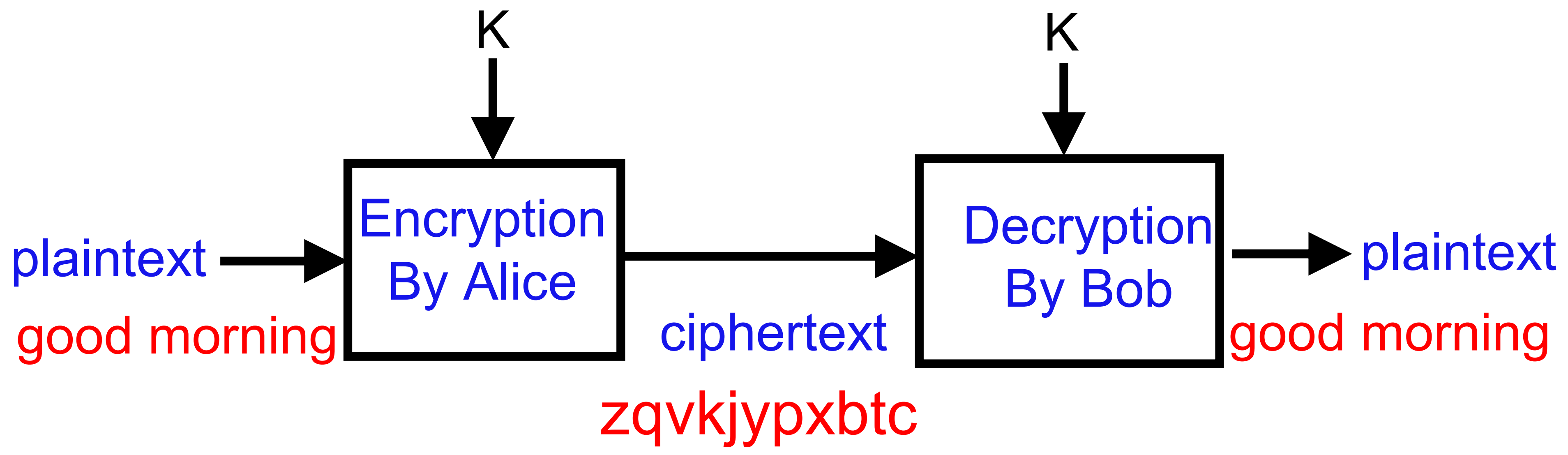
HOW CRYPTOSYSTEMS ARE REALLY BROKEN

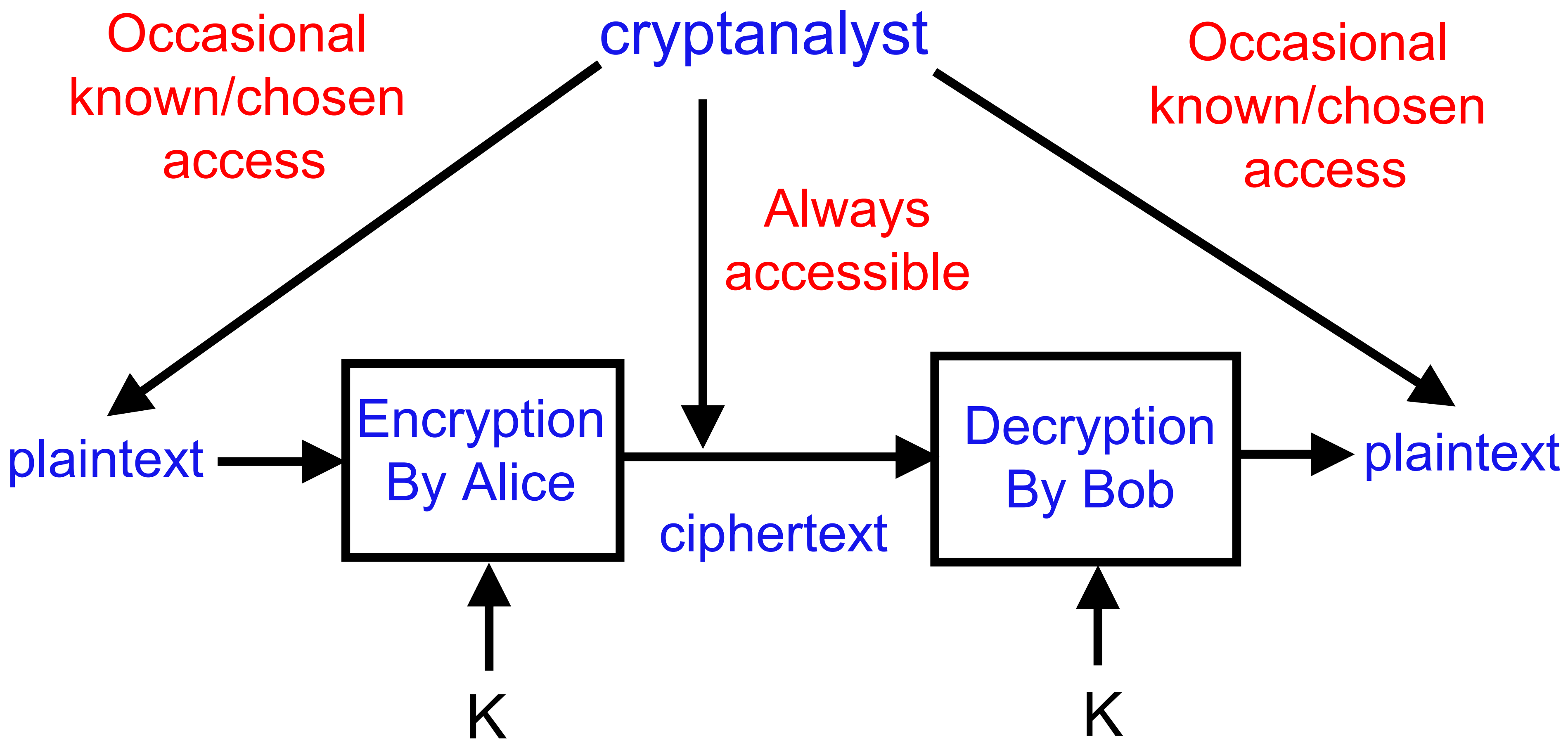
Adi Shamir
Computer Science
The Weizmann Institute
Israel

What is a cryptosystem?

Sending a plaintext securely from Alice to Bob:



The mathematical "black box" model of cryptography:



Mathematical cryptanalysis had a major impact on the outcome of WW2:

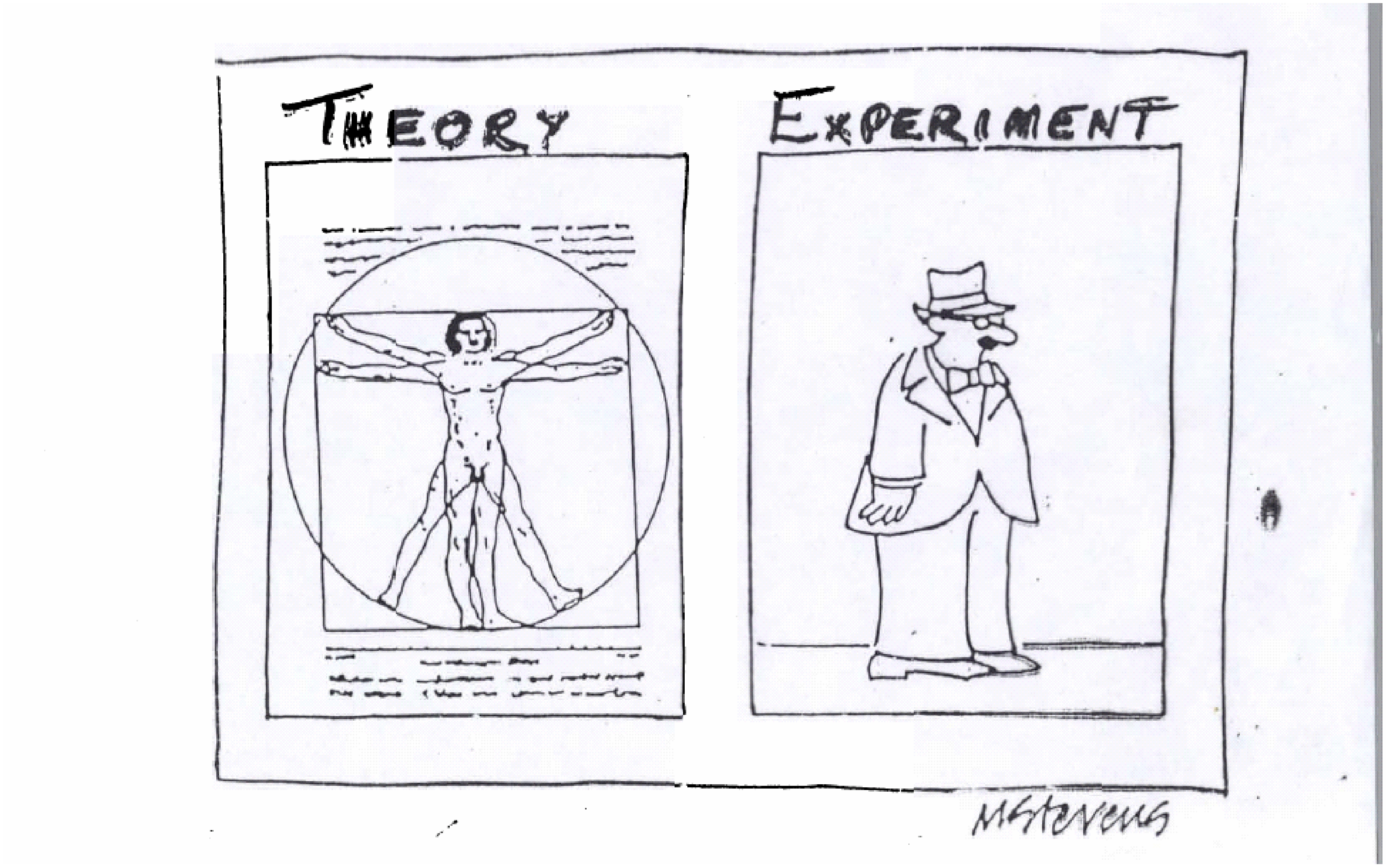
Breaking the German ENIGMA and Japanese PURPLE:

- Defeated the German bombing of England in 1940
- Almost prevented the Japanese attack on Pearl Harbor
- Helped the Americans defeat the Japanese navy at Midway
- Almost prevented the German invasion of the USSR in 1941
- Helped Montgomery stop Rommel at El Alamein in 1942

