IEEE COMPUTER SOCIETY

Enclave-Based Secure Programming with J_E

Aditya Oak — TU Darmstadt
Amir M. Ahmadian — KTH Royal Institute of Technology
Musard Balliu — KTH Royal Institute of Technology
Guido Salvaneschi — University of St.Gallen

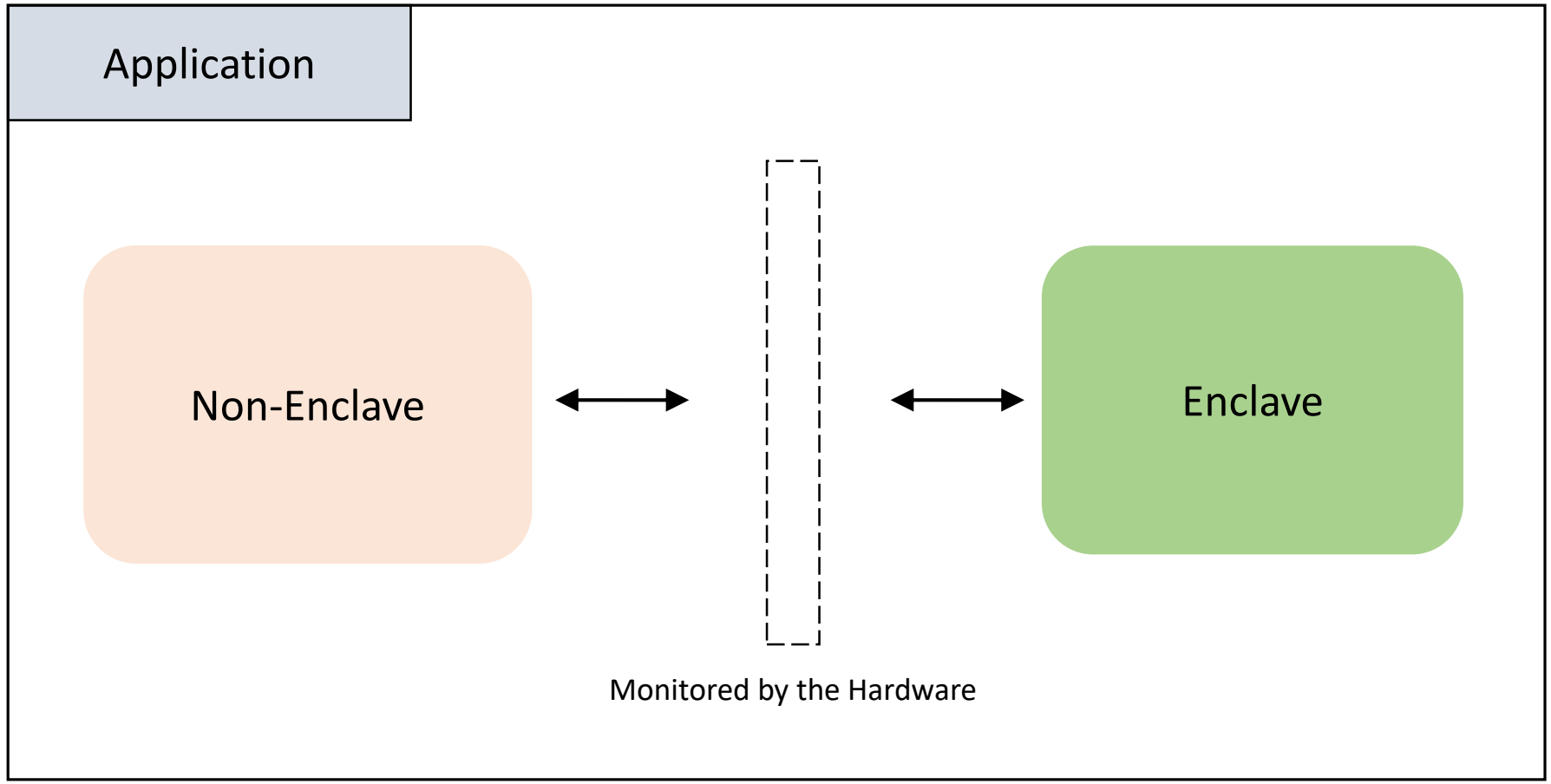
Trusted Execution Environments (TEEs)

