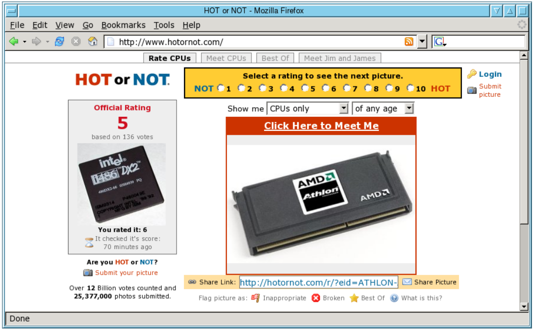
Detecting temperature through clock skew



Steven J. Murdoch (aka 253802)

`www.cl.cam.ac.uk/users/sjm217`

UNIVERSITY OF CAMBRIDGE

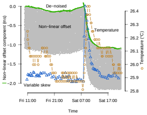
Computer Laboratory

OpenNet Initiative

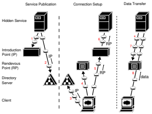
www.opennet.net

23rd Chaos Communication Congress, Berlin, Germany, 27–30 December 2006

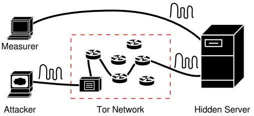
This presentation introduces clock skew and how it can introduce security vulnerabilities



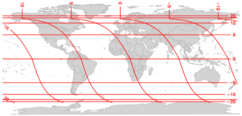
Clock skew, its link to temperature, and measurement



Tor and hidden services



Attacking Tor with clock skew



Other applications of clock skew

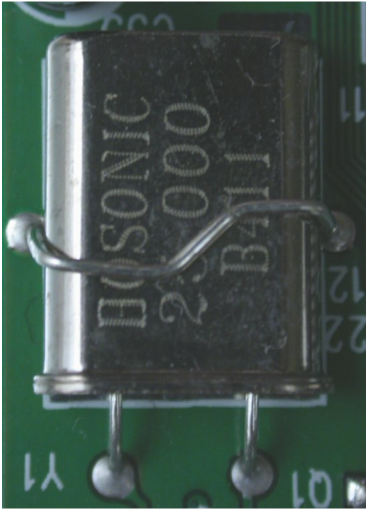
Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at an nominal frequency
- *Counter*, counts the number of ticks produce by the oscillator

On Linux there are several clocks available (other OS are similar):

- *Jiffy counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off and initialises the system clock



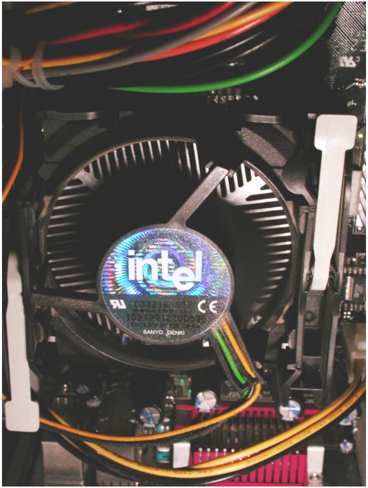
Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at an nominal frequency
- *Counter*, counts the number of ticks produce by the oscillator

On Linux there are several clocks available (other OS are similar):

- *Jiffy counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off and initialises the system clock



Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at an nominal frequency
- *Counter*, counts the number of ticks produce by the oscillator

On Linux there are several clocks available (other OS are similar):

- *Jiffy counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off and initialises the system clock

```
void do_timer(struct pt_regs *regs)
{
        (*(unsigned long *)&jiffies)++;
#ifndef CONFIG_SMP
        /* SMP process accounting uses

        update_process_times(user_mode
#endif
        mark_bh(TIMER_BH);
        if (TQ_ACTIVE(tq_timer))
                mark_bh(TQUEUE_BH);
}

#if !defined(__alpha__) && !defined(__

/*
 * For backwards compatibility?  This
 * and all newer ports shouldn't need
 */
asmlinkage unsigned long sys_alarm(uns
{
        struct itimerval it_new, it_ol
        unsigned int oldalarm;

        it_new.it_interval.tv_sec = it_
        it_new.it_value.tv_sec = second
        it_new.it_value.tv_usec = 0;
        do_setitimer(ITIMER_REAL, &it_
        oldalarm = it_old.it_value.tv_
```

Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

- *Oscillator*, controlled by a crystal, ticks at an nominal frequency
- *Counter*, counts the number of ticks produce by the oscillator

On Linux there are several clocks available (other OS are similar):

- *Jiffy counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off and initialises the system clock

```
/*
 * This version of gettimeofday has mi
 * and better than microsecond precisi
 */
void do_gettimeofday(struct timeval *t
{
        unsigned long flags;
        unsigned long usec, sec;

        read_lock_irqsave(&xtime_lock,
        usec = do_gettimeoffset();
        {
                unsigned long lost = j
                if (lost)
                        usec += lost *
        }
        sec = xtime.tv_sec;
        usec += xtime.tv_usec;
        read_unlock_irqrestore(&xtime_

        while (usec >= 1000000) {
                usec -= 1000000;
                sec++;
        }

        tv->tv_sec = sec;
        tv->tv_usec = usec;
}
```

Computers have multiple clocks which are constructed from hardware and software components

A clock consists of an:

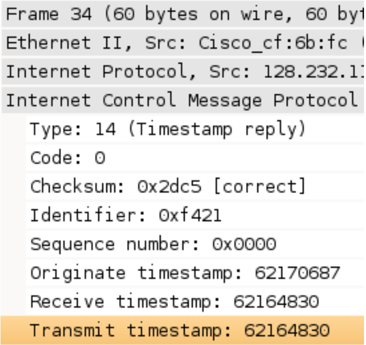
- *Oscillator*, controlled by a crystal, ticks at an nominal frequency
- *Counter*, counts the number of ticks produce by the oscillator

On Linux there are several clocks available (other OS are similar):

- *Jiffy counter:* An uncorrected clock used internally to the OS
- *System clock:* A clock corrected by NTP (used by applications)
- *BIOS clock* (also known as CMOS clock): runs even when PC is off and initialises the system clock

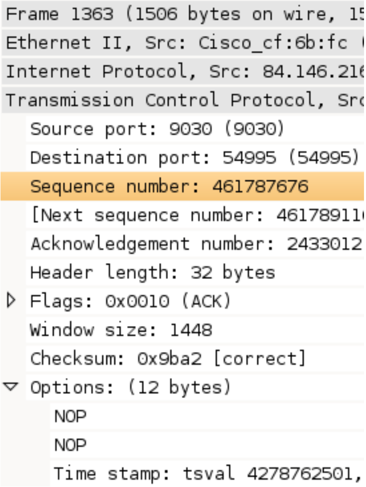
Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Only works for Linux, 1 MHz, generated from system clock, very hard to block on firewalls, details in my 22C3 talk on steganography
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from jiffy counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



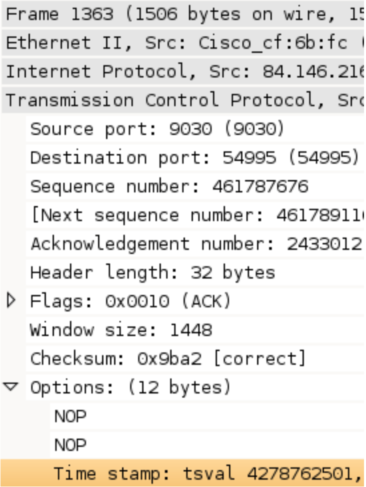
Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Only works for Linux, 1 MHz, generated from system clock, very hard to block on firewalls, details in my 22C3 talk on steganography
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from jiffy counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



