc_{\$}(-,-)} \approx_{(q,t,\varepsilon)} A^{\$(-,-)}$$

Two variants

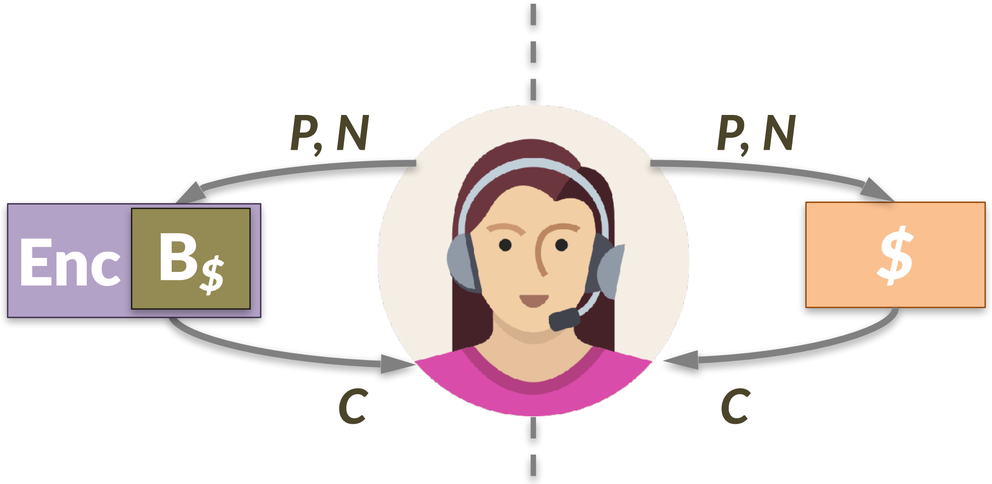
- *Standard:* Eve doesn't choose *N*, instead it is chosen randomly
- *Nonce-respecting:* Eve chooses *N*, but each choice must be distinct

Indistinguishability under a chosen plaintext attack (IND-CPA)

- Let's make an encryption game similar in style to the forgery one
- Alice provides many (P, C) pairs of Eve's choice
- Should still be difficult for Eve to distinguish between Enc(P_0) and Enc(P_1)