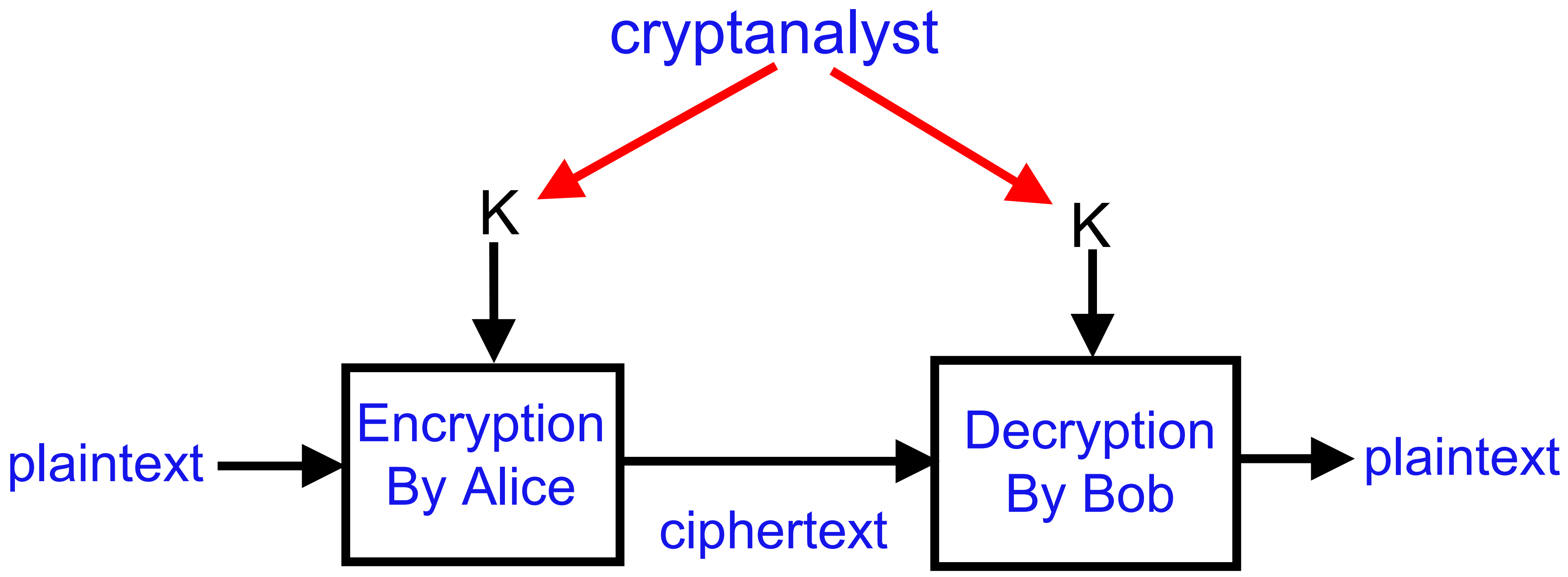
However, modern cryptosystems cannot be broken with such mathematical techniques

- Today, we have a much better understanding of how to construct cryptosystems which resist all the known types of mathematical attacks
- By using faster microprocessors, cryptographers can use more complicated cryptosystems with longer keys
- When the key length is doubled, the complexity of encryption is typically doubled, whereas the complexity of exhaustive search is squared.
- So in theory, cryptanalysts should be out of work…

The difference between theory and practice:



An unfair attack: stealing the keys



Some key stealing techniques:

- During wartime: The german U-571 submarine
- Espionage: The Walker family of spies
- Dirty tricks: Keys stolen from diplomatic mail and safes
- Trojan horses: Capturing passwords entered into PC's
- Tampered cryptosystems: The Swiss company CRYPTO AG

A new technique (published in 2008):

The "cold boot" technique to extract disk encryption keys:

- Assume that a lost or stolen laptop has important data protected by a disk encryption program such as bitlocker
- Assume that the encryption scheme is the strong AES
- Assume that the stolen laptop is in sleep mode, and that resuming operation requires a long unknown password
- The AES encryption key is kept in the volatile RAM inside the laptop, which is erased if we turn off the computer or when the battery runs out

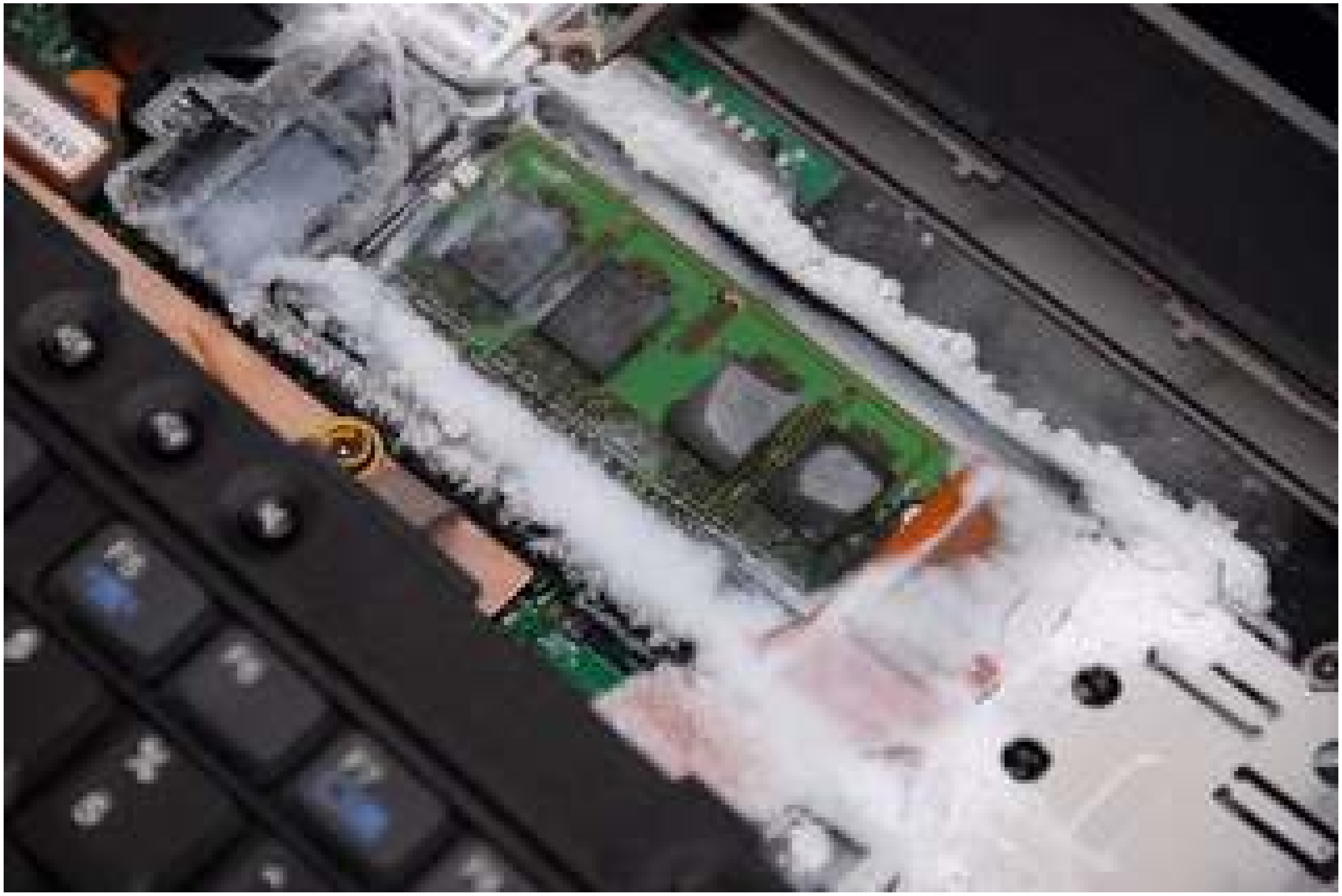
A new technique (February 2008):

The new observation:

- Data can be kept alive in unpowered RAM for tens of seconds if we cool it before cutting power
- The data deteriorates over time, but at a rate that depends on the temperature

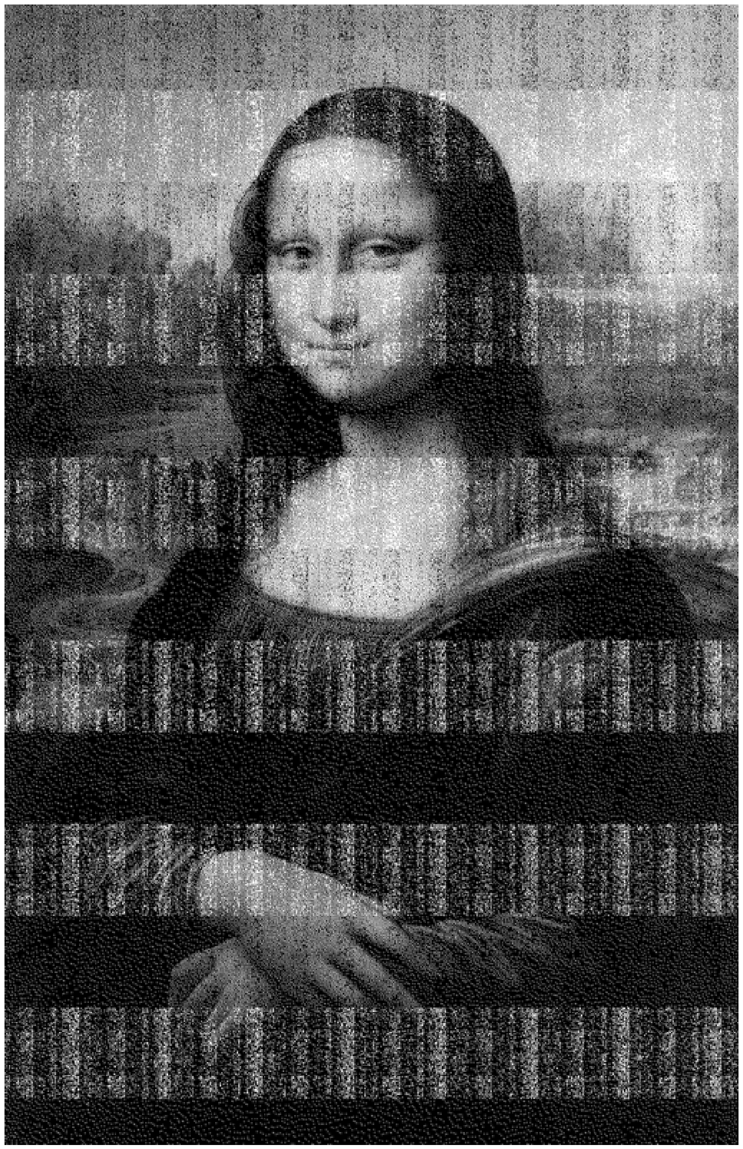
Data can be kept alive for many seconds in unpowered RAM by cooling it with a cheap can of Quick-Freeze:

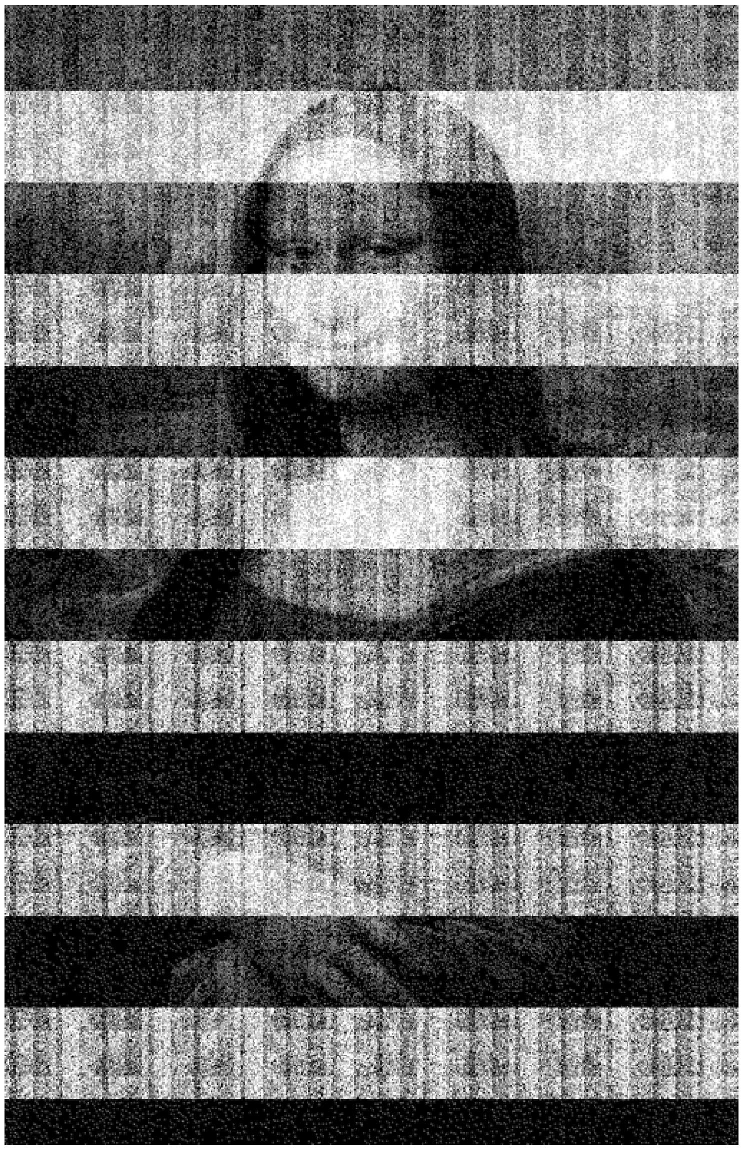
Data can be kept alive for many seconds in unpowered RAM by cooling it with a cheap can of Quick-Freeze:



