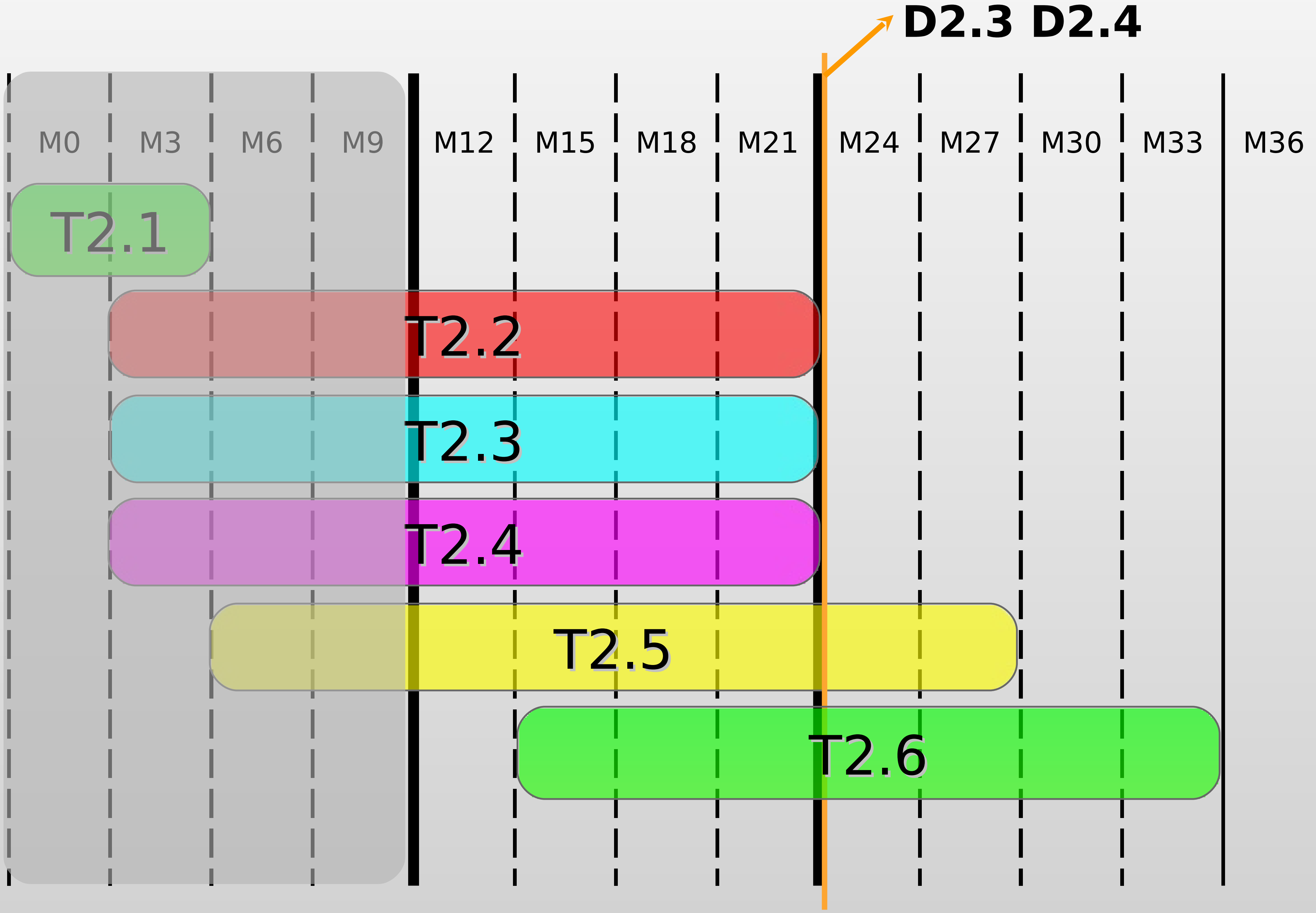
Second year review
WP2 overview
SW-based Method

