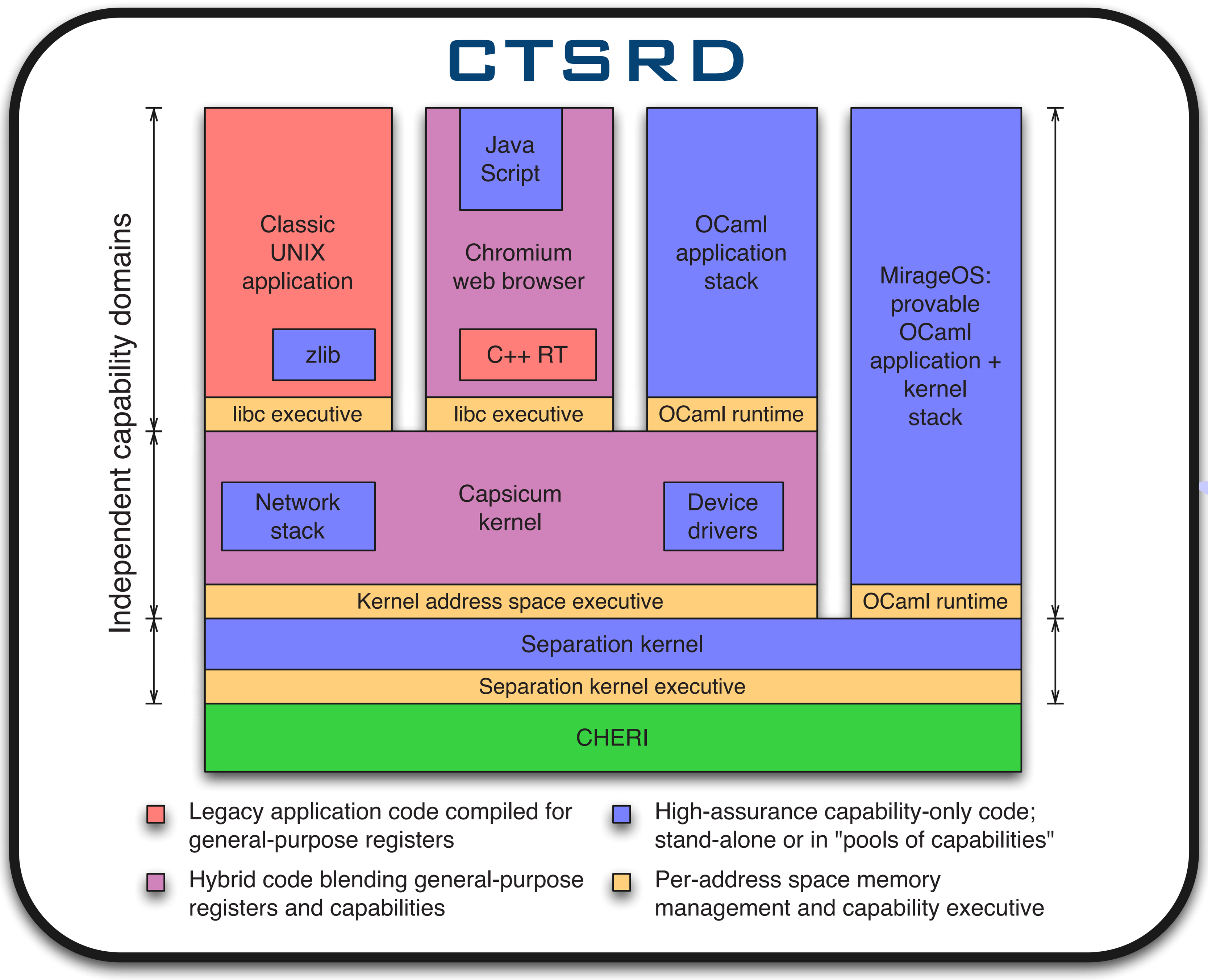
CTSRD

CRASH-worthy Trustworthy Systems Research and Development

SRI: P. Neumann, P. Lincoln, J. Rushby, H. Saidi   Cambridge: R. Watson, R. Anderson, J. Anderson, T. Finch, S. Hand, A. Madhavapeddy, A. Moore, S. Moore, S. Murdoch, P. Paeps, M. Roe, J. Woodruff.