Data can be kept alive for longer periods by using liquid nitrogen:

Picture kept in frozen unpowered RAM for 30 or 60 seconds:



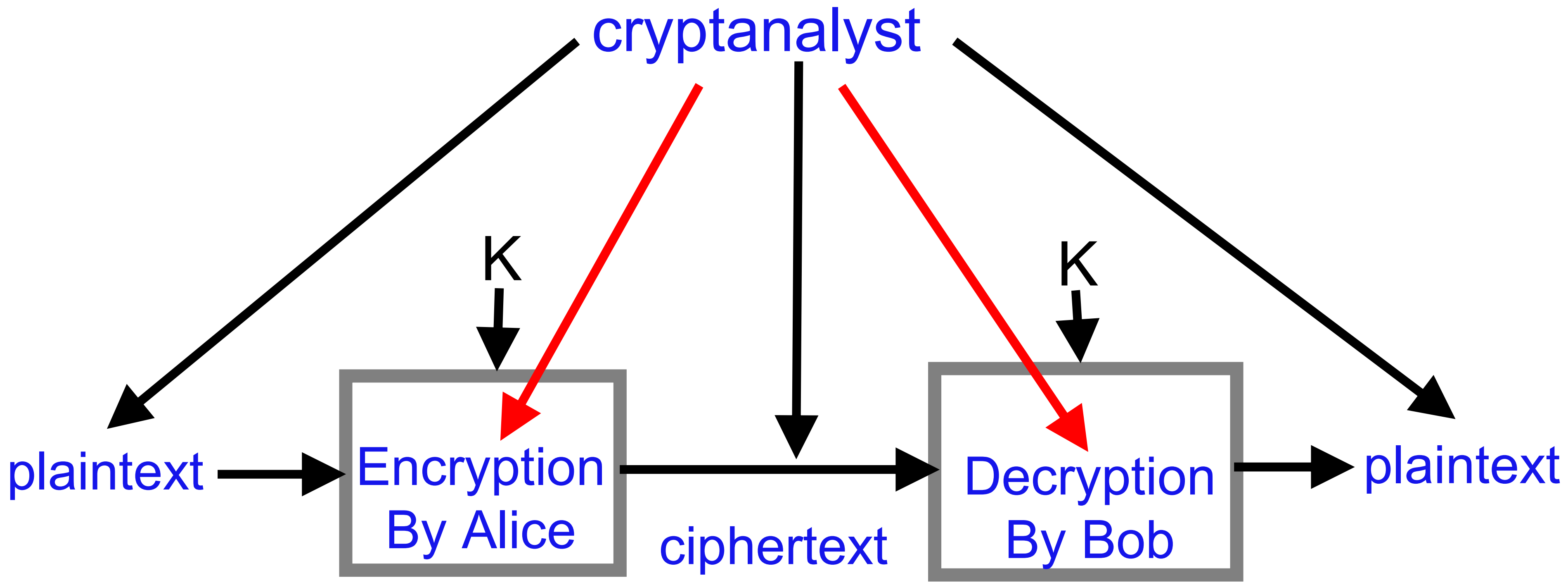


How to overcome all the known types of PC disk encryption techniques:

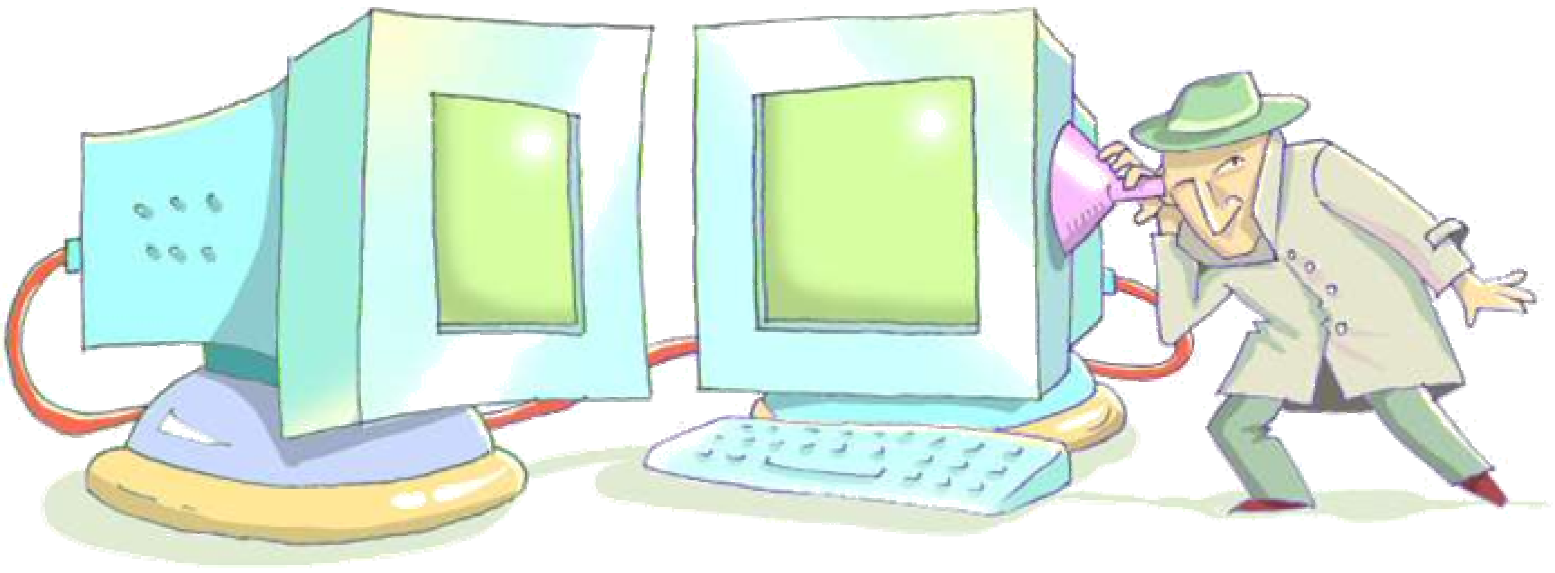
After cooling the RAM chips:

- Reboot the laptop via a small operating system located in a disk-on-key
- Quickly dump the memory contents into the disk-on-key
- Analyze the data to find a slightly corrupted AES key
- Use the fact that the 128-bit key is expanded in memory into 10 related 128-bit subkeys, which form an excellent error correcting code…

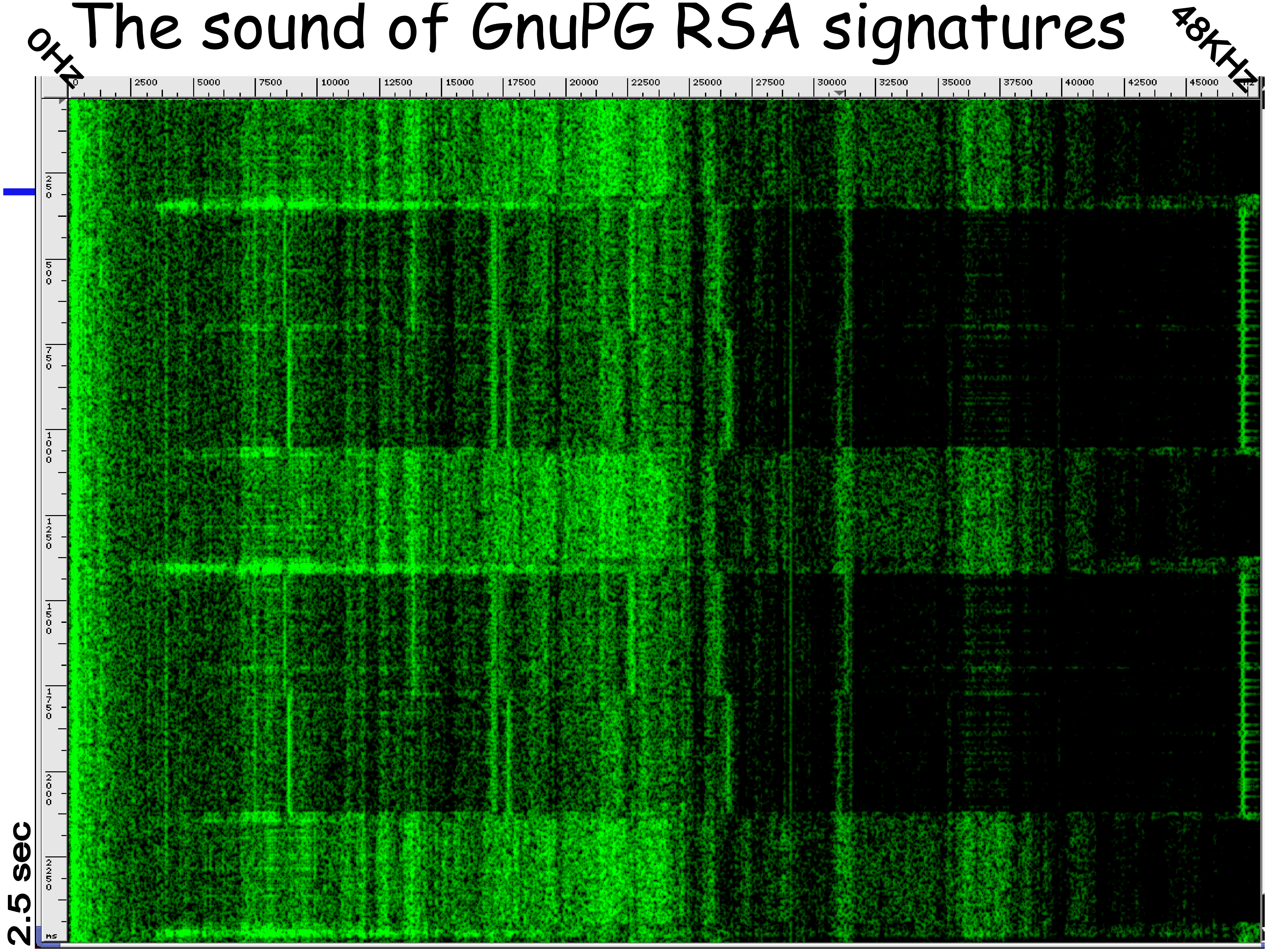
The “grey box” view of cryptography: Side channel attacks



