**SAD Lab**

**EXPERIMENT NO. 10**

**Aim**: Understanding the concepts of cryptography and guidelines for using encryption.

**Theory**:

1. What types of cryptography are symmetric and asymmetric?

**Symmetric encryption** is a type of encryption where only one key (a secret key) is used to both encrypt and decrypt electronic data. There are two types of symmetric encryption algorithms:

1. **Block algorithms**: Set lengths of bits are encrypted in blocks of electronic data with the use of a specific secret key.
2. **Stream algorithms**: Data is encrypted as it streams instead of being retained in the system’s memory.

Some examples of symmetric encryption algorithms include:

1. AES (Advanced Encryption Standard)
2. DES (Data Encryption Standard)
3. IDEA (International Data Encryption Algorithm)
4. Blowfish (Drop-in replacement for DES or IDEA)
5. RC4 (Rivest Cipher 4), RC5 (Rivest Cipher 5), RC6 (Rivest Cipher 6)

**Asymmetric encryption** is a type of encryption which uses mathematically linked public- and private-key pairs to encrypt and decrypt senders and recipients sensitive data. Some examples of symmetric encryption algorithms include:

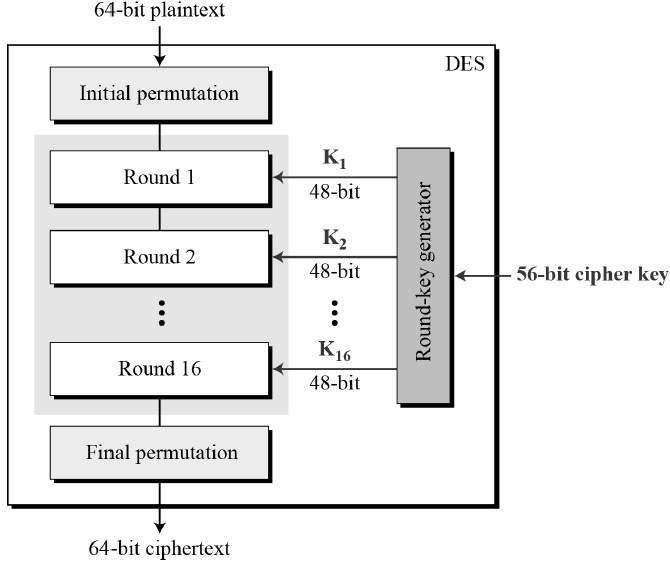
1. Rivest Shamir Adleman (RSA)
2. Digital Signature Standard (DSS)
3. Elliptical Curve Cryptography (ECC)
4. Diffie-Hellman exchange method
5. TLS/SSL protocol
6. What are the Cryptographic Best practices according to OWASP?

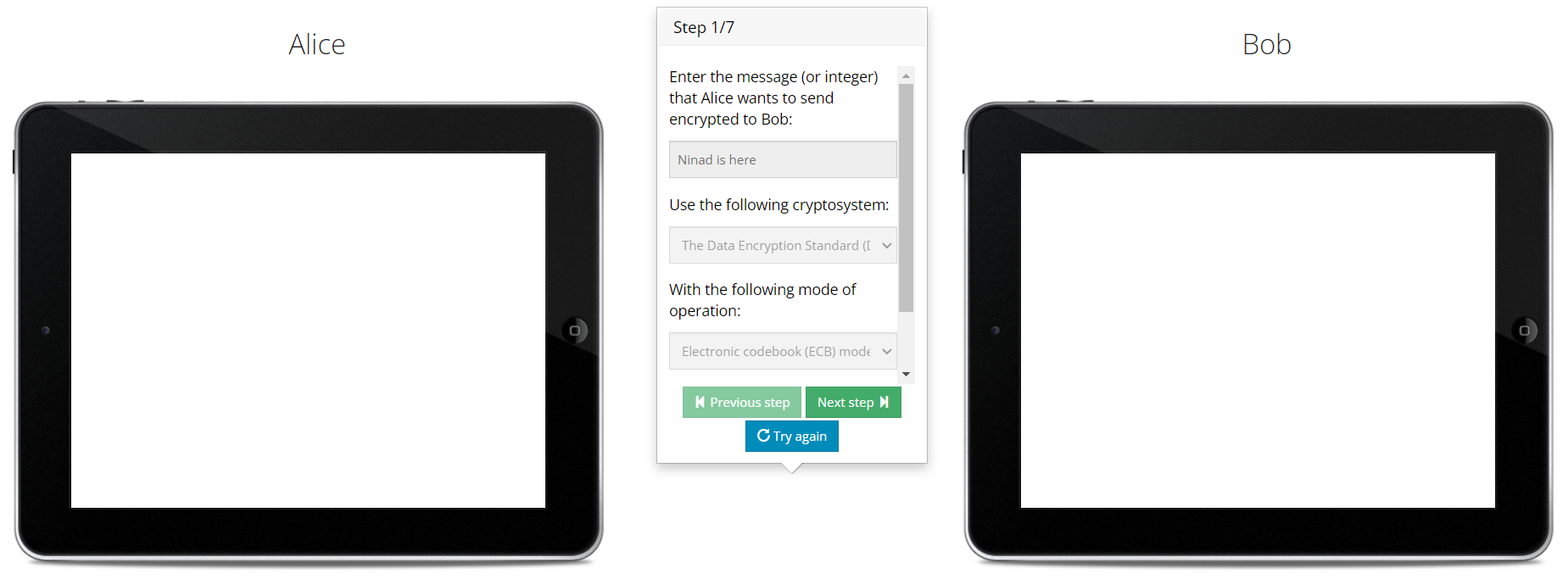
OWASP provides a **secure coding practices** checklist that includes 14 areas to consider in your software development life cycle. Of those secure coding practices, we’re going to focus on the top eight secure programming best practices to help you protect against vulnerabilities.

