10 Forget Not the Humble Timing Attack

by Colin O'Flynn

Judge not your neighbour's creation, as you know not under what circumstances they were created. And as we exploit the creations of those less fortunate than us, those that were forced to work under conditions of shipping deadlines or unreasonable managers, we give thanks to their humble offering of naïve security implementations.

For when these poor lost souls aim to protect a device using a password or PIN, they may choose to perform a simple comparison such as the following.

```
int password_loop(void){
  unsigned char master_password[6];
  unsigned char user_password[6];

  read_master_password_from_storage(master_password);
  wait_for_pin_entry(user_password);

  for (int i = 0; i < 6; i++){
    if (master_password[i] != user_password[i]){
      return 0;
    }
  }
  return 1;
}
```

Which everyone knows are subject to timing attacks. Such attacks can be thwarted of course by comparing a hash of the password instead of the actual password, but simple devices or small codes such as bootloaders may skip such an operation to save space.

10.1 A PIN-Protected Hard Drive

Let's look at a PIN-protected hard drive enclosure, which the vendor describes as a "portable security enclosure with 6 digit password." This enclosure formats the hard drive into two partitions, the Public partition and the secured Vault partition. The security of the Vault is entirely given by sacrilegious changes to the partition table, such that if you remove the hard disk from the enclosure and plug into a computer the OS won't recognize the disk, thinking it tainted. The data itself is still there however.

The PCB contains four ICs of particular interest: a Marvell 88SA8040 Parallel ATA to Serial ATA bridge, a JMicron JM20335 USB to PATA bridge, a WareMax WM3028A (no public information), and a SST 39VF010 flash chip connected to the WM3028A. There's also a number of discrete logic gates including two 74HCT08D AND devices and one 74HC00D NAND device. These logic gates are used to multiplex multiple parts from apparently limited IO pins of the WM3028A. It would appear that the system passes the Parallel ATA data through the WM3028A chip, which is presumably some microcontroller-based system responsible for fixing reads of the partition table once the correct password is put in.

The use of discrete logic chips for multiplexing IO lines ultimately makes our life easier. In particular one of the 74HCT08D chips, U10, provides us with a measurement point for determining when the password has failed the internal test.

Pin 3 of the switch is the multiplexing pattern from the microcontroller. Remember we must determine when the microcontroller has read the pin, not simply when the user pushed the pin. Knowing that this button was pressed, and thus caused the 'Wrong PIN' LED to come on, we can measure the time between when the microcontroller has read in the entire PIN and when the LED goes on.

We then break the system one digit at a time by measuring the time after the last button is pressed. First we enter 0-6-6-6-6-6, then 1-6-6-6-6-6, 2-6-6-6-6-6, etc. The delay between reading the button press and

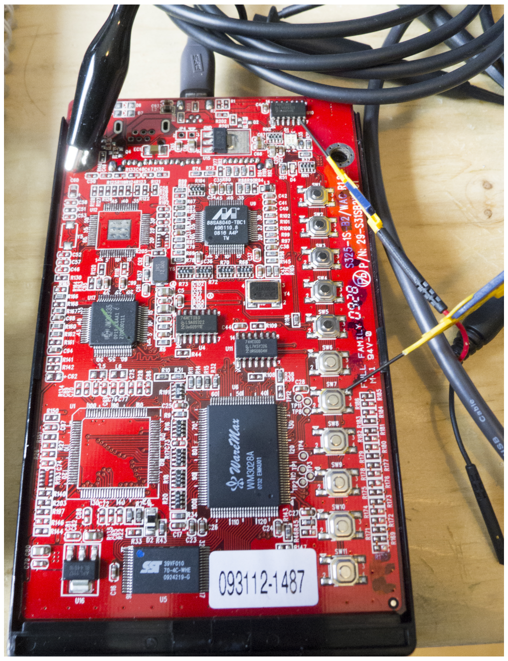


Figure 9: Pin-Protected Hard Disk

displaying the LED will be shortest if the first digit is wrong, longer if the first digit is right. A moving-picture version of this is available on the intertubes.[17]

An example of the oscilloscope capture of this is shown in Figure 10, where the correct password is 1-2-3-4-5-6. Note the jump in time delay between 0-6-6-6-6-6 and 1-6-6-6-6-6. This continues for each correct digit. Thus for a 6-digit pin, we guess only a worst case of $10 * 6 = 60$ options, instead of the million that would be required for brute-forcing the full pin.