- Privacy Enhancing Technology
- “Enclaves” protected by hardware
- Enclaves opaque to the OS
- Private computations on untrusted machines

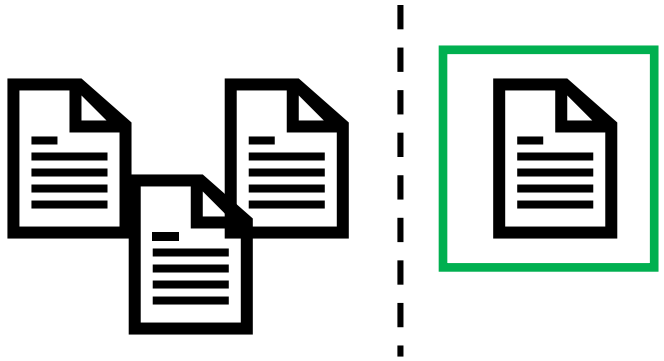
arm
TRUSTZONE

Apple Platform
Security

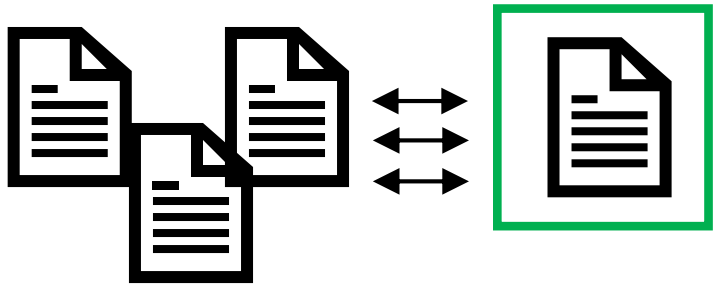
Programming with TEEs



Programming with TEEs: Limitations

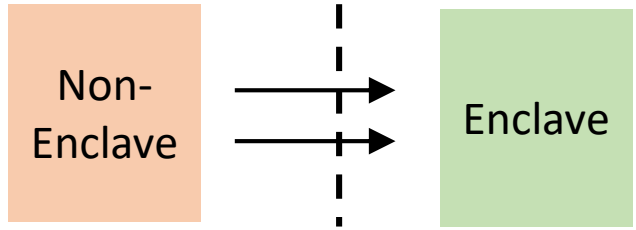


Manual application partitioning



Programming the Enclave–Non-enclave interface

Limited support for managed languages



No privacy guarantees in the presence of strong non-enclave attacks

Password Checker (C++) using Intel SGX

Enclave Code (enclave.cpp)

```
const char* password = "secret";
void checkPassword(char* guess, int* result, size_t len) {
strcmp(guess, password) == 0 ?
      *result = 1 : *result = 0;

}
```

Interface Code (enclave.edl)

```
enclave {
      trusted {
      public void checkPassword([in, size=len] char* guess, [out] int* result, size_t len);
      };

};
```

Non-Enclave Code (main.cpp)

```
#include "sgx_urts.h"
#include "enclave_u.h"
#define BUF_LEN 100
int main() {
    sgx_enclave_id_t eid;
    sgx_status_t ret = SGX_SUCCESS;
    sgx_launch_token_t token = {0};
    int updated = 0;
    ret = sgx_create_enclave(ENCLAVE_FILE, SGX_DEBUG_FLAG, &token, &updated, &eid, NULL);
    if (ret != SGX_SUCCESS) { ... /* exception */ }
        char* guess = ... // read guess from stdin
        int result = 0;
        checkPassword(eid, guess, &result, BUF_LEN);
        if (SGX_SUCCESS != sgx_destroy_enclave(eid)) {
                return 0;
        }
}
```

