TrustJS: Trusted Client-side Execution of JavaScript

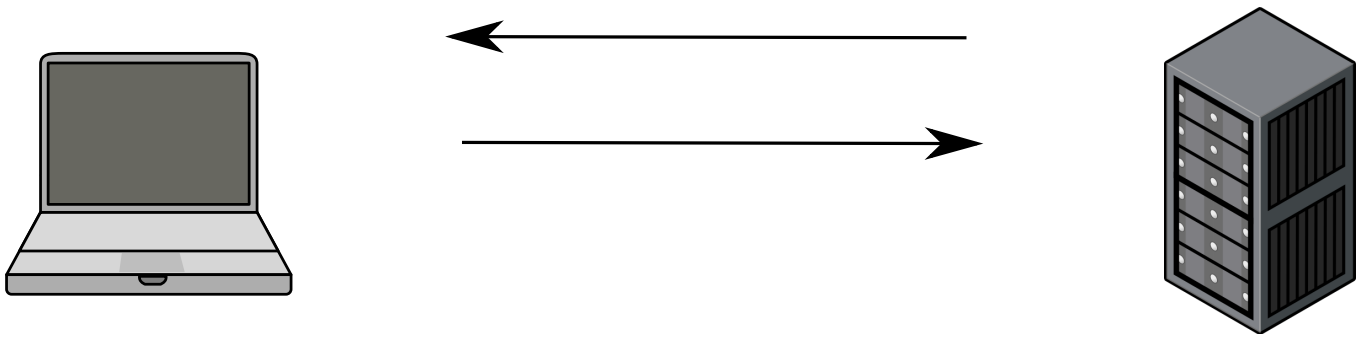
David Goltzsche[1], Colin Wulf[1], Divya Muthukumaran[2], Konrad Rieck[1], Peter Pietzuch[2] and Rüdiger Kapitza[1]

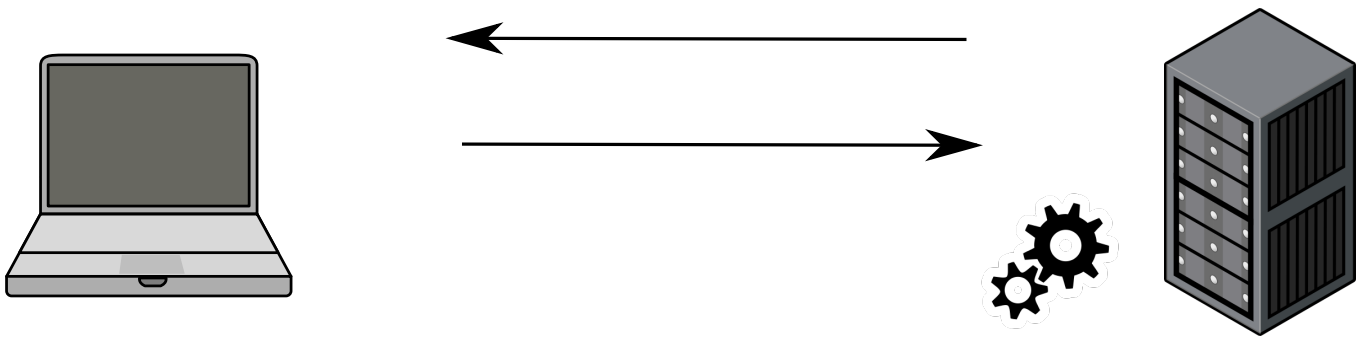
[1]TU Braunschweig, Germany
[2]Imperial College London, UK

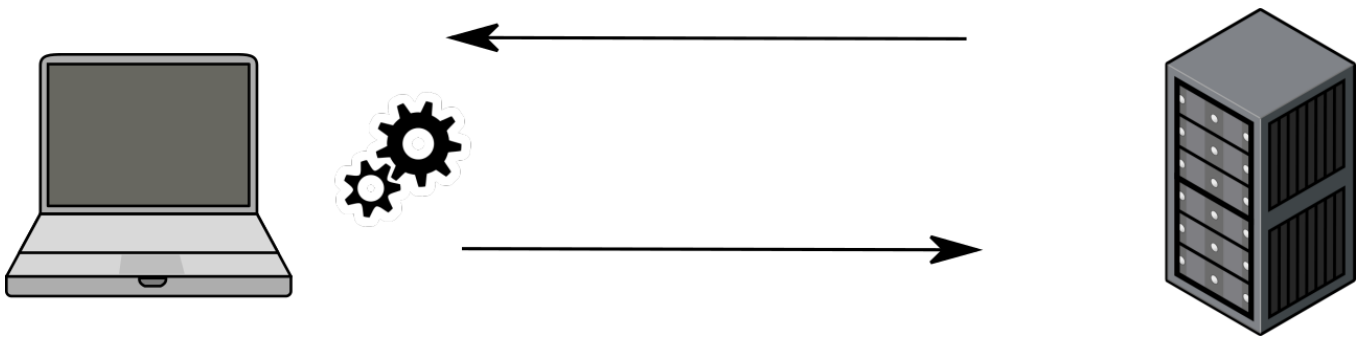
EuroSec'17, April 23, 2017, Belgrade, Serbia

