It wasn't me

How to Counter the Insecurity of Real World Cryptography

TECHNISCHE UNIVERSITÄT DARMSTADT

Eric Bodden

Mira Mezini

Karim Ali

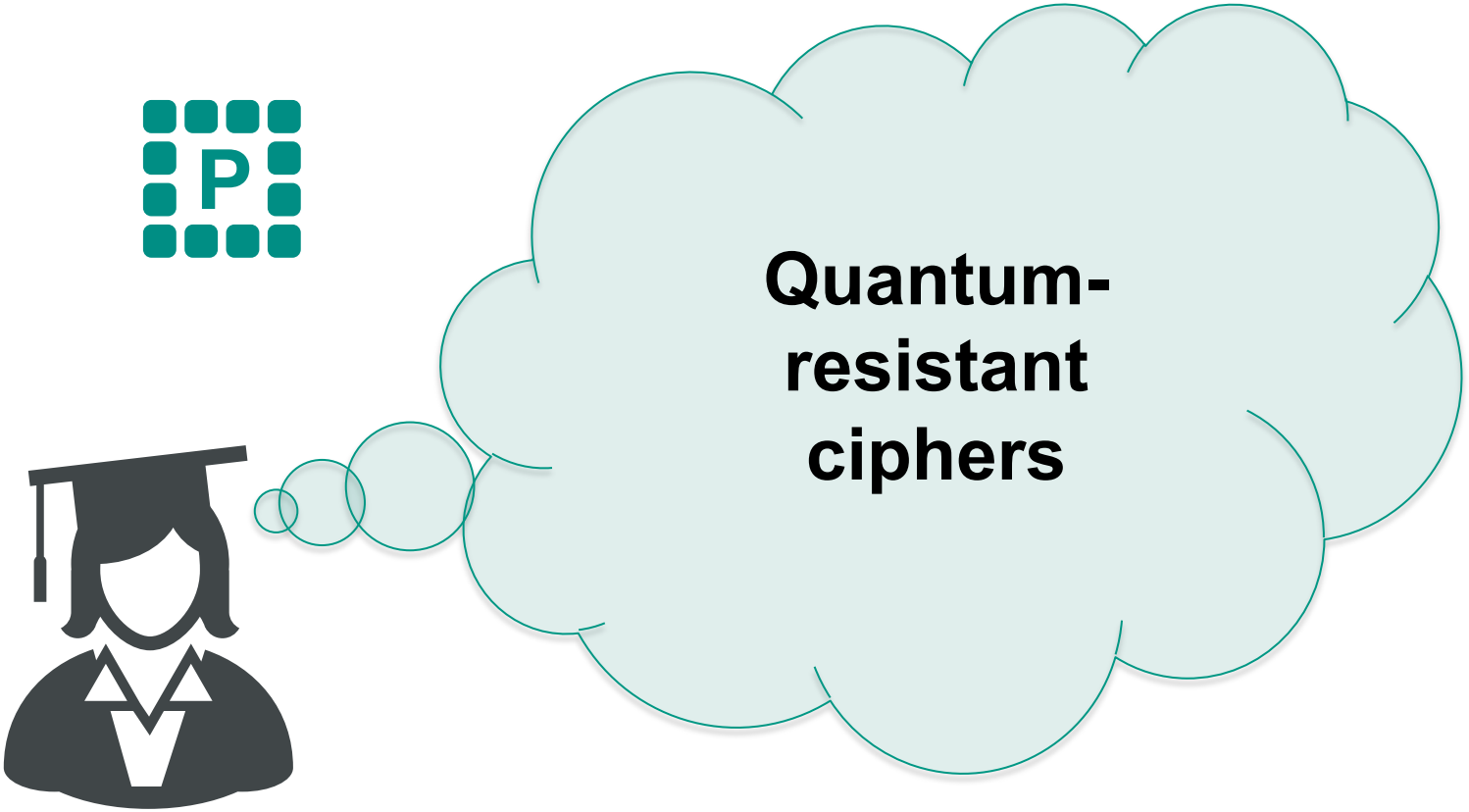
Stefan Krüger

Sarah Nadi

DFG Deutsche Forschungsgemeinschaft

