

Crypto Tools

By: Momopranto Amin, Viola Rreza, Sean Wang

Cryptographic Protocol

- Alice encrypts her message using a GUI
 - can choose between multiple encryption schemes
 - GUI generates key for Alice
 - GUI returns ciphertext
- Alice sends ciphertext to Bob through text or email
 - Alice mentions encryption scheme choice and number of bits used to generate key
- Bob uses same GUI to decrypt Alice's message

Encryption Scheme Options

- Hash
 - MD5
 - SHA
 - SHA1,
SHA224,
SHA256,
SHA384,
SHA512
- Asymmetric
 - Diffie-Hellman
- Symmetric
 - AES
 - CTR, ECB,
OFB, CBC
 - DES

GUI

- take the user's input such as:
 - choice of mode
 - key (copied and pasted from key generator offered by GUI)
 - designated number of bits (for key generator)
 - plaintext message (for encrypt)
 - ciphertext message (for decrypt)

GUI

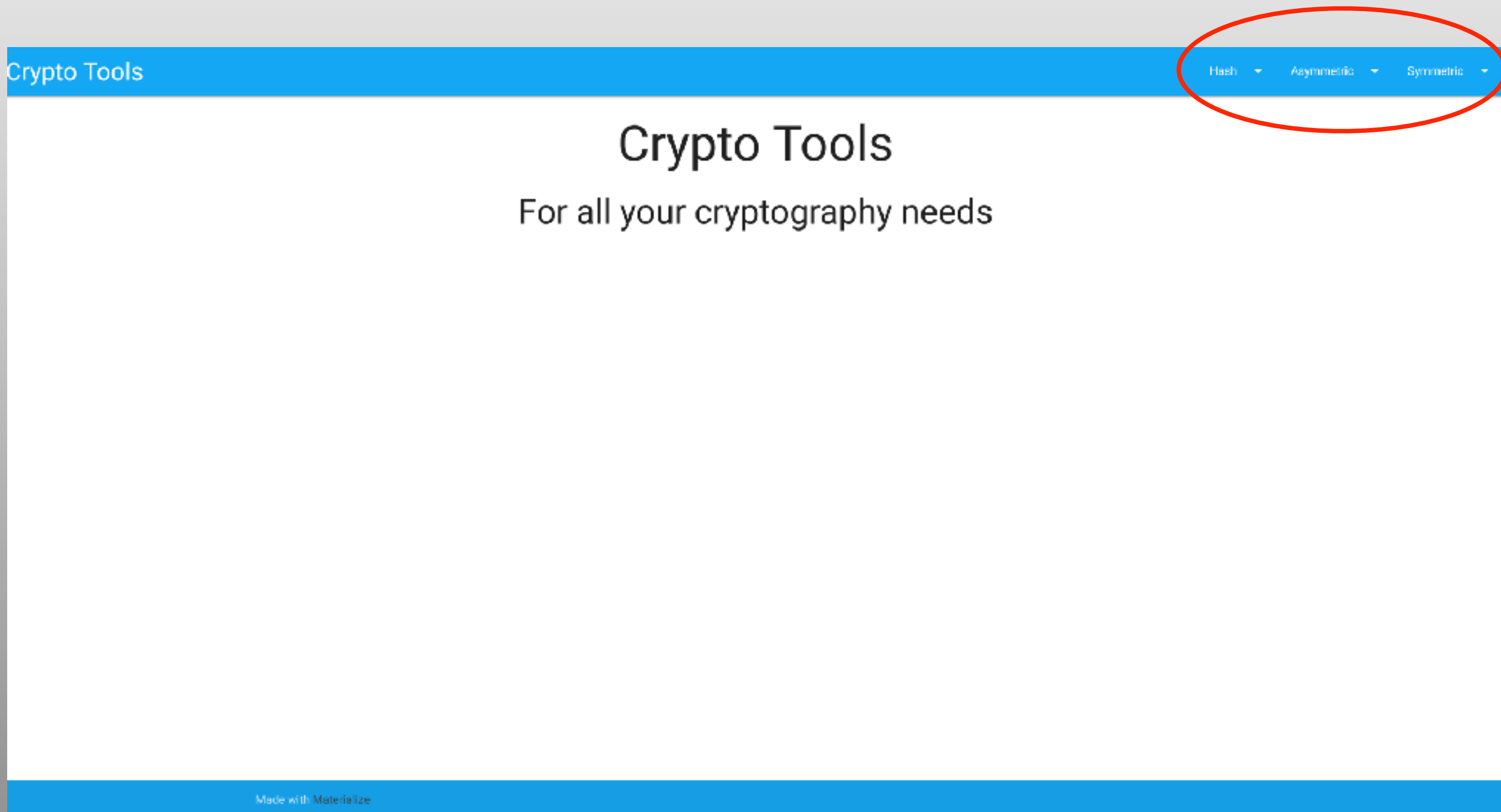
- process user's input
 - make calls to PyCrypto to obtain high-level functions along with their primitives
 - rendered output to front end
- incorporate back-end code into GUI functionality

```
from Crypto.Cipher import SHA
from Crypto.Cipher import SHA224
from Crypto.Cipher import SHA256
from Crypto.Cipher import SHA384
from Crypto.Cipher import SHA512

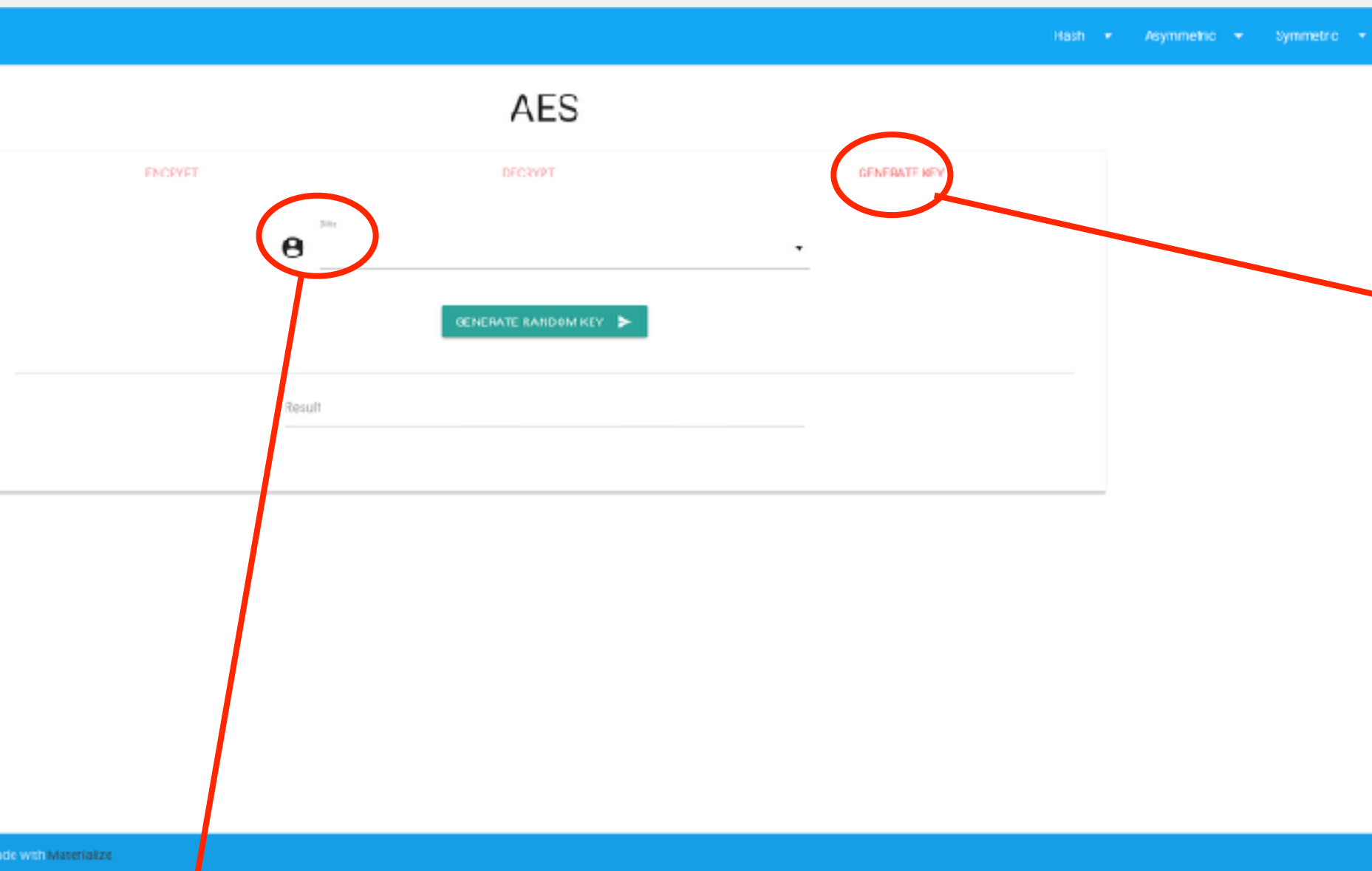
@app.route('/sha', method = ['GET','POST'])
def sha():
    option = request.form['mode']
    if request.message = 'POST':
        if (option == 'SHA'):
            message = request.form['message'].encode('UTF-8')
            return SHA.new(message).hexdigest()
        else if (option == 'SHA224'):
            message = request.form['message'].encode('UTF-8')
            return SHA224.new(message).hexdigest()
        else if (option == 'SHA256'):
            message = request.form['message'].encode('UTF-8')
            return SHA256.new(message).hexdigest()
        else if (option == 'SHA384'):
            message = request.form['message'].encode('UTF-8')
            return SHA384.new(message).hexdigest()
        else if (option == 'SHA512'):
            message = request.form['message'].encode('UTF-8')
            return SHA512.new(message).hexdigest()
    return render.template()
```