Trento - October 17th, 2008

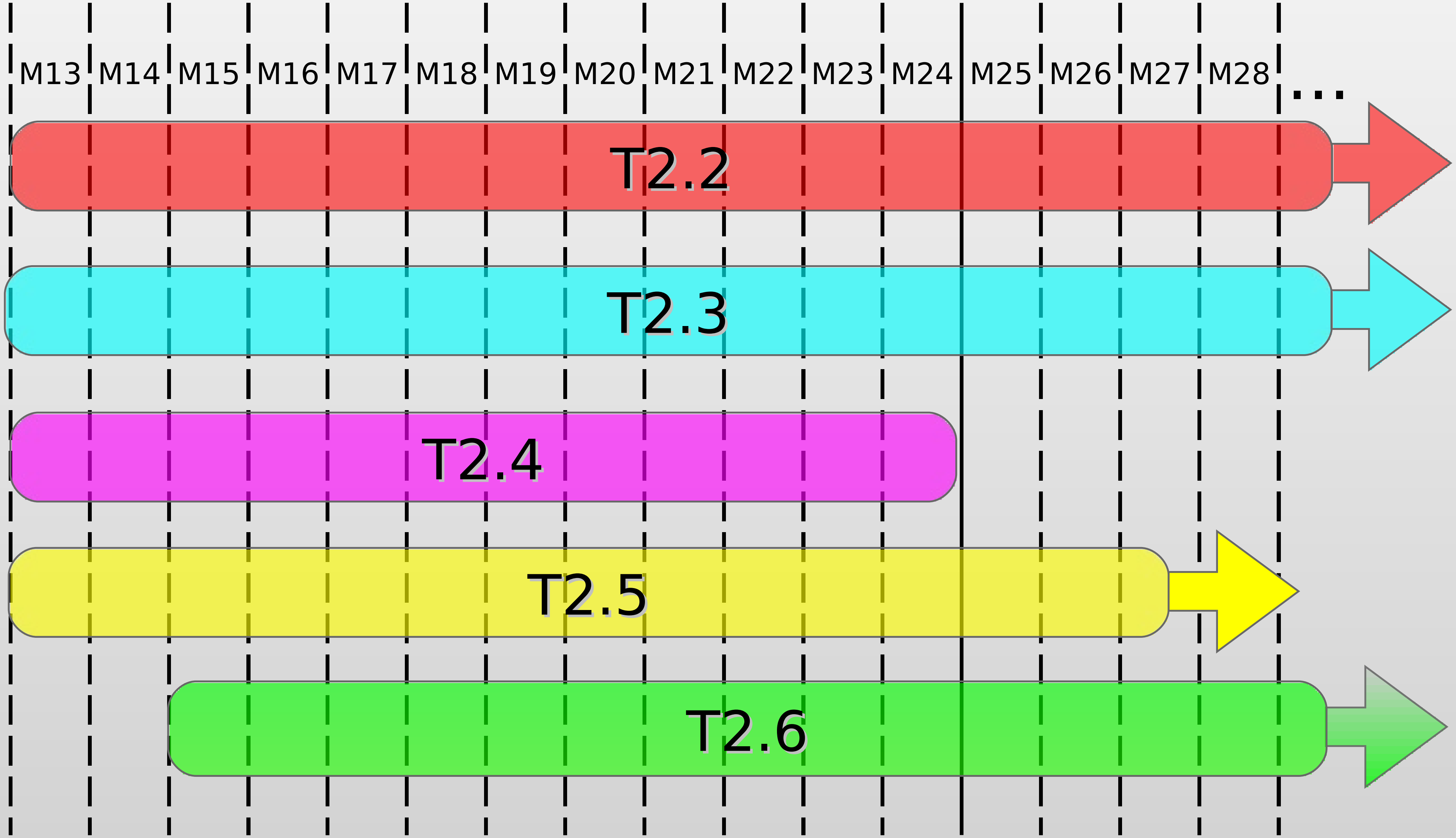
Goal

- To investigate software-only methodologies for remote entrusting implementation

