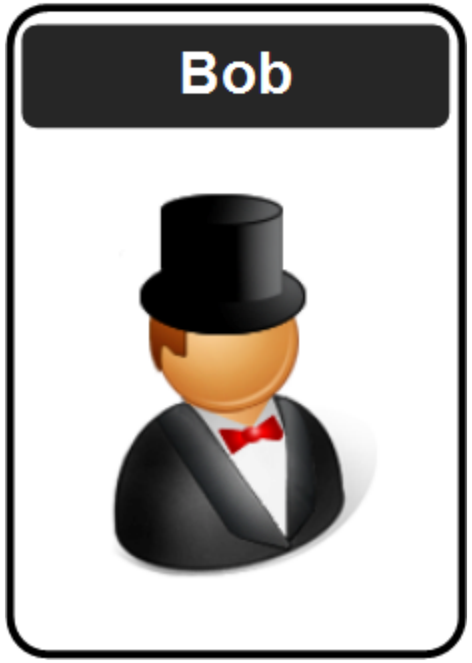


RSA Side Channel Attack

Prof Bill Buchanan OBE, The Cyber Academy

http://asecuritysite.com

CYBER ACADEMY

Say Hello To RSA

- Two primes p, q.
- Calculate N (modulus) as p x q eg 3 and 11. n=33.
- Calculate PHI as (p-1)x(q-1). PHI=20
- Select e for no common factor with PHI. e=3.
- Encryption key [e,n] or [3,33].
- (d x e) mod PHI = 1
- (d x 3) mod 20 = 1
- d= 7
- Decryption key [d,n] or [7,33]

Say Hello To RSA

- Encryption key [e,n] or [3,33].
- Decryption key [d,n] or [7,33]
- Cipher = M^e mod N

eg M=5.

- Cipher = 5^3 mod 33 = 26
- Decipher = C^d mod N
- Decipher = $(26)^7$ mod 33 = 5

RSA

$$Message = Cipher^d \pmod{N}$$

$A^xA^y=A^{x+y}$

$(A^x)^y=A^{xy}$

$(A^x)^2=A^{2x}$

$$5^2 \rightarrow 5^4 \rightarrow 5^8 \rightarrow 5^{16} \rightarrow 5^{32} \rightarrow 5^{64}$$

Square Square Square

```
def exp_func(x, y):
    exp = bin(y)
    value = x

    for i in range(3, len(exp)):
        value = value * value
        print i,":\t",value
        if(exp[i:i+1]=='1'):
            value = value*x
            print i,"*:\t",value
    return value
```

$5^8 \times 5 = 5^9$

1 - Square and multiply

0 - Square

Binary value of 12 is: 0b1100

Bit Result

2 : 25 (square)

2 : 125 (multiply)

