EAP-TLS Smartcards, from Dream to Reality

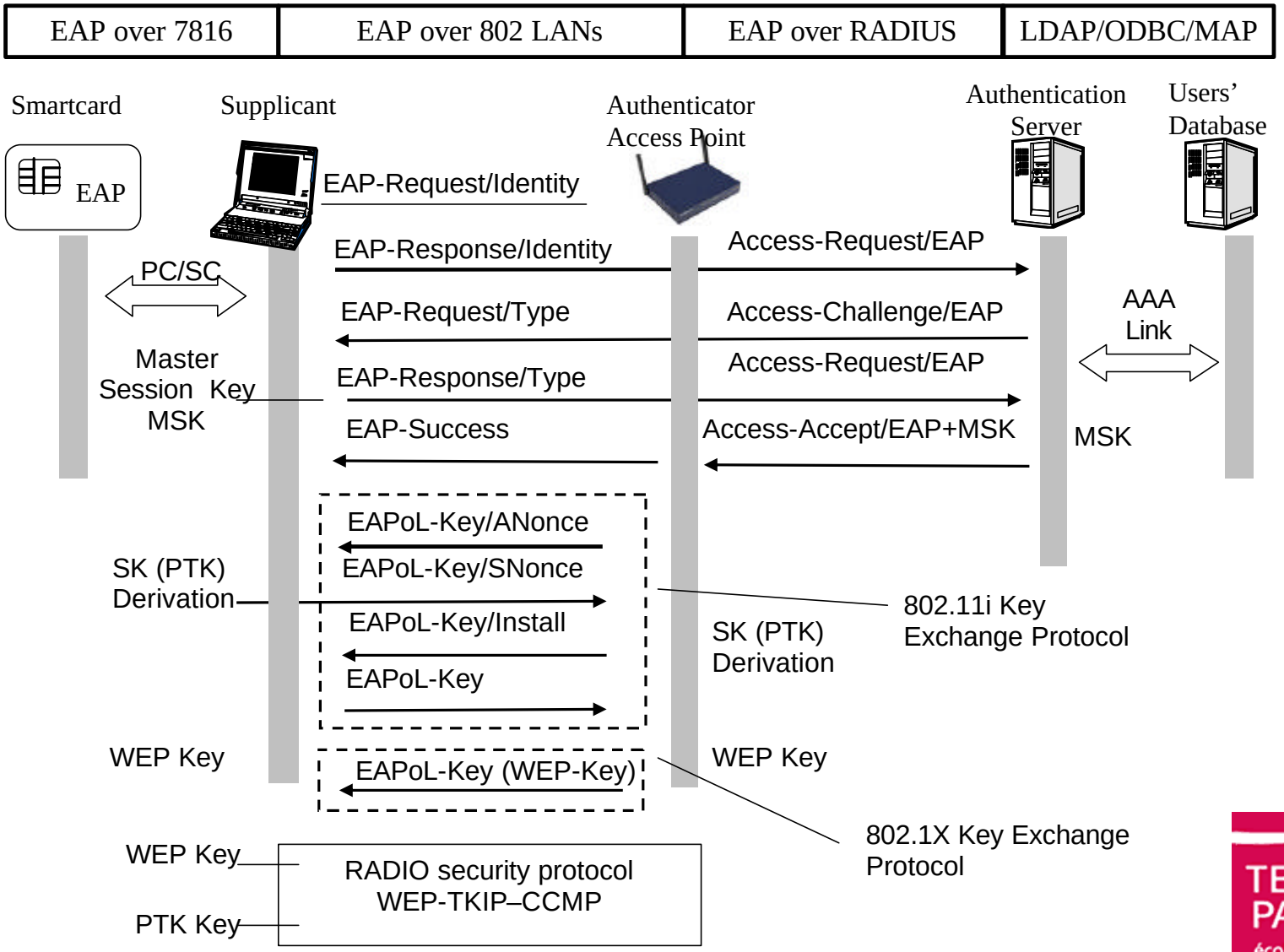
Pascal Urien, Mohamed Badra, Mesmin Dandjinou

4th Workshop on Applications and Services in Wireless Networks
Boston University
Massachusetts, USA
August 9th, 2004

Wi-Fi Security Model

- **User is authenticated according to the IEEE 802.1X model, based on the Extensible Authentication Protocol (EAP, RFC 2284bis)**
- **Authentication is performed between the supplicant (user's PC) and the (RADIUS) authentication server.**
  - The link between the users' database (LDAP, GSM HLR;..) and the RADIUS server is not specified.
- **At the end of this process, a Master Session Key (MSK) is computed by the supplicant and the authenticator**
- **As specified in 802.1X-REV-d8, MSK is a couple of two 32 bytes key named MS-MPPE-Send-Key and MS-MPPE-Recv-Key. These keys are securely sent (by the RADIUS server) to the access point as described in RFC 2548 (*Microsoft Vendor-specific RADIUS attributes*).**
- **A key exchange protocol (IEEE 802.1X, IEEE 802.11i) is used in order share a session key SK (for example a WEP key or a PTK key) between the Access Point an the Supplicant.**
- **According to the radio security protocol used between the Access Point and the Supplicant (WEP, TKIP, CCMP) various key are deduced from SK in in order to realize,**
  - 802 Frames privacy (data encryption)
  - 802 Frames integrity
  - 802 Frames authentication (data encryption + data integrity ⇔ symmetric signature).