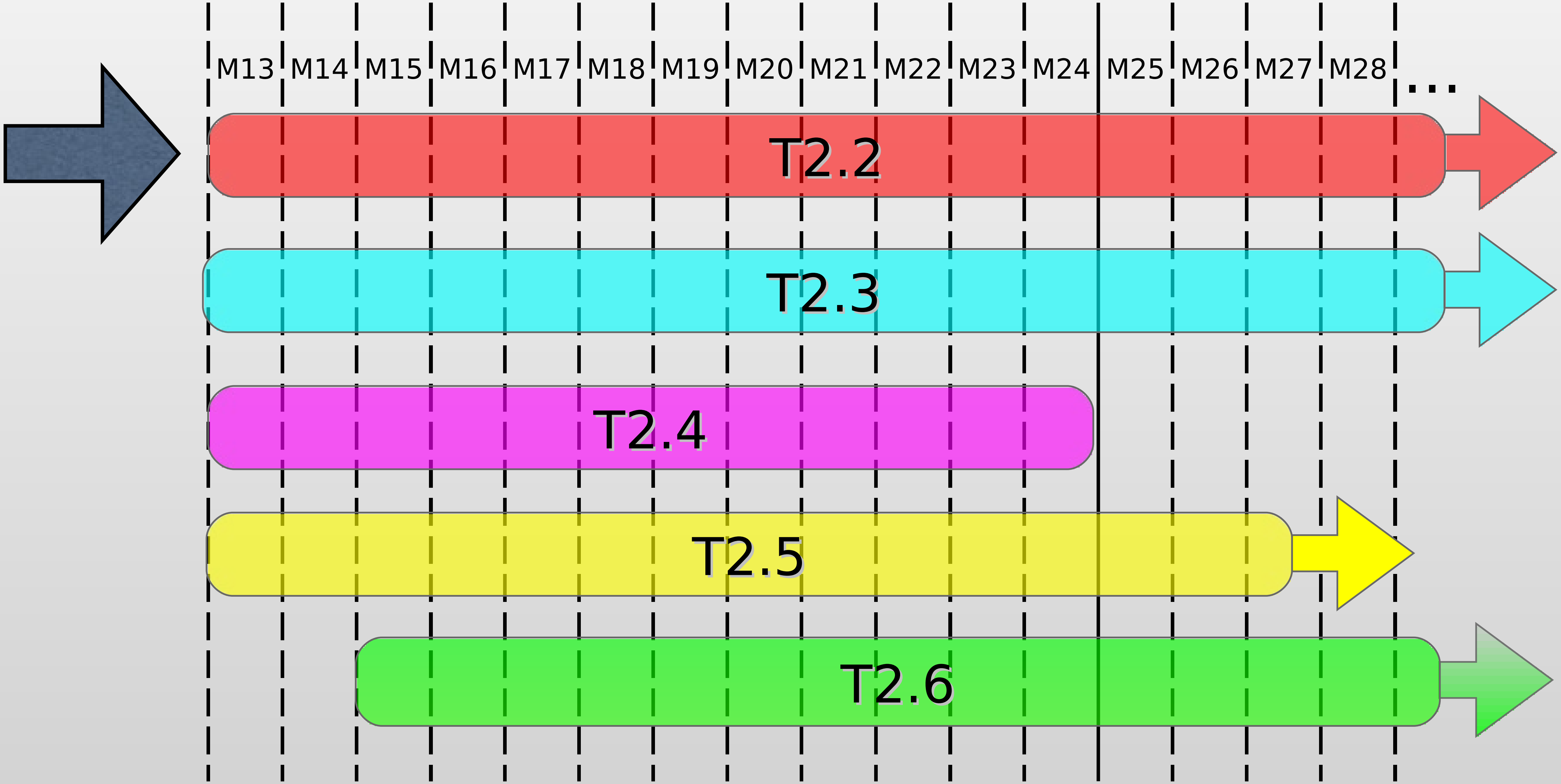
Tasks


D2.3 D2.4
M0
M3
M6
M9
M12
M15
M18
M21
M24
M27
M30
M33
M36
T2.1
T2.2
T2.3
T2.4
T2.5
T2.6

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

T2.2

Secure interlocking and authenticity checking

- Definition of software techniques to securely combine an application with different protection and authentication mechanisms

T2.2

Secure interlocking and authenticity checking

- Remote Invariants Monitoring (POLITO)
- Remote Control-Flow Checking (POLITO)
- White-Box Remote Procedure Call (UNITN,KUL)
- Barrier Slicing (UNITN)

T2.2

Secure interlocking and authenticity checking

- Remote Invariants Monitoring (POLITO)
- Remote Control-Flow Checking (POLITO)