Some of these clocks can be queried over the Internet through ICMP and TCP

- ICMP timestamp request
  - Generated from system clock, 1 kHz, commonly disabled or blocked by firewalls
- TCP sequence numbers
  - Only works for Linux, 1 MHz, generated from system clock, very hard to block on firewalls, details in my 22C3 talk on steganography
- TCP timestamp
  - Newer feature than ICMP timestamps, 2 Hz–1 kHz, generated from jiffy counter, enabled by default on all modern TCP stacks, hard to block on firewalls, required on fast networks



Measured clock skew acts as a fingerprint of a computer (Kohno *et al.*, 2005)

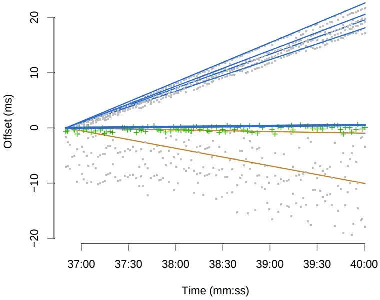
Offset:

- The difference between two clocks (ms)

↓

Skew:

- The rate of change of offset (ppm)
- Stable on one machine ($\pm$1–2 ppm), but varies over different machines (up to $\pm$50 ppm)
- Can give 4–6 bits of information on machine identity

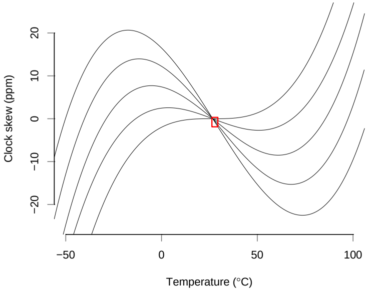


Fingerprinting computers allows identification of hosts and network traces, and detection of VMs

- Identify machines, as they change IP address, ISP and even location
- De-anonymise network traces
- Detecting whether a host is running on a virtual machine
- Confirming whether a group of hosts are running on the same hardware (e.g. a honeynet)
  - Honeyd has now been modified to produce different clock-skew fingerprints for virtual hosts
- Counting number of hosts behind a NAT
- The paper did note that clock skew can be affected by temperature, but did not explore the full potential

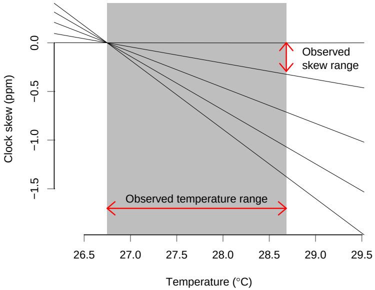
Temperature has a small, but remotely measurable, effect on clock skew

- Skew of typical clock crystal will change by ± 20 ppm over 150 °C operational range
- In typical PC temperatures, only around ± 1 ppm
- By requesting timestamps and measuring skews, an estimate of temperature changes can be derived
- Even in a well-insulated building, changes in temperature over the day become apparent



Temperature has a small, but remotely measurable, effect on clock skew

- Skew of typical clock crystal will change by ± 20 ppm over 150°C operational range
- In typical PC temperatures, only around ± 1 ppm
- By requesting timestamps and measuring skews, an estimate of temperature changes can be derived
- Even in a well-insulated building, changes in temperature over the day become apparent



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

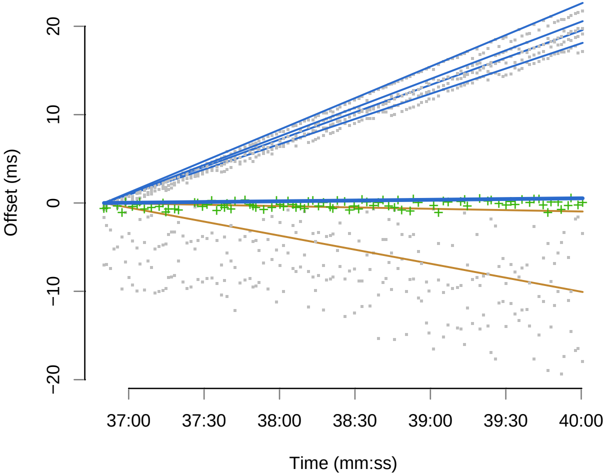
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