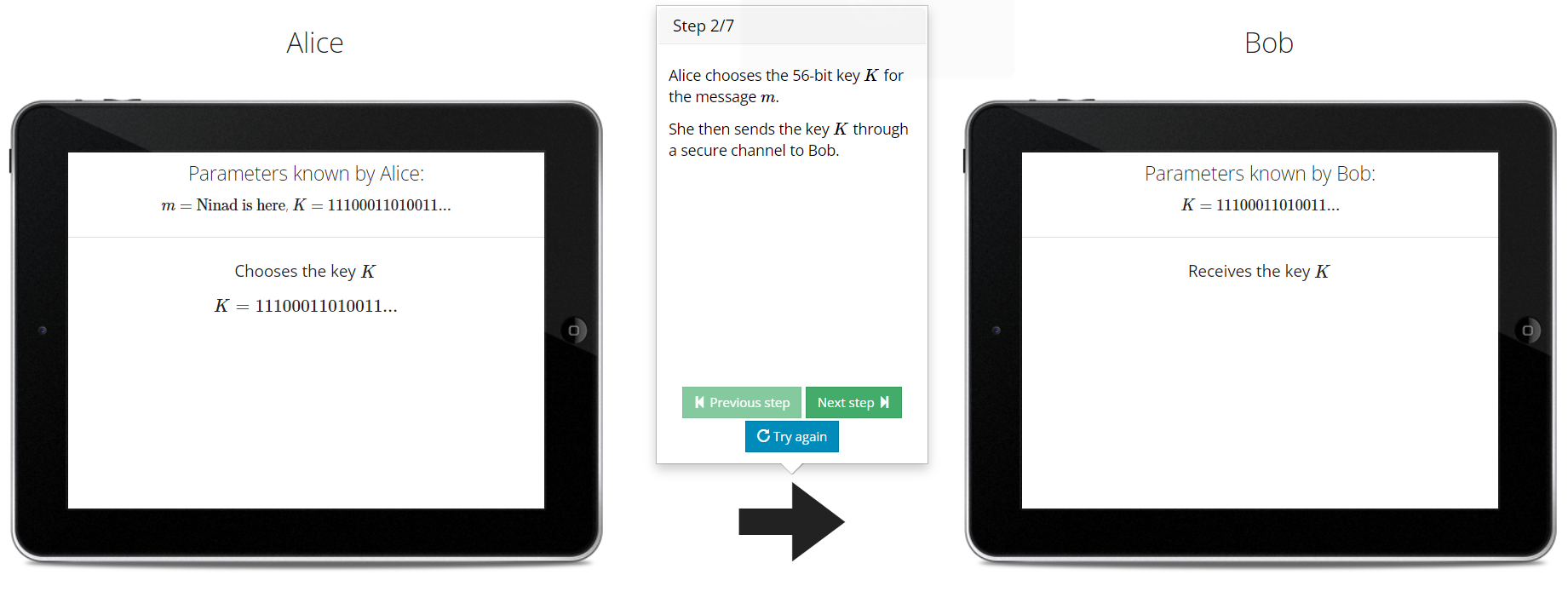
1. **Security by Design**: Security needs to be a priority as you develop code, not an afterthought. Organizations may have competing priorities where software engineering and coding are concerned.
2. **Password Management**: Passwords are a weak point in many software systems, which is why multi-factor authentication has become so widespread. Nevertheless, passwords are the most common security credential, and following secure coding practices limits risk.
3. **Access Control**: Take a “default deny” approach to sensitive data. Limit privileges and restrict access to secure data to only users who need it. Deny access to any user that cannot demonstrate authorization. Ensure that requests for sensitive information are checked to verify that the user is authorized to access it.
4. **Error Handling and Logging**: Software errors are often indicative of bugs, many of which cause vulnerabilities. Error handling and logging are two of the most useful techniques for minimizing their impact. Error handling attempts to catch errors in the code before they result in a catastrophic failure.
5. **System Configuration**: Clear your system of any unnecessary components and ensure all working software is updated with current versions and patches. If you work in multiple environments, make sure you’re managing your development and production environments securely.
6. **Threat Modeling**: Document, locate, address, and validate are the four steps to threat modeling. To securely code, you need to examine your software for areas susceptible to increased threats of attack.
7. **Cryptographic Practices**: Encrypting data with modern cryptographic algorithms and following secure key management best practices increases the security of your code in the event of a breach.
8. **Input Validation and Output Encoding**: These secure coding standards are self-explanatory in that you need to identify all data inputs and sources and validate those classified as untrusted. You should utilize a standard routine for output encoding and input validation.