Remote monitoring of program state, or program execution flow

Program traces sent from untrusted to trusted node

T2.2

Secure interlocking and authenticity checking

Program execution through an obfuscated virtual machine

Analysis of the application becomes not possible

(POLITO)

- White-Box Remote Procedure Call (UNITN, KUL)
- Barrier Slicing (UNITN)

T2.2

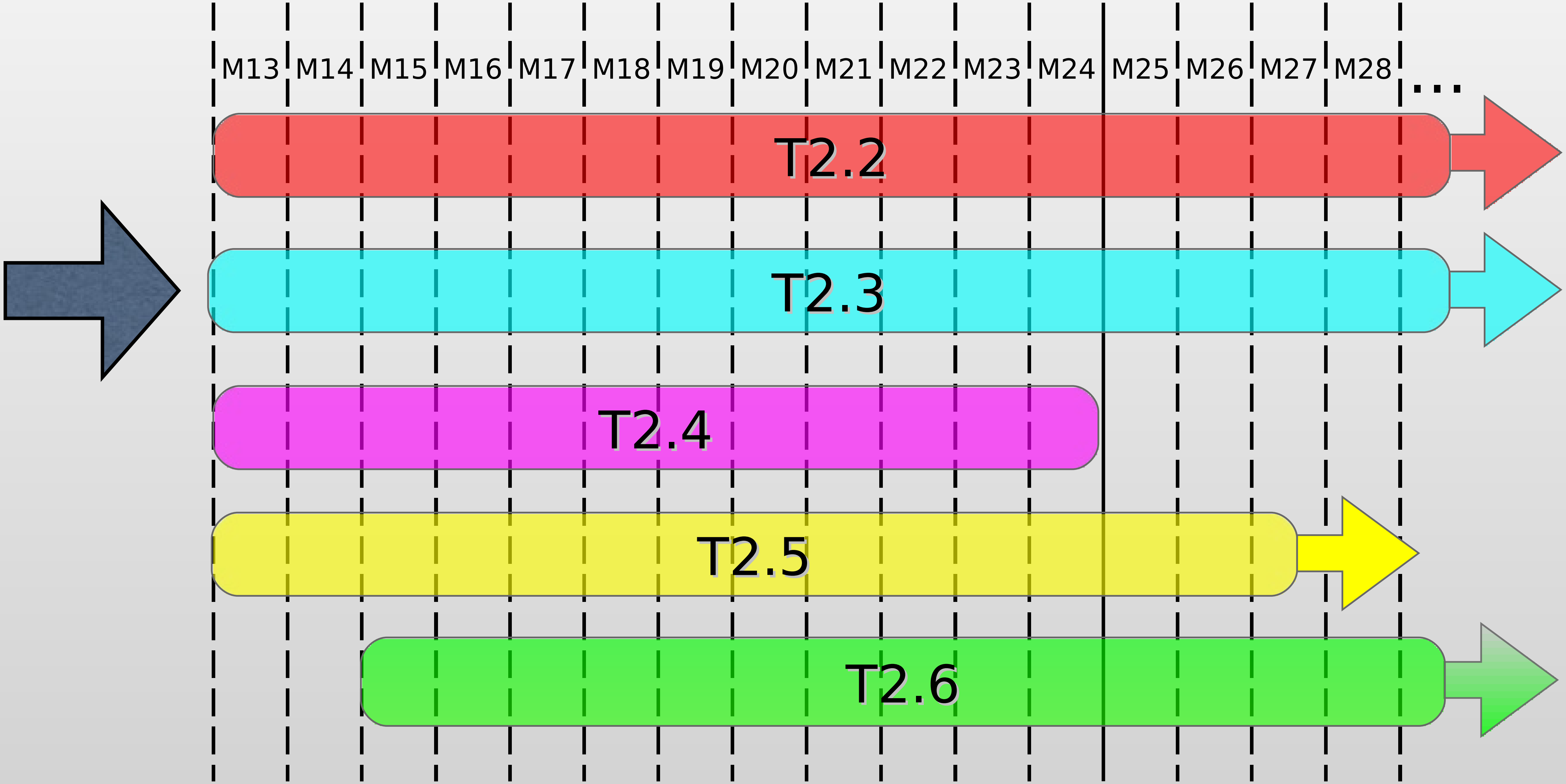
Secure interlocking and authenticity checking