Wi-Fi Security Model (suite)



THE EAP smartcard

- **What is the EAP smartcard ?**
  - A smartcard that processes EAP messages
  - It supports multiple authentication methods
    - EAP-SIM, EAP-TLS, EAP-MSCHAPv2, others
    - First EAP-TLS smartcard is operational since June 17th 2004.
- **What doest it look like ?**
  - It is an application written for Javacards.
- **Specified by an IETF draft**
  - “EAP-Support in smartcard”
    - draft-urien-eap-smartcard-05.txt
- **The EAP smartcard won two awards**
  - Sesame 2003, “Best Technological Innovation”, cartes’2003 exhibition, Paris, November 2003
  - Card Technology Magazine, Breakthrough Awards 2004, “Innovation”, CardTech/SecureTech exhibition, Washington DC, April 2004.
- **Are smartcards performances sufficient ?**
  - Usually smart cards include crypto-processors that compute the RSA 2048 bits algorithm in less than 0,5s.
  - Commercial Javacards memory size are around 32-64 Kb (available for code byte storage).
    - The size of an X509 certificate is about 1kb
    - As an illustration EAP-TLS applet size (processing EAP and TLS protocols) is around 20Kb.
  - New generation of smartcards based on FLASH technology, supports one megabyte of memory.

EAP-TLS Cryptographic costs for RC4-MD5 crypto suite

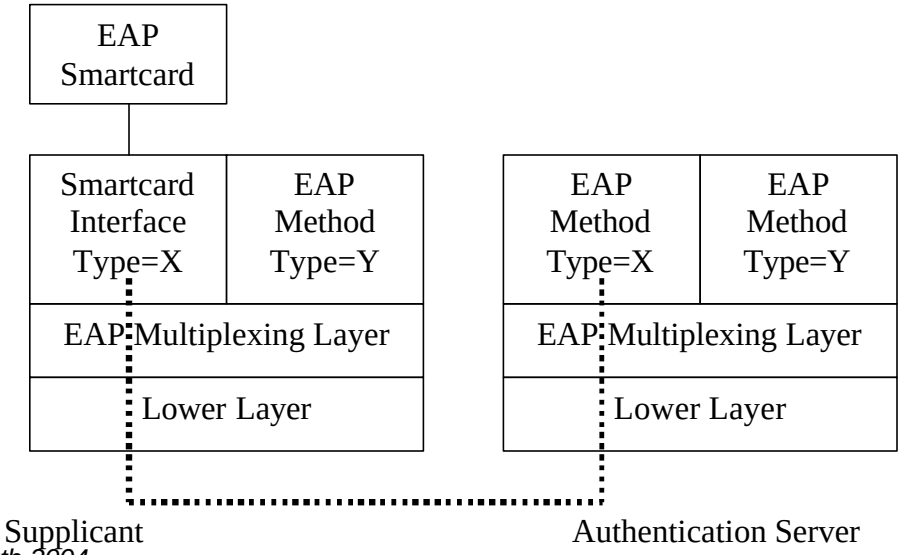
- **N = PRF, i= N/16, j=N/20**
  - PRF(2i+1 x HMAC-MD5, 2j+1 x HMAC-SHA-1)
  - HMAC-MD5 = 2 x MD5 (5 x blocks)
  - HMAC-SHA1 = 2 x SHA1 (5 x blocks)
- **Server Hello Message Processing**
  - Cost: 3xRSA, 500 x MD5-blocks, 500 x SHA1-blocks, 1xRC4
    - 3 x RSA
    - 3 x PRF = 20 x HMAC-MD5 (10 blocks)
    - 3 x PRF = 20 x HMAC-SHA1 (10 blocks)
    - 3 x MD5 (100 blocks)
    - 3 x SHA-1 (100 blocks)
    - 1 x RC4 (32 bytes)
- **Server Finished Message Processing**
  - Cost: 130 x MD5-blocs, 130 x HMAC-blocs, 1 x RC4
    - 2 x PRF = 13 x HMAC-MD5 (10 blocks)
    - 2 x PRF = 13 x HMAC-SHA1 (10 blocks)
    - 1 x RC4 (32 bytes)
- **Device A </block> = 23,5 ms**
  - Server Hello Processing > 1000 * 23,5 = 23,5s
- **Device B </block> = 11ms**
  - Server Hello Processing > 1000 * 11,0 = 11,0s

EAP Smartcard Services

- **The operating system interface.**
  - **Identity** is a pointer to an authentication triplet (EAP-ID, EAP-Type, Credentials) stored in the EAP-Smartcard.
  - Smartcard may manage several network accounts, the OS performs an **identity discovery process** in order to browse its content.
  - A **profile** is a collection of information, such as EAP-ID, EAP-Type, protocol version, list of preferred SSIDs, root certificates, user's certificates, or every data meaningful for operating systems in order to interoperate with the card or to select the right access point when multiple wireless networks are available.
- **The network interface.**
  - EAP messages are processed by the smartcard. At the end of the authentication method, a **Session Key** (PMK) is computed.
- **The user/issuer interface.**
  - The smartcard is protected by **two PIN codes** (Personal Identification Number), one is managed by the card bearer and the other by the card issuer. For example if the user's PIN is activated, the smartcard is locked (and can't be used) after three wrong PIN values presentation.
- **The management/personalisation interface.**
  - This service updates information (identities) stored in the smartcard.