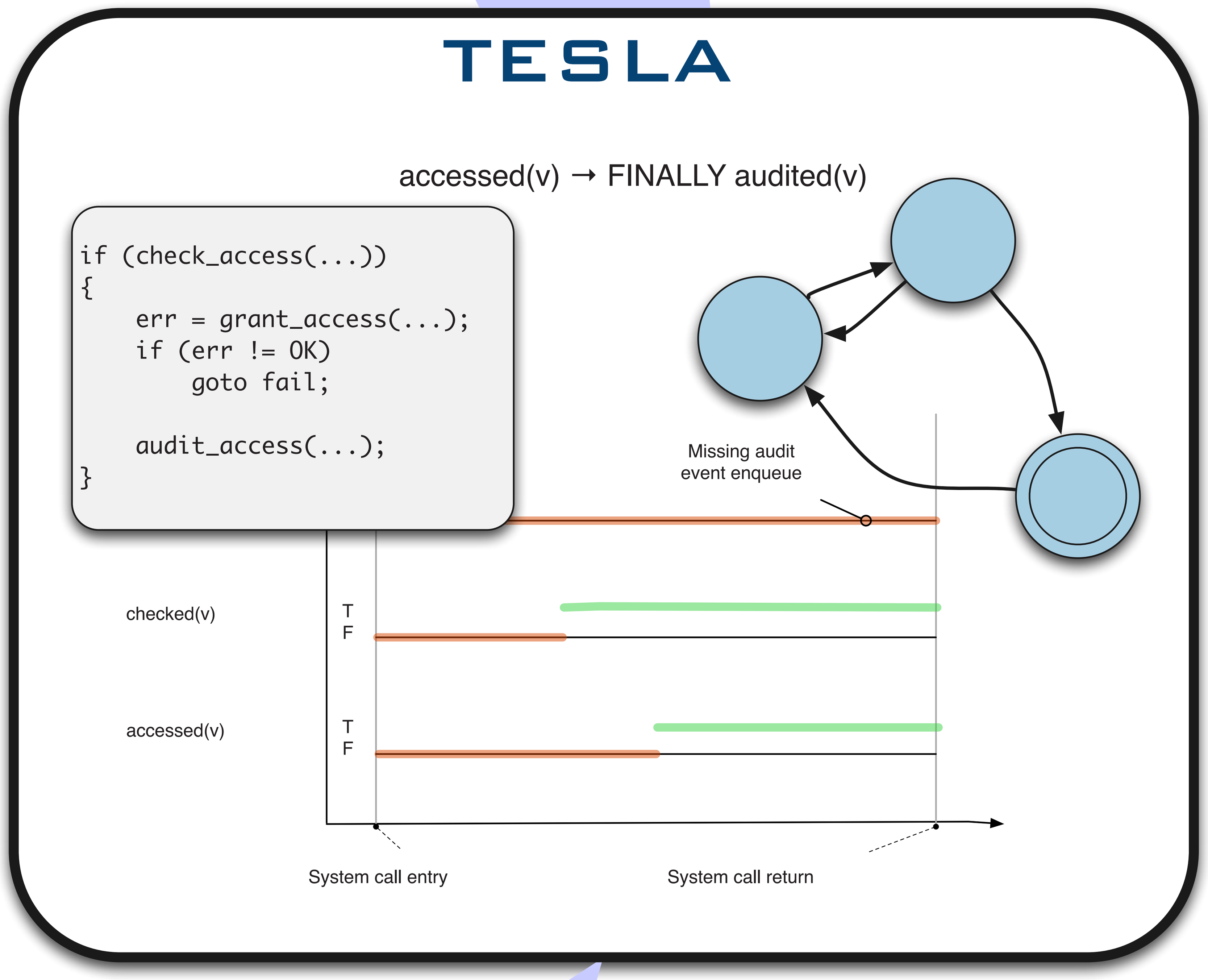
CTSRD is a principled, formally supported, and robust hardware/software platform designed for technology transfer. Security design principles and program security structure are reinforced by **Temporally Enforced Security Logic Assertions** (TESLA) and **Capability Hardware Enhanced RISC Instructions** (CHERI). The CTSRD architecture is hybrid design, able to run existing operating systems and applications while supporting a gradual adoption path for advanced security features.



CTSRD architecture is a **hybrid design** supporting high-assurance code compiled to use CHERI and TESLA, as well as legacy code:

- Code may be compiled to use general-purpose registers for addressing, capability registers, or both.
- Each address space has an *executive*, responsible for the memory model and capability creation.
- Thread contexts may be limited to only use capability addressing.
- "Pools of capabilities" allow capability code to operate within hybrid processes: the Capsicum kernel, libraries, script interpreters, etc.
- High-assurance components, such as the separation kernel and MirageOS stack, use only capability registers. They are also compiled with TESLA assertions to detect violations of design principles.

CTSRD supports critical TCB components: separation kernels, kernels, language runtimes, and particularly exposed and frequently vulnerable software components. Formal verification gives confidence in its design and implementation; TESLA picks up at runtime where proof leaves off.



