2 Way Key Verify:
An Ultra-Lightweight Authentication Scheme

B.R.Y.X.[1], J.F. Ho[1], R. Balaji[1]

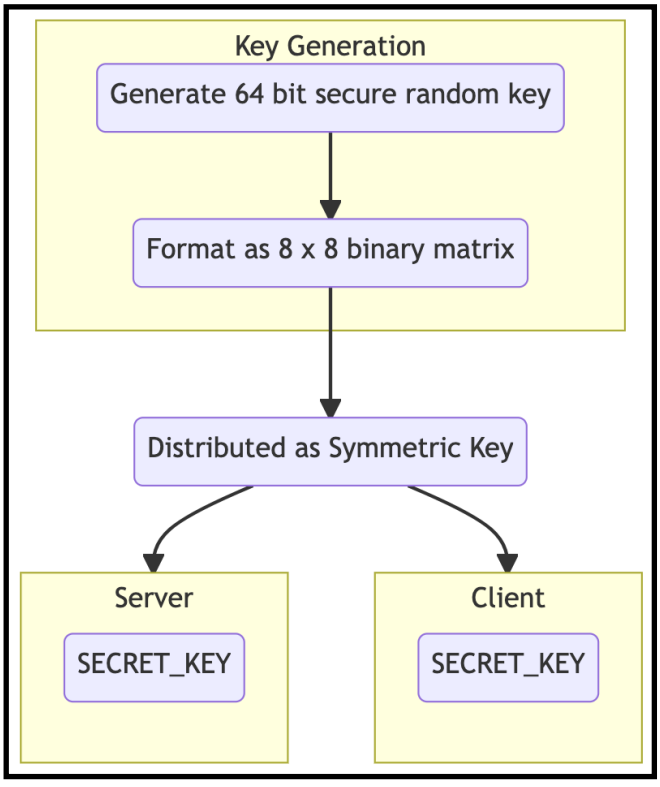
[1] Palindrome Research Labs, Singapore

1. Introduction

With the roll out of IOT devices in PALINDROME's botnet infrastructure and the need for fast and secure authentication has led to us developing a simple yet powerful method for 2 way proof of knowledge based key verification. In this paper, Palindrome Research Labs details the 2WKV scheme.

2. Key Generation

2WKV operates on the basis of symmetric key cryptography. A 64 bit key must be securely generated and distributed to both endpoints before the algorithm can perform any verification. This 64 bit key represents the 8×8 binary matrix that will be used to perform mathematical operations in GF(2) during the Key Verification procedures.



3. Key Verification

To verify the knowledge of the key without revealing the key to the other party, this scheme uses matrix multiplication in GF(2) as its proof of knowledge function.

Formally we can define a challenge and response vectors as having this relationship with the SECRET_KEY matrix:

$$\begin{bmatrix} SECRET_{11} & SECRET_{12} & \cdots & SECRET_{1n} \\ SECRET_{21} & SECRET_{22} & \cdots & SECRET_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ SECRET_{n1} & SECRET_{n2} & \cdots & SECRET_{nn} \end{bmatrix} \times \begin{bmatrix} challenge_1 \\ challenge_2 \\ \vdots \\ challenge_n \end{bmatrix} = \begin{bmatrix} response_1 \\ response_2 \\ \vdots \\ response_n \end{bmatrix}$$

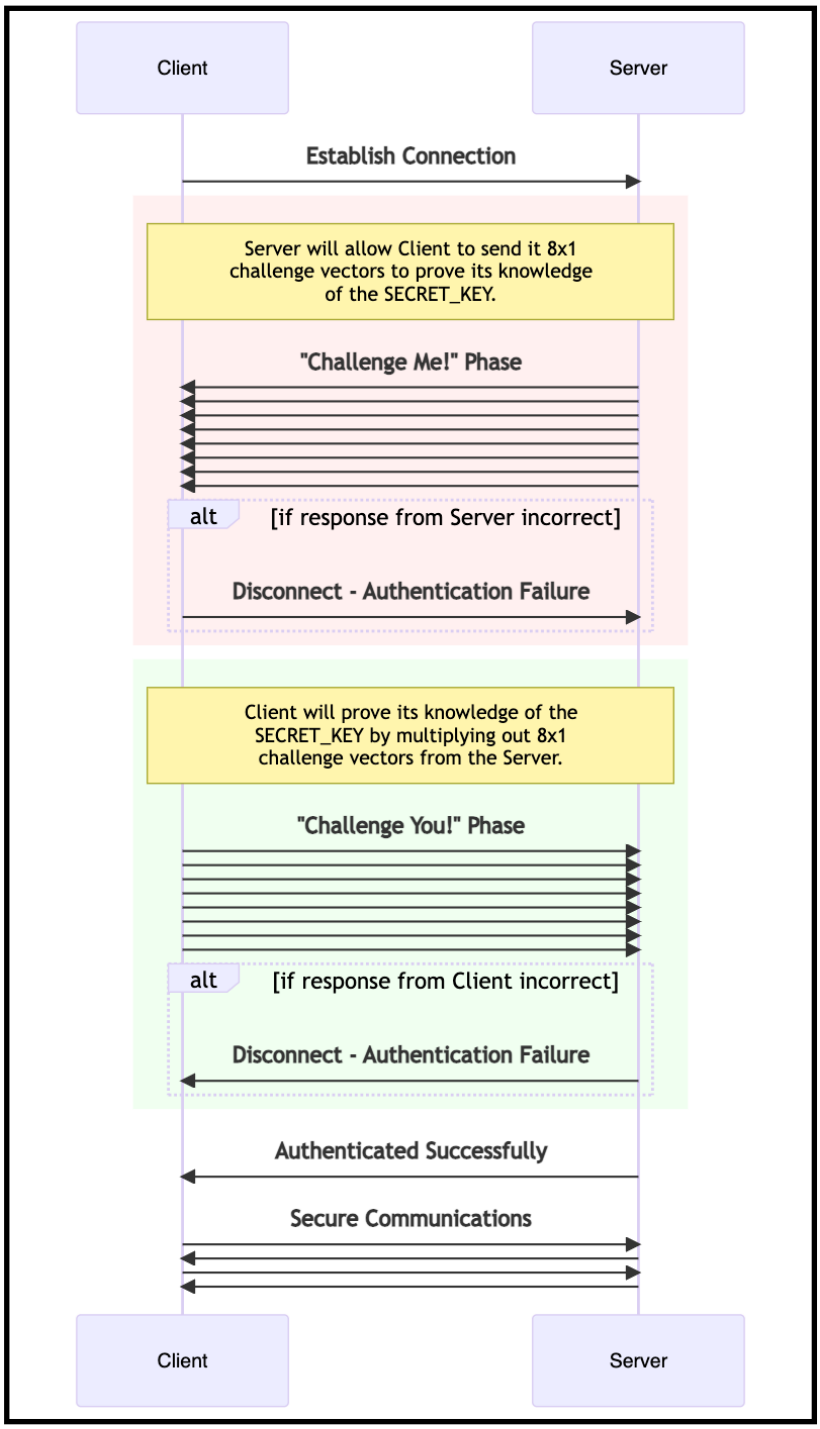
Do take note that all of these operations happen in GF(2).