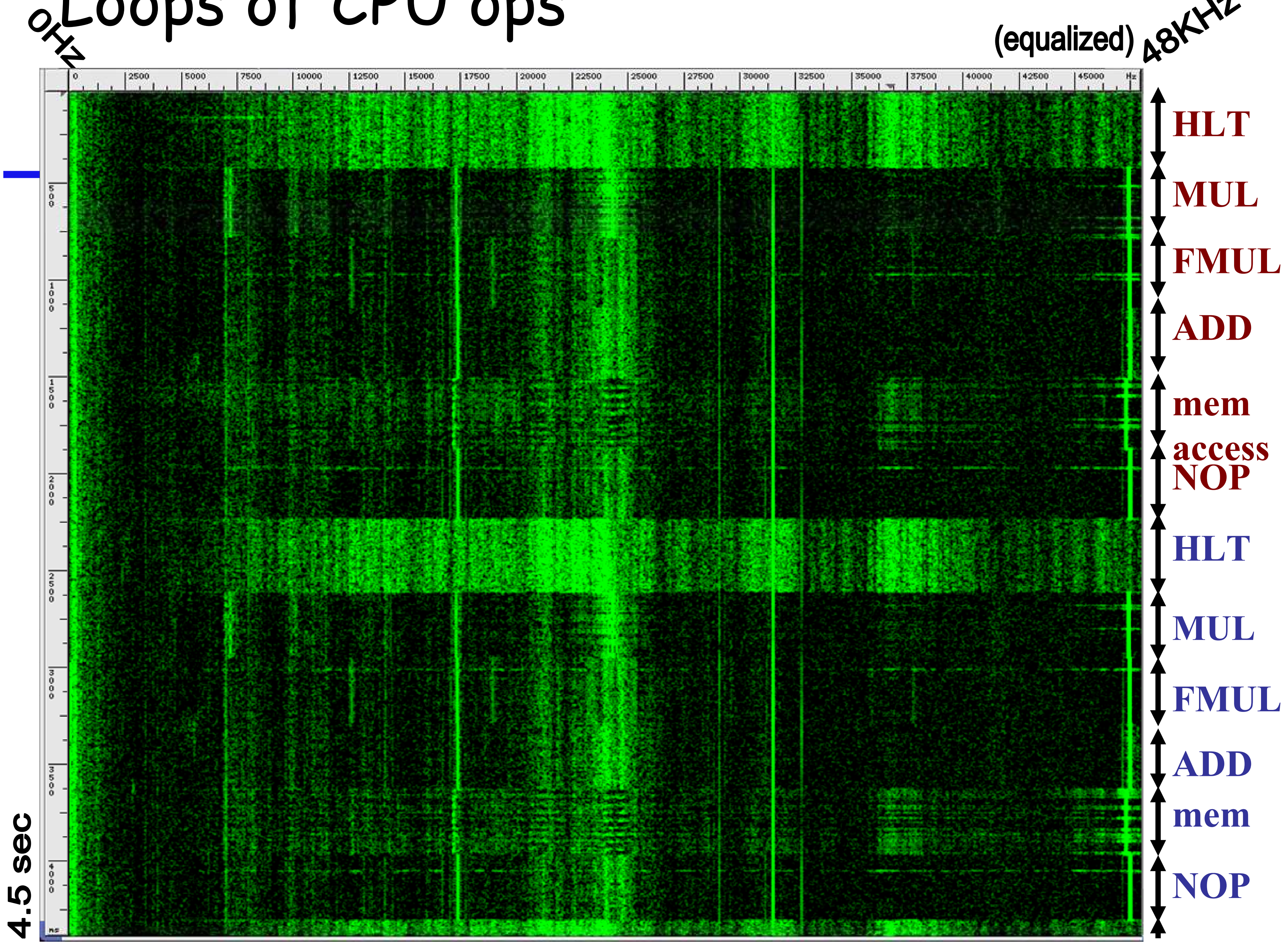
Acoustic Leakage from PC’s



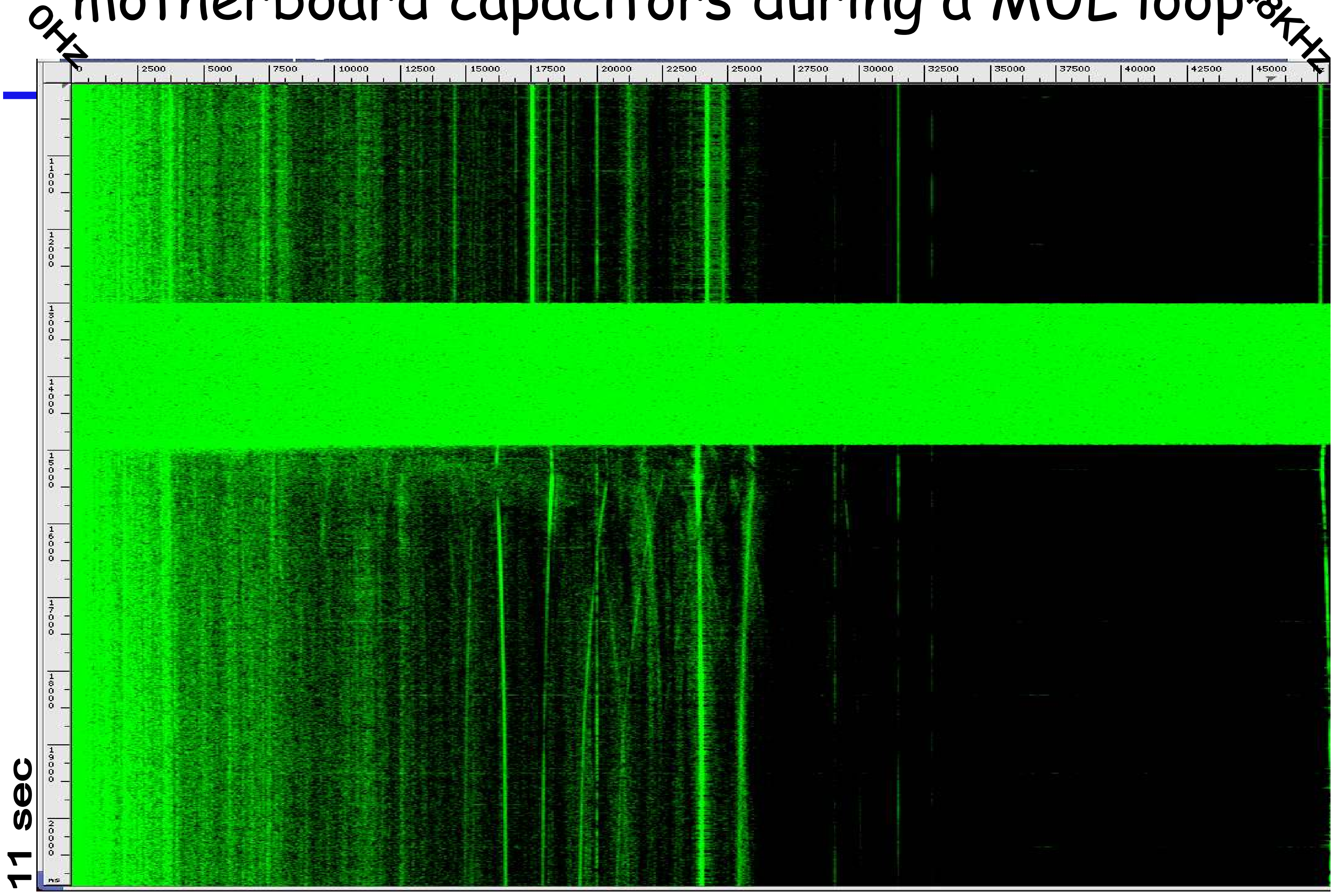
The sound of GnuPG RSA signatures



Loops of CPU ops



Application of Quick-Freeze to motherboard capacitors during a MUL loop

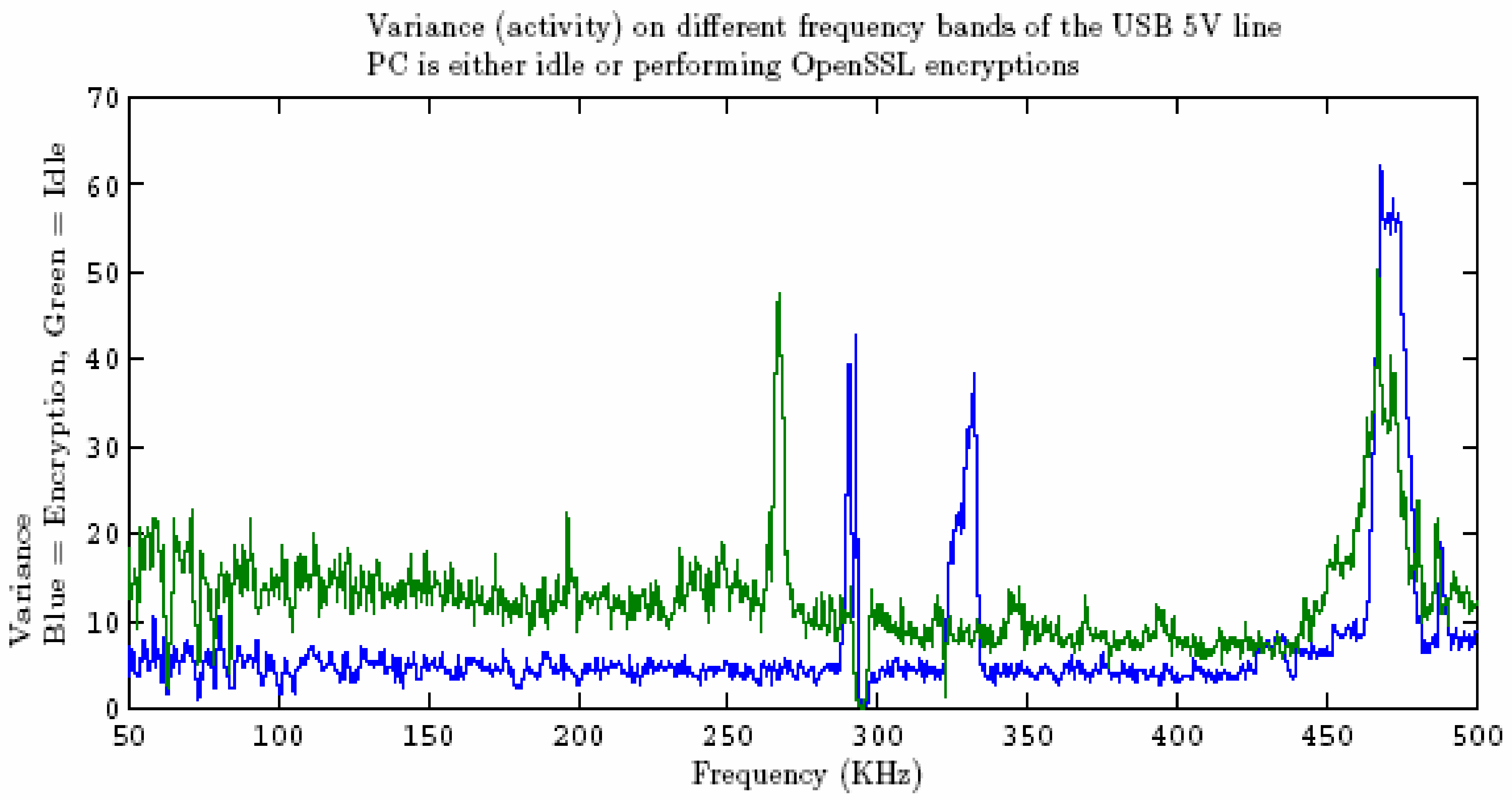


Example: How easy is it to record the power consumption of some target PC?

