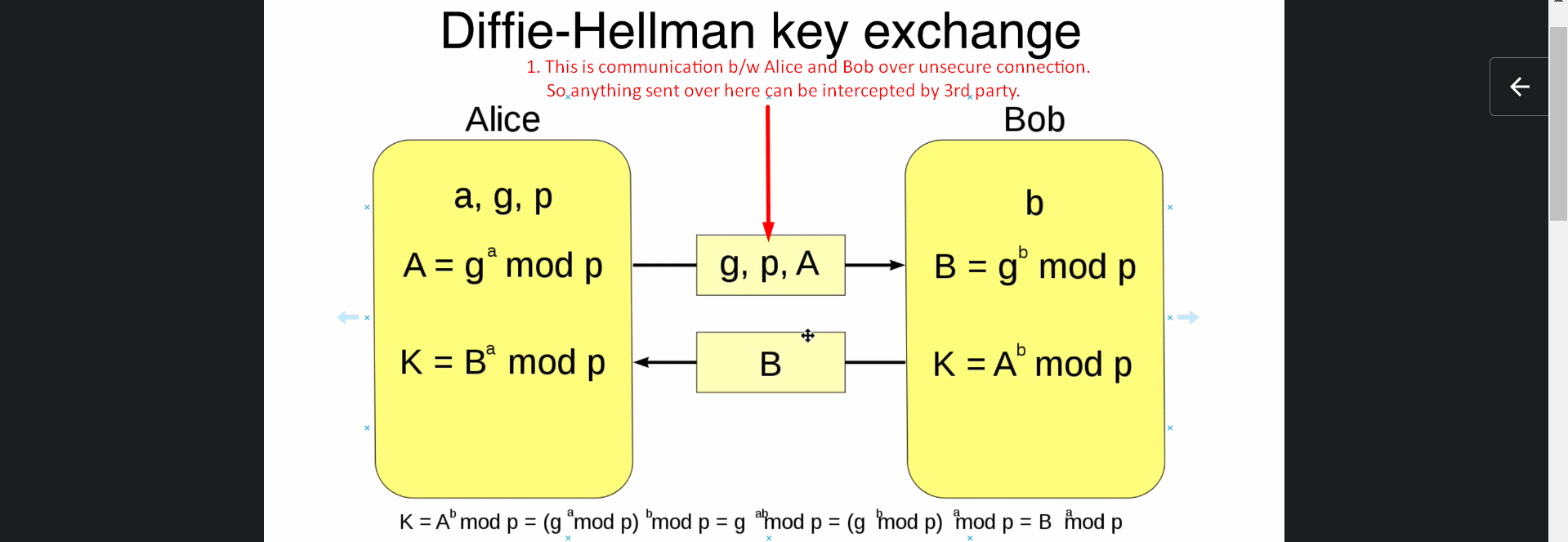
1. As we discussed in last lecture that Modulus is one-way function.
2. Agenda:
   1. Discuss how Diffie Hellman uses Modulus function to generate same key both sides.
3. 
4. Let’s discuss the assumed parameters, calculated values and all the operations (modulus, exponent and transmission) in this algo process.
5. First Alice and Bob who we are the end users, b/w whom the data transmission will occur requires a symmetric key for data encryption.
6. As we can see in the above snapshot, there are some calculations which require some input.
7. Let’s discuss the steps with non-optimization form.
   1. **Alice and Bob both of them, each will assume a separate pair of numbers**.
      1. Alice:
         1. “a”: **private** number and will not be shared with anyone.
         2. “g”: **public** number will be shared with Bob.
      2. Bob:
         1. “b”: **private** number and will not be shared with anyone.
         2. “p”: **public** number will be shared with Alice.
   2. **After sharing public numbers with each other**,
      1. Alice:
         1. will calculate A 🡪 A = ga mod p [Where P is public number from Bob]
         2. Alice will share the calculated A with Bob.
      2. Bob:
         1. will calculate B 🡪 B = g**b** mod p [Where P is public number from Bob]
         2. Bob will share the calculated B with Alice.
   3. **Now both of them has each other’s calculated value A, B and now they will calculate the key value K.**
      1. Alice side K = Ba mod p
      2. Bob side K = Ab mod p  
         Both K will be equal after all we are generating a Symmetric Key both sides not asymmetric key.
8. **But there is trick here** to optimize the number of transmission steps.  
   Bob’s public number will be created by Alice itself not by Bob and will be shared with Bob just to reduce the number of transmission steps.  
   Otherwise if Bob assumes public number p and shares with Alice then Alice calculates A and will share with Bob, then this process will have two steps.   
   But now Alice itself will assume Bob’s public number and will calculate A and then will share A (calculated value by Alice), p (Bob’s public number) and g (Alice’s public number). It will reduce one transaction step.
9. There is a formula at the bottom of the snapshot which is basically a proof to prove why this process (mathematical steps) will produce same K both sides.