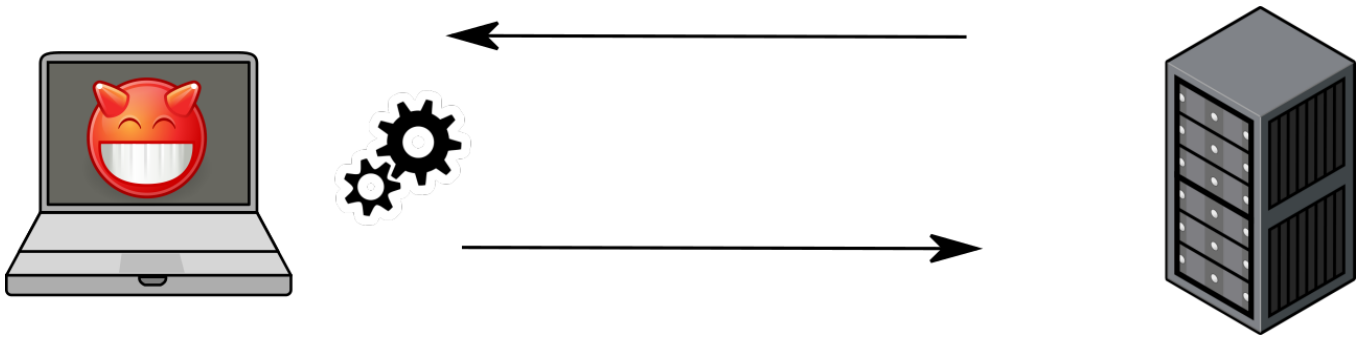
Imperial College London

Technische Universität Braunschweig

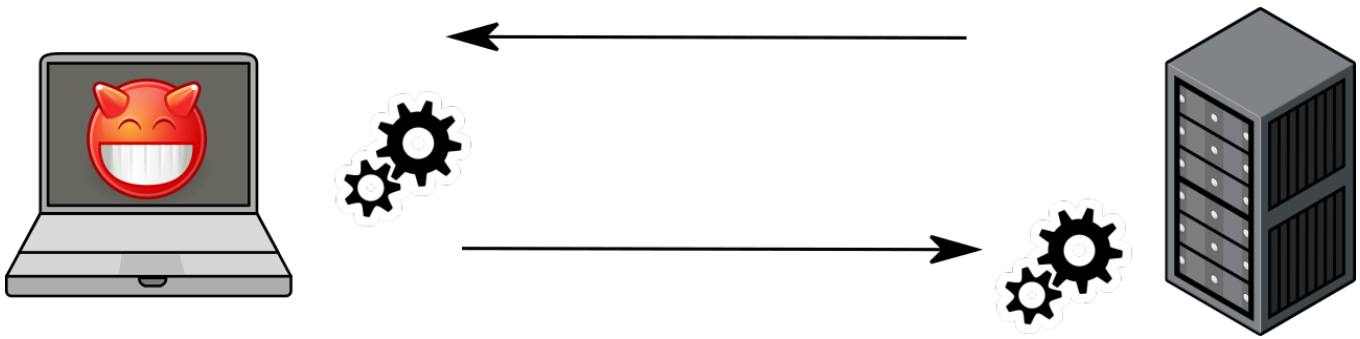




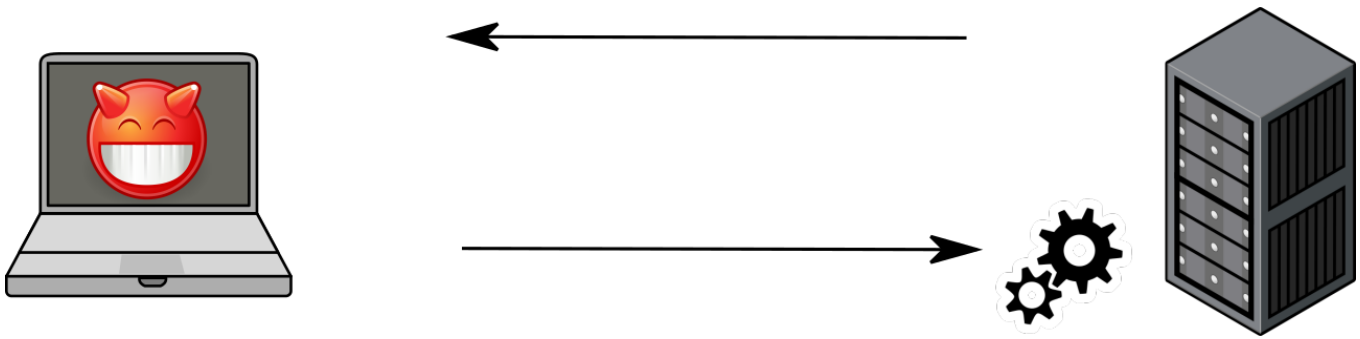




untrusted client-side computation



computation repeated



computation not offloaded

Agenda

Motivation

Approach of TrustJS

Evaluation

Conclusion

Motivation

- Web applications replace traditional desktop applications
- Providers **offload** computations **to clients**
  - Popular programming language: **JavaScript**
  - Minimise round trips
  - Reduce server-side resource demand
- Clients **not** assumed as **trustworthy**
  - Results can be faulty
  - No confidential code or data