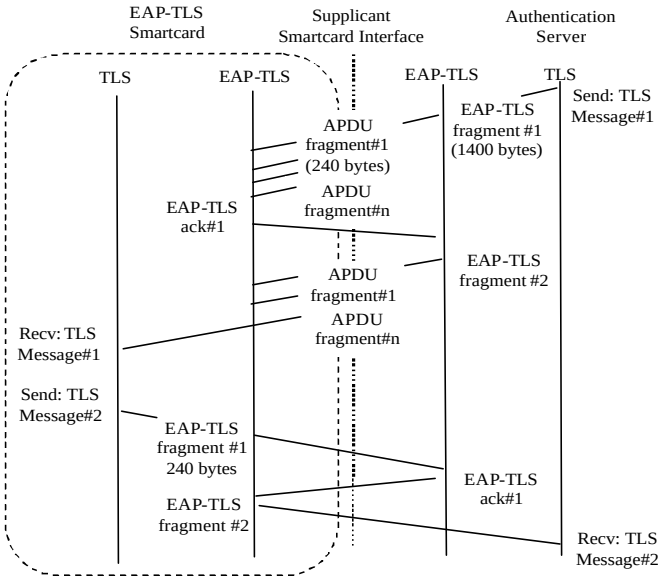
Integrating EAPSC in Operating Systems

- **EAP implementation conceptually consists of the three following components**
  - *Lower layer*. The lower layer is responsible for transmitting and receiving EAP frames between the peer and authenticator
  - *EAP multiplexing layer*. The EAP layer receives and transmits EAP packets via the lower layer, implements duplicate detection and retransmission and delivers and receives EAP messages to and from EAP methods.
  - *EAP method*. EAP methods implement the authentication algorithms and receive and transmit EAP messages via the EAP layer. Since fragmentation support is not provided by EAP itself, this is the responsibility of EAP methods.
- An EAP smartcard implements an EAP method and works in cooperation with a smartcard interface entity, which sends and receives EAP messages to/from this component. The simplest form of this interface is a software bridge that transparently forwards EAP messages to smartcard. According to EAP methods complexity and smartcard computing capacities, protocol sub-sets, which do not deal with security features may be computed by the smartcard interface entity.



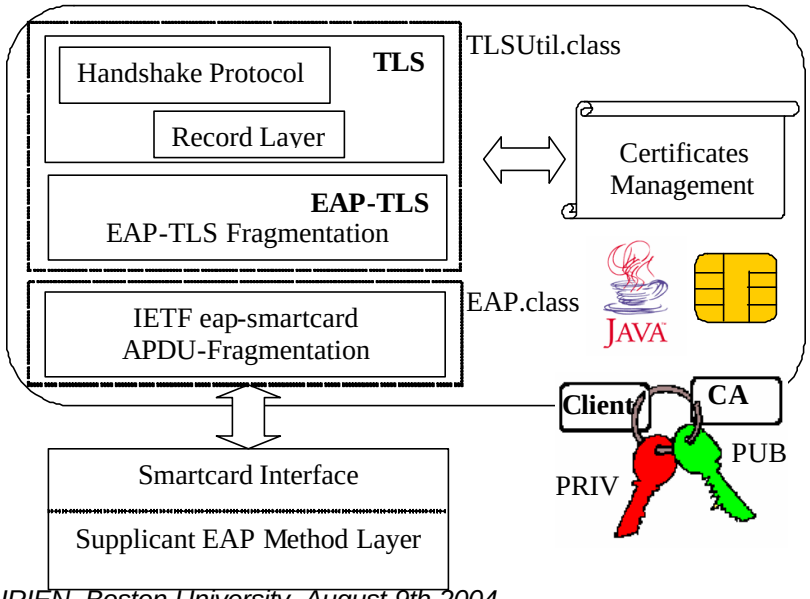
Double Segmentation

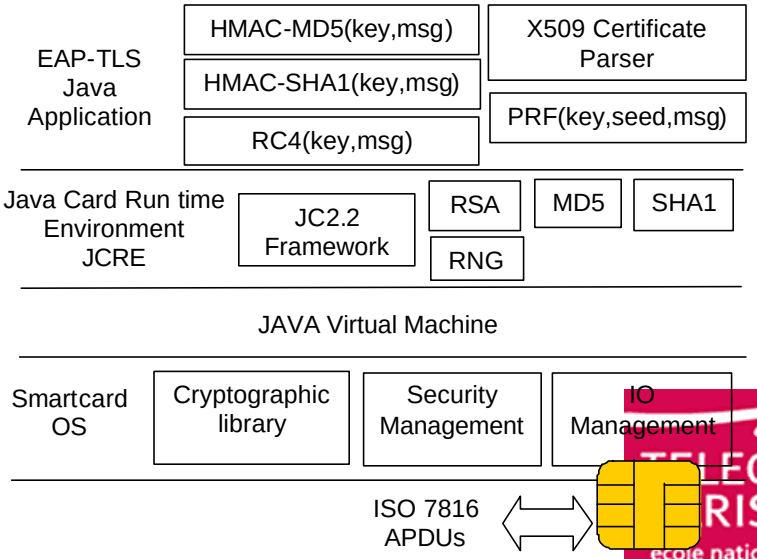
- According to a TLS record may be up to 16384 bytes in length, a TLS message may span multiple TLS records, and a TLS certificate message may in principle be as long as 16MB. Furthermore the group of EAP-TLS messages sent in a single round may thus be larger than the maximum LAN frame size. Therefore EAP-TLS [6] introduces a segmentation process that splits TLS messages in smaller blocs, acknowledged by the recipient.
- The RADIUS server generates acknowledgement requests and the supplicant acknowledgment responses.
- A double segmentation mechanism is necessary in order to forward TLS packets to smartcard. These messages are divided in smaller segments, whose size is typically 1400 bytes, and than encapsulated in EAP-TLS packets.