- Remote Invariants Monitoring

Portions of the application executed on a trusted node either local (smart card or secure hardware), or remote

- White-Box Remote Procedure Call (UNITN,KUL)
- Barrier Slicing (UNITN)

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

Dynamic replacement for increased tamper resistance

T2.3

- Investigation of innovative methods exploiting the “time dimension” to increase tamper resistance

T2.3

Dynamic replacement for increased tamper resistance

- Remote Tamper-Resistance with Continuous Replacement (UNITN, POLITO, prof. Christian Collberg)
- Increased reverse engineering complexity through continuous replacement and mutant code (POLITO)
- Orthogonal replacement (UNITN)

Dynamic replacement for increased tamper resistance

T2.3

- Remote Tamper-Resistance with Continuous Replacement (UNITN, POLITO, prof. Christian Collberg)

Program divided into blocks sent from the trusted node to the untrusted node

- The untrusted node never holds the complete application
- Each block obfuscated with different transformations including introduction of corrupted blocks

Implementation deployed on Java code

the trusted to the untrusted node and bound into the application through code mutations

- The untrusted node never holds the complete application
- Each block dynamically relocated during a single execution and over different executions (memory layout always different)

Implementation deployed on x86 binary code

- Increased reverse engineering complexity through continuous replacement and mutant code (POLITO)
- Orthogonal replacement (UNITN)