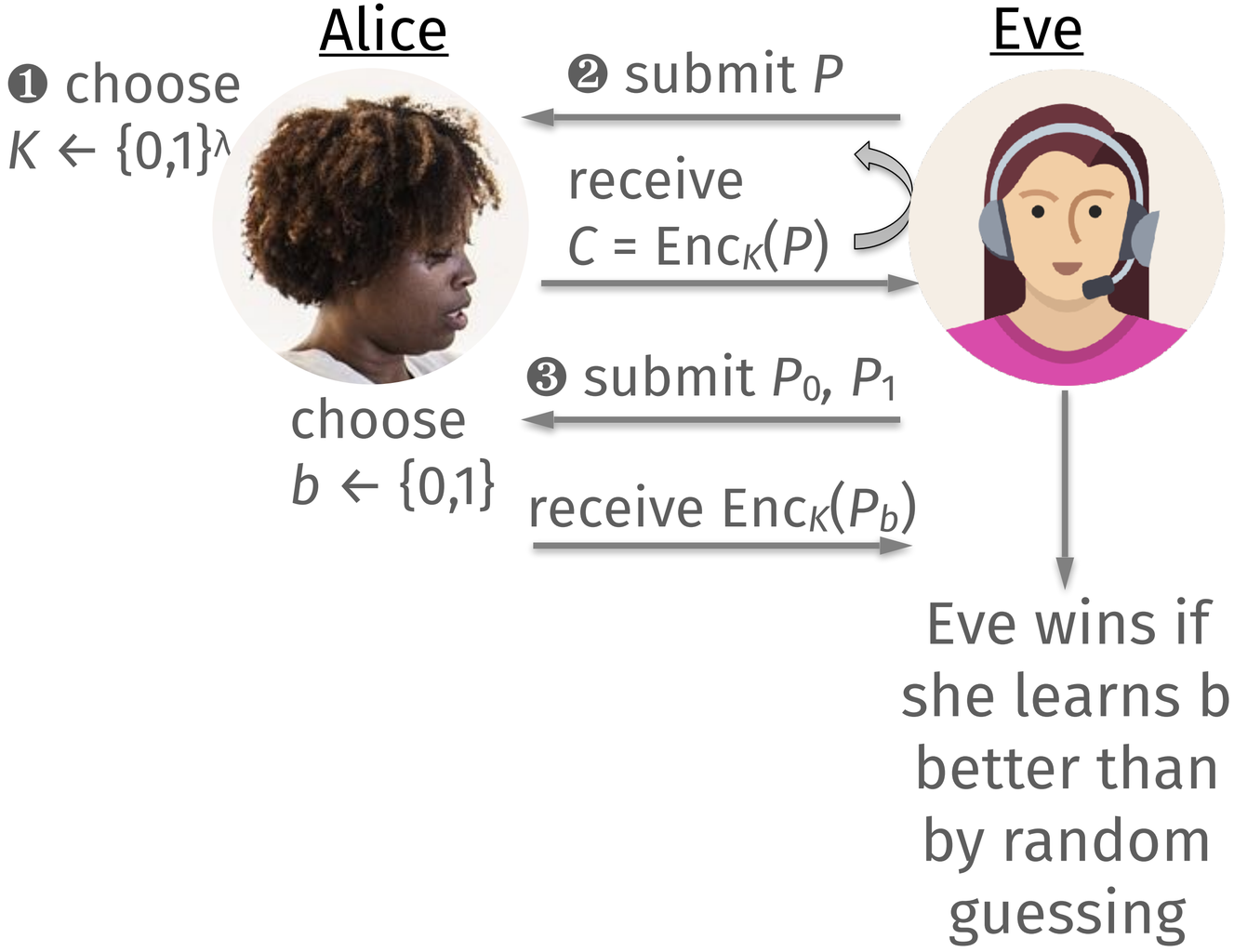


Thm. If Enc is IND$-CPA secure, then it is IND-CPA secure



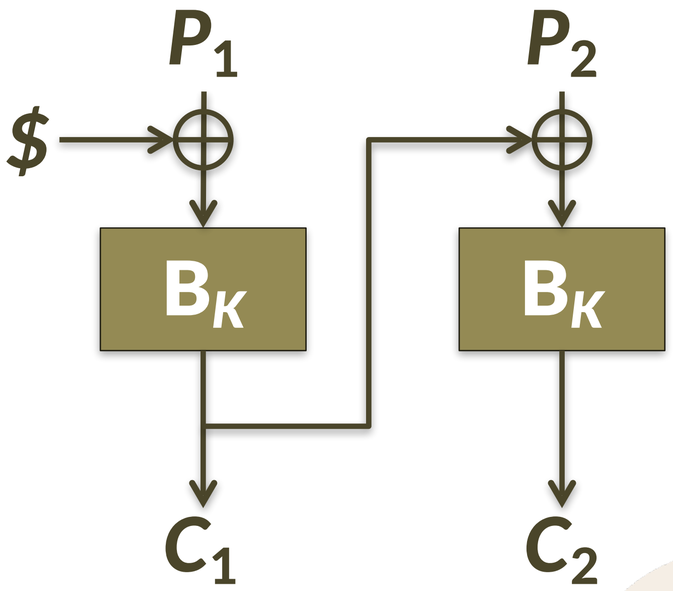
Proof sketch: even after making all of the *P* queries,