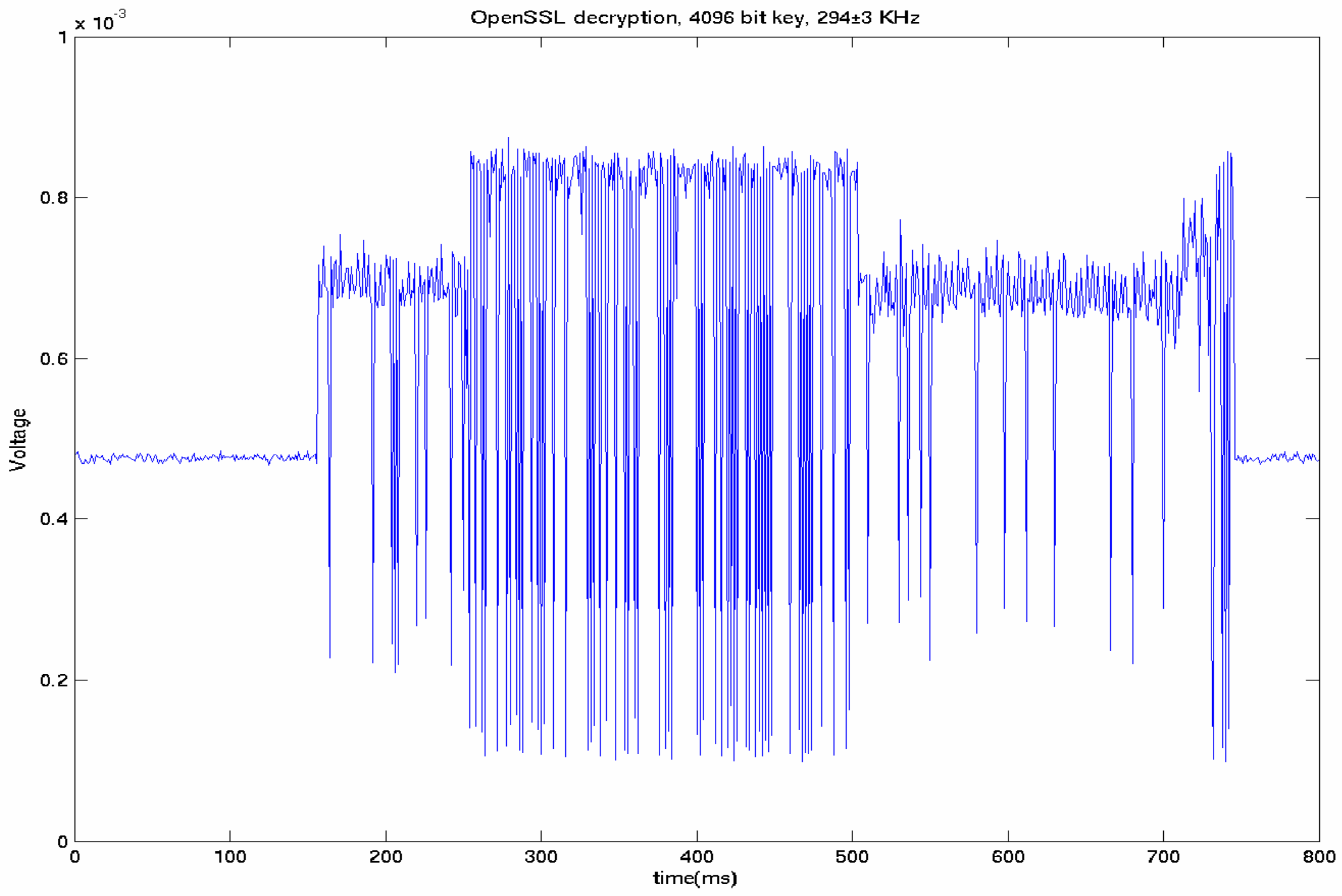
Cutting the power cord will reboot the PC, and openning a sealed PC enclosure will take too long

- A possible solution: the USB connector
- It supplies both power and data to external devices
- Many security programs control the USB connection

The Spectrum of USB power



The real-time signal of USB power at 294 KHz during OPENSSL decryption



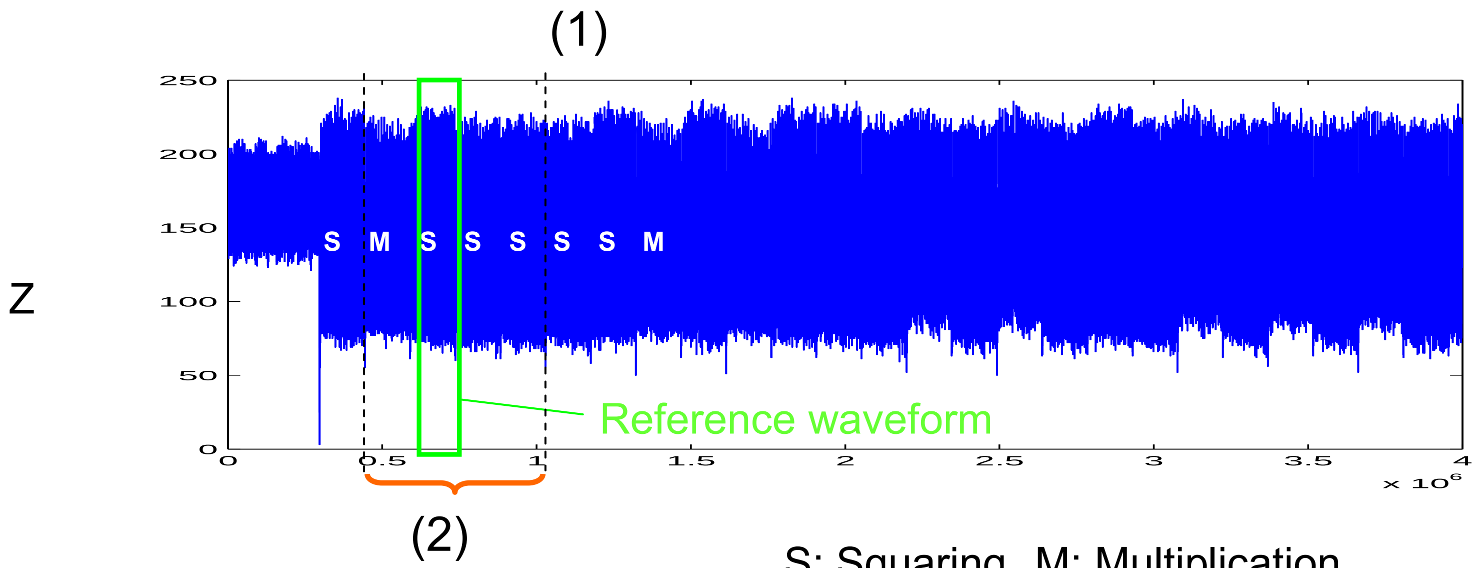
How to exploit such power traces: A new attack on the RSA scheme (Summer 2007):

- To decrypt ciphertexts or sign messages, the device computes $x^d \pmod{n}$ where d is the secret RSA key
- Since d is very large, the exponentiation is typically done by a sequence of squaring and multiplying:
- $X^{25}=((((x^2)*x)^2)^2)^2)*x$
- This can be summarized as S M S S S M

Can We Easily Distinguish Between S and M?

- In the past, they were implemented by very different algorithms, which made it easy to distinguish them by just looking at the power consumption curve
- This is no longer true, and to distinguish them we seem to need a large number of curves and sophisticated signal processing
- But now there is an exceptionally simple new attack…

Power Consumption Curves Look Like extremely Complicated Functions of the Numbers We Multiply:



S: Squaring M: Multiplication

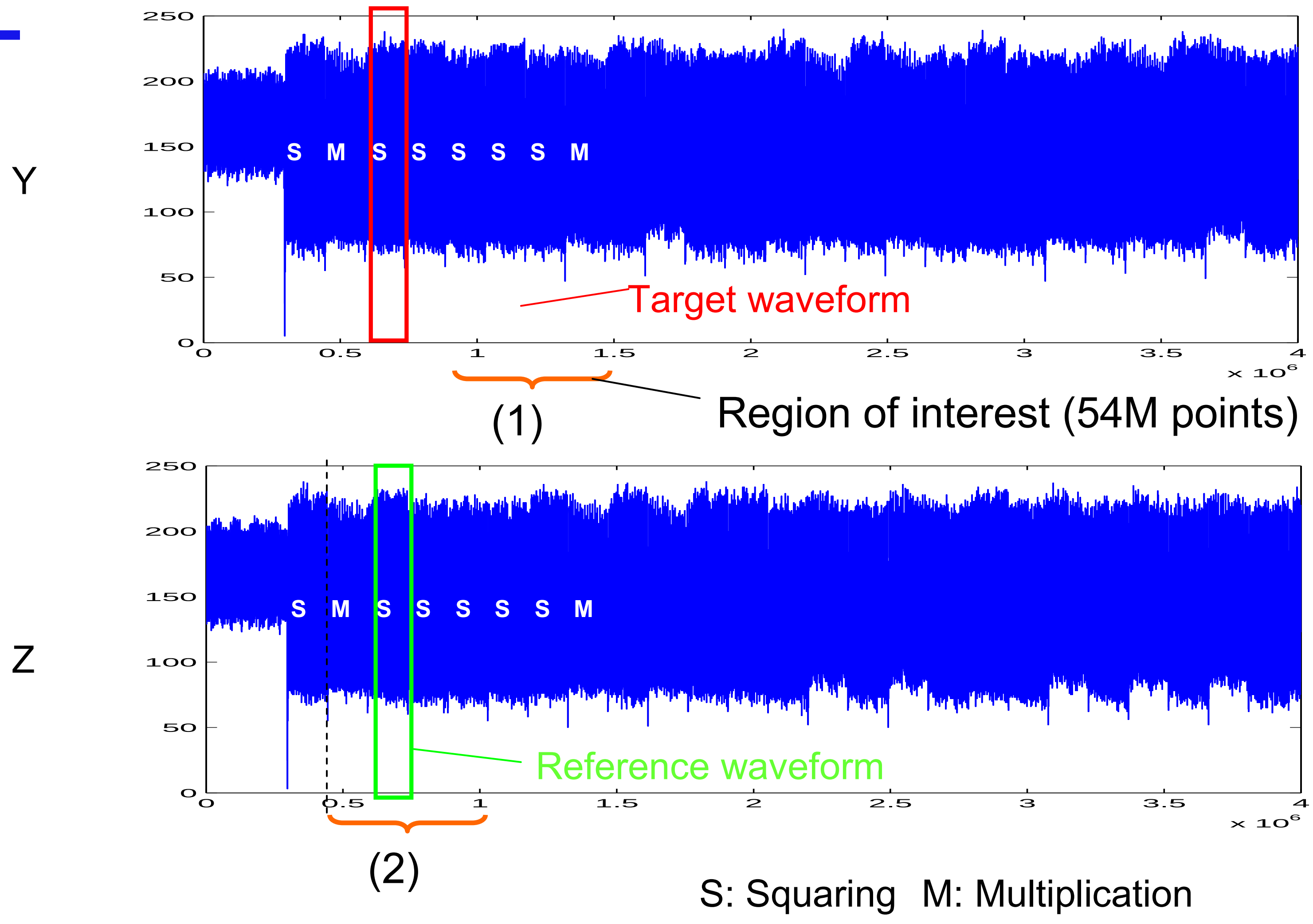
The new idea:

- Comparing two curves can serve as an equality oracle:
- If we multiply a*b and c*d, then the two curves will look similar if a=c and b=d, and different otherwise
- Our goal now is to perform only two exponentiations and compare the corresponding segments in the two power consumption curves