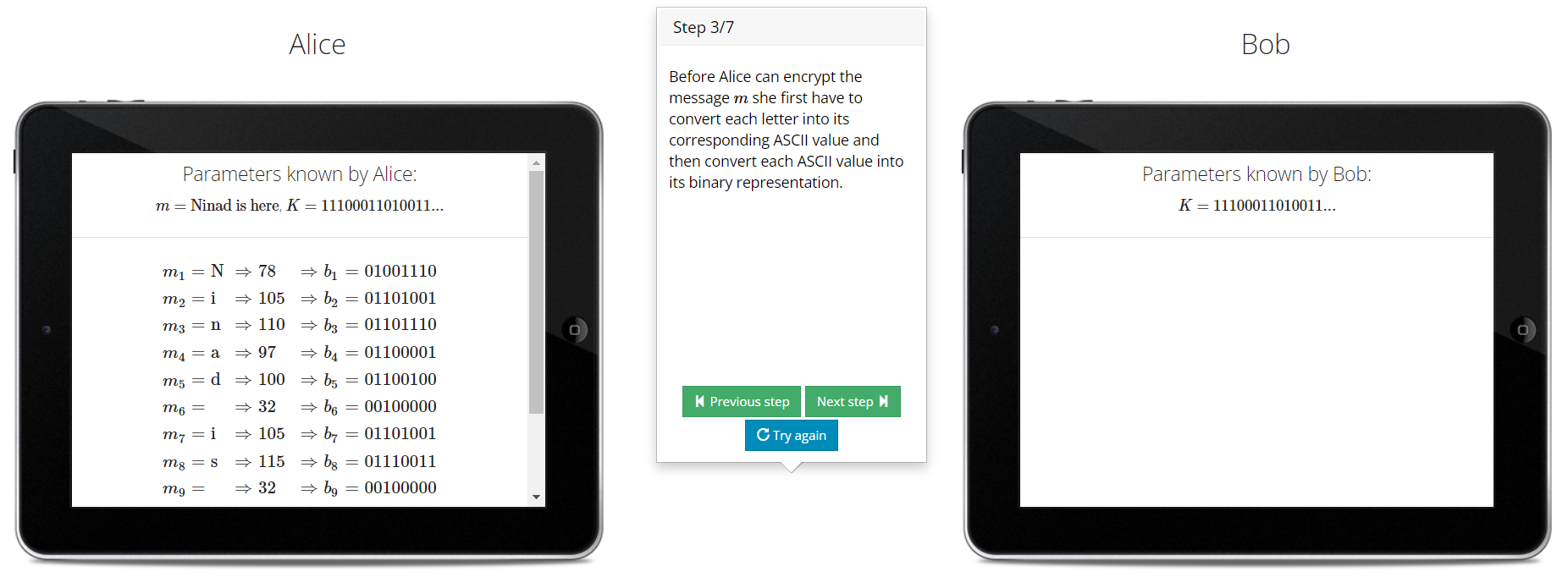
**Symmetric Cryptography Demonstration**