$$\mathrm{Enc}_K(P_0) \approx \$ \approx \mathrm{Enc}_K(P_1)$$

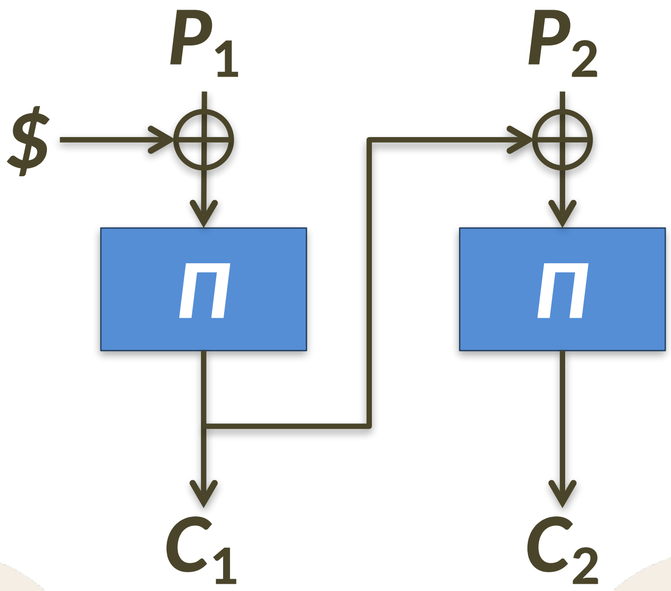


Informal proof by picture: CBC is IND$-CPA

Real CBC mode

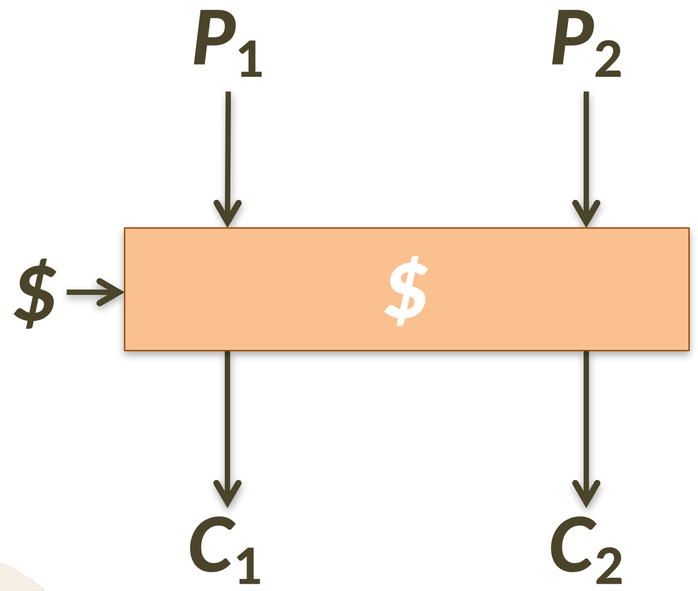


CBC with ideal cipher



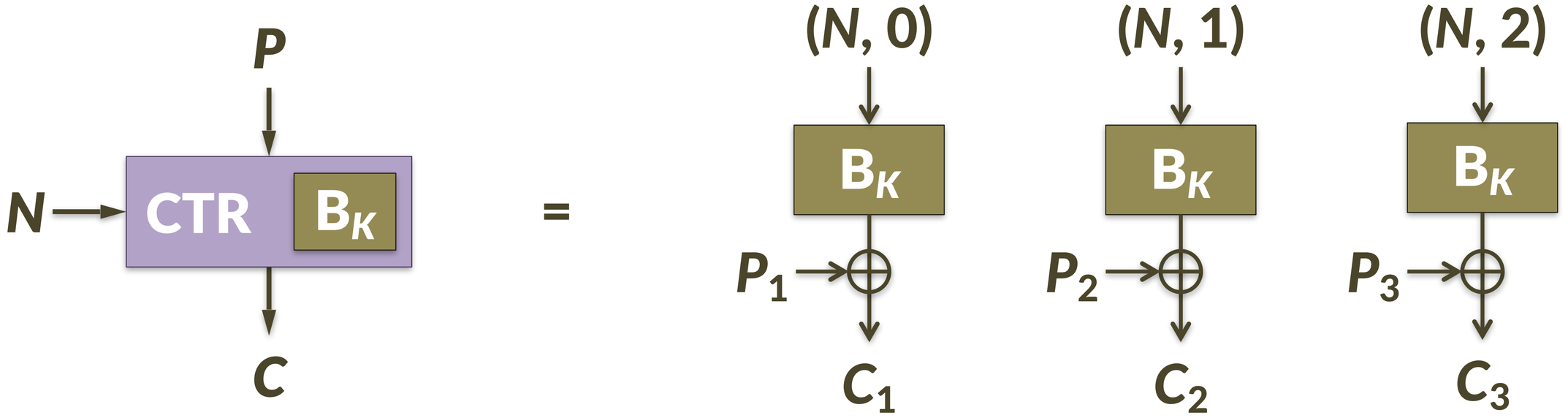
by pseudorandomness of the block cipher

Ideal encryption scheme



*input to every **Π** is "random"*

Counter (CTR) mode



Issues to consider

2^{96} 2^{32}

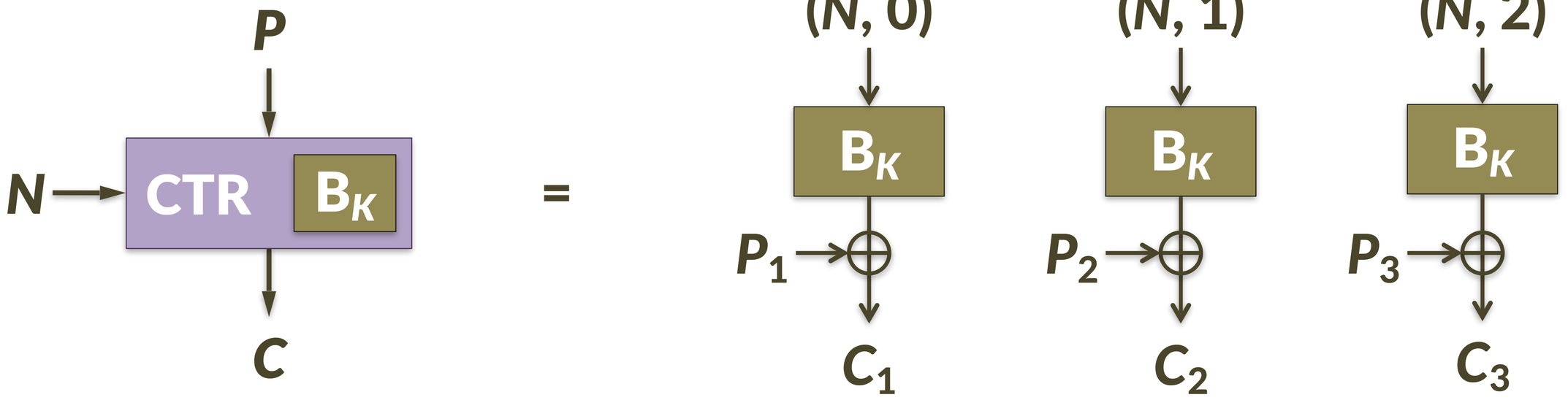
✔ 1. Tradeoff between the lengths of ***N*** and ***P***

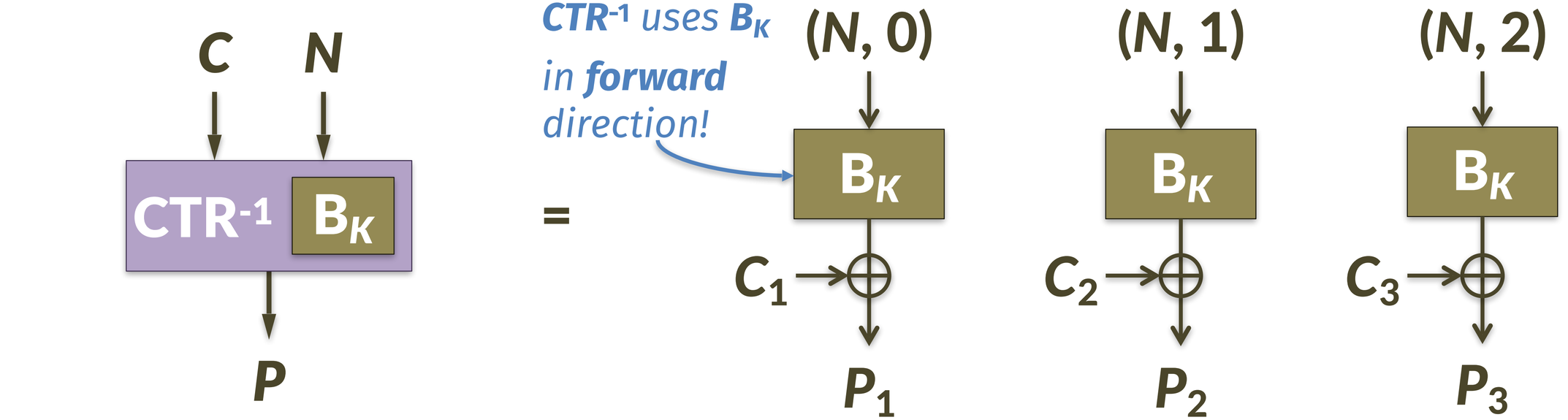
✔ 2. How do we choose ***N*** if the parties are stateless? → *choose randomly, rely on birthday bound*

? 3. How to prove that CTR satisfies IND$-CPA?

