This code provides a complete setup for using environment variables in PHP with a .env file.

Here's a breakdown of the steps:

- Create a .env file in your project root with your sensitive information.
- Create a config.php file that includes a function to load the .env file and set the variables in the environment.
- Create specific functions to retrieve different types of keys (database, login, reCAPTCHA).
- Use these functions in your application to access the sensitive information.

To use this in your project:

- Save the .env file in your project root and add your actual keys.
- Save the PHP code in a file named config.php.
- Include config.php at the beginning of your PHP scripts that need access to these variables.

Remember to add .env to your .gitignore file to prevent it from being committed to version control.

Step 1: Project Structure

First, create the following file structure:

```
/your_project_root
├── .env
├── config.php
├── db.php
├── functions.php
├── login.php
└── index.php
```

Step 2: Set up the .env file

Create a .env file in your project root with the following content:

```
DB_HOST=localhost
DB_NAME=your_database_name
DB_USER=your_database_user
DB_PASS=your_database_password
RECAPTCHA_SITE_KEY=your_recaptcha_site_key
```

RECAPTCHA_SECRET_KEY=your_recaptcha_secret_key
Replace the values with your actual database and reCAPTCHA credentials.

Step 3: Create config.php

This file will load the environment variables:

```
<?php
declare(strict_types=1);

function loadEnv(string $path): void {
    $lines = file($path, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    foreach ($lines as $line) {
        if (strpos(trim($line), '#') === 0) {
            continue;
        }
        list($name, $value) = explode('=', $line, 2);
        $name = trim($name);
        $value = trim($value);
        putenv(sprintf('%s=%s', $name, $value));
        $_ENV[$name] = $value;
        $_SERVER[$name] = $value;
    }
}

// Load .env file
loadEnv(__DIR__ . '/.env');
```

Step 4: Create db.php

This file will handle database connections:

```
<?php
declare(strict_types=1);

function getDbConnection(): PDO {
    $host = getenv('DB_HOST');
    $db   = getenv('DB_NAME');
    $user = getenv('DB_USER');
    $pass = getenv('DB_PASS');
    $charset = 'utf8mb4';

    $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
    $options = [
```

```
        PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES   => false,
    ];
    try {
        return new PDO($dsn, $user, $pass, $options);
    } catch (\PDOException $e) {
        throw new \PDOException($e->getMessage(), (int)$e->getCode());
    }
}
```

Step 5: Create functions.php

This file will contain utility functions:

```
<?php
declare(strict_types=1);

require_once 'db.php';

function verifyReCaptcha(string $recaptchaResponse): bool {
    $secret = getenv('RECAPTCHA_SECRET_KEY');
    $verifyResponse =
file_get_contents('https://www.google.com/recaptcha/api/siteverify?secret='.$secret.'&response
='.$recaptchaResponse);
    $responseData = json_decode($verifyResponse);
    return $responseData->success;
}

function login(string $username, string $password): bool {
    $pdo = getDbConnection();
    $stmt = $pdo->prepare('SELECT id, password FROM users WHERE username =
:username');
    $stmt->execute(['username' => $username]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];
        return true;
    }
    return false;
}
```

Step 6: Create login.php

This file will handle the login form and process:

```
<?php
declare(strict_types=1);

require_once 'config.php';
require_once 'functions.php';

session_start();

$error = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $recaptchaResponse = $_POST['g-recaptcha-response'] ?? '';
    if (verifyReCaptcha($recaptchaResponse)) {
        $username = $_POST['username'] ?? '';
        $password = $_POST['password'] ?? '';
        if (login($username, $password)) {
            header('Location: index.php');
            exit;
        } else {
            $error = "Invalid username or password";
        }
    } else {
        $error = "reCAPTCHA verification failed";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <script src="https://www.google.com/recaptcha/api.js" async defer></script>
</head>
<body>
    <h2>Login</h2>
    <?php if ($error): ?>
        <p style="color: red;"><?php echo htmlspecialchars($error); ?></p>
    <?php endif; ?>
```

```
    <form method="post">
        <div>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <div class="g-recaptcha" data-sitekey="<?php echo
htmlspecialchars(getenv('RECAPTCHA_SITE_KEY')); ?>"></div>
        <div>
            <button type="submit">Login</button>
        </div>
    </form>
</body>
</html>
```

Step 7: Create index.php

This will be the protected page that users see after logging in:

```
<?php
declare(strict_types=1);

session_start();

if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit;
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
</head>
<body>
    <h1>Welcome, you are logged in!</h1>
    <p>This is a protected page.</p>
    <a href="logout.php">Logout</a>
```

```
</body>
</html>
```

Step 8: Set up the database

Create a users table in your database:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

-- Insert a test user (password is 'password123')

INSERT INTO users (username, password) VALUES ('testuser',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi');
```

Step 9: Final steps

1. Make sure you have the PDO MySQL driver installed and enabled in your PHP configuration.
2. Ensure your web server is set up to serve PHP files.
3. Navigate to login.php in your web browser to test the login system.

Remember to enhance this system with features like CSRF protection, input sanitization, proper session management, secure password reset functionality, and a user registration system for a production environment.