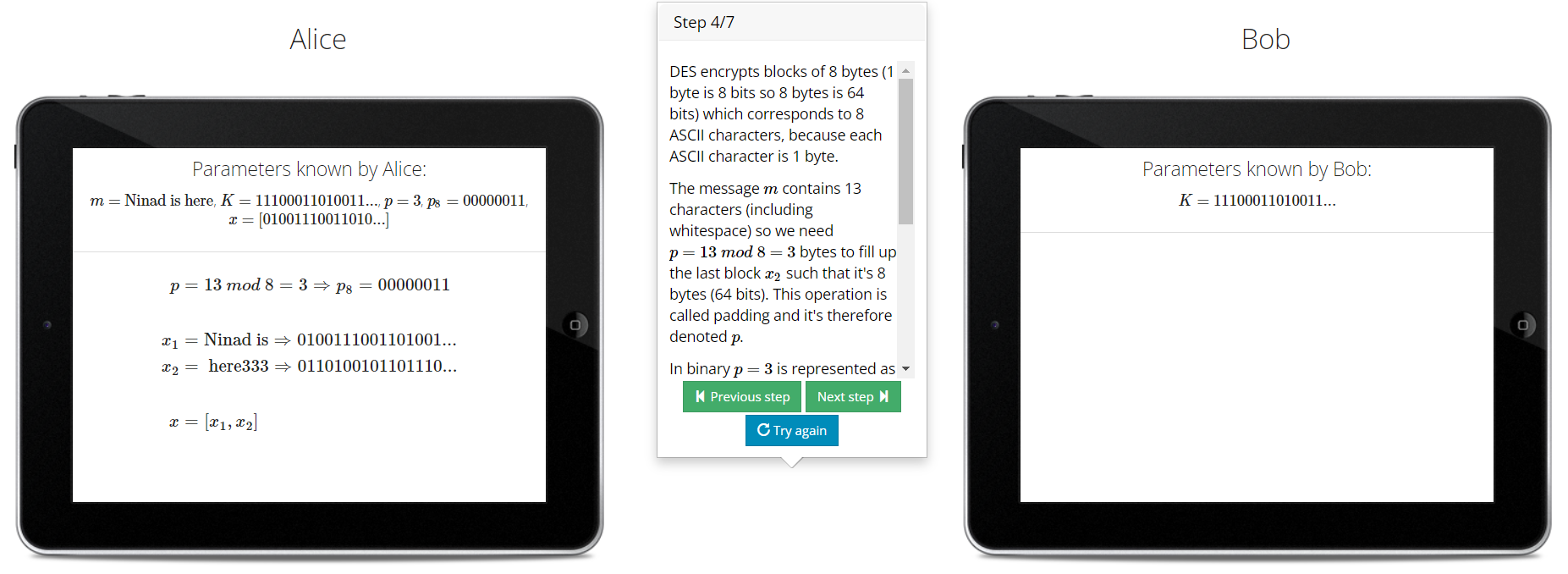
DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext.