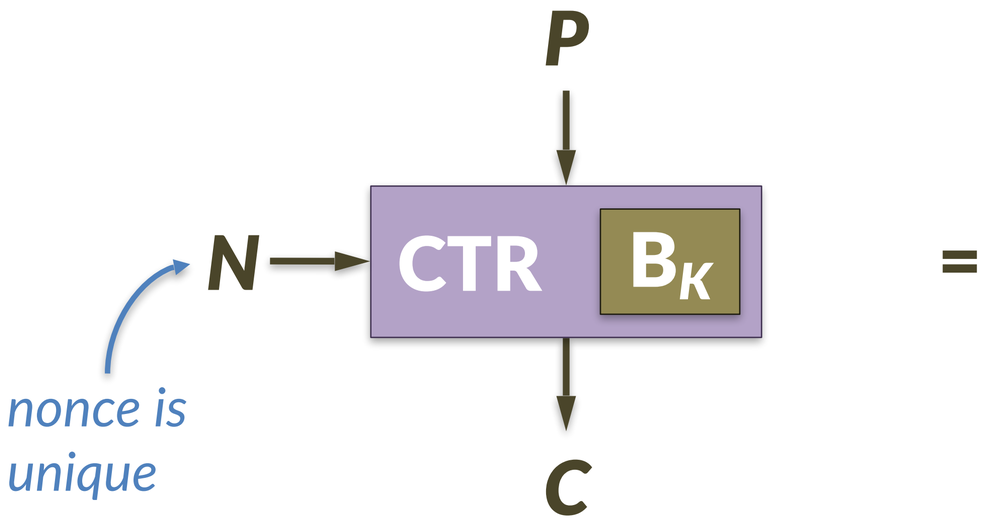
? 4. What to do if ***N*** is accidentally repeated?