ChatGPT Answer for better in-depth but short description:

In the **Diffie-Hellman key exchange** shown in the image, the two **public numbers** are:

1. **g** - The **base (generator)**.
2. **p** - The **prime number (modulus)**.

**Explanation:**

* Both **g** and **p** are **shared publicly** between Alice and Bob.
* These values are used to perform **modular arithmetic** for generating keys securely.
* The security of Diffie-Hellman relies on the **difficulty of computing discrete logarithms**, even if **g** and **p** are known.

**Private Values:**

* **a** (Alice's private key)
* **b** (Bob's private key)

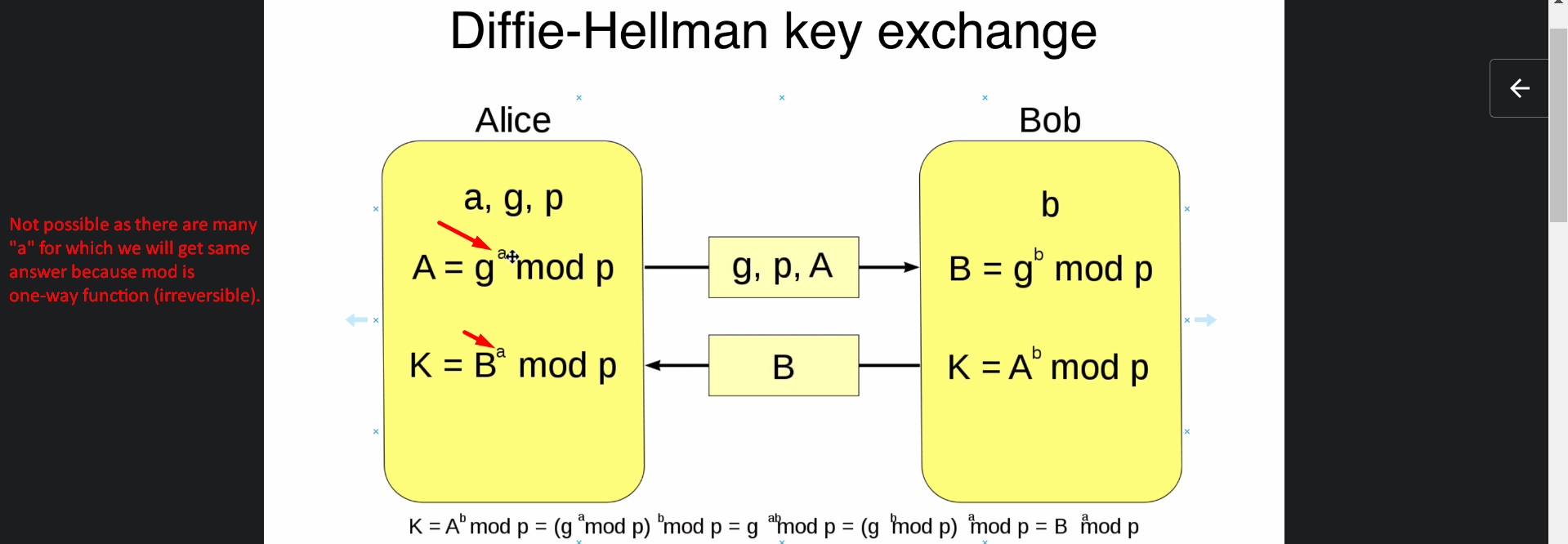
**Public Keys Sent Over the Network:**

* **A = g^a mod p** (Alice’s public key)
* **B = g^b mod p** (Bob’s public key)

**Shared Secret Key:**

* Both Alice and Bob compute the same **shared key (K)**:
* K = B^a mod p (for Alice)
* K = A^b mod p (for Bob)
* Even though **g, p, A, and B** are known publicly, computing the shared key **K** without knowing the **private keys (a or b)** is **computationally infeasible**.

Let me know if you need more explanations about this process!

1. Let’s discuss why someone except Alice and Bob can not create the same key even if they know A, g and p.
2. Answer: 
3. Private numbers a, b are extremely large 2000Bits.
4. **Question**: How long the computation **A = g^a mod** will take as g and a are very large numbers
5. **Answer**: There is a mechanism through which this calculation is performed with optimization.
6. This was all about Diffie-Hellman Key Exchange Algo to generate key both sides using unencrypted and unsecure public connection.