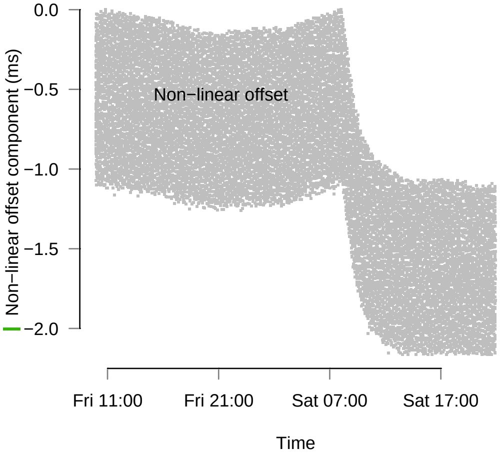
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

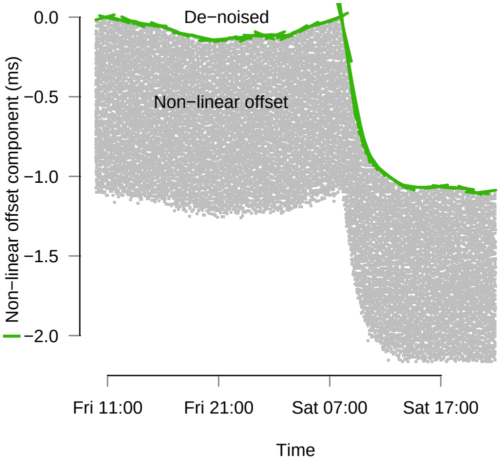
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

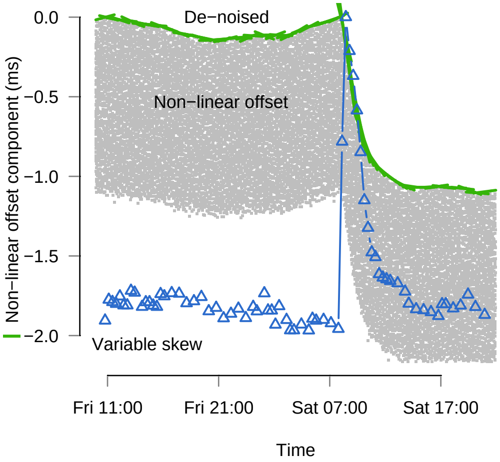
Remove noise

↓

Differentiate

↓

Compare to temperature



Clock skew variations are not visible in raw network traces, but can be extracted with numerical analysis

Measure offset of candidate machine(s)