Counter (CTR) mode

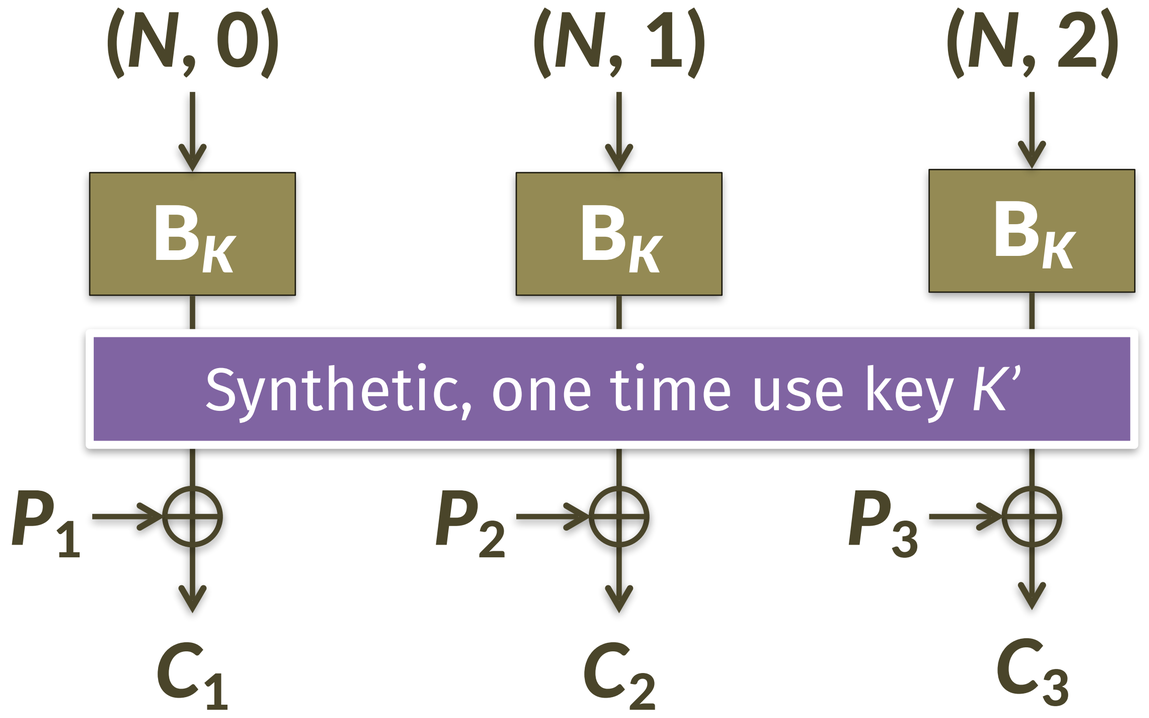


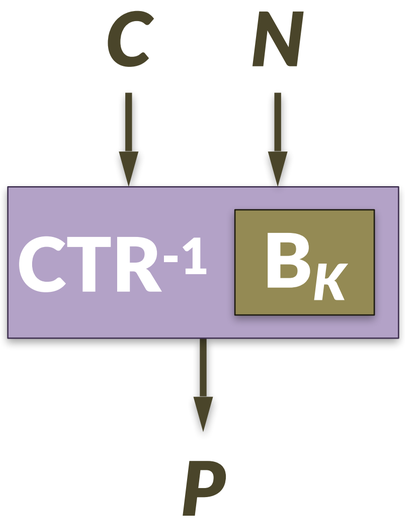


Counter (CTR) mode

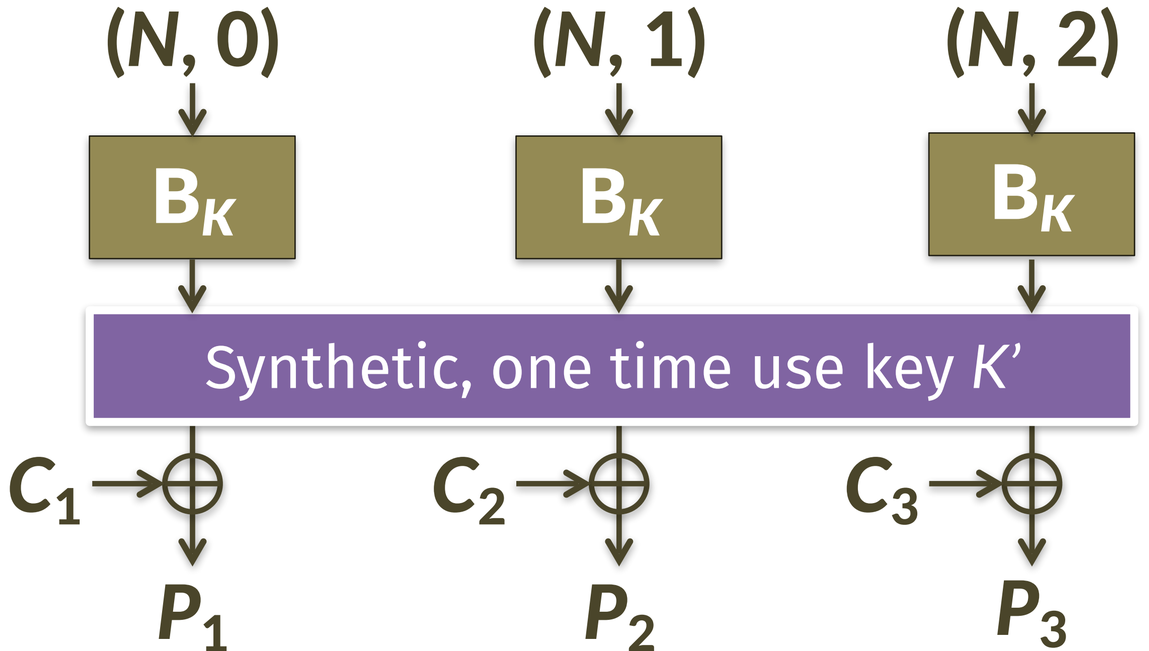


=

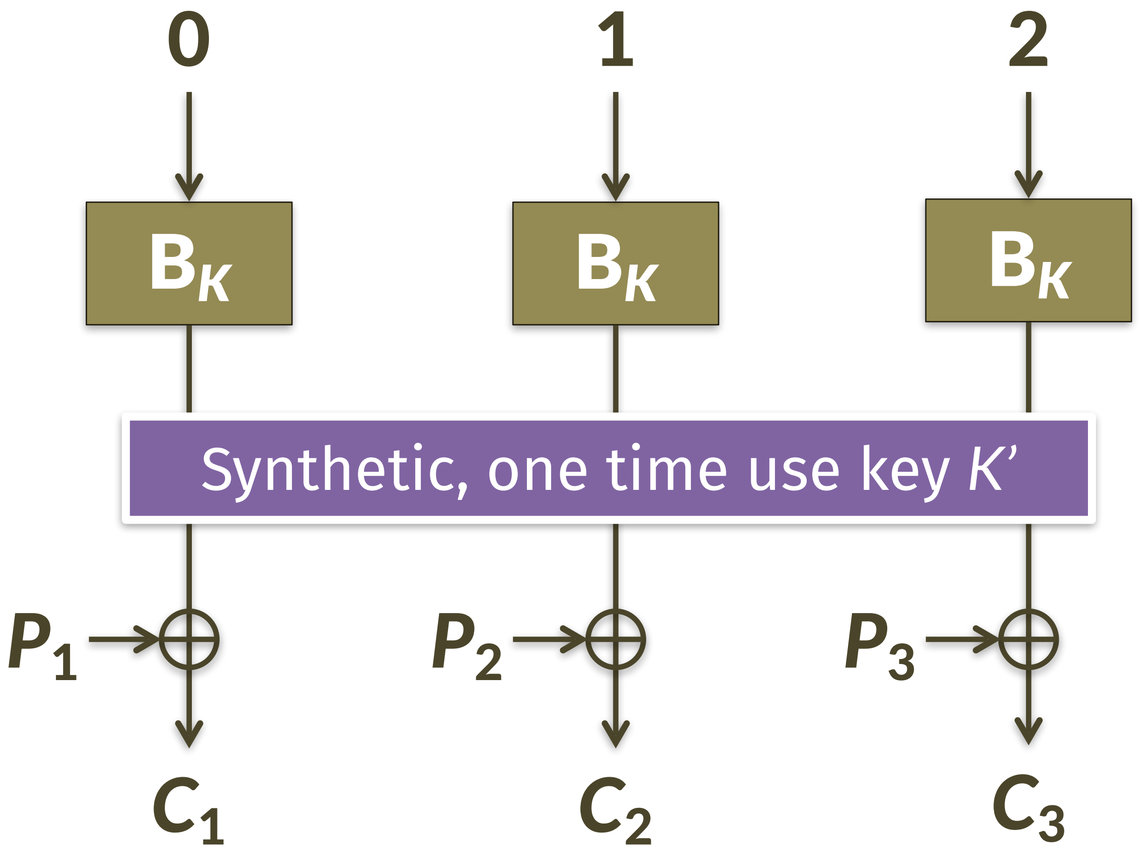




=



Counter mode ⇒ Stream cipher



Short ***K***

\+

Nonce ***N***

⇒