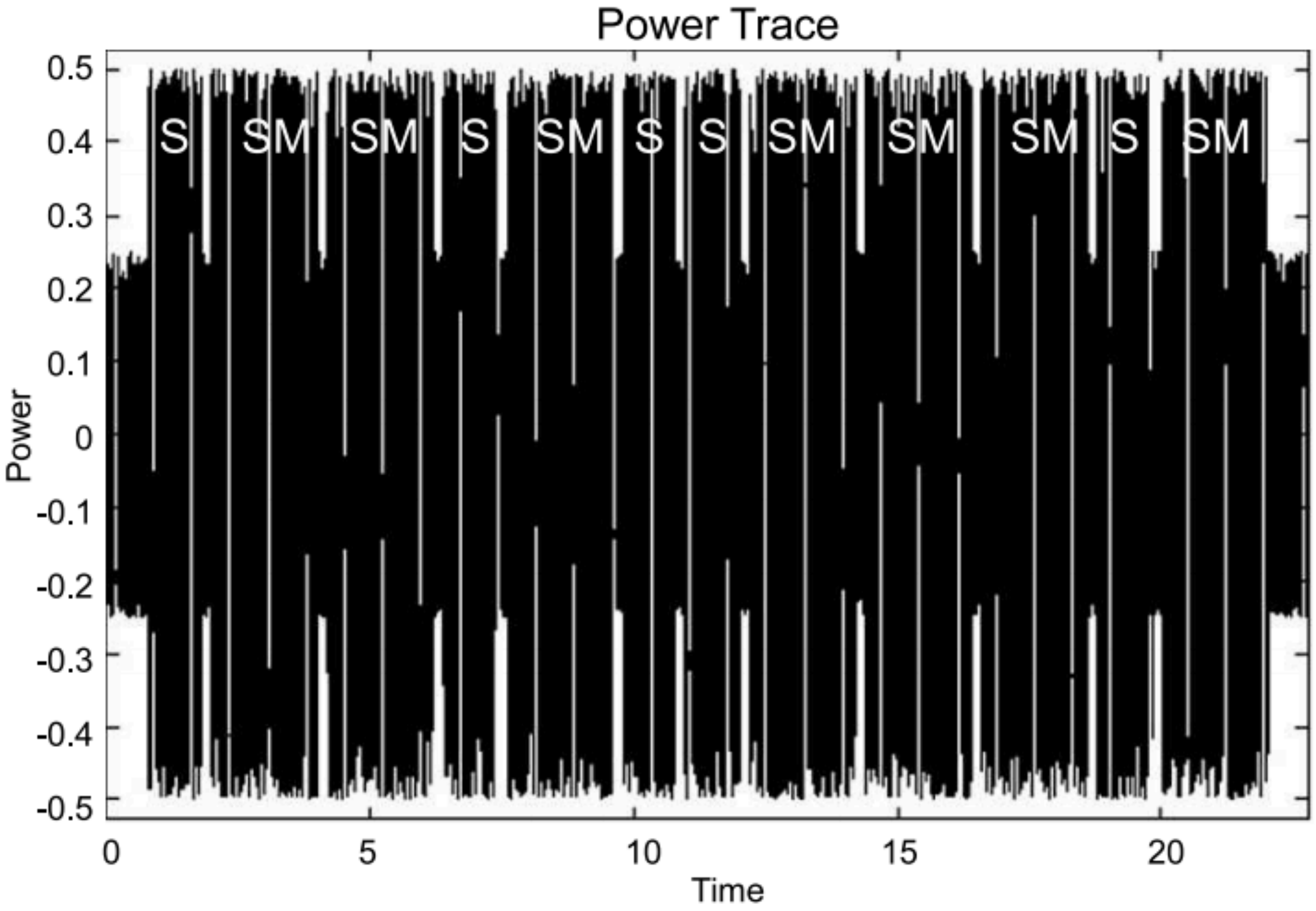
3 : 15625 (square)

4 : 244140625 (square)

Result: 244140625

Link

RSA



0 (S), 1(SM), 1(SM), 0(S), 1SM , 0(S), 0(S), 1(SM), 1(SM), 1(SM), 0(S), 1(SM). We have now revealed virtually all of the bits in the key:

Understanding Cryptography: A textbook for students, Page 197.

[Link]

RSA

One&Done: A Single-Decryption EM-Based Attack on OpenSSL's Constant-Time Blinded RSA

Monjur Alam, *Georgia Tech*
Haider A. Khan, *Georgia Tech*
Moumita Dey, *Georgia Tech*
Nishith Sinha, *Georgia Tech*
Robert Callan, *Georgia Tech*
Alenka Zajic, *Georgia Tech*
Milos Prvulovic, *Georgia Tech*

Abstract

This paper presents the first side channel attack approach that, without relying on the cache organization and/or timing, retrieves the secret exponent from a single decryption on arbitrary ciphertext in a modern (current version of OpenSSL) fixed-window constant-time implementation of RSA. Specifically, the attack recovers the exponent's bits during modular exponentiation from analog signals that are unintentionally produced by the processor as it executes the constant-time code that constructs the value of each "window" in the exponent, rather than the signals that correspond to squaring/multiplication operations and/or cache behavior during multiplicand table lookup operations. The approach is demonstrated using electromagnetic (EM) emanations on two