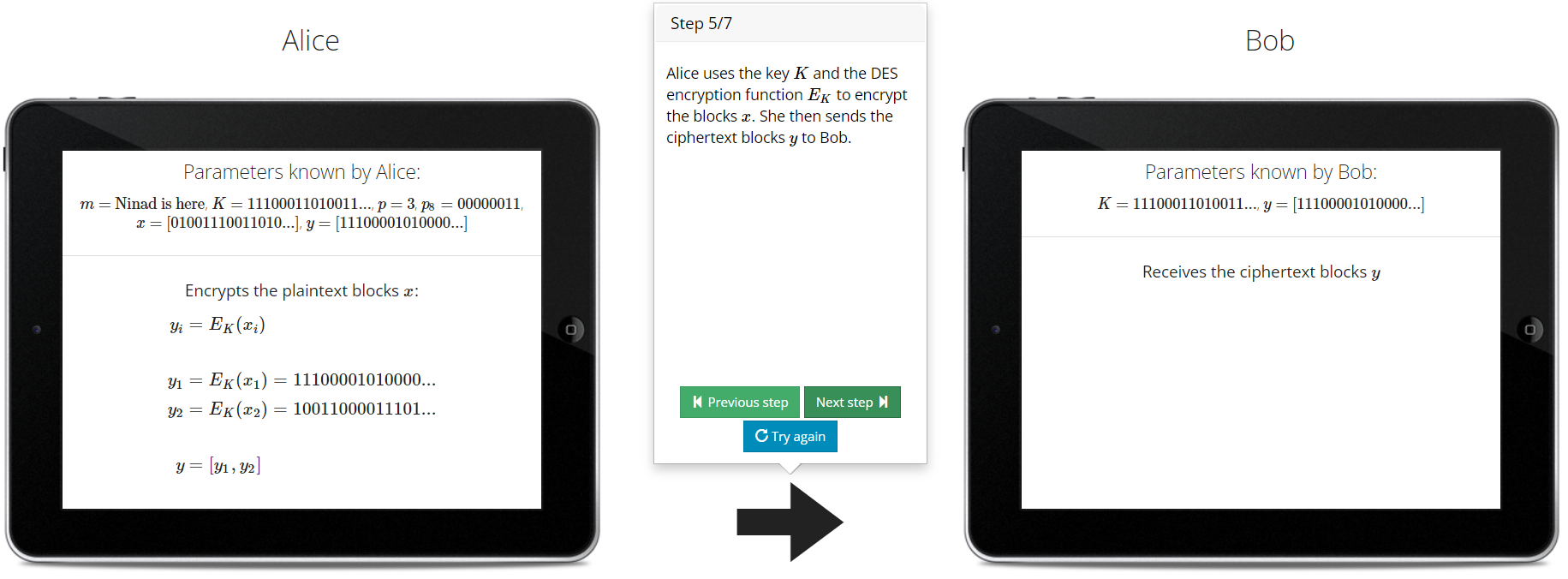


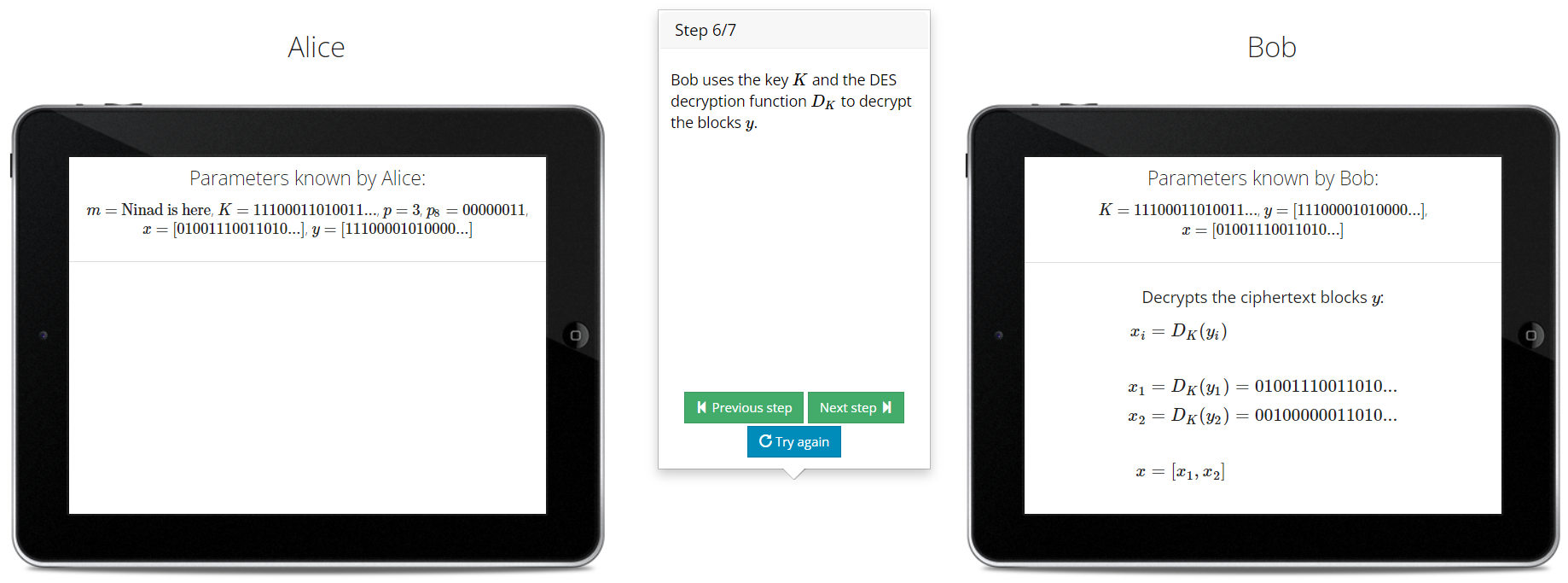
****

****

****

****

****

****

****

**Asymmetric Cryptography Demonstration**

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key.