The new idea:

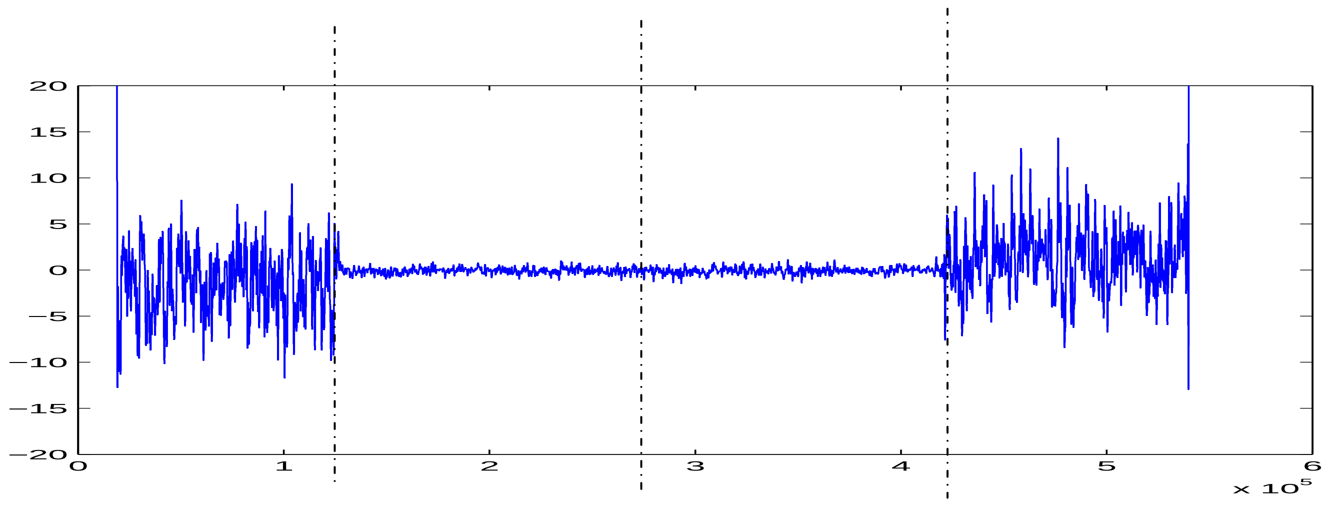
- Ask the smart card to exponentiate x and -x (mod n) (they look like totally different binary strings)
- Consider the sequence S S M S M S S S S M S S M
- All the multiplications M will look different
- Every S immediately after M will look different
- Every other S will look the same
- If we find all those equal S, we can simply fill the gaps between them with M S to find the secret d!

Exponentiating x and -x (mod n):



•Subtracting the two power consumption curves:

Subtraction after low pass filtering



Another side channel attack on RSA: Bug Attack (2008)

- Assume that a popular microprocessor has a subtle bug which affects all the manufactured chips due to a design error.
- The best known example is Intel's pentium division bug from 1994, but many other subtle bugs were discovered afterwards
- Even if Intel learned its lesson, there are many other manufacturers of microprocessors, and many designers of standard cell libraries for FPGA's.

A new side channel attack on RSA: Bug Attack

- Assume that a particular microprocessor (used in millions of devices which implement the RSA cryptosystem) has an extremely subtle multiplication bug: For a single pair of 64-bit integers a and b, their 128-bit product axb is computed incorrectly (eg, just in the least significant bit)
- This is extremely hard to detect experimentally
- Assume that the American NSA secretly discovers (or even asks the chip manufacturer to plant) such a and b
- We will now show that any such multiplication bug can lead to a devastating attack on the RSA cryptosystem

A new side channel attack on RSA: Bug Attack

Here is one way in which the NSA could breal any RSA key:

- Knowing the public key $n=pq$ of the faulty device (but not its factors p and q), the NSA can easily compute a number c which is guaranteed to be between p and q
- In particular, let c be the square root of n, rounded to the nearest integer. Then c is always located between the smaller prime p and the larger prime q.
- For example, if $n=7x11=77$, then $c=9$ satisfies $7<c<11$.

A new side channel attack on RSA: Bug Attack

- Any number which is sufficiently close to the square root c of n is also very likely to be between p and q
- In particular, the following half size number x whose low order words contain the problematic words a and b (which are improperly multiplied) is also very likely to be between p and b:

0	0	0	0	0	c	c	a	b	0

The NSA uses this x as a chosen ciphertext:

0	0	0	0	0	c	c	a	b	0

- The first step in RSA-CRT decryption is to reduce the input mod p and q. Since x is bigger than p but smaller than q, it gets randomized mod p but remains unchanged mod q.

- Each exponentiation always starts by squaring the input. This squaring almost certainly uses the natural division into the longest words which can be multiplied by the microprocessor's built-in multiplier.

- Consequently, the squaring mod q will perform the erroneous product axb, while the squaring mod p will be very unlikely to use this multiplication, and will be correct.

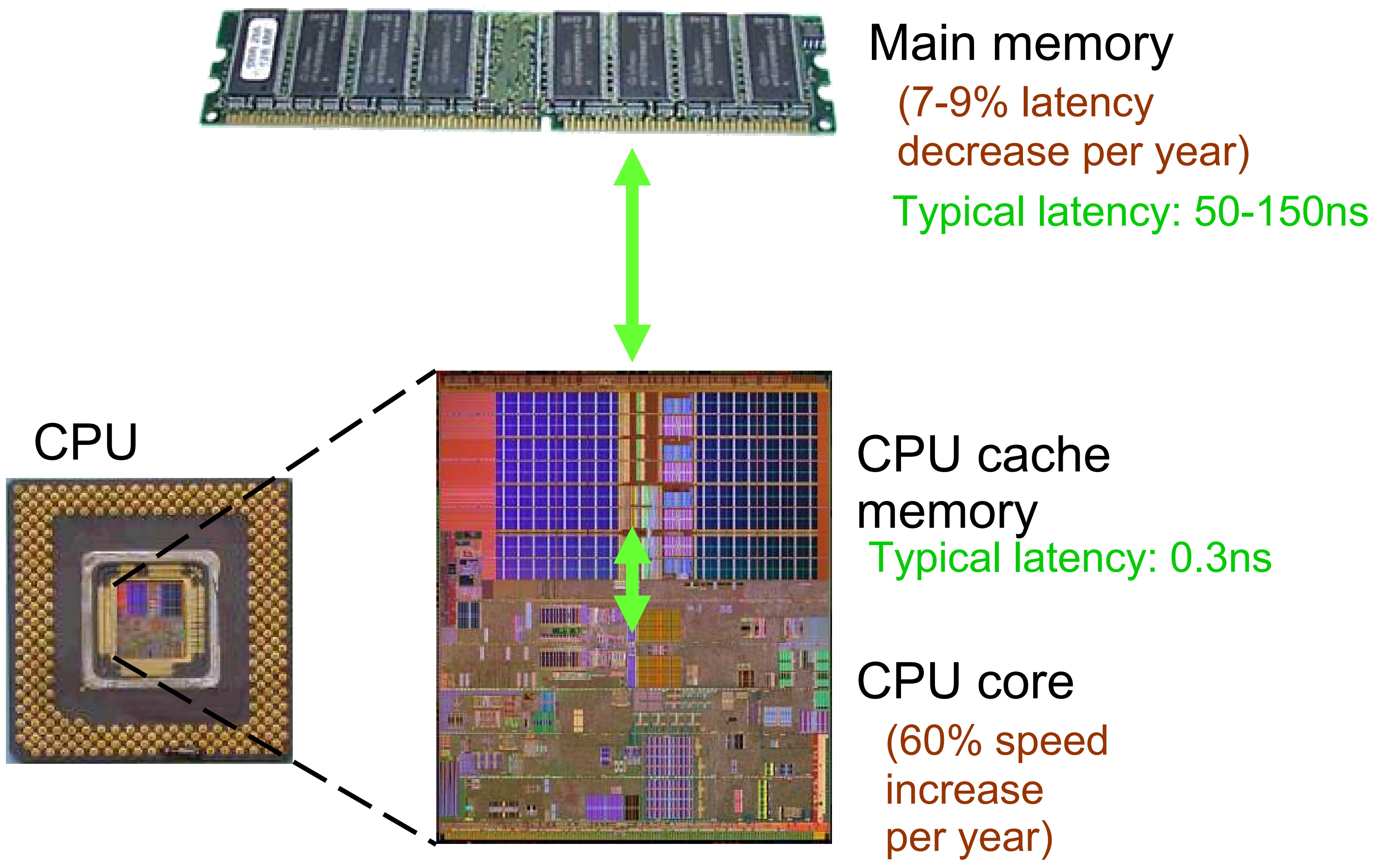
Factoring n given the wrong answer:

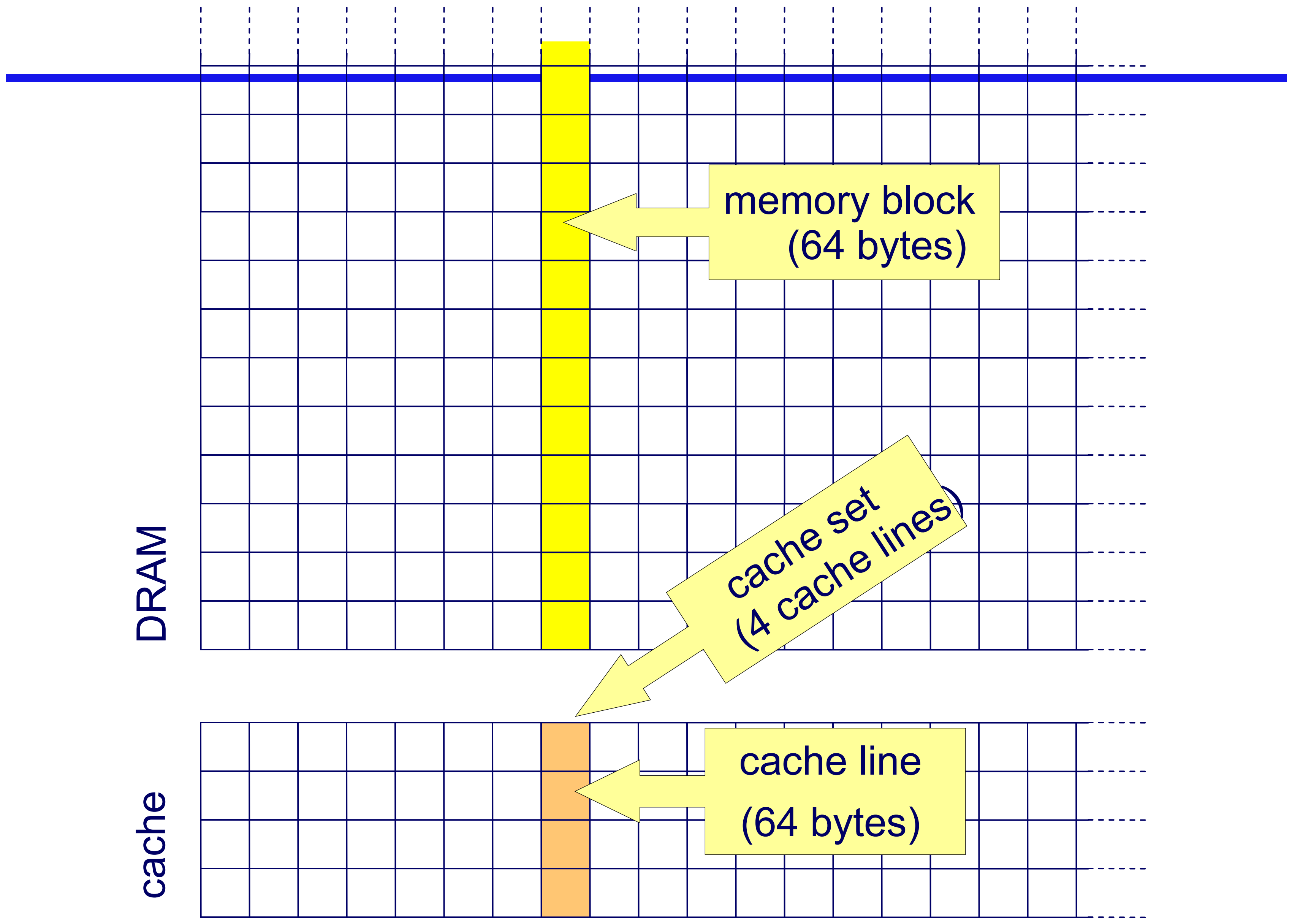
- Given the answer y and knowing the public exponent e, the NSA can compute $z=y^e-x \pmod{n}$. This z is zero mod p and nonzero mod q, so $\gcd(z,n)$ is very likely to be the secret p.
- This would enable an organization such as the NSA to break any key which is used in any RSA-based software running on any device whose microprocessor has any multiplication bug, using a single chosen message!
- I assume that many security organizations will now rush to test the multipliers of all the microprocessors they use…