1 Introduction

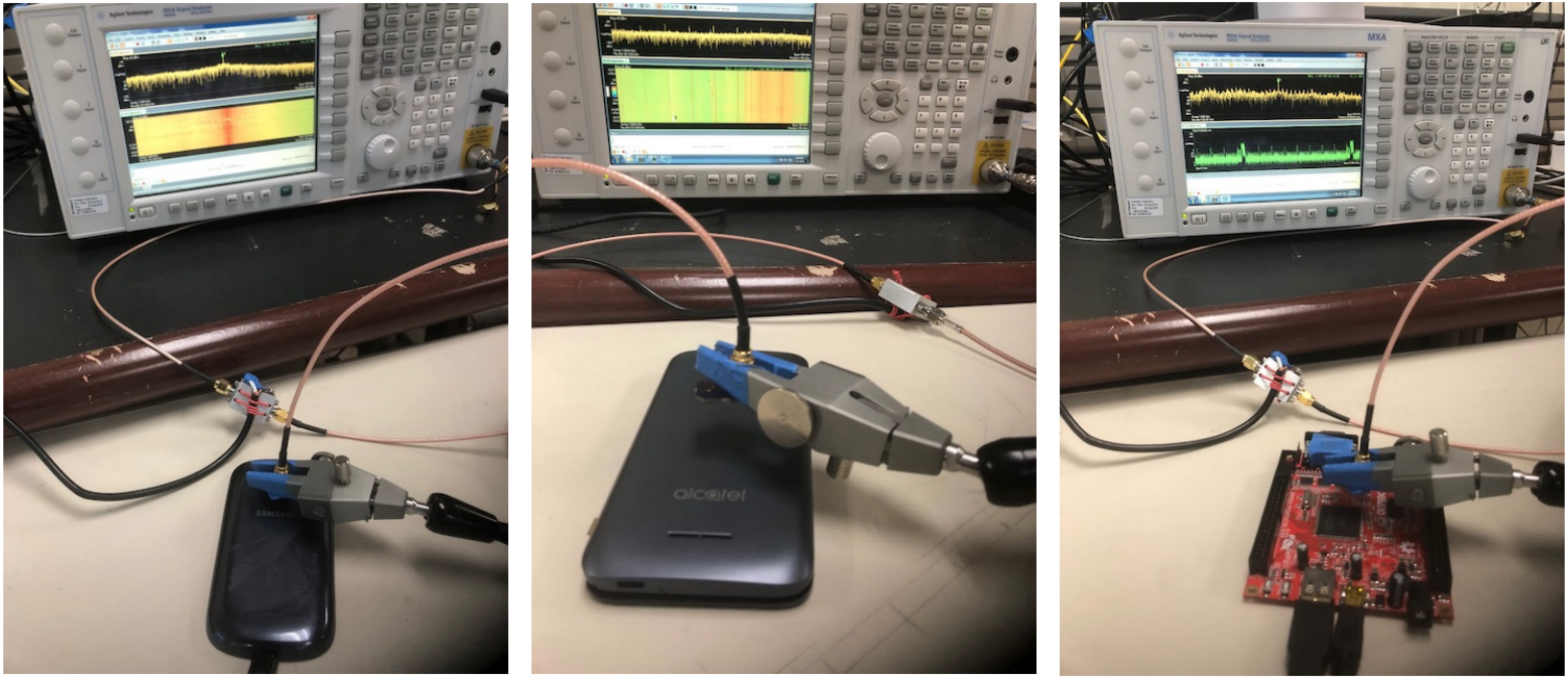
Side channel attacks extract sensitive information, such as cryptographic keys, from signals created by electronic activity within computing devices as they carry out computation. These signals include electromagnetic emanations created by current flows within the device's computational and power-delivery circuitry [2, 3, 14, 21, 33, 46], variation in power consumption [9, 12, 15, 17, 23, 26, 34, 35, 36, 41], and also sound [6, 16, 24, 42], temperature [13, 29], and chasis potential variation [23] that can mostly be attributed to variation in power consumption and its interaction with the system's power delivery circuitry. Finally, not all side channel attacks use analog signals: some use faults [11, 25], caches [8, 43, 44], branch predictors [1], etc.