EAP-TLS Javacards

- **JavaCard 2.x platform natively provides essential cryptographic services that are required by the TLS protocols; in particular:**
  - Random number generation.
  - MD5 and SHA1 digest functions.
  - RSA public key encryption and decryption.
  - RSA private key encryption and decryption.
  - DES or 3DES ciphering.
- **However some additional facilities that are not currently available in JC platforms are provided by the EAP-TLS application. For example:**
  - Keyed-hashing procedures (HMAC-MD5 and HMAC-SHA1).
  - The pseudo random function (PRF) defined by the TLS protocol.
  - The RC4 algorithm, which is often used by the TLS record layer.
  - An X509 certificate parser required for signature analysis and public key extraction.
- **Total code size, #22KB (8 KB of data)**

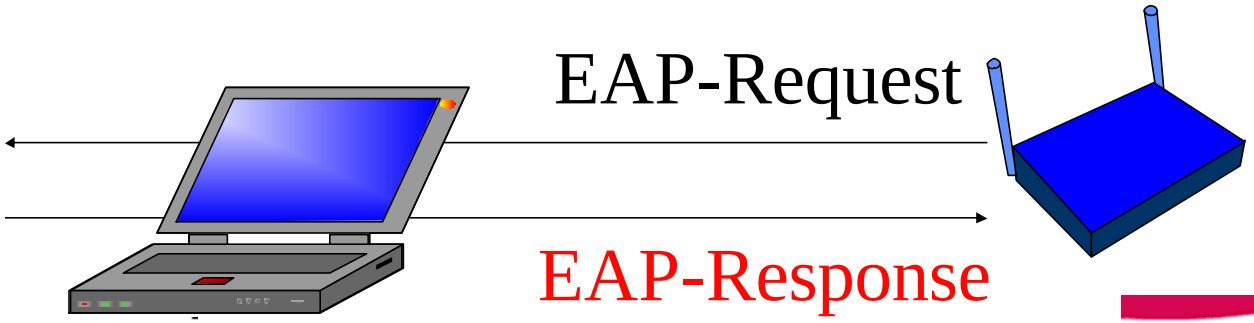




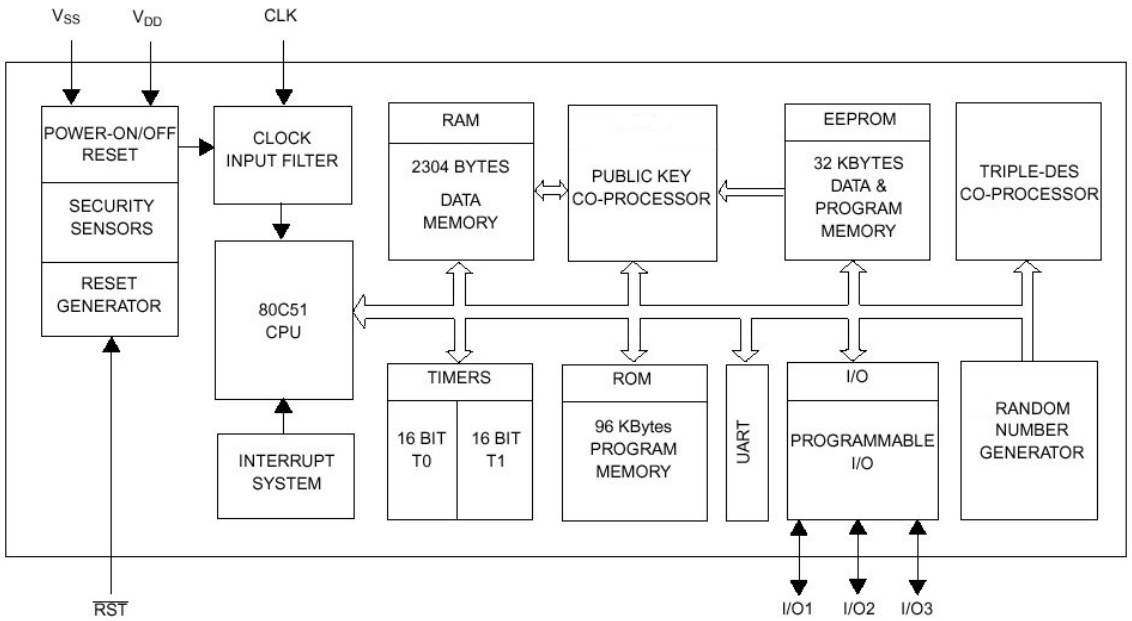
Performances Issue

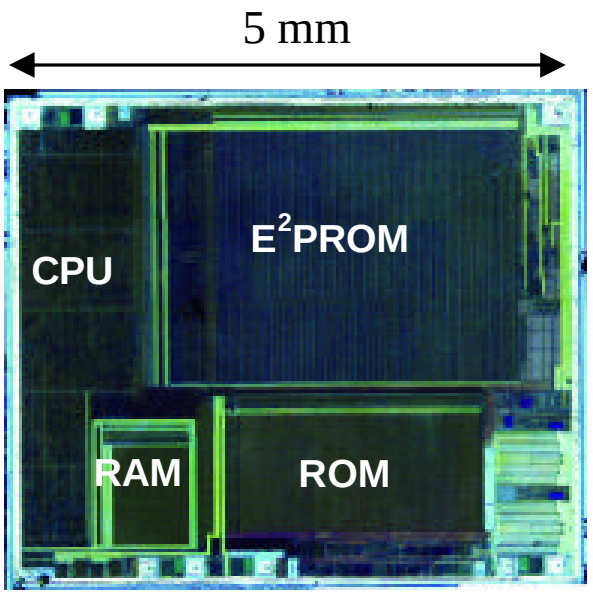
- **What performance is needed ?**
  - IEEE 802.1x, 2001, 8.5.4.1.2

  "*txPeriod. The initialization value used for the txWhen timer. Its default value is **30 s**; it can be set by management to any value in the range from 1 to 65535 s*."
- **Target.**
  - **Computing each sub-part of the EAP-TLS protocol in less than 30 seconds !**