GUI

- Alice opens the GUI
 - options in the top right: Hash, Asymmetric, Symmetric



GUI



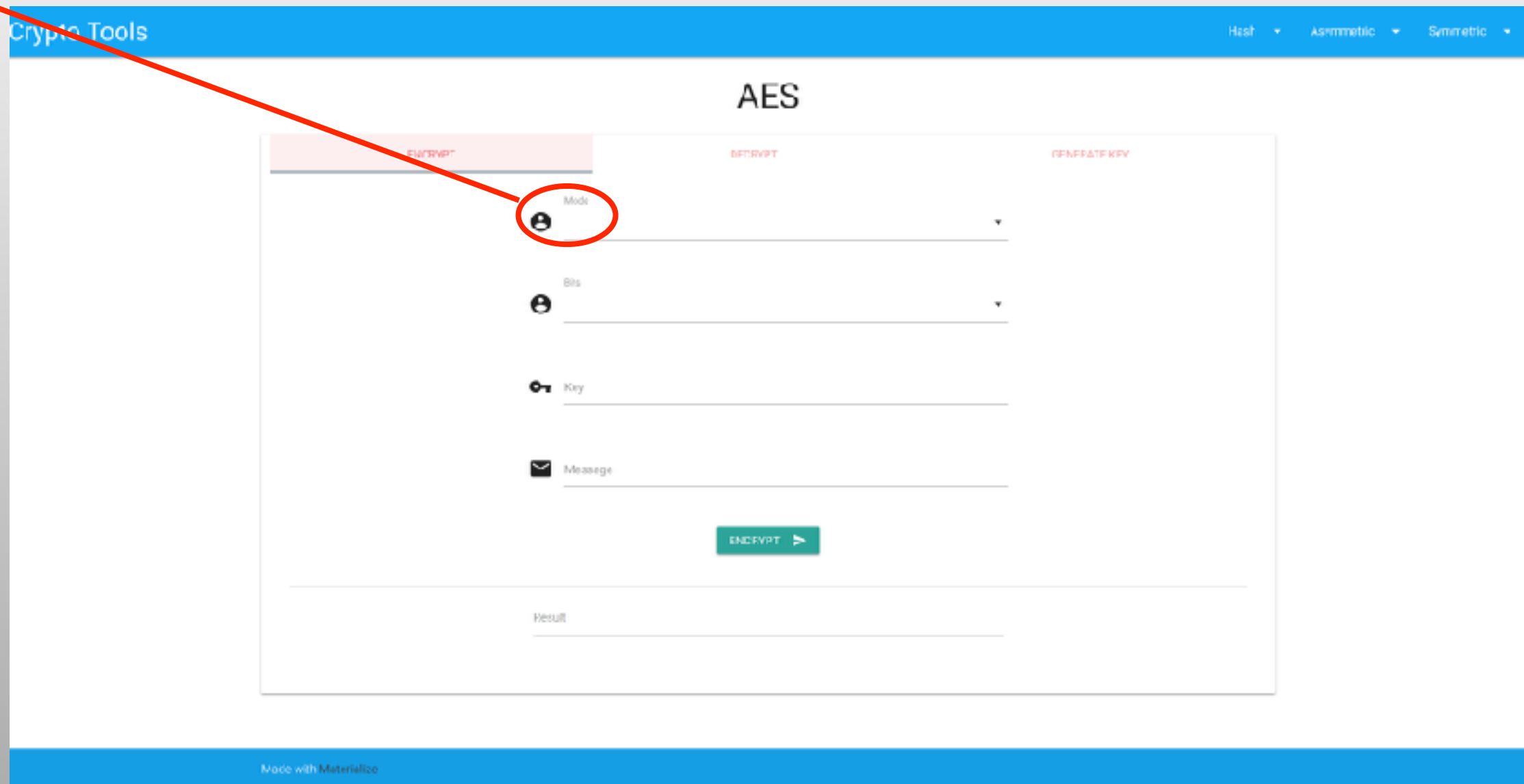
- Alice chooses AES under Symmetric

- clicks on “GENERATE KEY” tab

- chooses bits number then clicks “GENERATE RANDOM KEY” button
- copies the outputted key

GUI

- she then clicks on “ENCRYPT” tab
- chooses type of AES from “Mode” dropdown



The screenshot shows the 'Crypto Tools' website interface for AES encryption. The top navigation bar is blue with the text 'Crypto Tools' on the left and 'Hash', 'Asymmetric', and 'Symmetric' dropdown menus on the right. The main content area is titled 'AES' and features three tabs: 'ENCRYPT' (highlighted in pink), 'DECRYPT', and 'GENERATE KEY'. The 'ENCRYPT' tab is active, showing a form with four input fields: 'Mode' (with a dropdown arrow and a red circle around it), 'Bits' (with a dropdown arrow), 'Key' (with a key icon), and 'Message' (with an envelope icon). Below these fields is a green 'ENCRYPT' button with a right-pointing arrow. At the bottom of the form is a 'Result' label and an empty text area. The footer of the page is blue with the text 'Made with Materialize'.

- inputs same bits number she had chosen when generating key
- pastes generated key and writes out her message
- clicks “ENCRYPT” button

GUI

- Alice copies outputted ciphertext
- Alice texts Bob the ciphertext, which encryption scheme she used, and number of bits
- Bob copies ciphertext and opens GUI
- chooses correct scheme and clicks “DECRYPT” tab
- inputs necessary information
- is outputted the plaintext