Samsung Galaxy Centura SCH-S738C smart phone, an Alcatel Ideal smart phone, and an A13-OLinuXino board (Figure 3).

Link

RSA

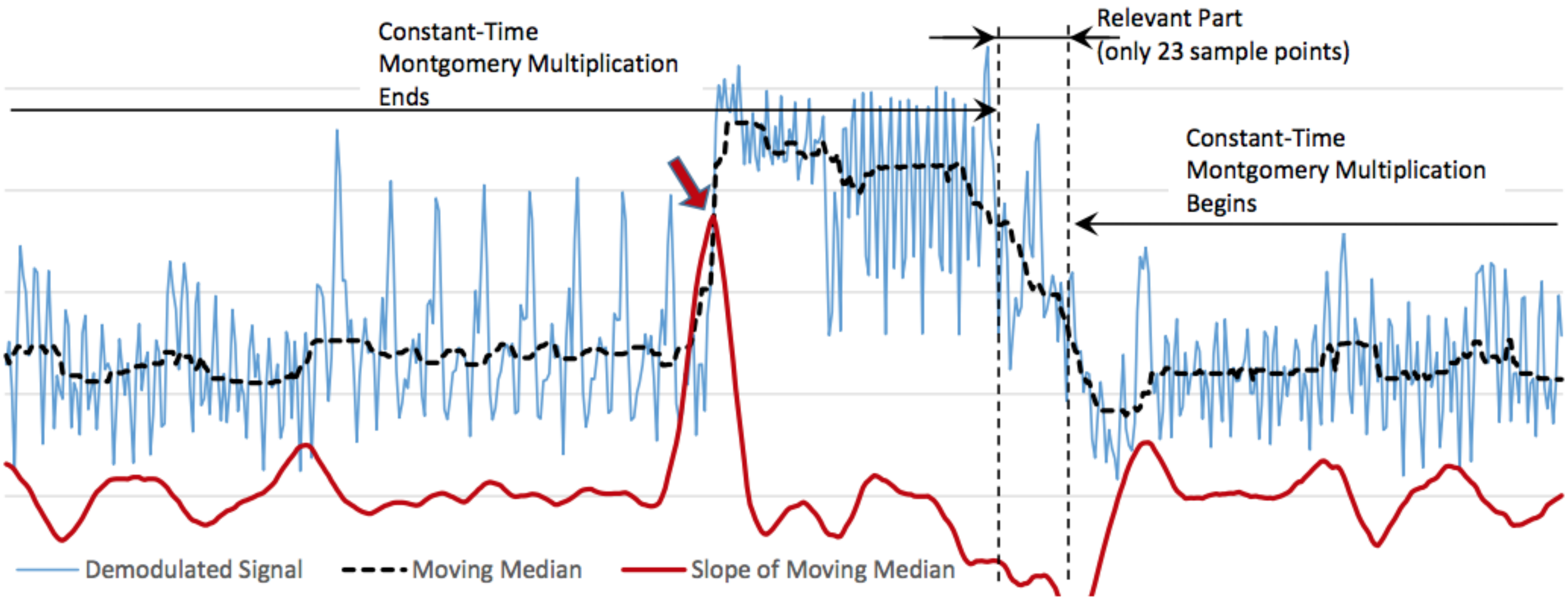
One&Done: A Single-Decryption EM-Based Attack on OpenSSL's Constant-Time Blinded RSA



Samsung Galaxy Centura SCH-S738C smart phone, an Alcatel Ideal smart phone, and an A13-OLinuXino board (Figure 3).

