**Option 2 — Put script in /system/etc/init.d (bypass /data)**

Since /system mounted fine for you, we can use it instead of /data.

From TWRP shell:

sh

CopyEdit

mkdir -p /system/etc/init.d

cat > /system/etc/init.d/99adb <<'EOF'

#!/system/bin/sh

setprop ro.adb.secure 0

setprop ro.secure 0

setprop service.adb.tcp.port 5555

stop adbd

start adbd

EOF

chmod +x /system/etc/init.d/99adb

**Note:** This only works if your ROM supports init.d scripts (many custom ROMs do, but stock Samsung firmware usually does **not**).

# Other method

Got it! Here’s a **clear step-by-step** to patch default.prop or build.prop in TWRP to disable ADB authorization on your J7 Prime:

**Step 1: Boot into TWRP**

* Power off your phone (hold Power+VolDown+Home 7–10 seconds if needed)
* Then boot into TWRP Recovery with **Power + Volume Up + Home** until it vibrates.

**Step 2: Connect phone to PC and open shell**

On your PC command prompt:

adb devices

You should see your device as recovery.

Now open a shell:

adb shell

**Step 3: Mount the system partition**

Inside the shell:

mount /system

**Step 4: Check if /system/default.prop exists**

Run:

ls /system/default.prop

If it shows the file, use it. Otherwise, check:

ls /system/build.prop

Usually Samsung uses build.prop.

**Step 5: Edit the file (we’ll use vi or sed)**

Try to edit with vi (if installed):

vi /system/default.prop

or

vi /system/build.prop

**Step 6: In vi editor, find the lines:**

ro.adb.secure=1

ro.secure=1

**Step 7: Change those lines to:**

ro.adb.secure=0

ro.secure=0

**Step 8: Save and exit vi**

* Press ESC
* Type :wq
* Press Enter

**Step 9: Reboot the phone**

Exit the shell:

exit

Then reboot:

adb reboot

**Step 10: Connect via ADB normally**

Wait for Android to boot. Then on PC:

adb devices

Your device should now show as device (authorized) instead of unauthorized.

**Notes:**

* If you don’t have vi or it’s hard to use, I can give you a sed command to do it automatically.
* If the file is read-only, you may need to remount with write permission:

mount -o remount,rw /system

before editing.

Want me to give you the exact sed command for quick patching instead of manual editing?

# New one

The “unauthorized” status means your phone still hasn’t accepted your PC’s ADB key, so it refuses to connect — and since your screen/touch are dead, you can’t tap “Allow” on the popup.

**Why this happens:**

* Samsung’s stock ROM usually enforces ADB authorization regardless of those property tweaks.
* The Magisk module or property changes sometimes don’t fully bypass this on Samsung devices.
* Without screen/touch, you cannot accept the PC key dialog.

**What you can do next:**

**1. Use TWRP to pull and manually add your PC’s adbkey to the device**

* Your PC’s adb key is at:  
  C:\Users\<YourName>\.android\adbkey.pub
* Copy your adb key to the device’s /data/misc/adb/adb\_keys file so your PC is automatically authorized.

**Step-by-step to add adb key manually via TWRP shell:**

1. Open adb shell in TWRP (recovery mode):

adb shell

1. Mount data partition with write access:

mount /data

1. Check if adb\_keys file exists:

ls -l /data/misc/adb/adb\_keys

1. If it exists, back it up:

cp /data/misc/adb/adb\_keys /data/misc/adb/adb\_keys.bak

1. Exit adb shell and push your PC’s adbkey.pub to the device:

adb push C:\Users\ACER-WIN10\.android\adbkey.pub /sdcard/adbkey.pub

1. Back to adb shell:

adb shell

1. Remount data partition (if not already):

mount /data

1. Append the adbkey.pub content to adb\_keys:

cat /sdcard/adbkey.pub >> /data/misc/adb/adb\_keys

1. Fix permissions:

chmod 600 /data/misc/adb/adb\_keys

chown root:shell /data/misc/adb/adb\_keys

1. Reboot your phone normally:

reboot

**Now try adb devices again. Your PC should be authorized.**

If you want, I can help you with exact commands based on your environment — just let me know!

# New none1\_1

Alright — let’s walk through **Option 2** from the very beginning so you can try it without missing any steps.

We’ll basically **store your PC’s ADB key in /system/etc/adb/adb\_keys**, which older Samsung builds sometimes accept.