10.2 TinySafeBoot for the Atmega328P

But what if the clever developer decided to not tell the user when they've entered a wrong password? A security-conscious bootloader might wish to avoid being vulnerable to timing attacks, but is attempting to avoid adding hash code for size reasons. An example of this is pulled from a real bootloader which has a password feature. When a wrong password is entered jumps into an endless loop, effectively avoiding providing information that would be useful for a timing attack.

In particular, let's take a look at TinySafeBoot, which is a very small bootloader for most AVR microcontrollers.[18] This wonderful bootloader has many features, such as using a single IO pin, auto-calibrating baud rate, and automatically build a bootloader image for you. And, as already mentioned, it contains a password feature.

But compare the measurements of the power signatures shown in Figure 11, which is the bootloader running on an AtMega328P. The correct password is `{0x61, 0x52, 0x77, 0x6A, 0x73}`. If we measure the power consumption of the device, we observe clear differences between the correct and incorrect guesses. This can be done by using a resistor in-line with the microcontroller power supply, such as by lifting a TFQP package pin.

The code for the password feature looks as in the following listing. Note when you receive an incorrect

[17]`http://tinyurl.com/pintiming`

[18]You can find more information about this bootloader at `http://jtxp.org/tech/tinysafeboot_en.htm`.

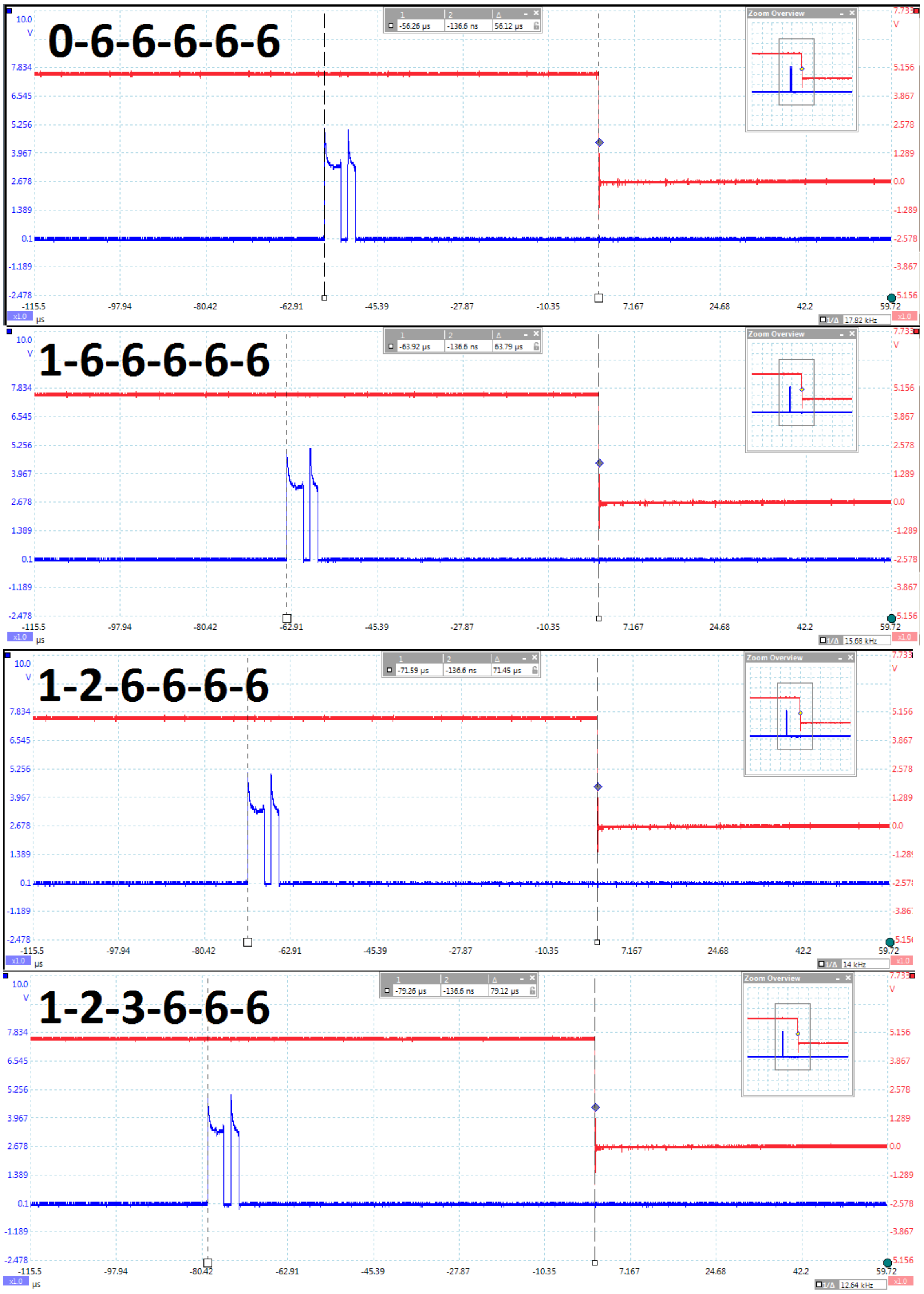


Figure 10: Timing Results

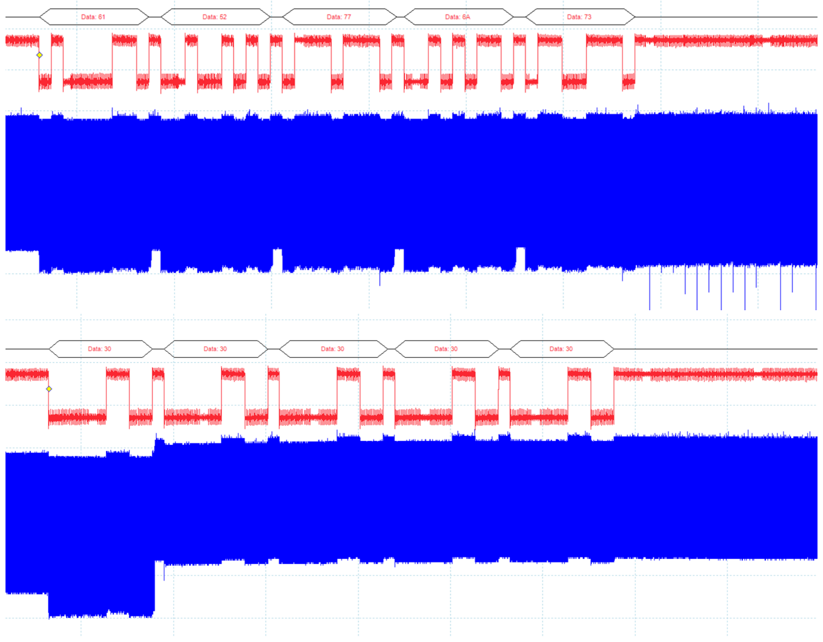


Figure 11: Power Analysis. Above is a correct guess, Below is incorrect.