  $\rightarrow$ Strong **limitation** for offloading approach
- Results of untrusted clients typically **verified** at server-side
  - Requires recomputation, that can lead to vulnerabilites[1]

  $\rightarrow$ Waste of resources

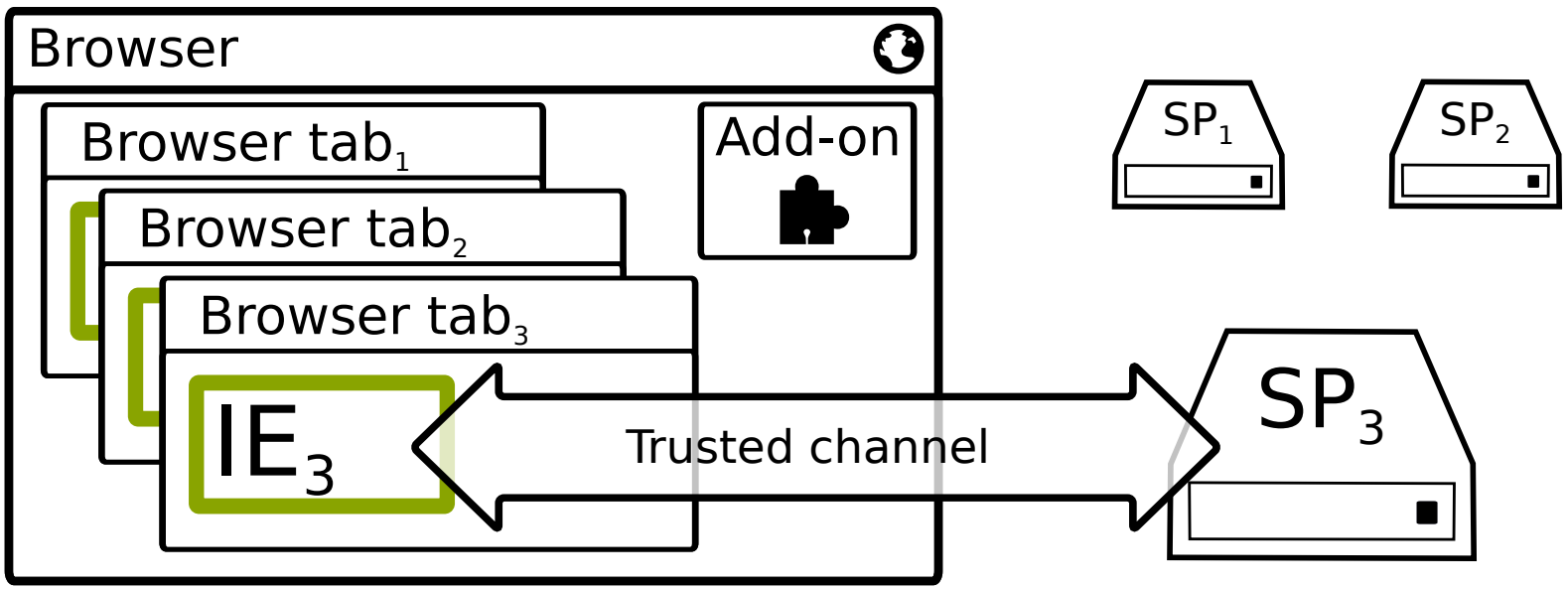
[1] P. Bisht, et el. NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications. CCS, 2010.

TrustJS

Trusted client-side execution of JavaScript

- **General purpose** JavaScript
- Integration in **commodity** browsers
- Protecting code and data
  - Integrity
  - Confidentiality (optional)
- **Remote verification** of computation results