J_E Programming Model

J_E

Programming with
Enclaves made simple

J_E Programming Model

```
const char* password = "secret";
void checkPassword(char* guess, int* result, size_t len) {
strcmp(guess, password) == 0) ?
      *result = 1 : *result = 0;
}

enclave {
      trusted {
      public void checkPassword([in, size=len] char* guess, [out] int*
                  result, size_t len);
      };
};
#include "sgx_urts.h"
#include "enclave_u.h"
#define BUF_LEN 100
int main() {
    sgx_enclave_id_t eid;
    sgx_status_t ret = SGX_SUCCESS;
    sgx_launch_token_t token = {0};
    int updated = 0;
    ret = sgx_create_enclave(ENCLAVE_FILE, SGX_DEBUG_FLAG, &token,
&updated, &eid, NULL);
    if (ret != SGX_SUCCESS) { ... /* exception */ }
        char* guess = ... // read guess from stdin
        int result = 0;
        checkPassword(eid, guess, &result, BUF_LEN);
        if (SGX_SUCCESS != sgx_destroy_enclave(eid)) {
                  return 0;
        }
}
```

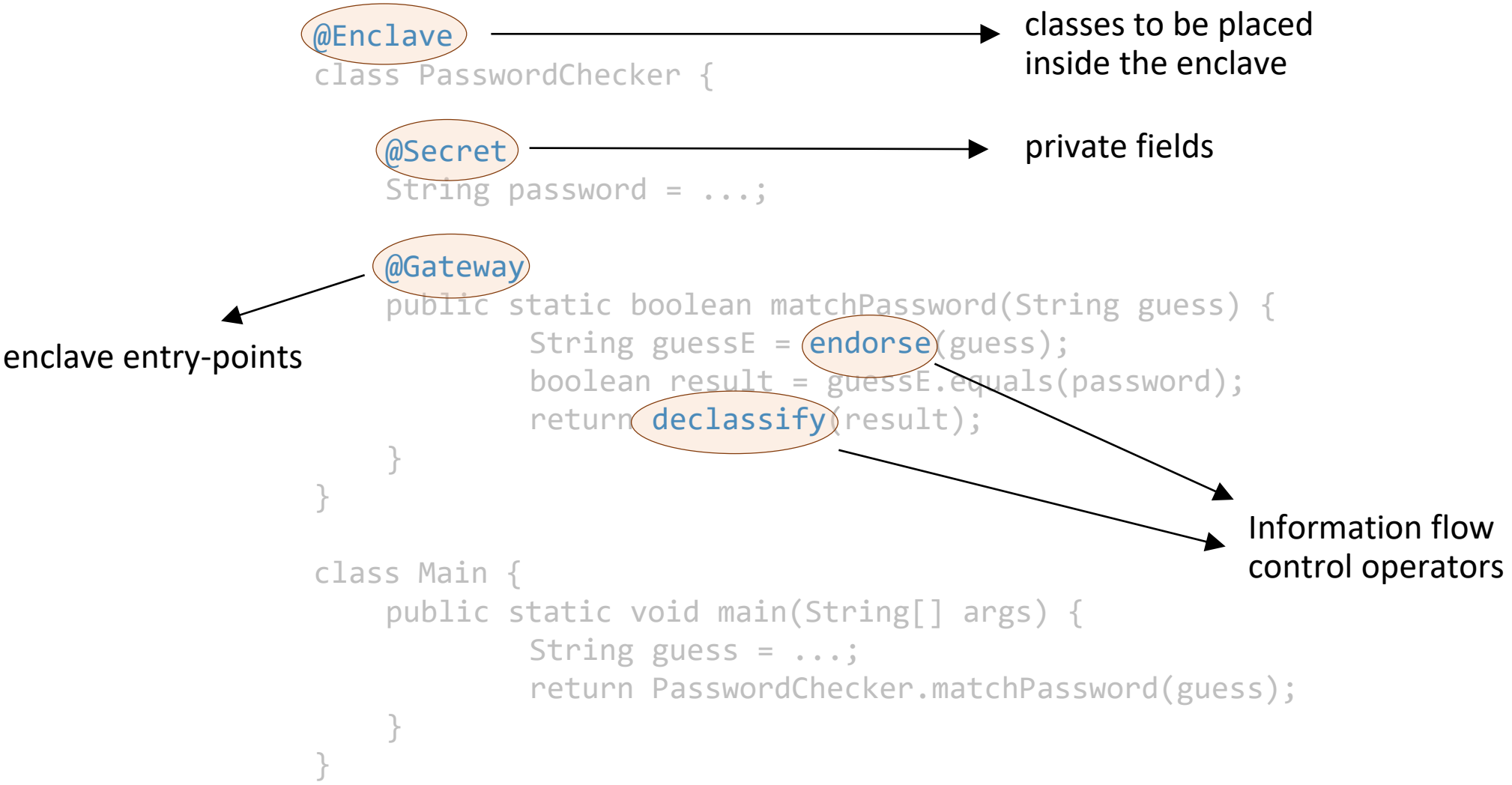
```
@Enclave
class PasswordChecker {
      @Secret
      String password = ...;

      @Gateway
      public static boolean matchPassword(String guess) {
                  String guessE = endorse(guess);
                  boolean result = guessE.equals(password));
                  return declassify(result);
      }
}

class Main {
      public static void main(String[] args) {
                  String guess = ...;
                  return PasswordChecker.matchPassword(guess);
      }
}
```