character the system jumps into an infinite loop at the `chpwl` label, meaning a reset is required to try another password.

```
CheckPW:
chpw1:
          lpm tmp3, z+                    ; load character from Flash
          cpi tmp3, 255                   ; byte value (255) indicates
          breq chpwx                      ; end of password -> exit
          rcall Receivebyte               ; else receive next character
chpw2:
          cp tmp3, tmp1                   ; compare with password
          breq chpw1                      ; if equal check next character
          cpi tmp1, 0                     ; or was it 0 (emergency erase)
chpwl:    brne chpwl                      ; if not, loop infinitely
          rcall RequestConfirmation       ; if yes, request confirm
          brts chpa                       ; not confirmed, leave
          rcall RequestConfirmation       ; request 2nd confirm
          brts chpa                       ; cannot be mistake now
          rcall EmergencyErase            ; go, emergency erase!
          rjmp Mainloop
chpa:
          rjmp APPJUMP                    ; start application
chpwx:
;         rjmp SendDeviceInfo             ; go on to SendDeviceInfo
```

We can immediately see the jump to the infinite loop in the power trace! It happens as soon as the device receives an incorrect character of the password. Thus despite the original timing attack failing, with a tiny bit of effort we again find ourselves easily guessing the password.

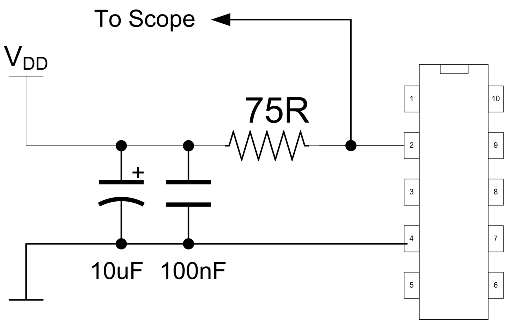


Figure 12: Tapping VCC for Power Analysis

Measuring the power consumption of the microcontroller requires you to insert a resistor into the power supply rail. Basically, this requires you to perform the schematic as shown in Figure 12. Note you can insert it either into the VCC or the GND rail. It may be that the GND rail is cleaner for example, or it may be that it's easier to physically get at the VCC pin on your device.

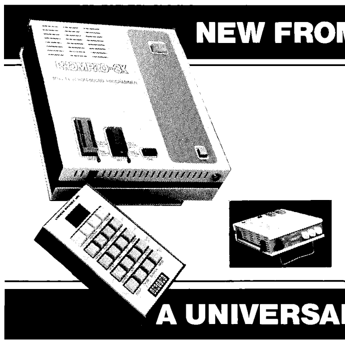
For a regular oscilloscope you may need to build a Low Noise Amplifier (LNA) or Differential Probe. I've got some details of that in my previous talk and whitepaper.[19] You can expect to make a probe for a pretty low cost, so it's a worthwhile investment!

In terms of physically pulling this off, the easiest option is if you build a breadboard circuit with the AVR and a resistor inserted in the power line. Be sure you have lots of decoupling after the resistor, which will give you a much cleaner signal. If you're looking to use an existing board, you can make a 'cheater' socket with a resistor inline, as in Fig B, which was designed for an Arudino board.

Real devices are likely to be SMD. If you're attacking a TQFP package, you might find it easiest to lift a lead and insert a 0603 or 0402 resistor inline with the power pin. You might wish to find a friendly neighbour with a steady hand and a stereo microscope for this if you aren't of strong faith in your soldering!

Thus when attacking embedded systems, the timing attacks often present a practical entry method. Be sure to carefully inspect the system to determine the 'correct' measurement you need to use, such as measuring the point in time when the microcontroller reads an I/O pin, not simply when an external event happens.

When designing embedded systems, store the hash of the users password, lest ye be embarrassed by breaks in your device.



NEW FROM LOGICAL DEVICES INC:

PROMPRO-8X™ Model II

A stand-alone programmer starting at $895.00 can put you in business to program EE/EPROMs PAL/PLDs,* Single Chip micros,* and Bipolar PROMs,* + EPROM IN-CIRCUIT EMULATION* capability that can speed up your development time considerably and an RS-232 communications port that lets you integrate it with your IBM PC as a total firmware and Logic development station.

All from a company with an excellent reputation for quality and service.

A UNIVERSAL DEVICE PROGRAMMER

[19] http://newae.com/blackhat