Architecture of TrustJS



High-level architecture of TrustJS

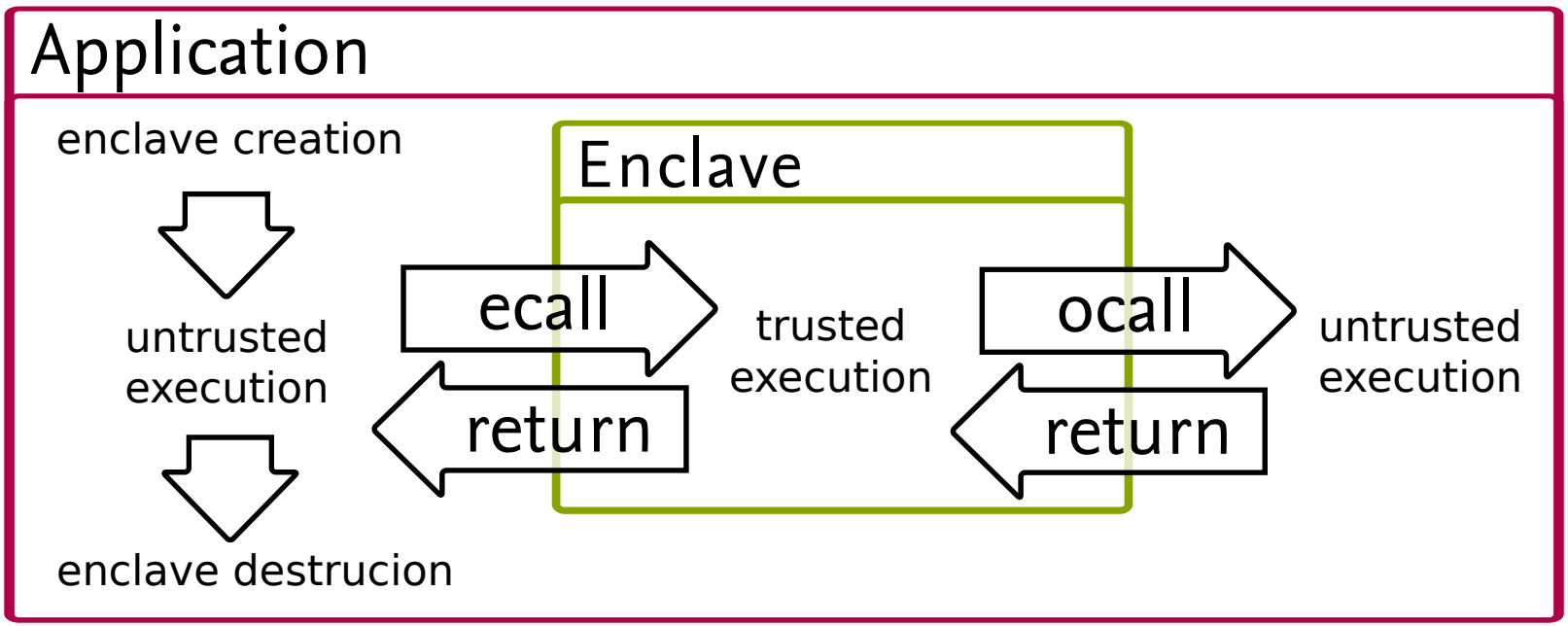
IE: interpreter enclave

SP: service provider

Intel SGX (1/2)

- **S**oftware **G**uard E**x**tensions
- Extension of x86 instruction set
- Creation of isolated compartments $\rightarrow$ **enclaves**
- Execution **isolated** from untrusted OS
- Transparent **memory encryption**
  - Pages stored in EPC
- Support for **remote attestation**
  - based on Intel-provided service **IAS**

Intel SGX (2/2)



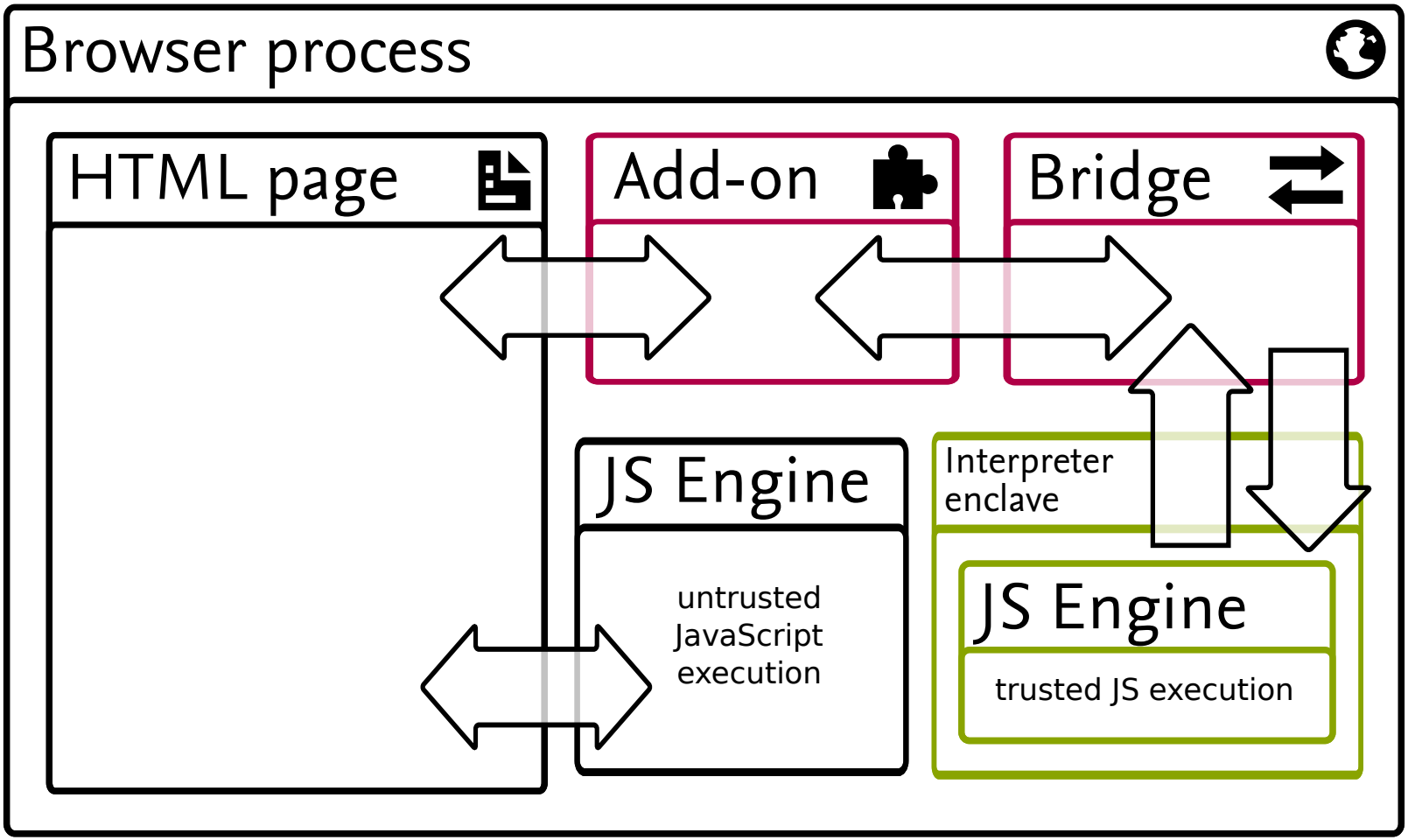
Basic interaction pattern between application and enclave