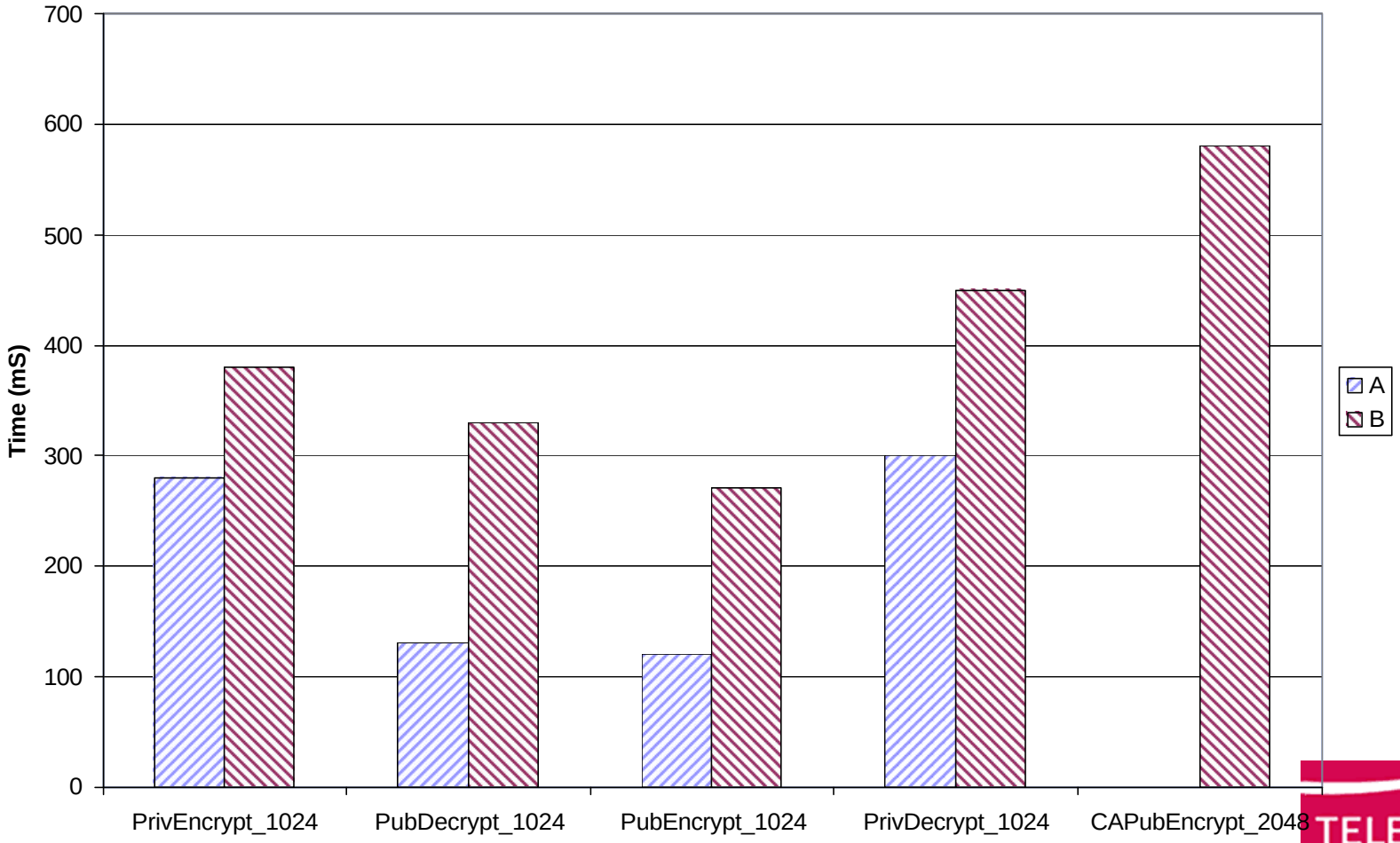
Tamper-Resistant Microcontrollers





Device	CPU	RAM bytes	ROM Kbytes	E²PROM Kbytes	Max. Clock	Max Data Rate	RSA Processor	RNG
A	8 bits	2304	96	32	10 MHz	424 kbit/s	1088 bits	yes
B	8 bits	4096	96	34	8 MHz	1000 kbit/s	4032 bits	no