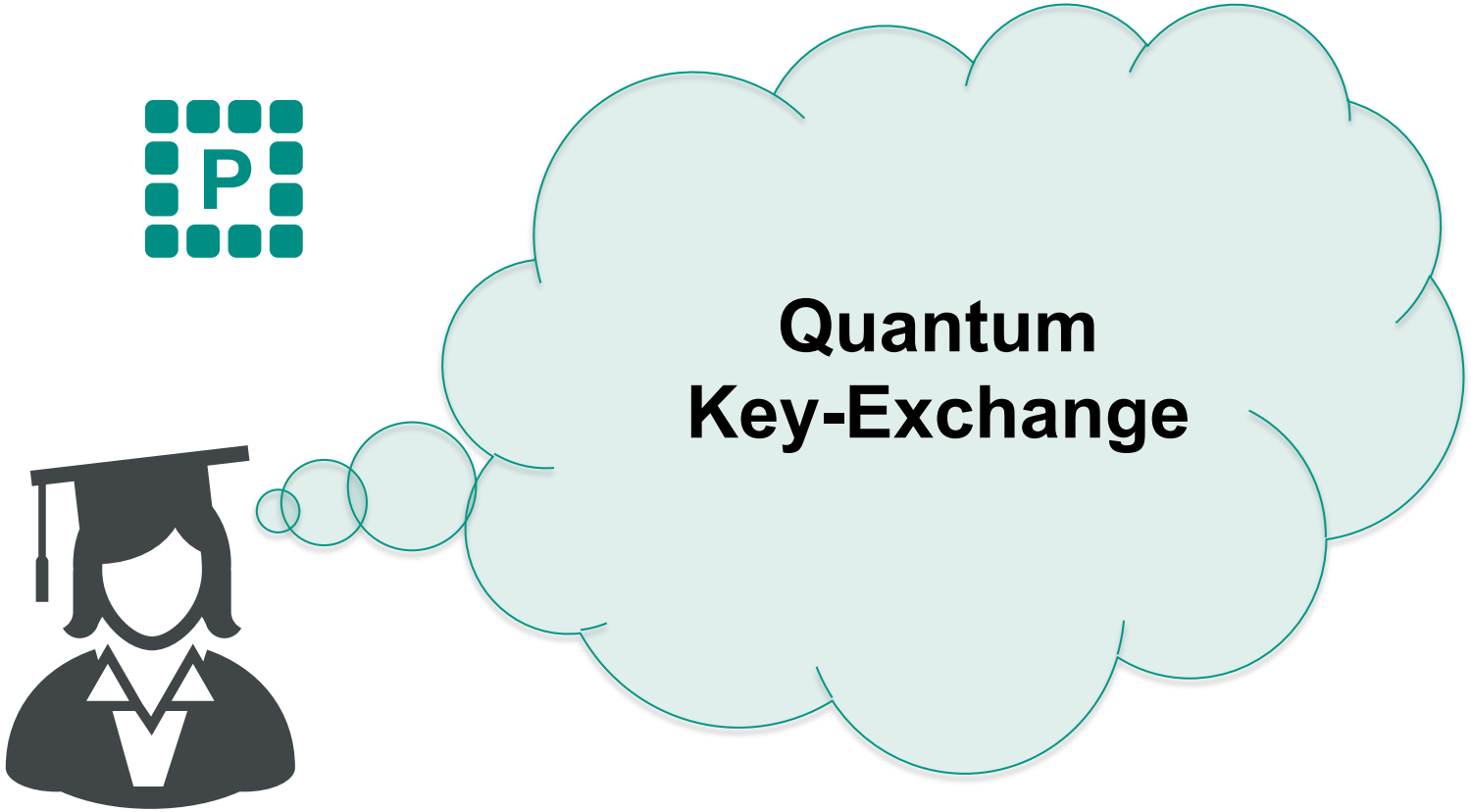
CROSSING

Meet the mighty cryptographer...



Meet the mighty cryptographer...

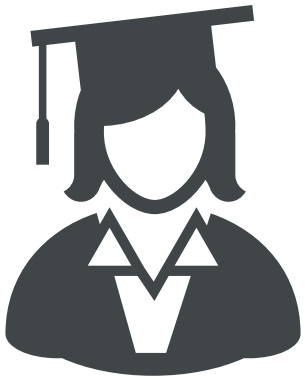
Meet the mighty cryptographer...



Meet the mighty cryptographer...