Dynamic replacement for increased tamper resistance

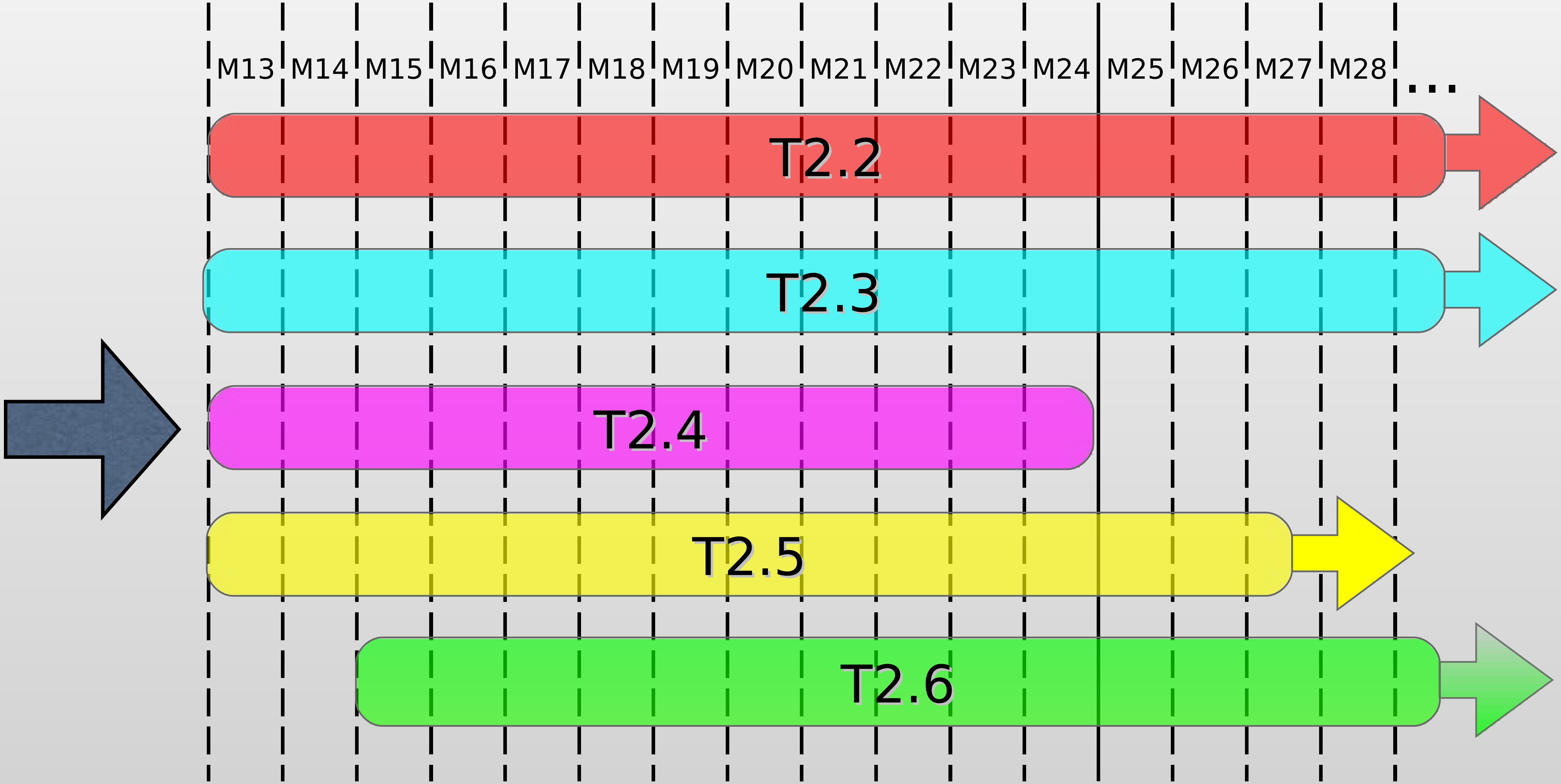
T2.3

- Remote Tamper-Resistance with

A theoretical model to build different versions of a program and/or a program block in such a way that a given version does not provide information to reverse engineering future versions

- Orthogonal replacement (UNITN)

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

T2.4

Increased reverse engineering complexity for software protection

- Definition of pure software solutions to increase reverse engineering complexity

T2.4

Increased reverse engineering complexity for software protection

- Crypto guards (KUL)
- Fuzzing (KUL)
- White-Box Cryptography (KUL)
- Obfuscation of Java byte code (GEM)
- Obfuscation Techniques (KUL)
- SProT (KUL)

T2.4

Increased reverse engineering complexity for software protection

- Cryptoguards (KUL)
- Fuzzing (KUL)
- White-Box Cryptography (KUL)

(KUL)

A technique to protect software against analysis and against tampering as well

Deployed on a binary level by using the Diablo binary rewriter

T2.4

Increased reverse engineering complexity for software protection

- Cryptoguards (KUL)
- Fuzzing (KUL)
- White-Box Cryptography (KUL)
- Obfuscation of Java byte code

Software testing technique

Submitting random or unexpected data to an application and monitoring it for any resulting error

STILL IN A PRELIMINARY PHASE

T2.4

Increased reverse engineering complexity for software protection

- Cryptoguards (KUL)
- Fuzzing (KUL)
- White-Box Cryptography (KUL)
- Obfuscation of Java byte code (GEM)

Deep analysis of state-of-the-art in WBC

Proposal of a secure encryption scheme, designed to be white-boxing

Java byte code obfuscation

• Layout obfuscation: debug information and identifier names removed • Data obfuscation: the way data is stored and encoded changed• Control flow obfuscation: the way the program runs changed (e.g., method invocation, loops, branches)• Preventive obfuscation: identification of weakness in current de-obfuscation and de-compilers to make them crash or fail

- Obfuscation of Java byte code (GEM)
- Obfuscation Techniques (KUL)
- SProT (KUL)

T2.4

Increased reverse engineering complexity for software protection

Class containing control flow obfuscation techniques such as control flow graph flattening and opaque predicates

Implemented in Txl a code transformation language

(GLM)

- Obfuscation Techniques (KUL)
- Software Protection Tool - SProT (KUL)

T2.4

Increased reverse engineering complexity for software protection

- Crypto guards (KUL)