Cache Attacks:

- Pure software attacks, developed by Osvik Shamir and Tromer in 2006
- Very efficient (e.g., full AES key extraction from Linux encrypted file system in 65 ms)
- Require only the ability to run untrusted code (e.g., ActiveX, Java applets, managed .NET, JavaScript) in parallel to the privileged encryption code on the same target machine
- Can be used to attack virtualized machines in cloud computing systems

Basic cache technology


Main memory
(7-9% latency decrease per year)
Typical latency: 50-150ns
CPU
CPU cache memory
Typical latency: 0.3ns
CPU core
(60% speed increase per year)


memory block
(64 bytes)
cache set
(4 cache lines)
cache line
(64 bytes)
DRAM
cache

A typical software implementation of AES

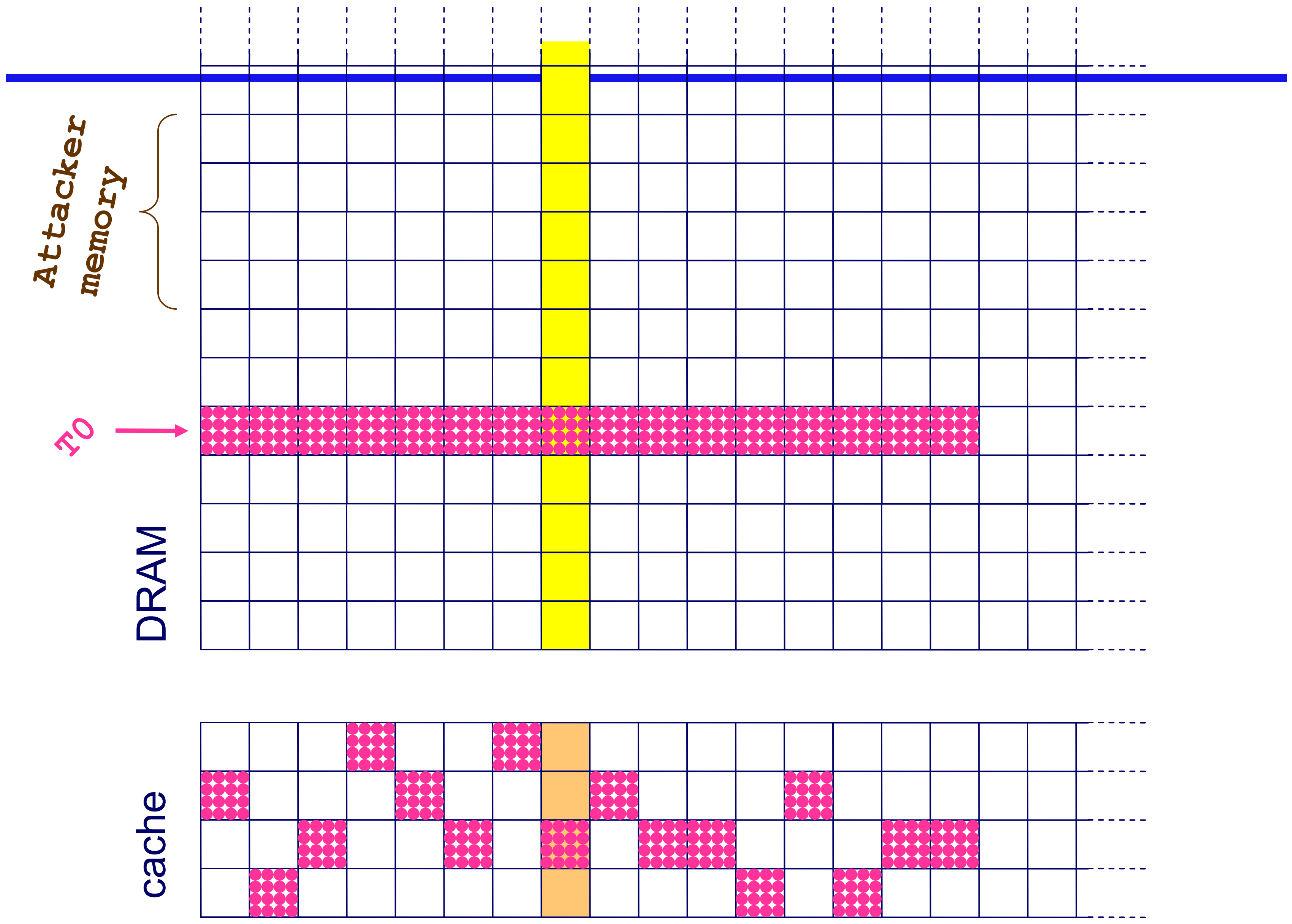
```
char p[16], k[16];                          // plaintext and key
int32 T0[256],T1[256],T2[256],T3[256]; // lookup tables
int32 Col[4];                               // intermediate state
...

/* Round 1 */
Col[0]← T0[p[ 0]⊕k[ 0]] ⊕ T1[p[ 5]⊕k[ 5]] ⊕
        T2[p[10]⊕k[10]] ⊕ T3[p[15]⊕k[15]];
Col[1]← T0[p[ 4]⊕k[ 4]] ⊕ T1[p[ 9]⊕k[ 9]] ⊕
        T2[p[14]⊕k[14]] ⊕ T3[p[ 3]⊕k[ 3]];
Col[2]← T0[p[ 8]⊕k[ 8]] ⊕ T1[p[13]⊕k[13]] ⊕
        T2[p[ 2]⊕k[ 2]] ⊕ T3[p[ 7]⊕k[ 7]];
Col[3]← T0[p[12]⊕k[12]] ⊕ T1[p[ 1]⊕k[ 1]] ⊕
        T2[p[ 6]⊕k[ 6]] ⊕ T3[p[11]⊕k[11]];
```

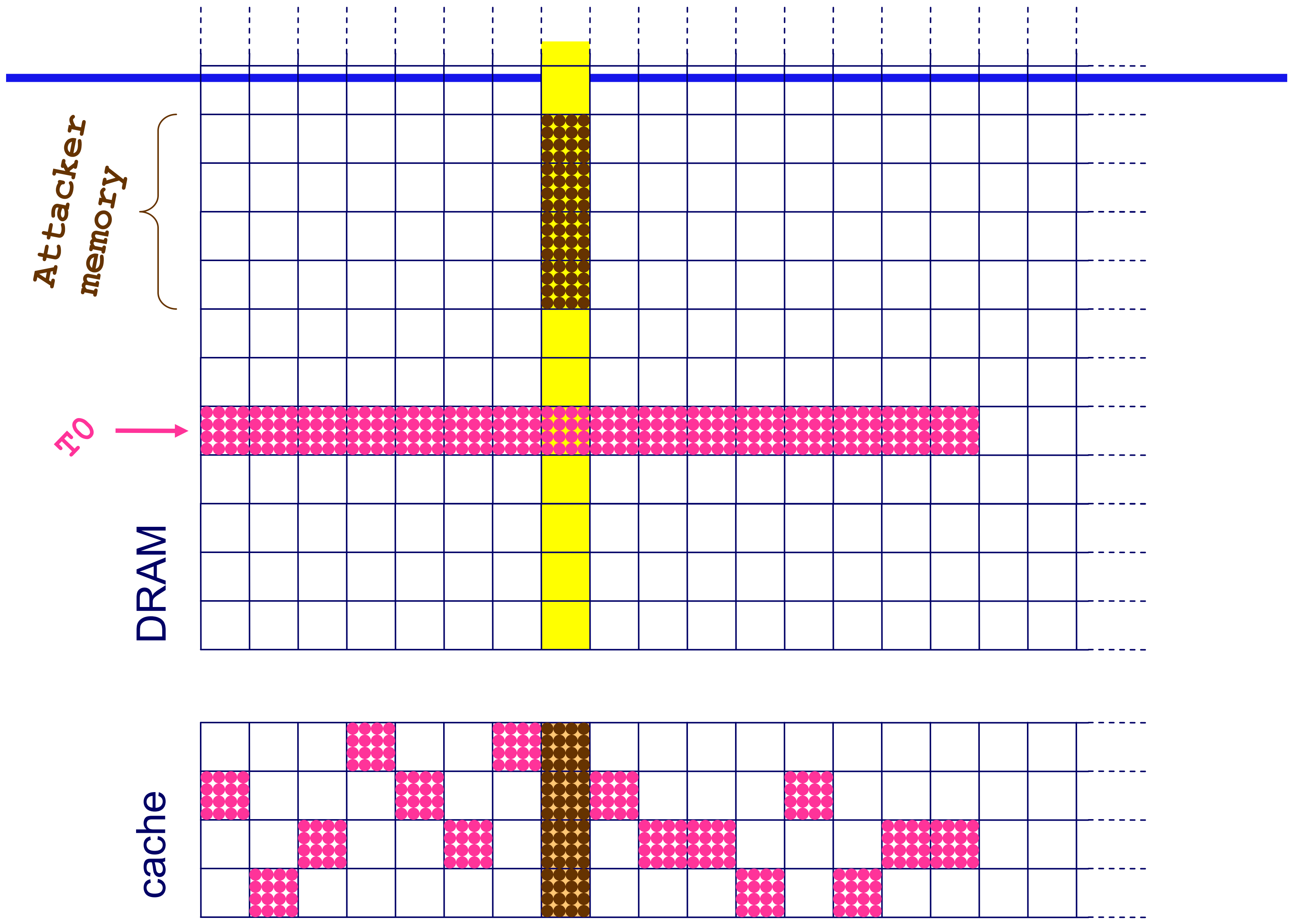
lookup index = plaintext ⊕ key

(and the parameters are favorable to the attack)