CROSSING

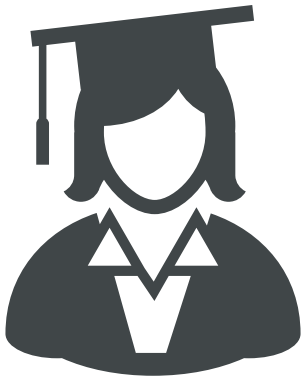
Strongly Secure Connection Establishment



Meet the mighty cryptographer...

CROSSING

Long-Term
Secure Archiving



Meet the mighty developer...

CROSSING

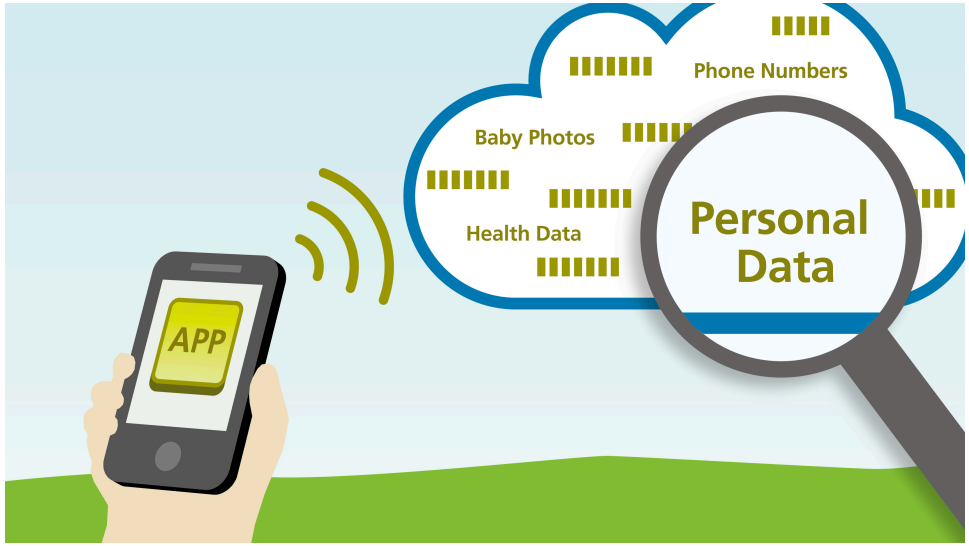
The most dangerous code in the world: validating SSL certificates in non-browser software
Georgiev et al., CCS 2012

Why eve and mallory love Android: an analysis of Android SSL (in)security
Fahl et al., CCS 2012

An empirical study of cryptographic misuse in Android applications
Egele et al., CCS 2013

Meet the mighty developer...

CROSSING



26.05.2015

Technische Universität Darmstadt and Fraunhofer SIT: App Data Vulnerability Threatens Millions of Users

DFG Deutsche Forschungsgemeinschaft

TECHNISCHE UNIVERSITÄT DARMSTADT

Meet the mighty developer...

CROSSING

- How to securely connect to the server?
- How to encrypt app data on disk?
- How to send a message securely?

- What does (a)symmetric mean?
- What is an initialization vector?
- Do I need to ask for a password?
- How can I maintain compatibility?

Bridging the gap

CROSSING

DFG Deutsche Forschungsgemeinschaft

TECHNISCHE UNIVERSITÄT DARMSTADT

Traditional library

CROSSING

Functional interface

```
Cipher.getInstance(String)
```

ORACLE Java SE Documentation

Oracle Technology Network Software Downloads Documentation

Java ™ Cryptography Architecture (JCA) Reference Guide

for Java™ Platform Standard Edition 6

Introduction

Design Principles
Architecture
JCA Concepts

Core Classes and Interfaces

55 A4 pages!!!

JCA, BouncyCastle,
FlexiProvider,
OpenSSL, NaCL

New concept: *active* library

CROSSING

- Helps choose the right algorithm(s) and parameters
- Generates example code
- Assists and monitors the secure integration into your software
- Alerts of component changes
- Alerts of maintenance mistakes in application code