↓

Remove constant skew from offset

↓

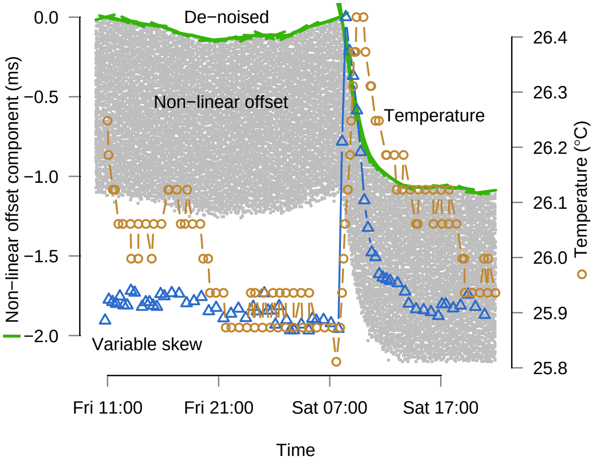
Remove noise

↓

Differentiate

↓

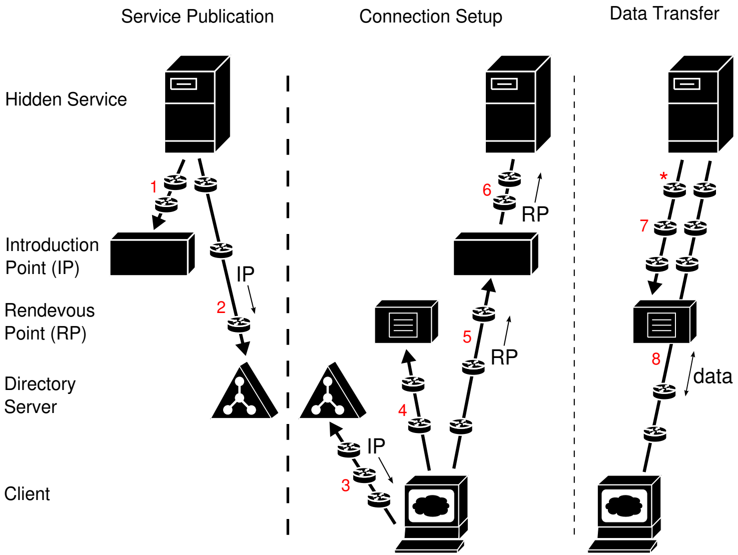
Compare to temperature



Tor is a low-latency, distributed anonymity system

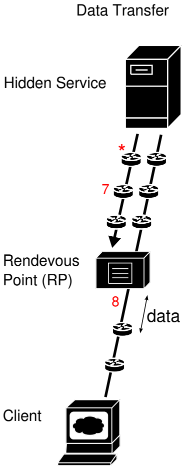
- Real-time TCP anonymisation system (e.g. web browsing)
- Supports anonymous operation of servers (hidden services)
- These protect the user operating the server and the service itself
- Constructs paths through randomly chosen nodes (around 800 now)
- Multiple layers of encryption hide correlations between input and output data
- No intentional delay introduced

Hidden services are built on top of the anonymity primitive the Tor network provides



The IP address of hidden services can be found through traffic analysis (Øverlier, Syverson, 2006)

- One Tor node (∗), selected at random by the hidden service knows, the hidden service's real IP address
- If a malicious client also controls a Tor node (easy), then eventually his node will be selected on that path
- Data is encrypted, so the malicious Tor node cannot trivially detect when it is being used to access the hidden service
- However enough timing patterns remain to identify this even, and so allowing the malicious client to locate the hidden server
- This attack is now resisted by the hidden service selecting fixed *guard* nodes for ∗

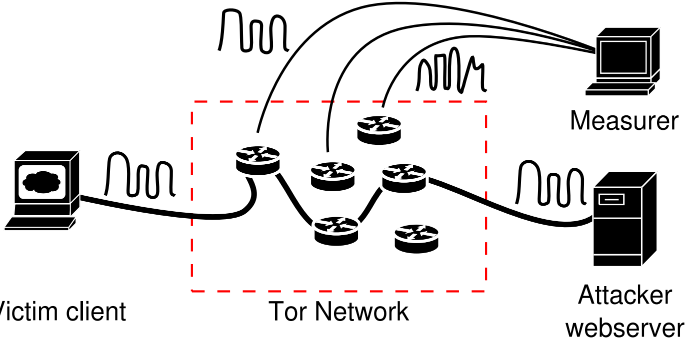


Even if an attacker cannot observe the network, traffic analysis is still possible (Murdoch, Danezis, 2005)

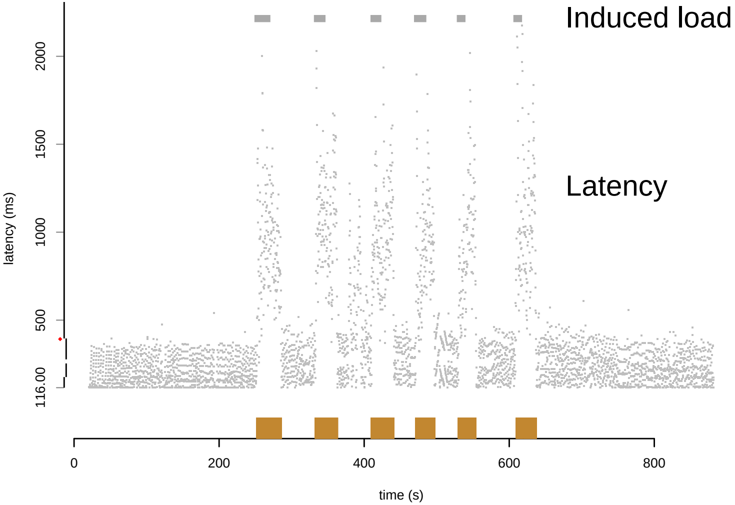
Attacker inserts traffic pattern into anonymous stream
Measurer probes all Tor nodes for their latency