Link

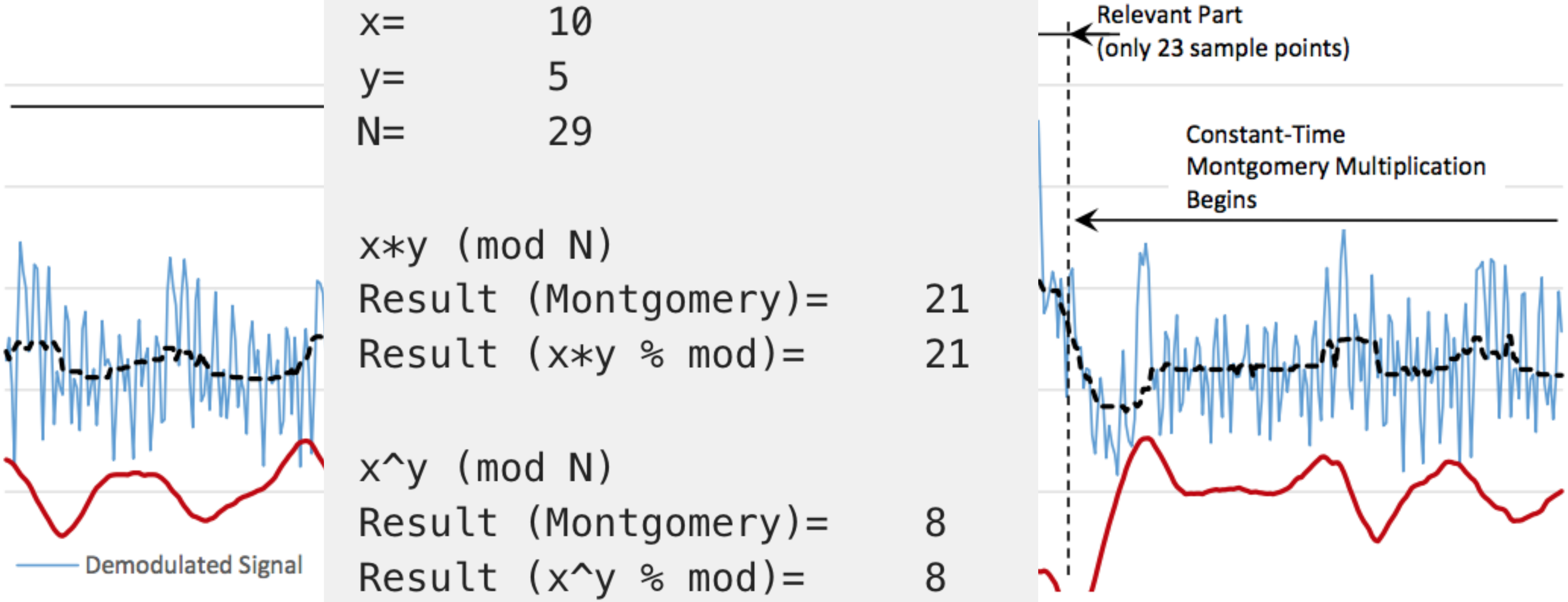
RSA



In **Montgomery reduction** we *add* multiples of *N* in order to simply the multiplication.