New concept: *active* library

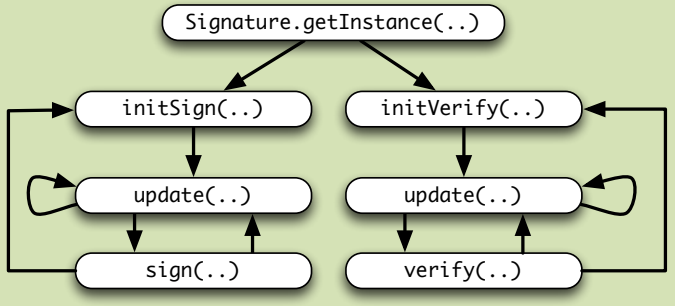
CROSSING

Functional interface

```
Cipher.getInstance(String)
```

Security interface

Usage pattern



DFG Deutsche Forschungsgemeinschaft

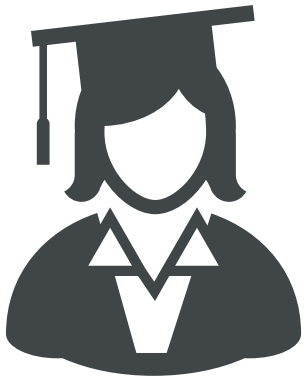
TECHNISCHE UNIVERSITÄT DARMSTADT

Encoding expert knowledge on variability space with CLAFER

CROSSING

```
1  abstract Algorithm
2      name -> string
3      performance ->> integer
4      xor status
5        secure
6        insecure
7
8  abstract Digest : Algorithm
9      outputSize ->> integer //in bits
10
11 abstract KeyDerivationAlgorithm : Algorithm
12
13 abstract Task
14   name -> string
15 DigestAlgorithms
16   md5: Digest
17     [name = "MD5"]
18     [performance = 4 ]
19     [ insecure ]
20     [outputSize = 128]
21
22   sha_224 : Digest
23     [name = "SHA-224"]
24     [outputSize = 224 ]
25     [secure]
26     [performance = 2 ]
27
28   sha_256: Digest
29     [name = "SHA-256"]
30     [outputSize = 256 ]
31     [secure]
32     [performance = 2 ]
33   sha3_384: Digest
34     [name = "SHA3-384"]
35     [outputSize = 384 ]
36     [secure]
37     [performance = 2 ]
38
39
40 KeyDerivationAlgorithms
41   pbkdf : KeyDerivationAlgorithm
42     [name = "PBKDF"]
43     [performance = 2 ]
44     [secure]
45
46   bcrypt : KeyDerivationAlgorithm
47     [name = "Bcrypt"]
48     [performance = 1 ]
49     [secure]
50
51   scrypt : KeyDerivationAlgorithm
52     [name = "Scrypt"]
53     [performance = 1 ]
54     [secure]
55
56 PasswordStoring : Task
57   [name = "Password Storing"]
58   digestToUse -> Digest ?
59   kdaToUse -> KeyDerivationAlgorithm ?
60     [kdaToUse = pbkdf => digestToUse]
61     [kdaToUse != pbkdf => no digestToUse]
62   [kdaToUse || digestToUse]
63   [no digestToUse.status.insecure]
64
```

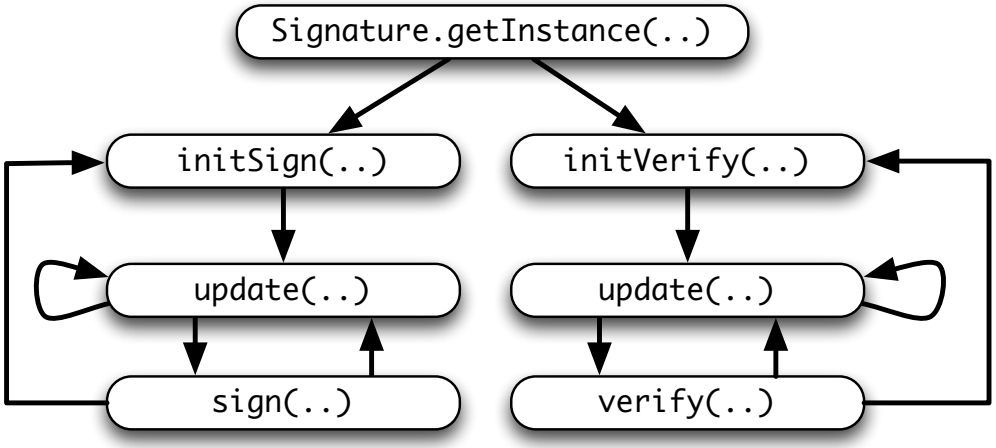
Variability Model



DFG Deutsche Forschungsgemeinschaft

TECHNISCHE UNIVERSITÄT DARMSTADT

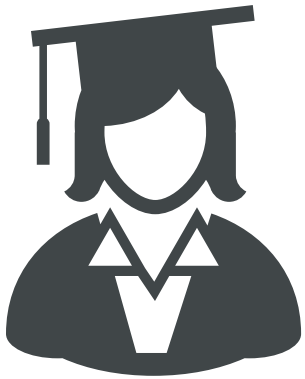
Encoding expert knowledge on usage protocols with TS4J