Approach of TrustJS

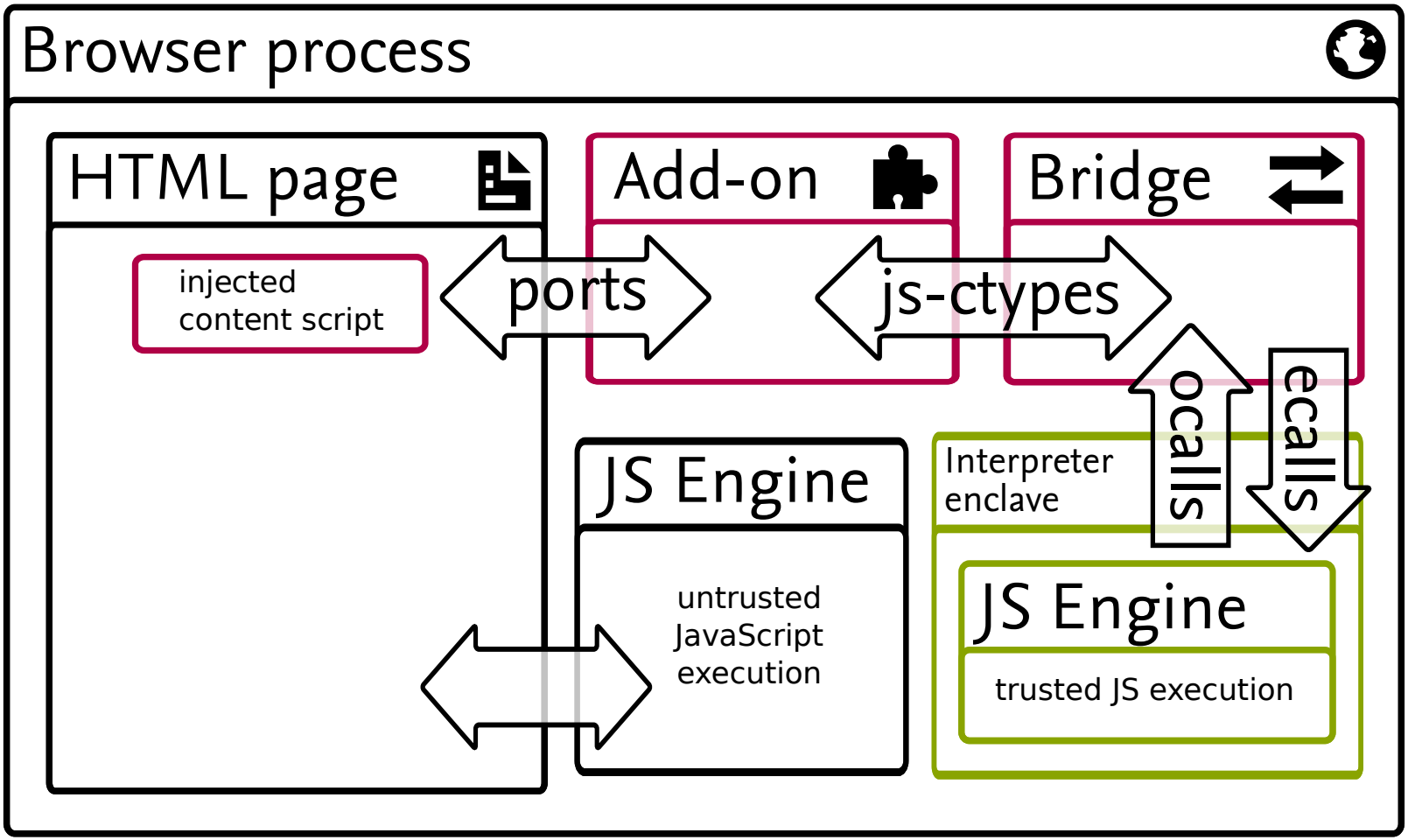
- Use **Intel SGX enclaves** at client-side
  - Put JavaScript interpreter **MuJS** in it
  - **"Interpreter enclave"**
- Integration as **browser add-on**
  - **Generic enclave binary** shipped
- Additional **trusted** JavaScript interpreter in browser
  - **Untrusted**: GUI rendering, user interaction
  - **Trusted**: execution of integrity-protected/encrypted code
    - Remote attestation to generate **verifiable responses**