↓

Nodes along path that the anonymous stream takes will exhibit the same pattern

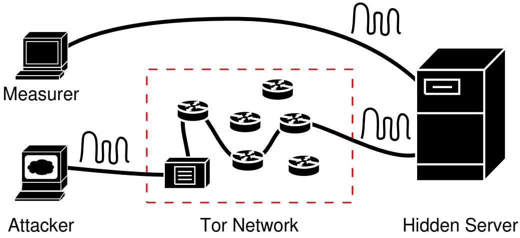


The latency of one connection going through a Tor node is strongly affected by its network load



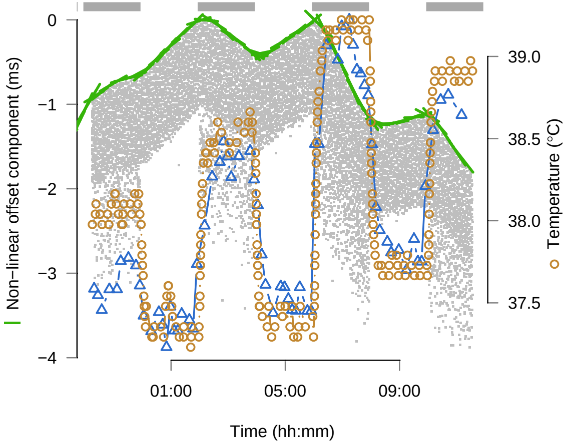
The attack can be resisted with QoS features but this creates a temperature covert channel

- Prevent one stream going through another node from interfering with any others
- Hard QoS guarantee on every stream, and no more connections accepted than there is capacity
- When one stream is not used, no other streams may use the resources released, so CPU will be idle
- This will cause the CPU to cool down and the clock skew will change accordingly, allowing connections to be tracked
- Validated with Tor hidden services on a private Tor network



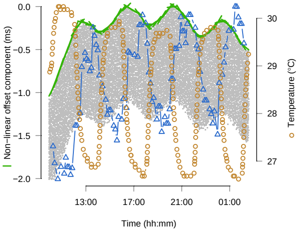
The load of a hidden service can be estimated by measuring temperature induced clock skew

- Attacker induces load by making requests to the hidden server
- Here, a periodic 2 hour on, 2 hour off pattern was used
- Measurer records clock offset and derives temperature



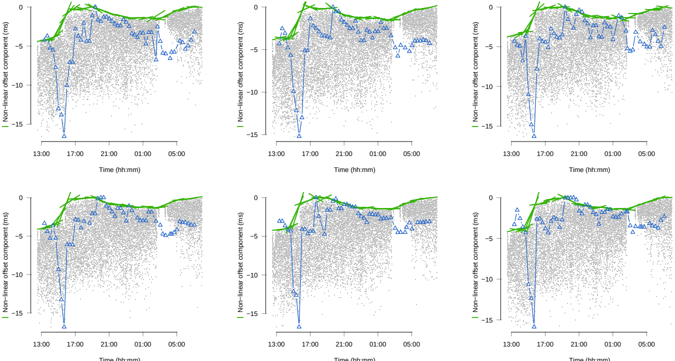
This temperature covert channel can be applied in a variety of different situations

- Inter-process communication through modulating temperature load
    - Fixed scheduling will not defend against this
    - Relies on second time source, affected differently by temperature; could be remote (NTP) or local (sound card)
- Temperature effects can cross “air-gap” security barriers
    - Confirmed in rack-mount computers; plausible for “blade” arrangements too
- Detecting temperature of Sputnik tags?