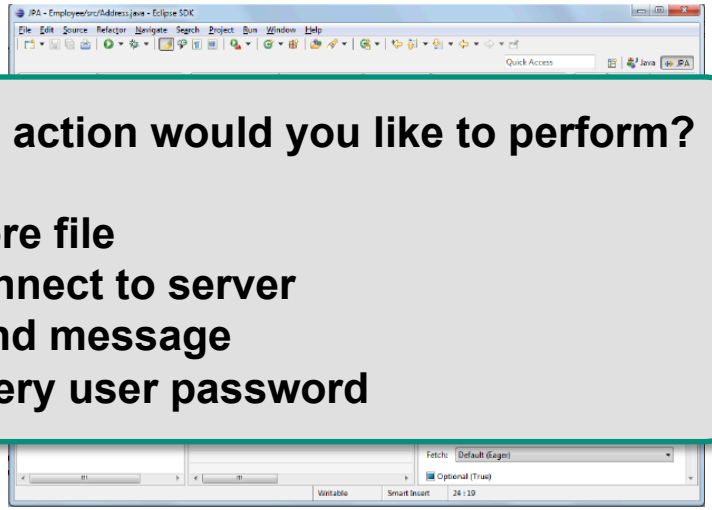


Variability Model

Protocol Model

Rules encoded in lightweight syntax, allowing for simple and direct encoding



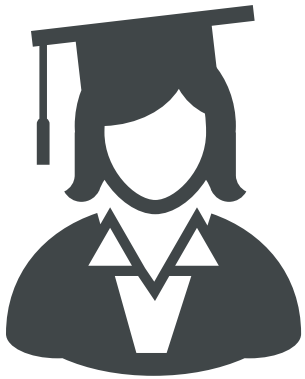


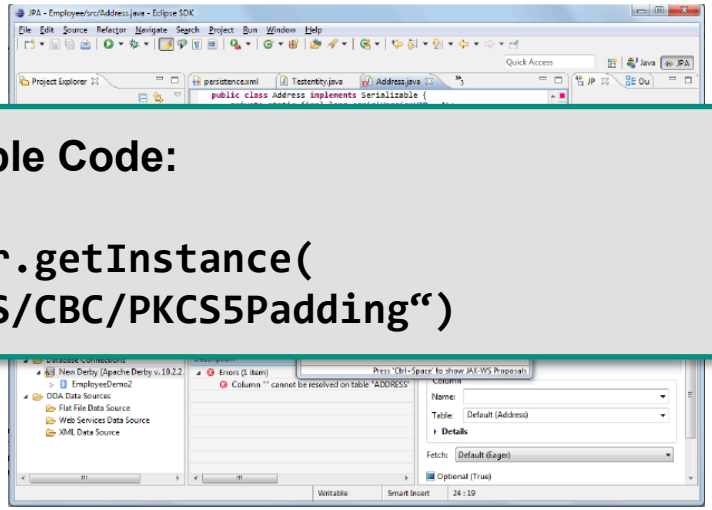
Which action would you like to perform?