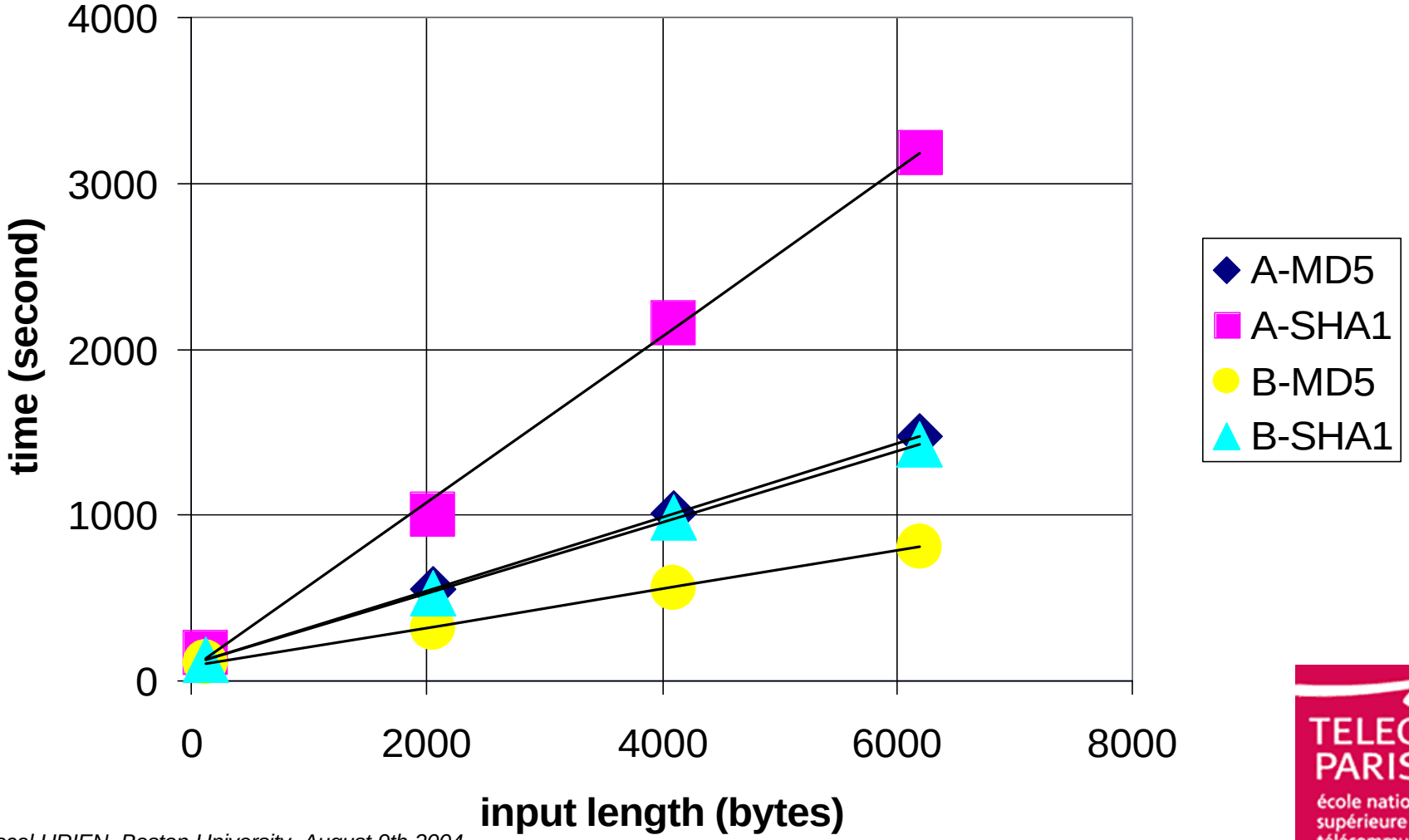
RSA computing time



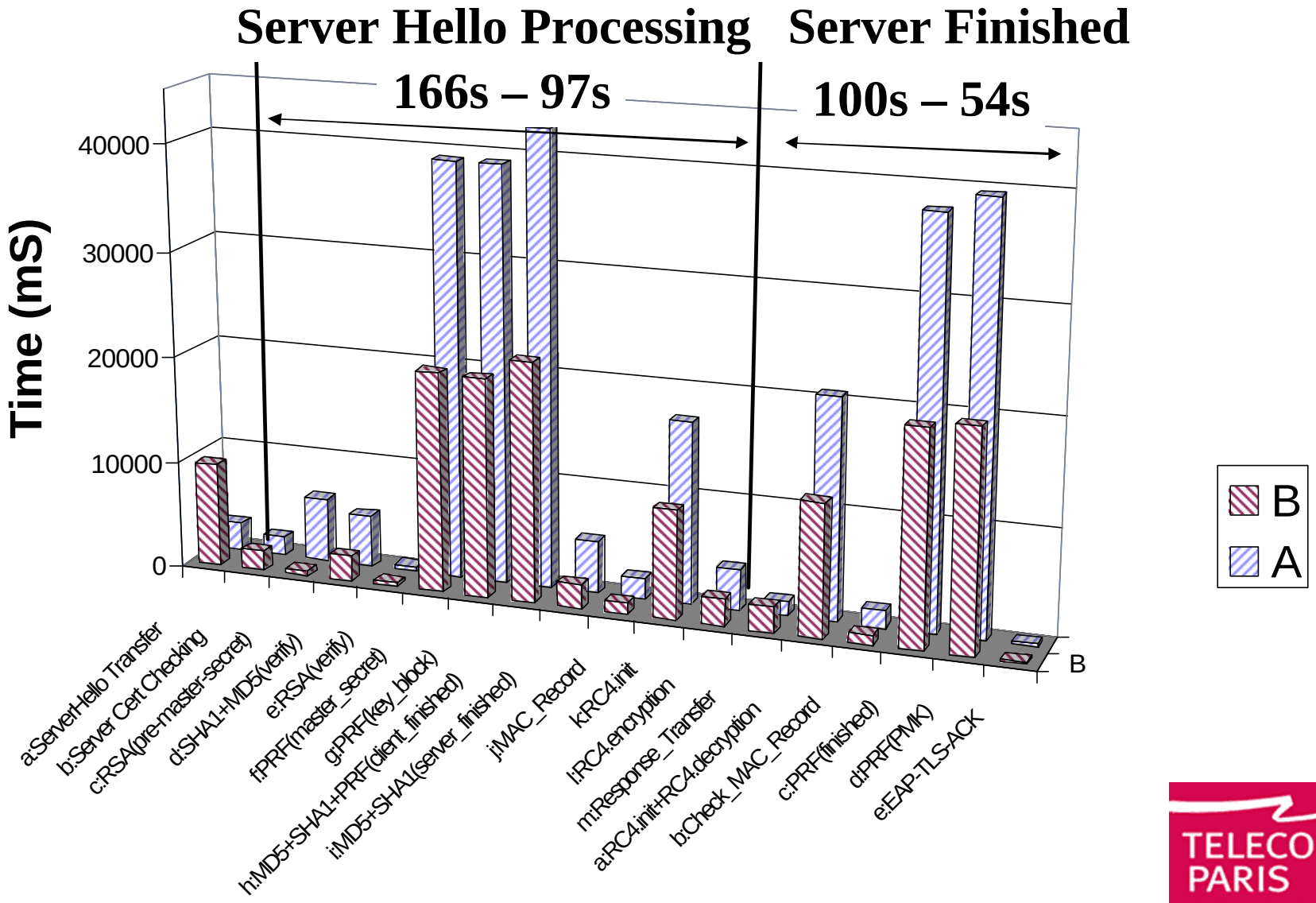
MD5 & SHA-1 Performances

From “SSL and TLS”chapter 1,p32
OPENSSL FreeBSD Pentium II 400
MD5…. 65 MB/s
SHA1…31 MB/s