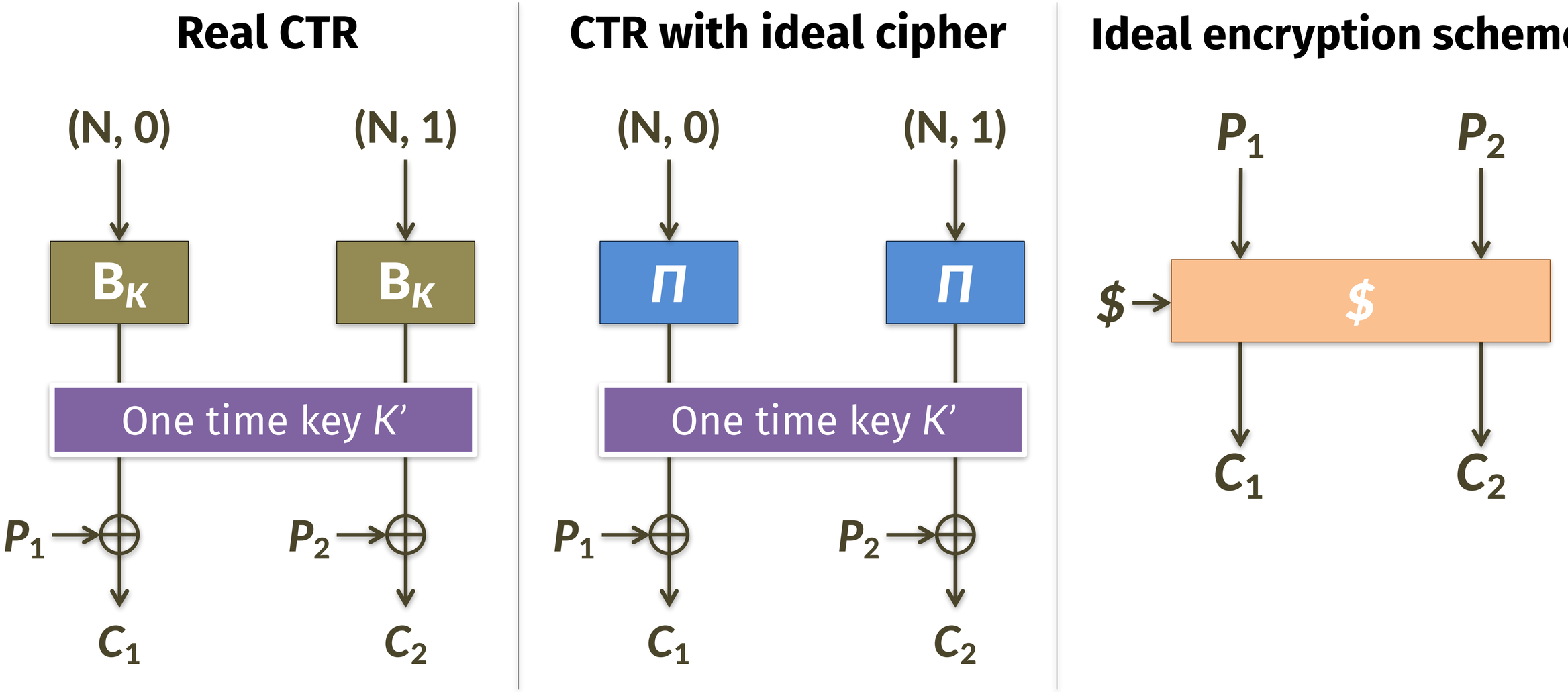
Synthetic, one time use key *K'*

- Roughly ~2x better performance
- Competitions held by European standards body: NESSIE, eSTREAM
- Only recently has anything gained much adoption

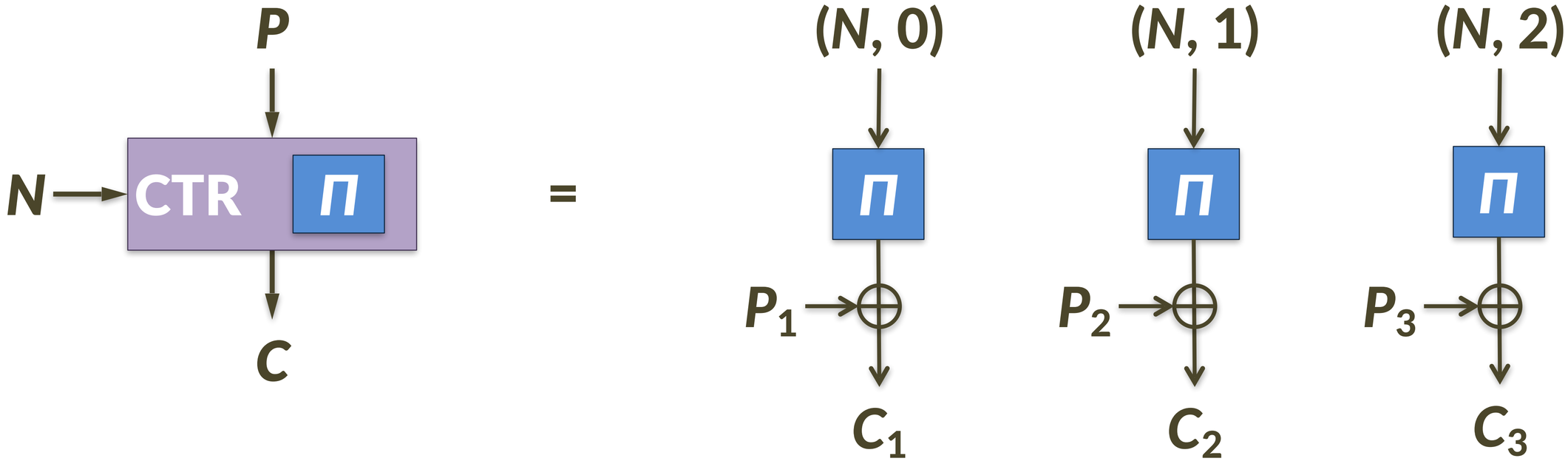
Random functions R: $\{0,1\}^{in} \rightarrow \{0,1\}^{out}$