- ○ Store file
- ○ Connect to server
- ○ Send message
- ○ Query user password

Variability Model

Protocol Model



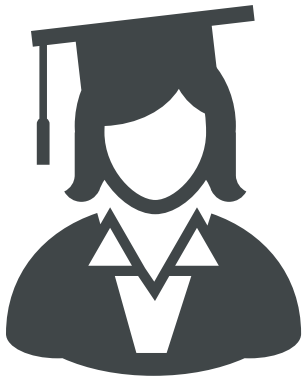


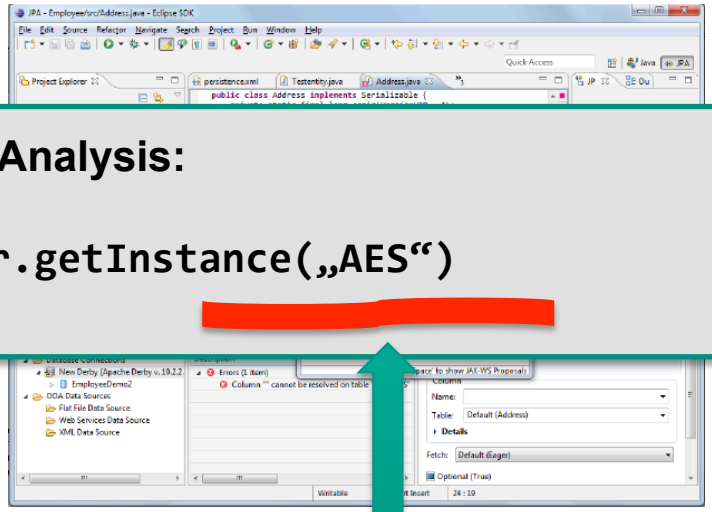
Example Code:

```
Cipher.getInstance(
   „AES/CBC/PKCS5Padding“)
```

Variability Model

Protocol Model



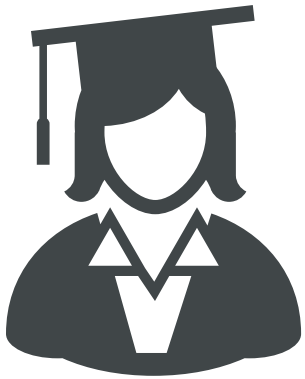


Static Analysis:

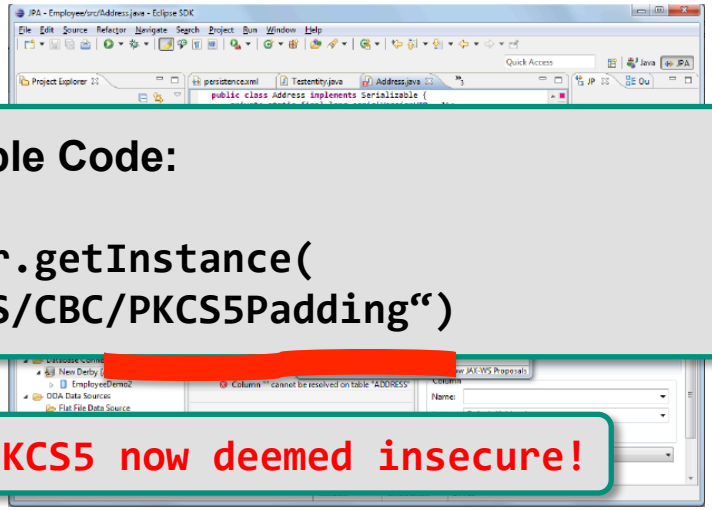
`Cipher.getInstance(„AES“)`

Variability Model

Protocol Model



DFG Deutsche Forschungsgemeinschaft
TECHNISCHE UNIVERSITÄT DARMSTADT



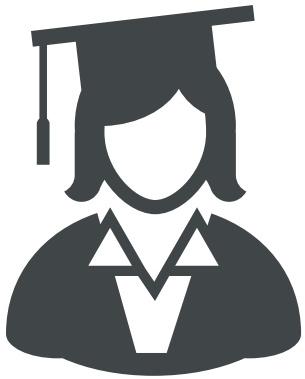
Example Code:

```
Cipher.getInstance(
  „AES/CBC/PKCS5Padding“)
```

PKCS5 now deemed insecure!

Variability Model

Protocol Model



Open questions & important challenges

CROSSING

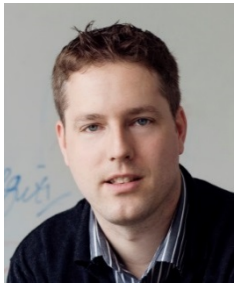
- What common crypto-usage scenarios exist?
- What are the common misuses?

- How can one comprehensively model correct crypto-API usages?
- How to auto-generate example code from those models?
- How to auto-generate static-analysis engines?

- How can different modes/parameters impact the API protocols?
- How to adapt to protocol and/or code changes?

Interested? Visit our poster & demo!

CROSSING



Eric Bodden

Mira Mezini

Karim Ali

Stefan Krüger

Sarah Nadi

DFG Deutsche Forschungsgemeinschaft

TECHNISCHE UNIVERSITÄT DARMSTADT