    $\rightarrow$ Verification at server
- Developers **annotate** code parts for trusted execution
  - Automated transition into interpreter enclave

TrustJS Client



TrustJS Client



TrustJS Server

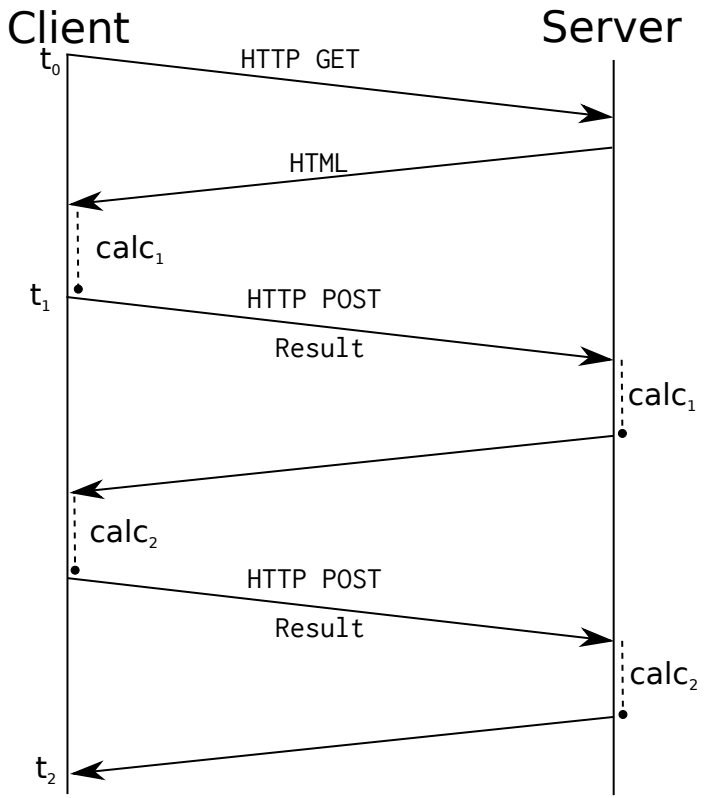
```
<script trustjs-encrypt="yes">
  /* @exposed confidentialFunction 1 */
  function hiddenFunction(y) { ...
  }
  function confidentialFunction(x) { ...
    hiddenFunction(x);
  }
</script>
<script>var a = confidentialFunction(42);</script>
```

TrustJS Server

```
<script trustjs-encrypt="yes">
  /* @exposed confidentialFunction 1 */
  function hiddenFunction(y) { ...
  }
  function confidentialFunction(x) { ...
    hiddenFunction(x);
  }
</script>
<script>var a = confidentialFunction(42);</script>
```

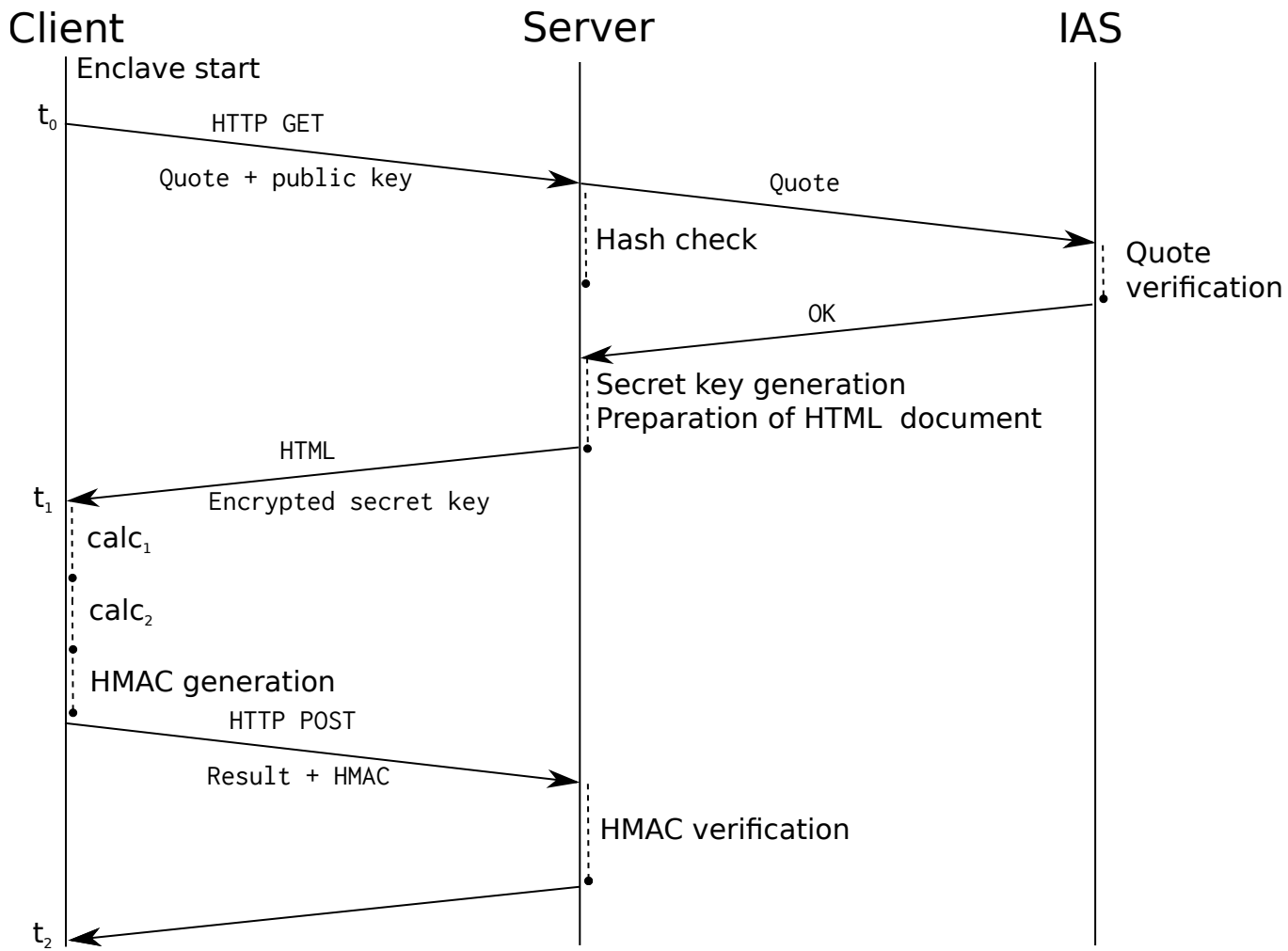
```
<script trustjs-encrypt="yes"
  trustjs-blob="X6YXkazAVA7oBZYC..9CkX0Tq9I="/>
<script>var a = confidentialFunction(42);</script>
```