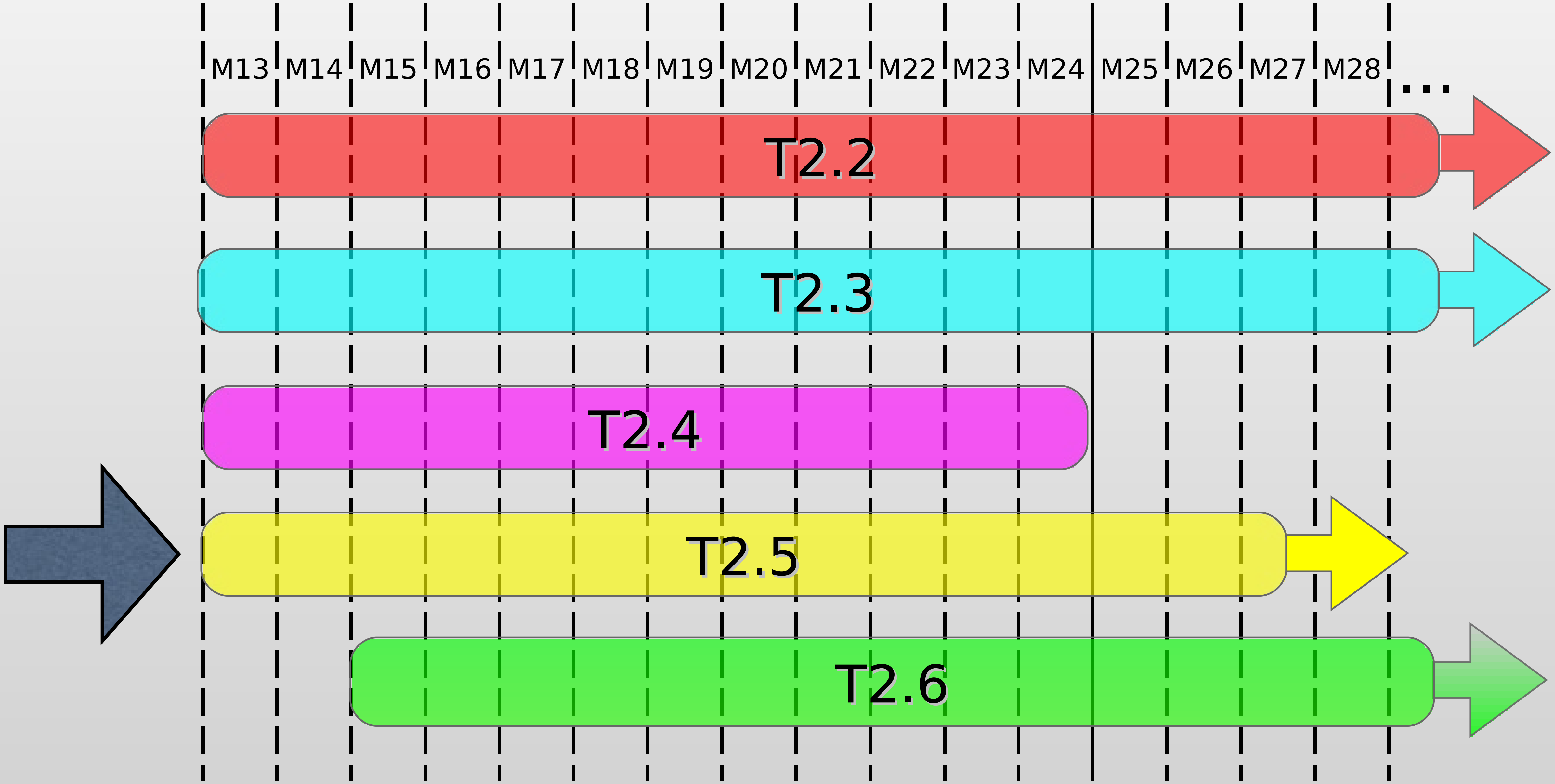
Several analysis resistance and tamper resistance techniques integrated into a single tool

• WBC be means of white-box DES and white-box AES • Obfuscation techniques • Crypto Guards

- Obfuscation Techniques (KUL)

- Software Protection Tool - SProT (KUL)

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

T2.5

Design of entrusting protocol

- Cryptographic and synchronization concerns of the communication protocol employed between trusted and untrusted node

T2.5

Design of entrusting protocol

- Preliminary design of the entrusting protocol (SPIIRAS)
- Analysis of the entrusting protocol (SPIIRAS)