**Step 1: Generate the RSA modulus**

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown: *N=p\*q*

Here, let N be the specified large number.

**Step 2: Derived Number (e)**

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

**Step 3: Public key**

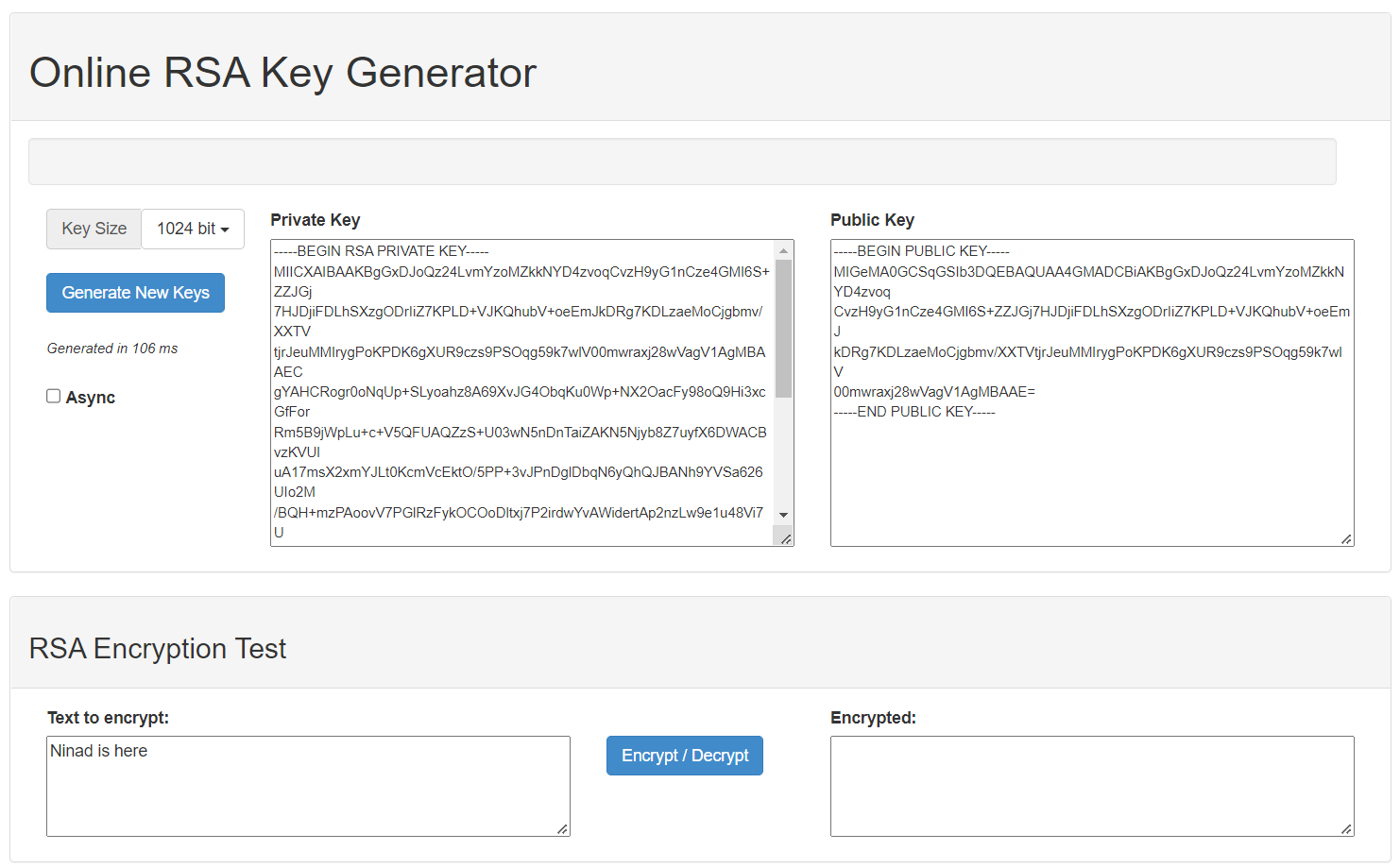
The specified pair of numbers n and e forms the RSA public key and it is made public.