Evaluation 1/4



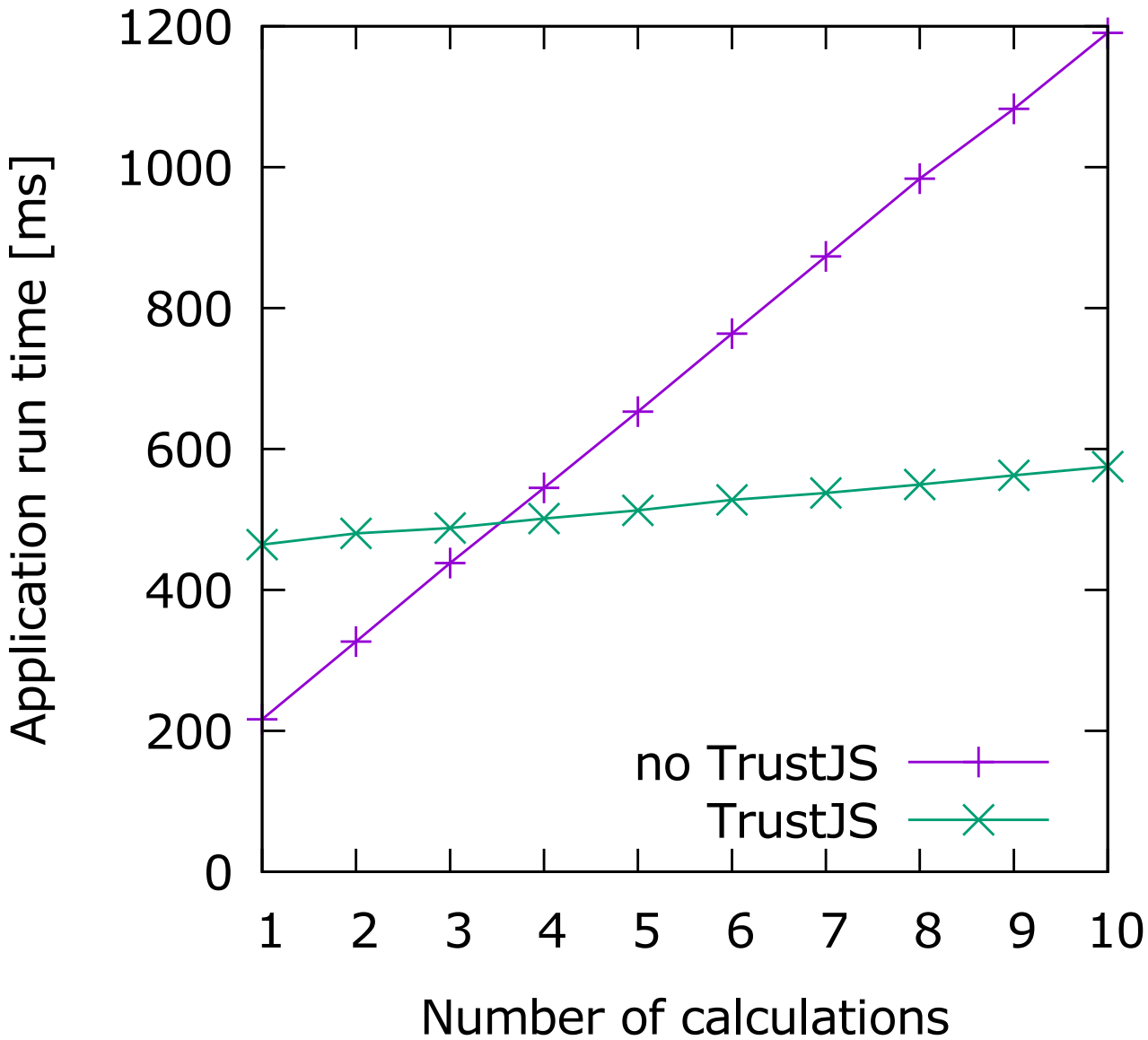
Latency build-up of traditional web application with server-side recalculations