J_E

J_E Language Abstractions

```
@Enclave
class PasswordChecker {

    @Secret
    String password = ...;

    @Gateway
    public static boolean matchPassword(String guess) {
            String guessE = endorse(guess);
            boolean result = guessE.equals(password);
            return declassify(result);
    }
}

class Main {
    public static void main(String[] args) {
            String guess = ...;
            return PasswordChecker.matchPassword(guess);
    }
}
```

J_E Language Abstractions



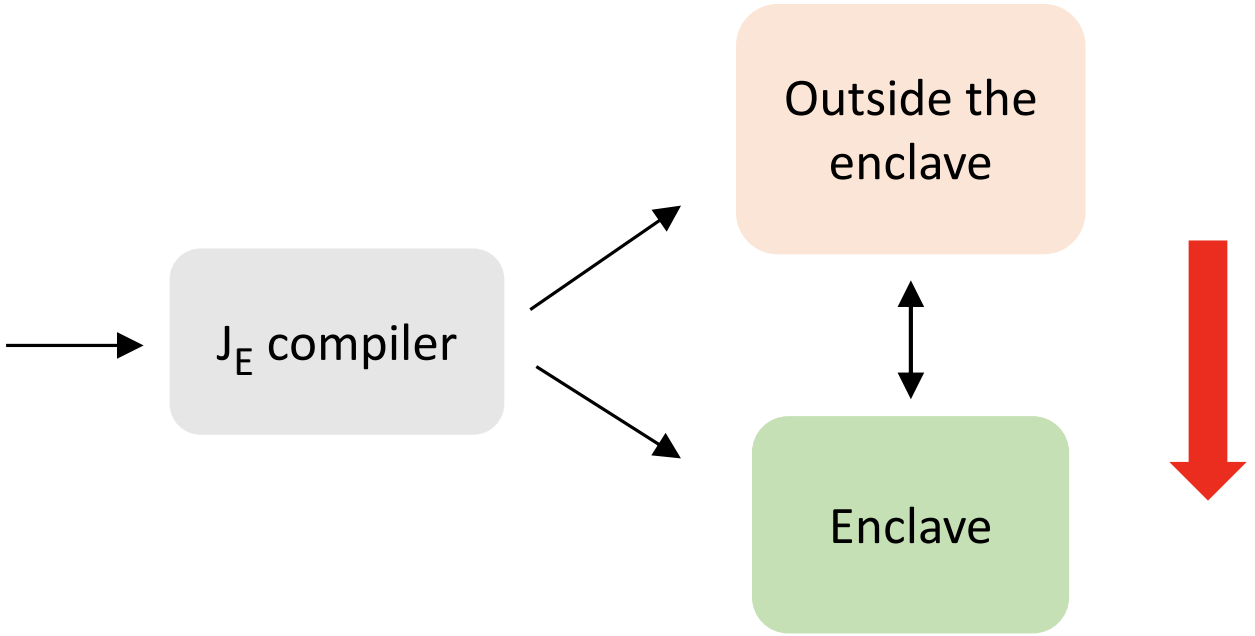
J_E: Program Partitioning and Attacker Model

```
@Enclave
class PasswordChecker {

    @Secret
    String password = ...;

    @Gateway
    public static boolean matchPassword(String guess) {
            String guessE = endorse(guess);
            boolean result = guess.equals(password);
            return declassify(result);
    }
}

class Main {
    public static void main(String[] args) {
            String guess = ...;
            return PasswordChecker.matchPassword(guess);
    }
}
```