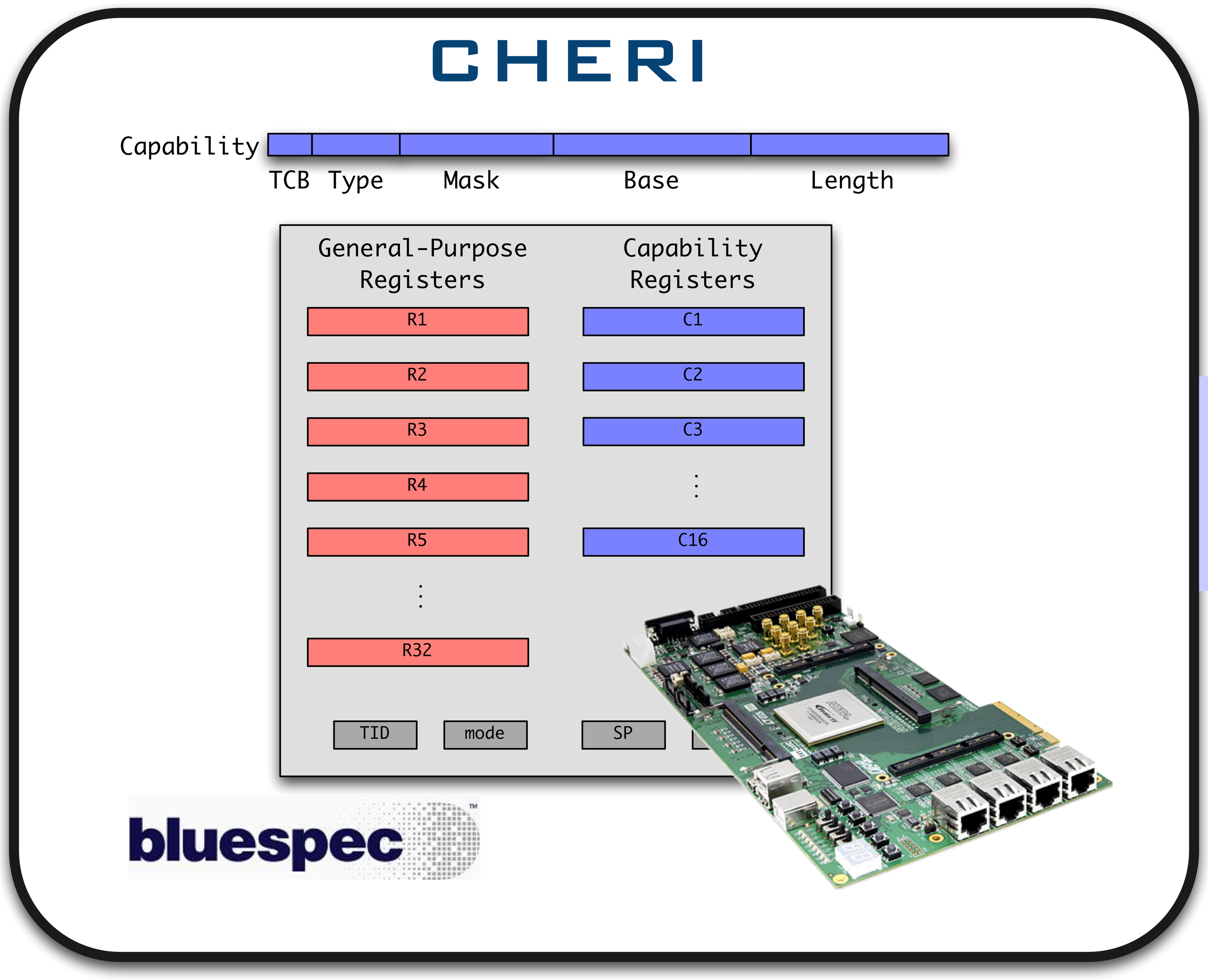
Temporally Enforced Security Logic Assertions (TESLA) applies ideas from model checking to runtime software validation:

- Assertions employ C and DTrace language constructs: types, etc.
- Temporal quantifiers capture temporal security and safety principles:

  accessed(v) → FINALLY audited(v)

  ¬accessed(c, v) UNTIL checked(c, v)
- TESLA enhancements to the clang/LLVM compiler suite mechanically instrument code with DTrace probes supporting continuous validation.
- Hardware-enhanced TESLA employs tightly coupled hardware threads within a single core to improve performance, robustness.

TESLA can be used in testing, but also in production. TELSA will fail-stop the system on violation of design principles — or in supporting runtimes, an exception can be thrown that can be caught and handled.



Capability Hardware Enhanced RISC Instructions (CHERI) is a hybrid FPGA soft core blending a paged virtual memory (VM) design with hardware capabilities:

- Program security structure is exposed by the compiler to (and enforced by) hardware: general-purpose RISC registers supplemented by capability registers and tagged memory.
- Capabilities are scoped by address spaces; each address space has an executive that manages allocation and capability semantics.
- Massive multithreading implements procedure capabilities with hardware message passing rather than expensive virtual memory context switches.
- Hybrid design allows individual address spaces to blend general-purpose registers and capabilities, or be capability-only.

The CHERI development platform is the Terasic DE4 Altera FPGA board combined with the Cambridge TIGER MIPS soft core. The DE4 board can be inserted in a PC, or used as a stand-alone computer.

Approved for Public Release, Distribution Unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

SRI International   UNIVERSITY OF CAMBRIDGE