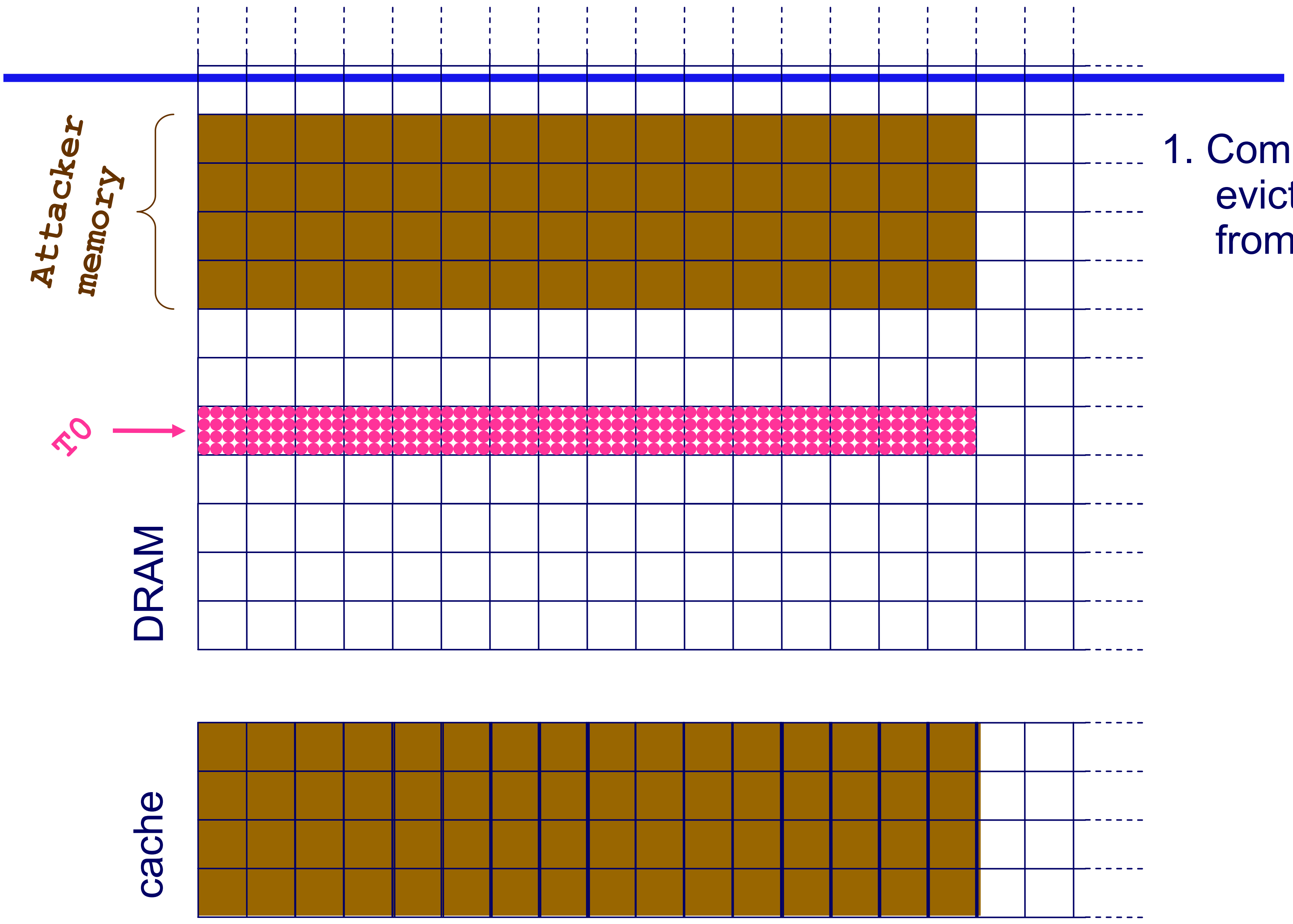
The effect of encryptions on the cache:



Programs compete for cache locations:

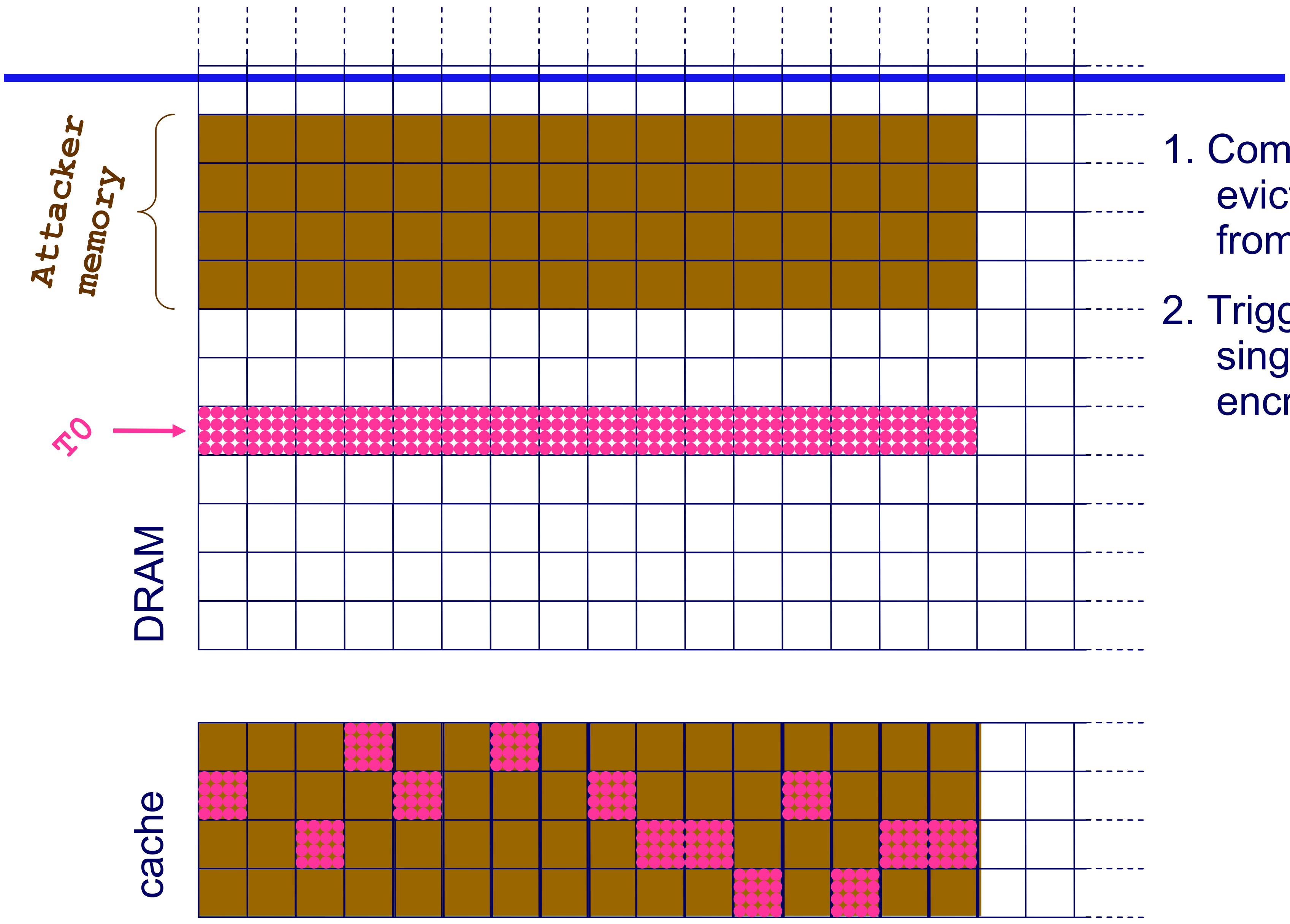


Exploiting the effect of encryption on cache:



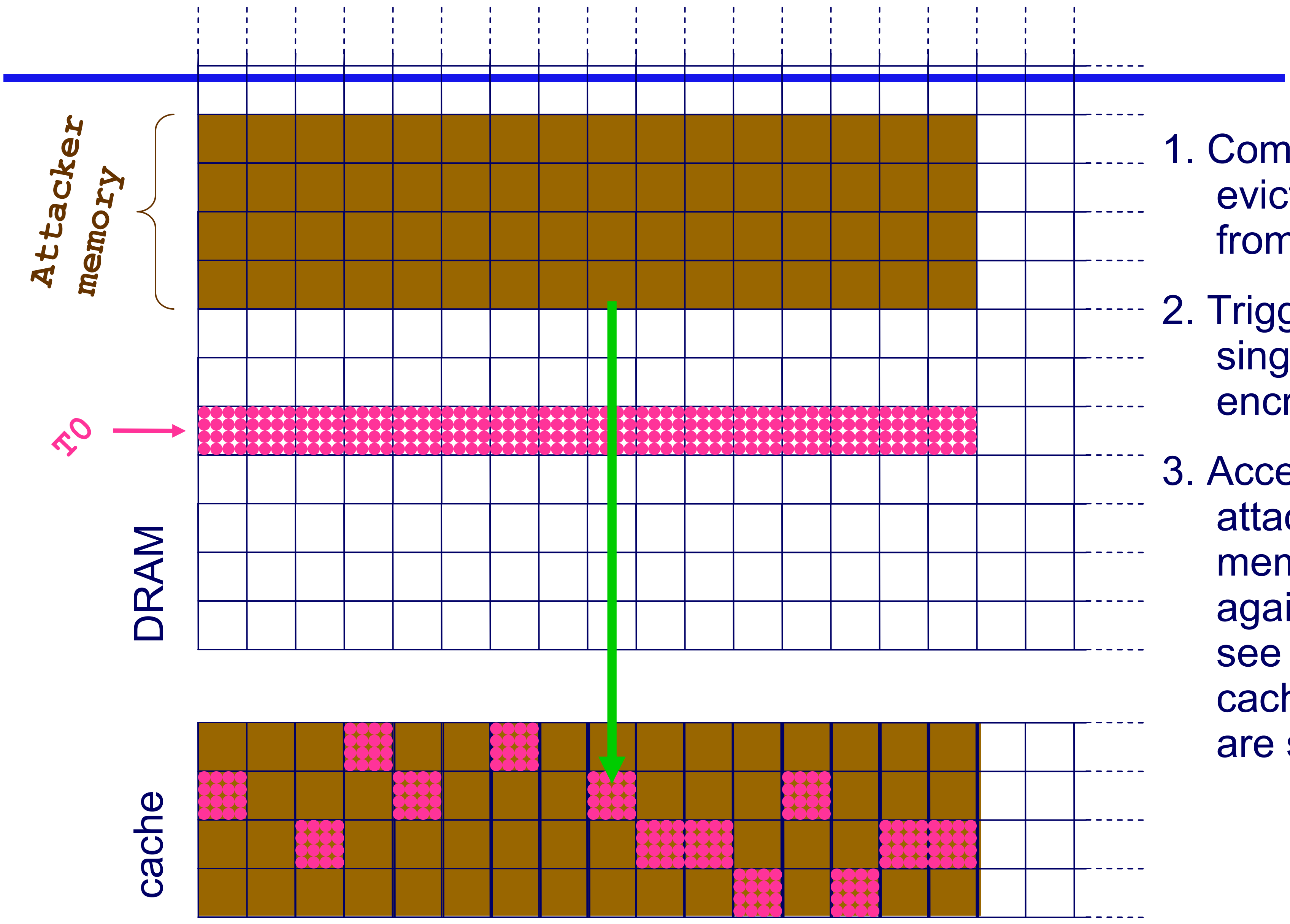
1. Completely evict tables from cache

Measurement via effect of encryption on cache



1. Completely evict tables from cache
2. Trigger a single encryption

Measurement via effect of encryption on cache



1. Completely evict tables from cache
2. Trigger a single encryption
3. Access attacker memory again and see which cache sets are slow

Summary:

- New types of side channel attacks are found and published every few months. Recent discoveries of side channel attacks far outnumber those of classical cryptanalytic attacks

- Side channel attacks are much more practically significant than classical mathematical attacks

- We should completely rethink the issues of how to develop and implement new crypto applications, and how to formally prove their security