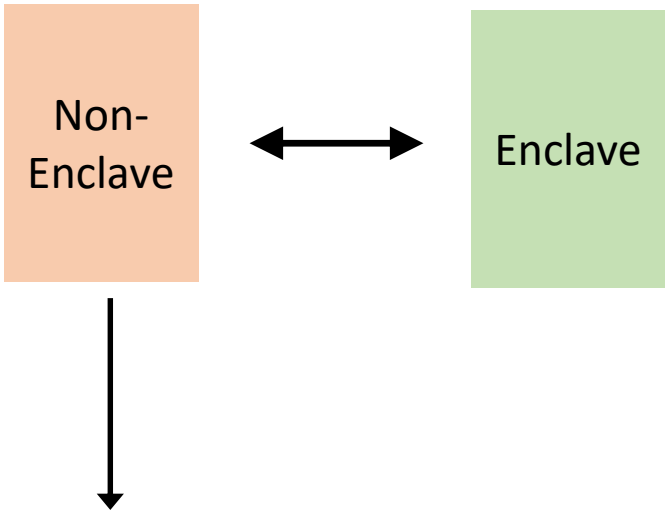
J_E: Program Partitioning and Attacker Model

```
@Enclave
class PasswordChecker {

   @Secret
   String password = ...;

   @Gateway
   public static boolean matchPassword(String guess) {
           String guessE = endorse(guess);
           boolean result = guess.equals(password);
           return declassify(result);
   }
}

class Main {
  public static void main(String[] args) {
           String guess = ...;
           return PasswordChecker.matchPassword(guess);
           }
}
```

J_E code

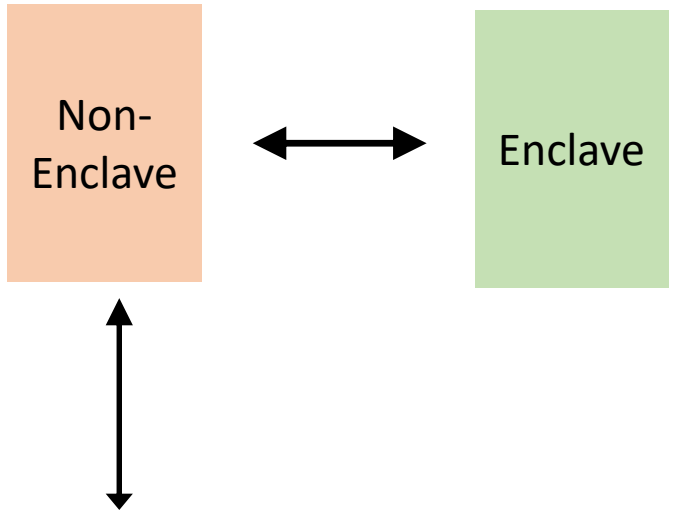


J_E: Attacker Models



Read-only

Passive Attacker



Read and Write

Active Attacker
(two types)