Case:	in < out	in = out (permutation)	in > out	in = ∞
One *public* function	Stream cipher	Codebook	Compression function	Hash function/ random oracle
Family of functions indexed by *private* key	CTR	Block cipher	Message auth code	

Informal CTR reduction by picture



CTR mode with $\Pi \Rightarrow$ one time pad



Recall: How formal reductions work

<u>If we begin with:</u>

a block cipher B_K
that is $(q_B, t_B, \varepsilon_B)$ pseudorandom

<u>Then we can construct:</u>

symmetric key enc scheme Mode B_K
that is $(q_C, t_C, \varepsilon_C)$ indistinguishable from pseudorandom under chosen plaintext attack

Contrapositive*:** If an adversary ***A can break Mode B_K, then we can construct an adversary ***A'*** that breaks B_K almost as effectively.

Formal CTR mode reduction

<u>If we begin with:</u>

Adversary $\boldsymbol{A_{CTR}}$ who can distinguish

with probability > ε_C given time t_C and queries that total q_C blocks of data

<u>Then we can construct:</u>

Adversary $\boldsymbol{A_{BC}}$ who can distinguish

with probability > ε_B given time t_B and a total of q_B queries

Formal CTR mode reduction

If we begin with:

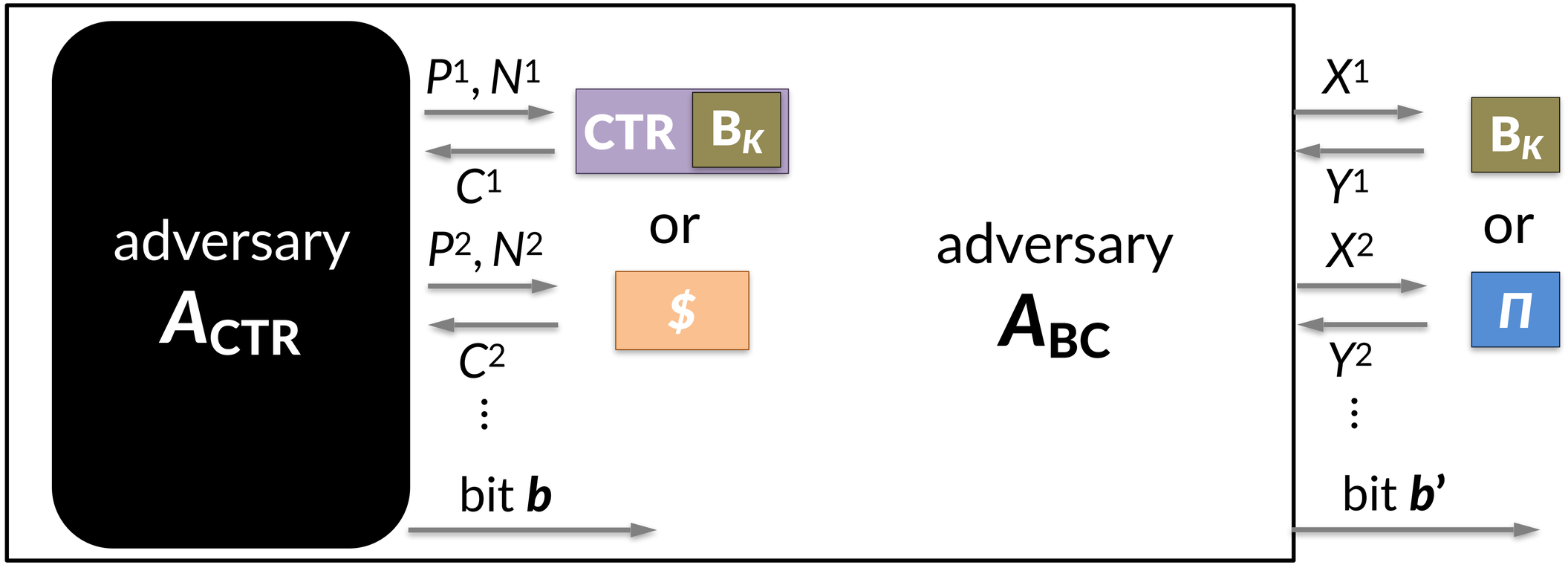
Adversary A_{CTR} who can distinguish

CTR B_K from $

Then we can construct:

Adversary A_{BC} who can distinguish

B_K from Π



How A_{BC} operates

Step 1:

Wait for A_{CTR} to output a (P, N) pair

Step 2:

Query A_{BC}'s own oracle on (N,0), (N,1), …, (N, |P|-1)

Step 3:

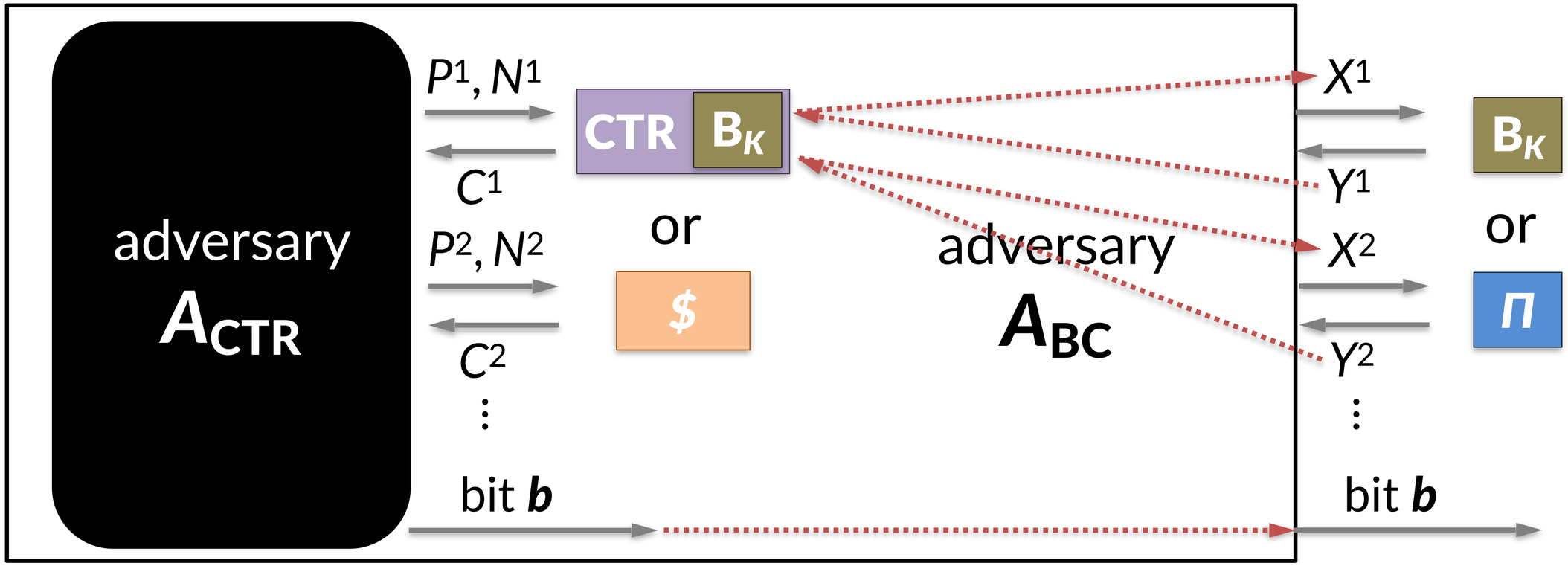
Concatenate response blocks, then xor with P

Step 4:

Repeat

Step 5:

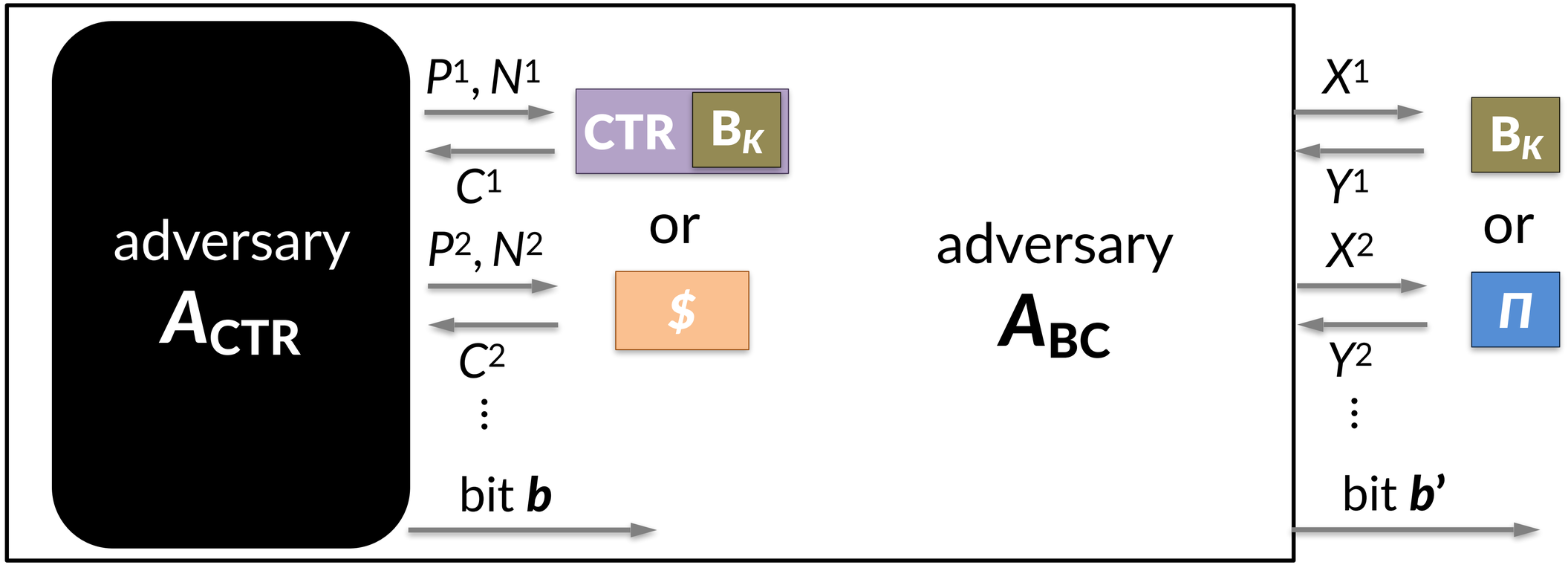
Output the same bit as A_{CTR}



Why this reduction works

If A_{BC} is talking to B_K, then this procedure faithfully yields CTR B_K

If A_{BC} is talking to Π, then this procedure faithfully yields CTR Π, identical to $\$$ by non-repetition



Our final result

<u>If we begin with:</u>

Adversary $\boldsymbol{A}_{\mathbf{CTR}}$ who can distinguish

with probability > ε_C given time t_C and queries that total q_C blocks of data

<u>Then we can construct:</u>

Adversary $\boldsymbol{A}_{\mathbf{BC}}$ who can distinguish

with probability > ε_C given time $t_C + q_C$ and a total of q_C queries