Evaluation 2/4

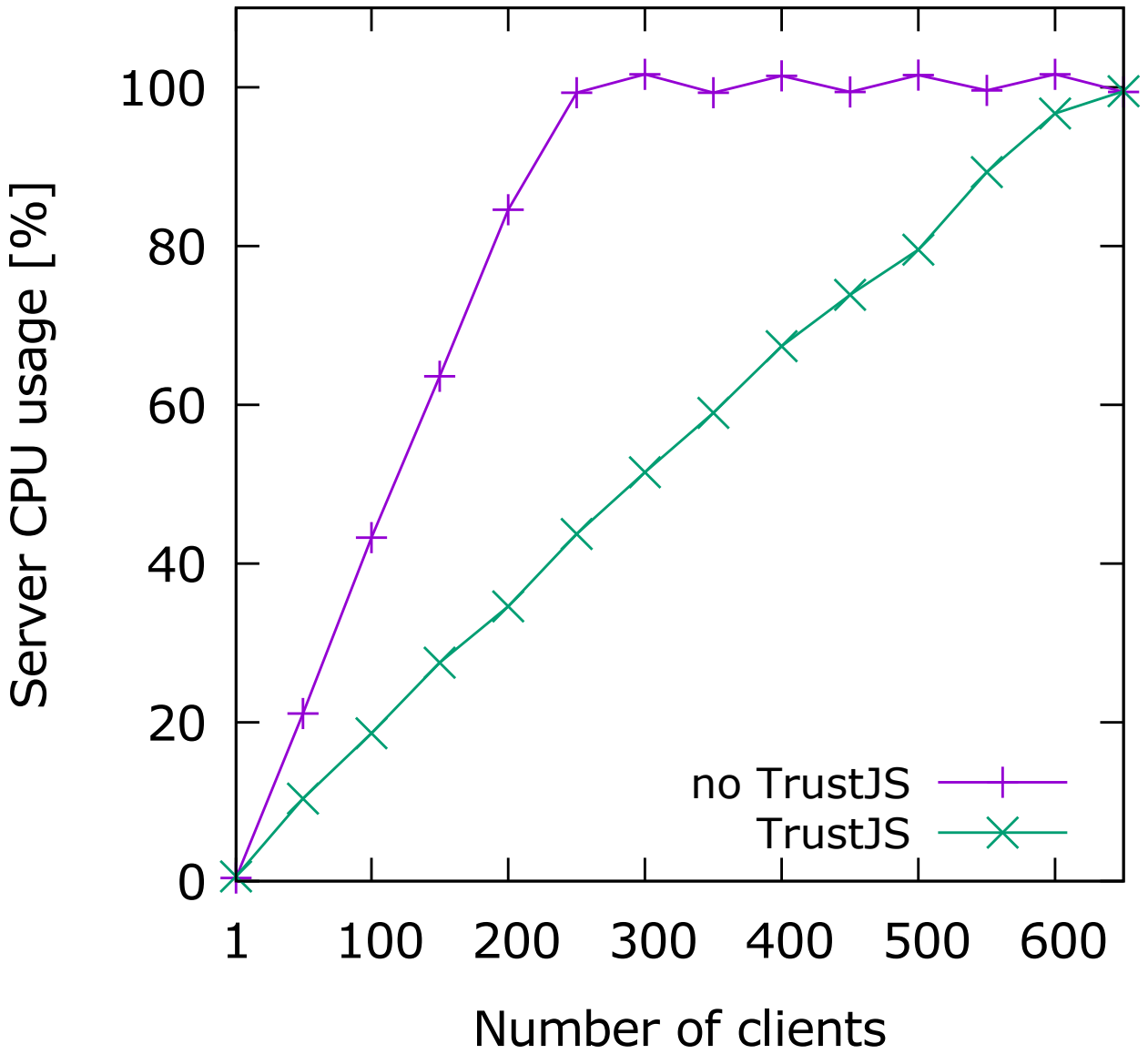


Latency build-up of web application using TrustJS with single server-side verification

Evaluation 3/4



Evaluation 4/4



Conclusion and Future Work

- **TrustJS** enables...
  - **trusted execution** of JavaScript in commodity browsers seamlessly integrated as an add-on
  - service providers to **save resources** by removing (re)computations on server-side
  - developers to remove **unnecessary round trips**

- With **future work** TrustJS may...
  - support more sophisticated JavaScript engines
  - make parts of the Node.js API available in enclave