4. Typical Use Case

Here we detail an example of the typical use case for our 2WKV scheme. There should be 2 phases of verification so that both the server and client can bidirectionally verify each other. In the event that either side receives an incorrect response vector to its challenge vector, that side should immediately stop communications.

If successful authentication is possible, both clients can begin secure data exchange.

In the next page, we also provide a real sample of an implementation we use in our live production infrastructure.



Banner String

Helper & Win Functions

SECRET KEY (Symmetric)

Client → Server Challenges

Server → Client Challenges

Authenticated!

files — bat main.py — bat — less ◂ bat main.py — 83×80

File: main.py
Size: 2.7 KB

```
import sys
import numpy as np

banner = """
 ::::::::        :::       :::     :::     :::   :::
:+:    :+:       :+:       :+:   :+: :+:   :+:   :+:
      +:+        +:+       +:+  +:+  +:+   +:+ +:+
    +#+          +#+  +:+  +#+ +#++:++#++:   +#++:
  +#+            +#+ +#+#+ +#+ +#+     +#+    +#+
 #+#              #+#+# #+#+#  #+#     #+#    #+#
##########         ###   ###   ###     ###    ###

:::    ::: :::::::::: :::   :::          :::               :::
:+:   :+:  :+:        :+:   :+:         :+:                :+:
+:+  +:+   +:+         +:+ +:+         +:+                 +:+
+#+++:+    +#++:++#     +#++:         +#+    +#++:++#++:++    +#+
+#+  +#+   +#+           +#+          +#+                    +#+
#+#   #+#  #+#           #+#          #+#                   #+#
###    ### ##########    ###          ###                ###

:::     ::: :::::::::: :::::::::  :::::::::: :::::::::: ::::    :::
:+:     :+: :+:        :+:    :+: :+:        :+:        :+:+:   :+:
+:+     +:+ +:+        +:+    +:+ +:+        +:+        :+:+:+ +:+
+#+     +:+ +#++:++#   +#++:++#:  +#+        :#::+::#    +#+ +:+
 +#+   +#+  +#+        +#+    +#+ +#+        +#+        +#+  +#+
  #+#+#+#   #+#        #+#    #+# #+#        #+#        #+#   #+#
    ###     ########## ###    ### ########## ########## ###    ###

"""

def sysout(fstr):
  sys.stdout.write(fstr)
  sys.stdout.flush()

def prompt(fstr):
  guards = "=" * len(fstr)
  sysout(f"{guards}\n{fstr}\n{guards}\n")

def vectostr(v):
  return "".join(map(str, v.reshape(-1)))

def strtovec(s, rows=8, cols=1):
  return np.fromiter(list(s), dtype="int").reshape(rows, cols)

def win():
  flag = open("/flag.txt").read().strip()
  prompt(f"Here is your flag: {flag}")

SECRET_KEY = np.round(np.random.rand(8, 8)).astype("int")

if __name__ == "__main__":
  sysout(banner)

  prompt("Challenge Me!")
  for i in range(8):
    input_vec = input(f"Challenge Me #{i+1:02} <-- ")
    assert len(input_vec) == 8
    assert input_vec.count("1") + input_vec.count("0") == 8
    input_vec = strtovec(input_vec)
    output_vec = (SECRET_KEY @ input_vec) & 1
    sysout(f"My Response --> {vectostr(output_vec)}\n")

  prompt("Challenge You!")
  for i in range(8):
    input_vec = np.round(np.random.rand(8, 1)).astype("int")
    sysout(f"Challenge You #{i+1:02} --> {vectostr(input_vec)}\n")
    test_vec = input(f"Your Response <-- ")
    assert len(test_vec) == 8
    assert test_vec.count("1") + test_vec.count("0") == 8
    test_vec = strtovec(test_vec)
    answer_vec = (SECRET_KEY @ input_vec) & 1
    assert (answer_vec == test_vec).all()

  prompt("All challenges passed :)")
  win()
```