Design of

T2.5

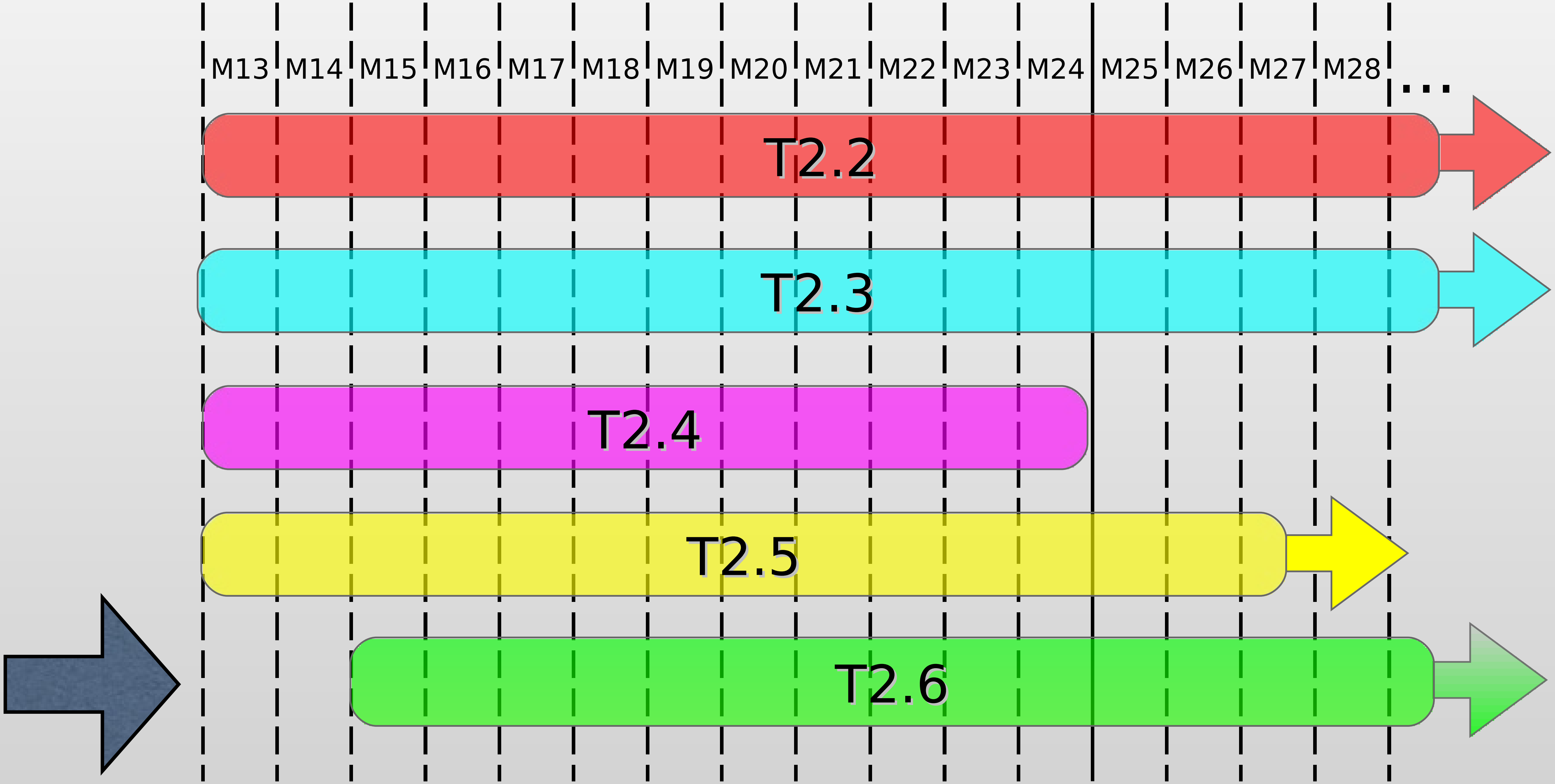
Broad analysis of existing protocol formal design and verification means

Selection of two verification tools AVISPA and Isabelle

Verification of the correctness of the entrusting protocol

- Analysis of the entrusting protocol

Tasks


M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
M27
M28
...
T2.2
T2.3
T2.4
T2.5
T2.6

T2.6

Proof of concepts

- Preliminary discussions about final proof of concept application:
  - Gemalto IP Multimedia Subsystem (IMS) server platform
  - On line games
  - VoIP
  - .....

T2.6

Proof of concepts

- Proof of concept meeting (Trento May 29th 2008):
  - On line gaming application as target
    - Candidate game: car race game
    - Definition of basic requirements: distributed application, DRM, licensing

Conclusions

- All tasks in a healthy state
- Focus during the last year of the project on:
  - Entrusting protocol (T2.5)
  - Proof of concept (T2.6)