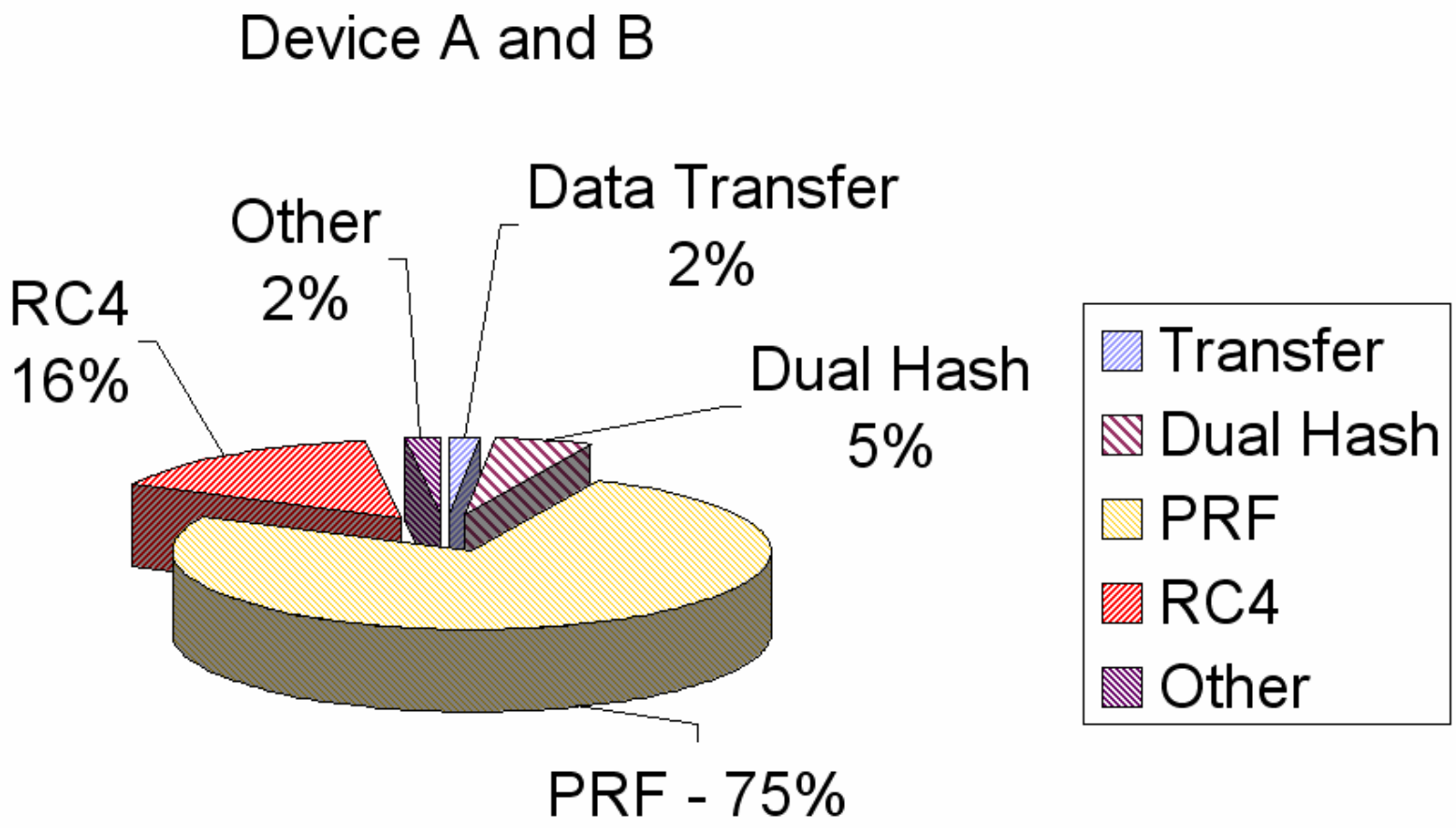
A: 23,5ms/block
B: 11,0ms/block



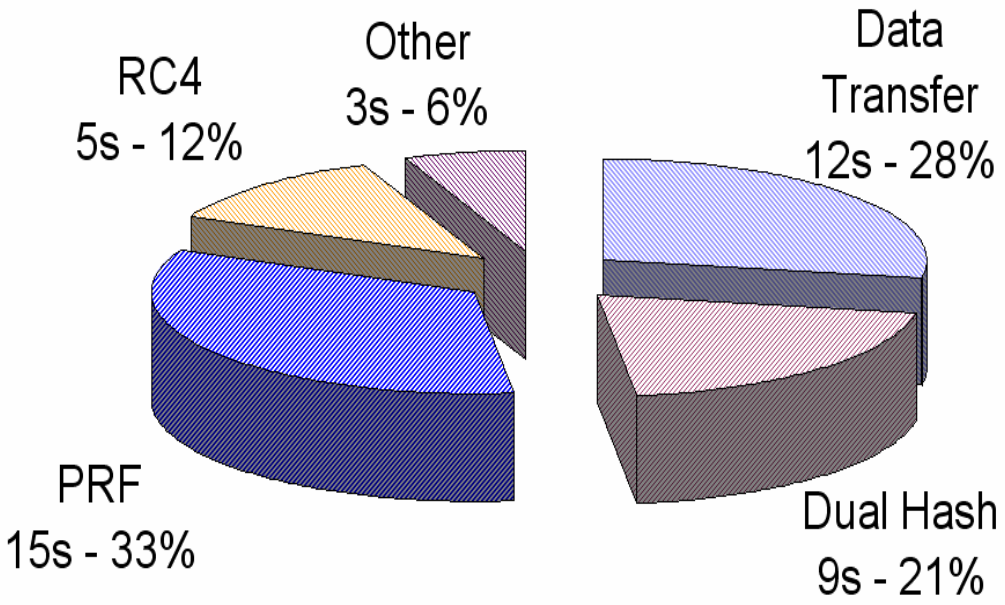
Version 1 performances



Version 1 performances



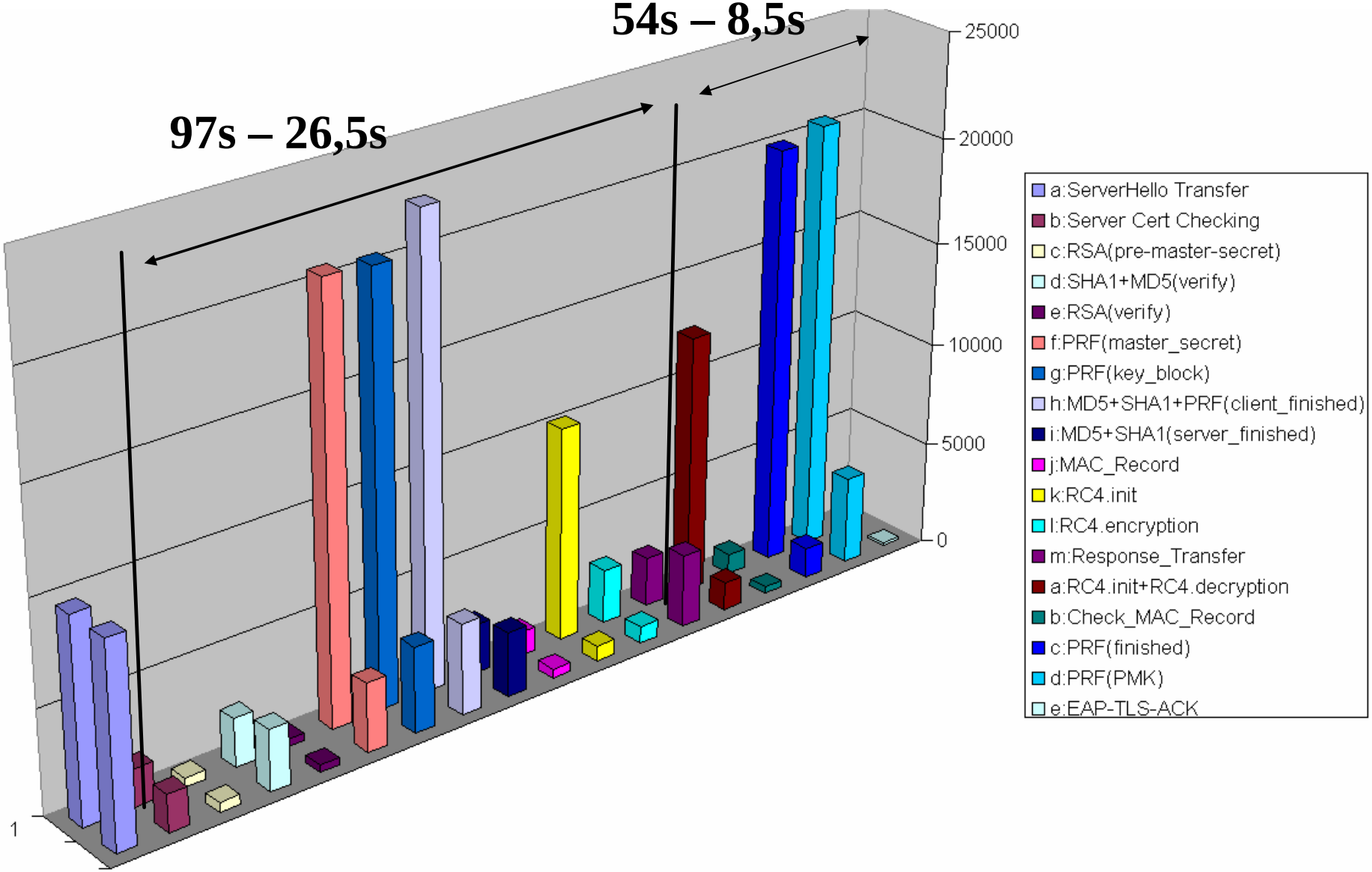
Version 2 optimization



Server Hello Message Transfer to SC…….. 10,0s
Server Hello Message Processing.....24,0s
SC Response Transfer……2,5s
Server Finished Message Processing …….8,5s

Total……..45,0s

Version 2 optimization



Real performances with 1024 RSA keys

- EAP-TLS-Start / Client-Hello……..1,5s
- Server- Hello 1st frag / EAP-TLS-ACK……2,0s
- Server Hello 2nd frag / Client Response 1st frag…. .26,0s
- Server EAP-ACK / Client Response 2nd Frag…….1,0s
- Server Finished / EAP-TLS-ACK……7,0s

Conclusion