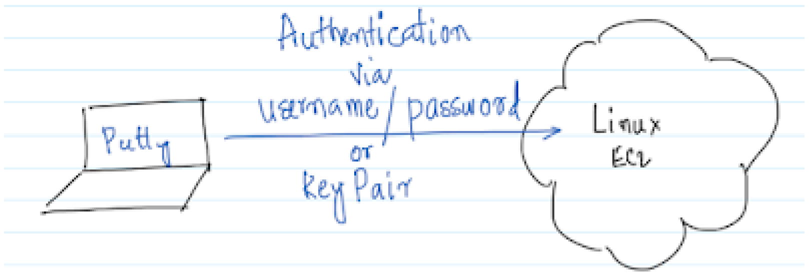
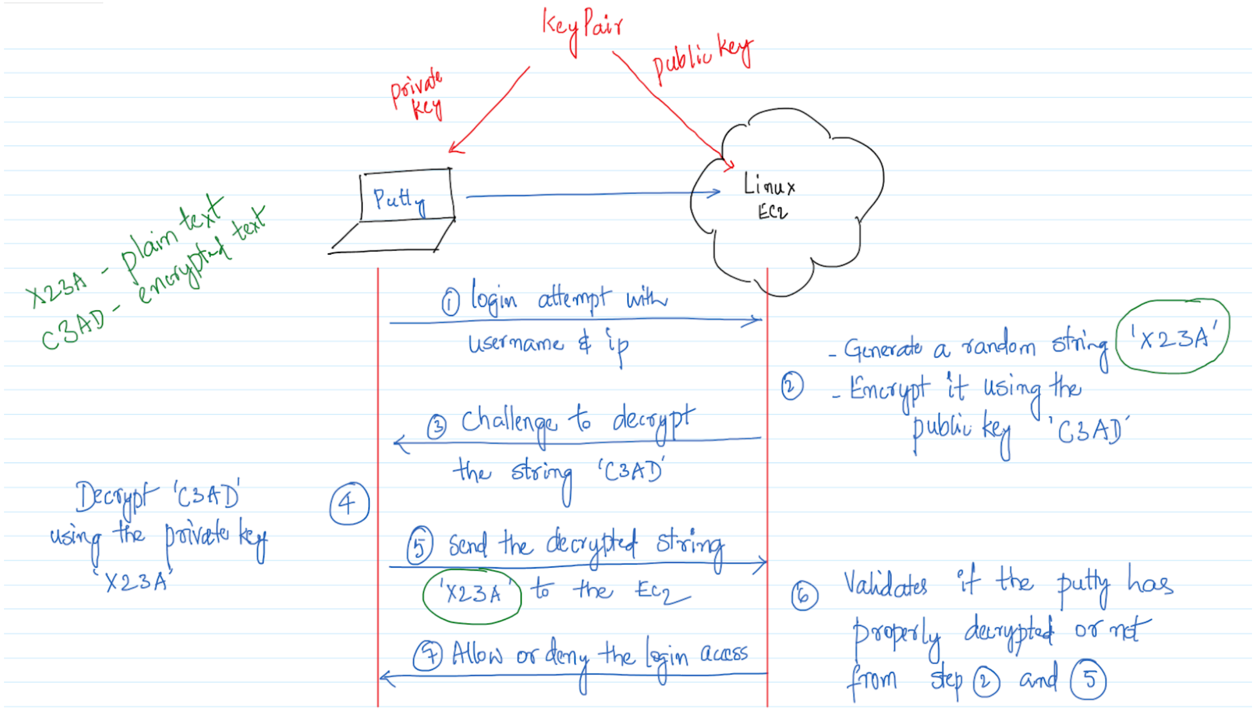
How does Key Pair work behind the scenes for Linux EC2 authentication

Different ways of authenticating against Linux EC2

Once a Linux EC2 instance has been created, the same can be accessed via Putty or some other SSH client. To access the EC2, first we need to authenticate the user. Either the Username/Password or the KeyPair can be used for authentication. There are pros and cons of each of them. AWS has chosen to go with the KeyPair way of authentication by default. If required this can be disabled and the Username/Password can be enabled.



How does KeyPair work behind the scenes for Linux EC2 authentication


Key Pair
private key
public key
Putty
Linux EC2
X23A - plain text
C3AD - encrypted text
① login attempt with username & ip
② - Generate a random string 'X23A' - Encrypt it using the public key 'C3AD'
③ Challenge to decrypt the string 'C3AD'
④ Decrypt 'C3AD' using the private key 'X23A'
⑤ Send the decrypted string 'X23A' to the EC2
⑥ Validates if the putty has properly decrypted or not from step ② and ⑤
⑦ Allow or deny the login access

A picture conveys more than words, so is the above workflow. A KeyPair consists of a Private Key and a Public Key. The Private Key goes onto the Laptop and the Public Key automatically goes into the EC2 instance. Go through the above workflow to get to know what happens behind the scenes.

Note that the Private Key never leaves the laptop, this is one of the advantages of using the Key Pairs. Also, the way we never share the passwords with anyone, we should never share the Private Key with anyone. This would allow them to access the EC2. Also, the way we never use the same password across multiple services, never we should use the same Key Pair across multiple EC2 instances for the obvious reasons.

Also, it's always better to create a different set of Key Pairs for multiple users accessing the same EC2 instance, very similar to different passwords.

we will write a Lambda function that will create an EC2 instance. This Lambda function will be written in Python using the Boto3 library. We will also create a custom Lambda execution policy for our IAM role. When we’re done, we will be able to log in to the new EC2 instance via SSH.

Create a Lambda Function

1. Navigate to Lambda.
2. Click **Create a function**.
3. Choose **Author from scratch** and use the following settings:
    - *Name*: **CreateEC2**
    - *Runtime*: **Python 3.7**
    - *Role*: **Create a custom role**
4. Expand *Choose or create an execution role*.
5. Set *Execution role* to **Create a new role with basic Lambda permissions**.
6. Copy the execution role name that appears.
7. Click **Create function**.
8. Navigate to IAM.
9. Search for and select your newly created role.

10. Edit the policy to replace its existing policy with this file on GitHub.

11. Back in the Lambda console, scroll to the *Function code* section and paste in the Python source code from this file on GitHub.

IAM Role

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "ec2:RunInstances"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Python 3.7

```
import os
import boto3

AMI = os.environ['AMI']
INSTANCE_TYPE = os.environ['INSTANCE_TYPE']
KEY_NAME = os.environ['KEY_NAME']
SUBNET_ID = os.environ['SUBNET_ID']

ec2 = boto3.resource('ec2')


def lambda_handler(event, context):

    instance = ec2.create_instances(
        ImageId=AMI,
        InstanceType=INSTANCE_TYPE,
        KeyName=KEY_NAME,
        SubnetId=SUBNET_ID,
        MaxCount=1,
        MinCount=1
    )

    print("New instance created:", instance[0].id)
```

1. Set four environment variables:
    - `AMI`: The `ami-` value of an Amazon Linux 2 instance
    - `INSTANCE_TYPE`: t2.micro
    - `KEY_NAME`: The name of your EC2 key pair
    - `SUBNET_ID`: The ID of one of the public subnets in your VPC
2. Save the Lambda function.

Test Lambda Function

1. Click **Test**.
2. Define an empty test event. Its contents can simply be `{}`.
3. Give it any name you like.
4. Click **Create**.
5. Click **Test** again for a second test.
6. Observe that an EC2 instance is initializing.