Link

RSA

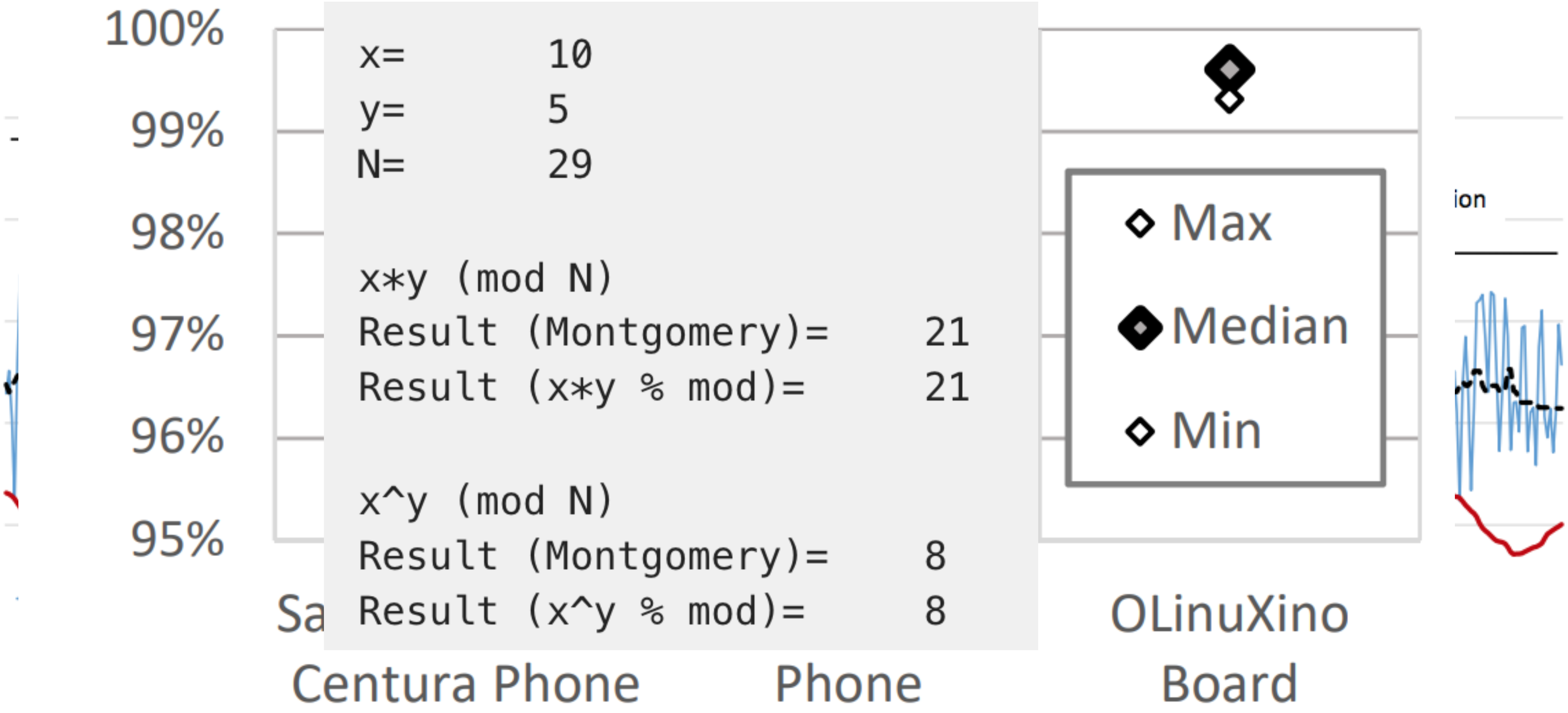



```
x=        10
y=        5
N=        29

x*y (mod N)
Result (Montgomery)=     21
Result (x*y % mod)=      21

x^y (mod N)
Result (Montgomery)=     8
Result (x^y % mod)=      8
```

In **Montgomery reduction** we *add* multiples of *N* in order to simply the multiplication.

Link

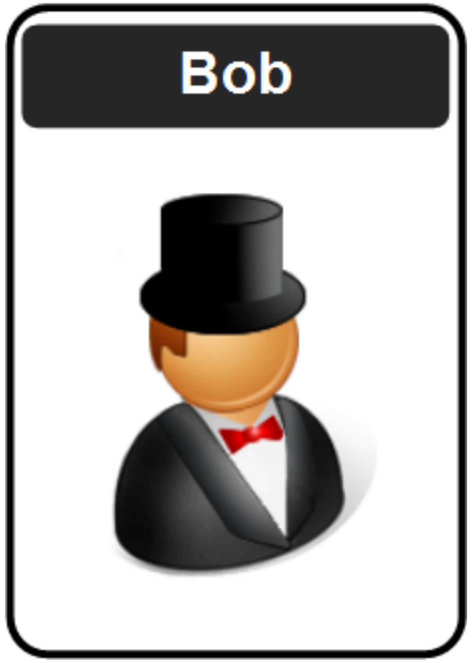
RSA




```
x=         10
y=         5
N=         29

x*y (mod N)
Result (Montgomery)=     21
Result (x*y % mod)=      21

x^y (mod N)
Result (Montgomery)=     8
Result (x^y % mod)=      8
```

simply the multiplication.

Link



RSA Side Channel Attack

Prof Bill Buchanan OBE, The Cyber Academy

http://asecuritysite.com

CYBER ACADEMY