**Step-by-step**

**1️⃣ Get your PC’s public ADB key**

On your PC:

* Open this file in Notepad:

C:\Users\<YourWindowsUser>\.android\adbkey.pub

* Leave it open — we’ll push it to your phone in the next step.

**2️⃣ Boot phone into TWRP recovery**

* Power off the phone.
* Boot into TWRP: **Power + Volume Up + Home**.

**3️⃣ Enable ADB in TWRP**

In TWRP, ADB is usually enabled automatically — verify by:

adb devices

It should show:

xxxxxxxx recovery

**4️⃣ Push your PC’s adbkey.pub into /sdcard**

On your PC terminal:

adb push "C:\Users\<YourWindowsUser>\.android\adbkey.pub" /sdcard/adbkey.pub

Replace <YourWindowsUser> with your actual Windows username.  
Example:

adb push "C:\Users\ACER-WIN10\.android\adbkey.pub" /sdcard/adbkey.pub

**5️⃣ Enter ADB shell in TWRP**

adb shell

**6️⃣ Mount /system partition**

mount /system

(If you get an error, try mount -o rw,remount /system instead.)

**7️⃣ Create adb folder in /system/etc**

mkdir -p /system/etc/adb

**8️⃣ Append your key to adb\_keys**

cat /sdcard/adbkey.pub >> /system/etc/adb/adb\_keys

**9️⃣ Fix permissions and ownership**

chmod 644 /system/etc/adb/adb\_keys

chown root:root /system/etc/adb/adb\_keys

**🔟 Exit shell and reboot**

exit

adb reboot

When Android boots, run:

adb devices

If your ROM honors /system/etc/adb/adb\_keys, it should now show:

xxxxxxxx device

(no unauthorized message).

If this still doesn’t authorize, that means your Android build **only checks /data/misc/adb/adb\_keys**, so we’ll have to either decrypt /data or use a patched recovery/boot image.

Do you want me to also **add the mount -o rw,remount** and SELinux relabel commands just in case your /system is read-only?  
That will make this method more likely to work.

# Wipted daqta

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb devices

List of devices attached

33002dfea0af64ed recovery

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell twrp format data

E:Unrecognized script command: 'format'

Done processing script file

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell.mkfs.ext4 /dev/block/mmcblk0p25

adb.exe: unknown command shell.mkfs.ext4

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell mkfs.ext4 /dev/block/mmcblk0p25

/sbin/sh: mkfs.ext4: not found

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell mkfs.ext4 /dev/block/mmcblk0p25

/sbin/sh: mkfs.ext4: not found

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>ls

'ls' is not recognized as an internal or external command,

operable program or batch file.

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell twrp format data

E:Unrecognized script command: 'format'

Done processing script file

C:\Users\ACER-WIN10\Downloads\scrcpy-win64-v3.3.1>adb shell

make\_ext4fs /dev/block/mmcblk0p25

~ # make\_ext4fs /dev/block/mmcblk0p25

Creating filesystem with parameters:

Size: 27577548800

Block size: 4096

Blocks per group: 32768

Inodes per group: 8176

Inode size: 256

Journal blocks: 32768

Label:

Blocks: 6732800

Block groups: 206

Reserved block group size: 1024

Created filesystem with 11/1684256 inodes and 149746/6732800 blocks

~ # mount /data

~ # mount | grep "/data"

/dev/block/mmcblk0p25 on /data type ext4 (rw,seclabel,relatime,data=ordered)

# Other

I see exactly what’s happening — /data is mounted, but your system doesn’t have the /data/misc/adb/ directory at all, so the key file has nowhere to go.  
That means the system is either:

* **Freshly formatted** but hasn’t booted Android yet to create the folder structure, or
* You’re in a minimal recovery environment where /data is just a bare ext4 volume.

We can fix this by **manually creating the folder structure** before placing your adb\_keys.

Here’s what to do step-by-step in your current shell:

adb shell

mount /data

mkdir -p /data/misc/adb

cat /sdcard/adbkey.pub > /data/misc/adb/adb\_keys

chmod 600 /data/misc/adb/adb\_keys

chown root:shell /data/misc/adb/adb\_keys

exit

adb reboot

**Why this will work:**  
mkdir -p /data/misc/adb creates the exact folder path Android uses for ADB key storage.  
Then we write your public key there, set correct permissions, and reboot so Android will recognize it.

⚠️ **Important:** When Android boots, it must keep /data intact — if you still boot into recovery instead of Android, the key won’t be used.

Do you want me to also give you a **backup trick using /system/etc/adb/adb\_keys** so even if /data gets wiped again, your key will still be trusted? That would make it persistent.