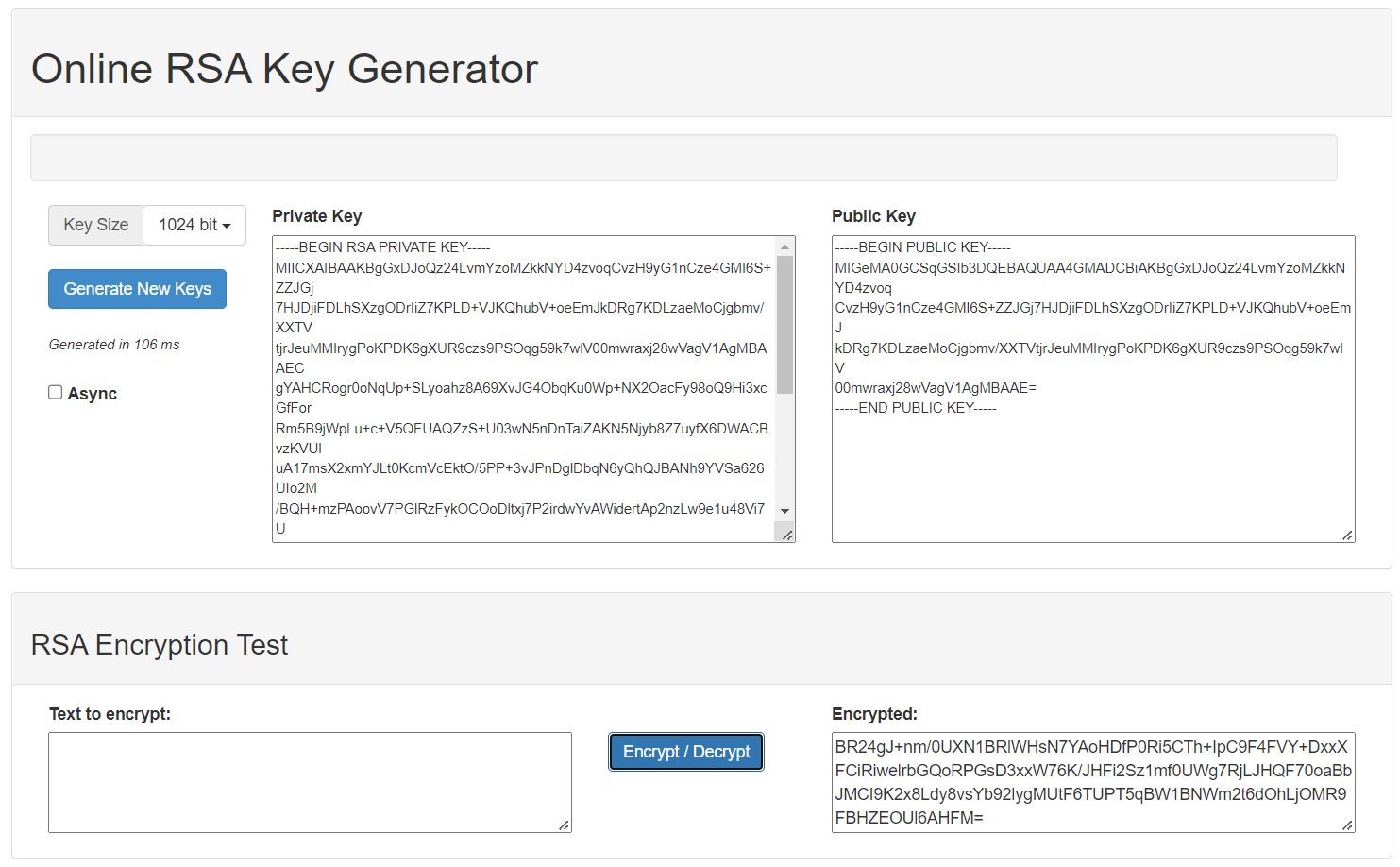
**Step 4: Private Key**

Private Key d is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows: *ed = 1 mod (p-1) (q-1)*

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

**Decryption Formula**: The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver C has the private key d, the result modulus will be calculated as *Plaintext = Cd mod n*





**Conclusion**:

Thus we have studied how to perform different types of Symmetric and Asymmetric cryptography.