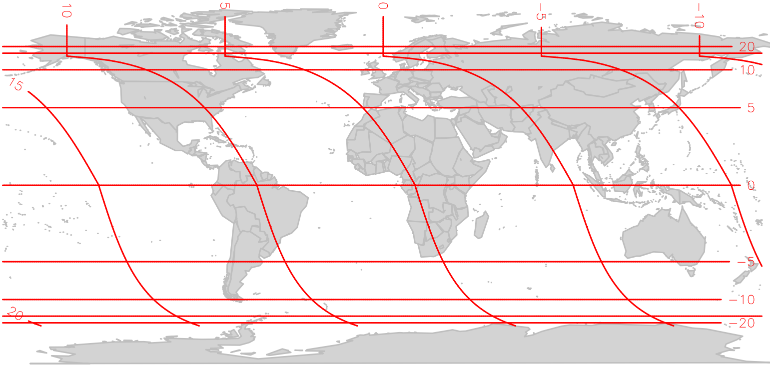
Clock skew identifies both machine identity (absolute skew) and environment (relative)

- Kohno *et al.* already showed how to identify computers through clock skew
- Temperature information can indicate environment
- Applied to investigate suspected “Sybil” attack on Tor, to discover than the 30 suspicious Tor nodes were actually 2 physical machines



From the changes in temperature of a machine, we can even estimate its location

- If length of day and middle/start/end of day can be found, locations of measurement can be found
- Imprecise, time-consuming and affected by local conditions (air conditioning) but perhaps could provide coarse-grained coordinates

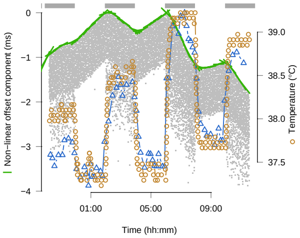


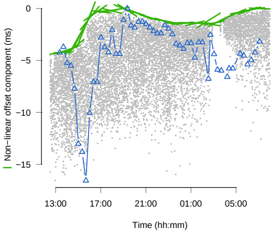
Defending against clock skew attacks is difficult and doesn't come for free

Defence	*Limitations*
Block timing information	Many low-level events are triggered on timer interrupts and this could be detected remotely
Run CPU at full load	Inefficient and must be done with care since different types of tasks can have varying temperature effects
Install a temperature compensated clock crystal	These might not have an adequate $< \pm 1$ ppm accuracy
Install an oven compensated crystal	Expensive, power hungry, but better accuracy

In summary, temperature covert channels are a viable attack even when other vectors are blocked

- Through clock skew, temperature and thus CPU load can be remotely measured, over tens of router hops
- By inducing load on a Tor hidden server and measuring the resulting clock skew, the hidden service pseudonym can be linked to its IP address
- Thermal covert channels are applicable in several other situations
- Even when a system is secure in one model of abstraction, stepping outside these limits can reveal additional attacks





You can find out more about this topic in other 23C3 talks and on the web

Talks at 23C3

- Roger Dingledine (Tor project leader), “Tor and China”: How Tor can be modified to circumvent censorship. Day 2, Saal 1, 14:00
- George Danezis (anonymity researcher), “An Introduction to Traffic Analysis”: How the volume and timing of encrypted traffic can be exploited to break anonymity. Day 3, Saal 1, 17:15

Online resources

- My paper “Hot or Not: Revealing Hidden Services by their Clock Skew” (see this talk’s page on the 23C3 Fahrplan)
- Blog posting summarising key points (linked to from Fahrplan)
- “Remote physical device fingerprinting” (Kohno *et al.* 2005), linked to from the blog posting above
- Slides and source code on my website (linked to from Fahrplan)

USENIX Security, Boston, MA. Deadline 1 February 2007