J_E: Security Guarantees

Security Guarantees:

Indistinguishable traces

Robustness: Limit the attacker's influence on declassification (release of secret data)

Active Attackers: Data Attacker

Robustness under **Data Attacker**

- Can change **Data**
- Parameters to Gateways can be changed
- Parameters are untrusted
- Untrusted values should not affect declassification

```
@Enclave
class Controller {

    @Secret
    String quote = ...;

    @Gateway
    public static String getQuote(int val) {
            if (val > 500)
                    return declassify(quote);
            else ...
    }
}
```

```
class Main {
    public static void main(String[] args) {
    int val = ...;
    String quote = Controller.getQuote(val);
    }
}
```

Active Attackers: Data and Code Attacker

Robustness under **Data and Code Attacker**

- Can change **Data** and **Code**
- Thus can change control flow outside the enclave
- The order determines which value will be declassified
- Change in the order and frequency of

  Gateways should not leak more information

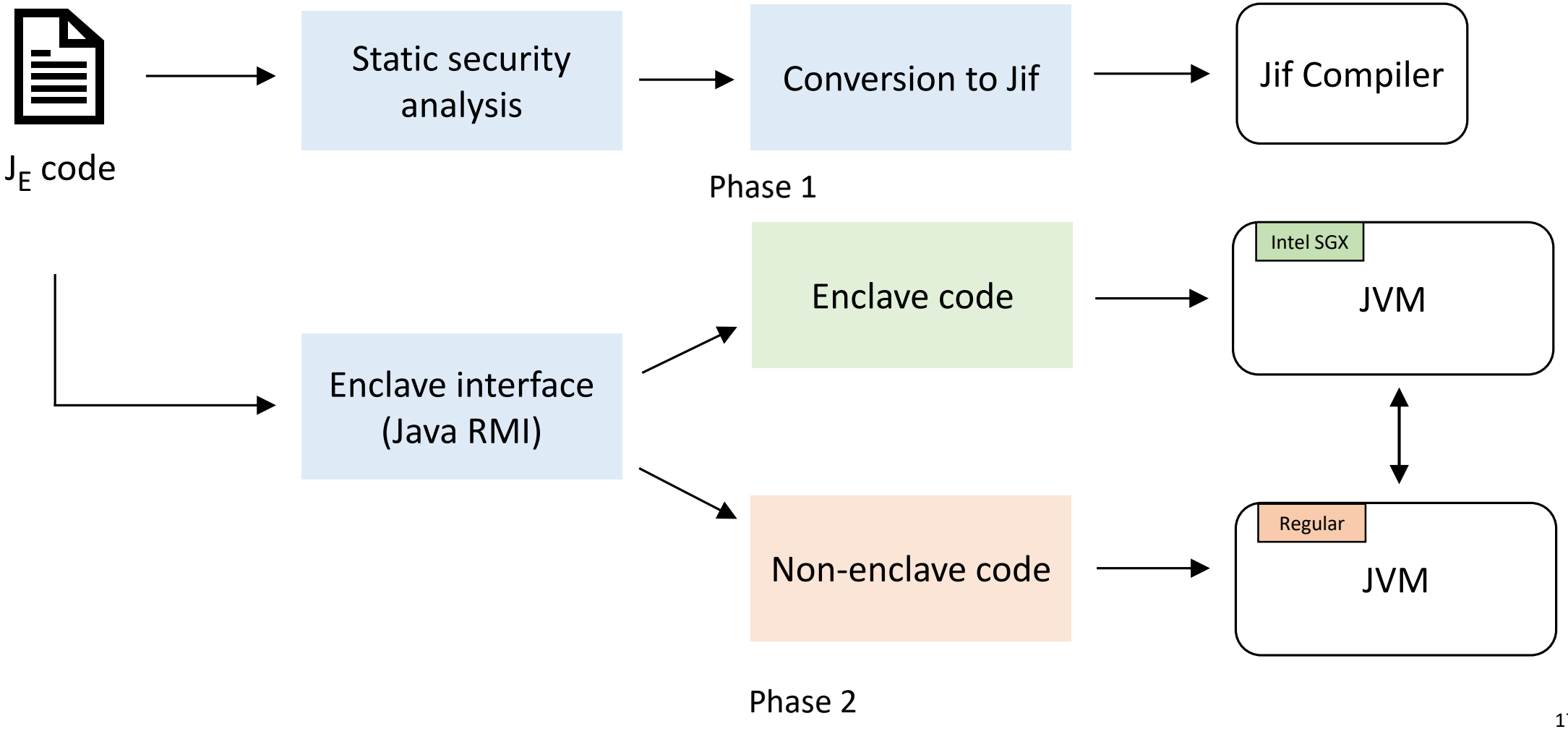
```
@Enclave
class Timer  {

    @Secret static int secret1, secret2;

    static boolean releaseTrigger = false;

    @Gateway
    public static void setTrigger() {
              releaseTrigger = true;
    }

    @Gateway
    public static int release() {
             int res = 0;
             if (releaseTrigger) {
                res = declassify(secret1);
             } else {
                res = declassify(secret2);
             }
             return res;
    }  }
```

J_E: Type System

Static Type System

- Ensures robustness against the Passive and Active attackers
- Formal details presented in: *"Language Support for Secure Software Development with Enclaves" CSF 2021*

J_E Implementation and Workflow



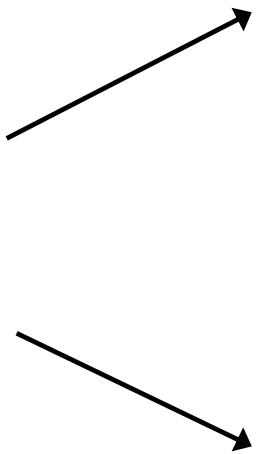
J_E Case Study: Password Checker

```
class Main {
    public static void main (String[] args) {
            String guess = ...;
            PasswordChecker.matchPassword(guess);
    }
}

@Enclave
class PasswordChecker {

    @Secret static private String password;

    @Gateway
    public static boolean matchPassword(String guess) {
       String guessE = endorse(guess);
       boolean result = guessE.equals(password);
       return declassify(result);
    }
}
```

J_E Case Study: Password Checker (cont.)

J_E code

```
class Main {
      public static void main (String[] args) {
          String guess = ...;
          PasswordChecker.matchPassword(guess);}  }

@Enclave
class PasswordChecker {
      @Secret
      String password;

      @Gateway
      public static boolean matchPassword(String guess) {
        String guessE = endorse(guess);
        boolean result = guessE.equals(password);
        return declassify(result);
}     }
```



Non-Enclave Partition

```
class Main {
      public static void main (String[] args) {
         String guess = ...;
          PasswordChecker.matchPassword(guess);
      }
}
```


Enclave Partition

```
@Enclave
class PasswordChecker {
@Secret
String password;

@Gateway
public static boolean matchPassword(String guess) {
          String guessE = endorse(guess);
          boolean result = guessE.equals(password);
          return declassify(result);

}     }
```

J_E Case Study: Password Checker (cont.)

Enclave Partition

```
@Enclave
class PasswordChecker {
      @Secret
      String password;

      @Gateway
      public static matchPassword(String guess) {
            String guessE = endorse(guess);
           boolean result = guessE.equals(password);
           return declassify(result);
}   }
```

Enclave Partition: Jif

```
class PasswordChecker [principal Enclave] authority(Enclave) {
      private {Enclave->*; Enclave<-*} password;

      public boolean{Enclave->_; Enclave<-*} matchPassword{Enclave<-*}(String{} guess)
      where authority(Enclave) {
                  String guessE = endorse(plaintext, {} to {Enclave<-*});
                  boolean{Enclave->*; Enclave<-*} result =
                  guessE.equals(password);
                  return declassify (result, {Enclave->*; Enclave<-*}
                                          to {Enclave->_;Enclave<-*});
      }
}
```

J_E Case Study: Password Checker (cont.)

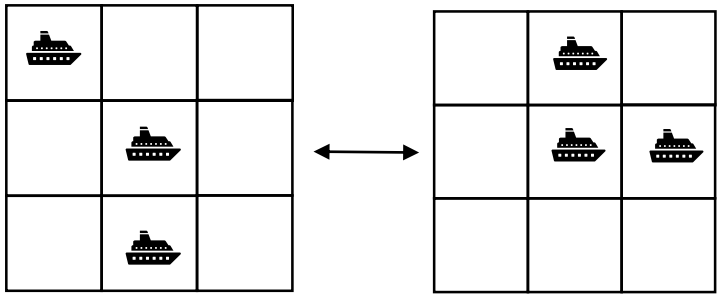
Non-Enclave Partition: RMI Communication

```
class Main {
      public static void main(String[] args) {
         String guess = ...;

        Remote remoteServer =
Naming.lookup("rmi://ipAddr/RemotePwdChecker");

       RemotePwdChecker remPwdCheckerStub = (RemotePwdChecker)
remoteServer;
        boolean result = remPwdCheckerStub.wrapMatchPwd(guess);
      }
}
```

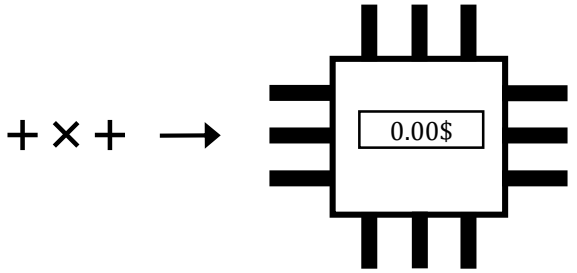
Enclave Partition: RMI Communication

```
interface RemotePwdChecker extends Remote {
      public String wrapMatchPwd(String guess);
}
class PwdCheckerWrapper extends UnicastRemoteObject implements
RemotePwdChecker {

      @Override
      public boolean wrapMatchPwd(String guess) {
            return PasswordChecker.matchPassword(guess);
      }
}
class PasswordChecker {. . .}
```

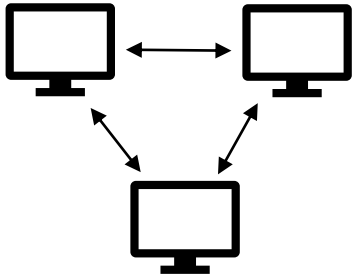
Evaluation



Battleship Game



Secure Calculator



Complex Event Processing

J_E : *Programming with Enclaves made simple*

J_E Language Abstractions

```
@Enclave                                   classes to be placed
class PasswordChecker {                    inside the enclave

    @Secret                                private fields
    String password = ...;

    @Gateway
    public static boolean matchPassword(String guess) {
            String guessE = endorse(guess);
            boolean result = guessE.equals(password);
            return declassify(result);
    }
}

class Main {
    public static void main(String[] args) {
            String guess = ...;
            return PasswordChecker.matchPassword(guess);
    }
}
```

enclave entry-points

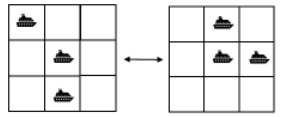
Information flow control operators

7

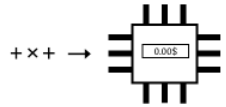
J_E: Attacker Models

J_E Implementation and Workflow

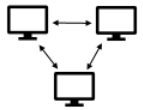
Evaluation



Battleship Game



Secure Calculator



Complex Event Processing

19

https://prg-grp.github.io/je-lang

@prg_grp