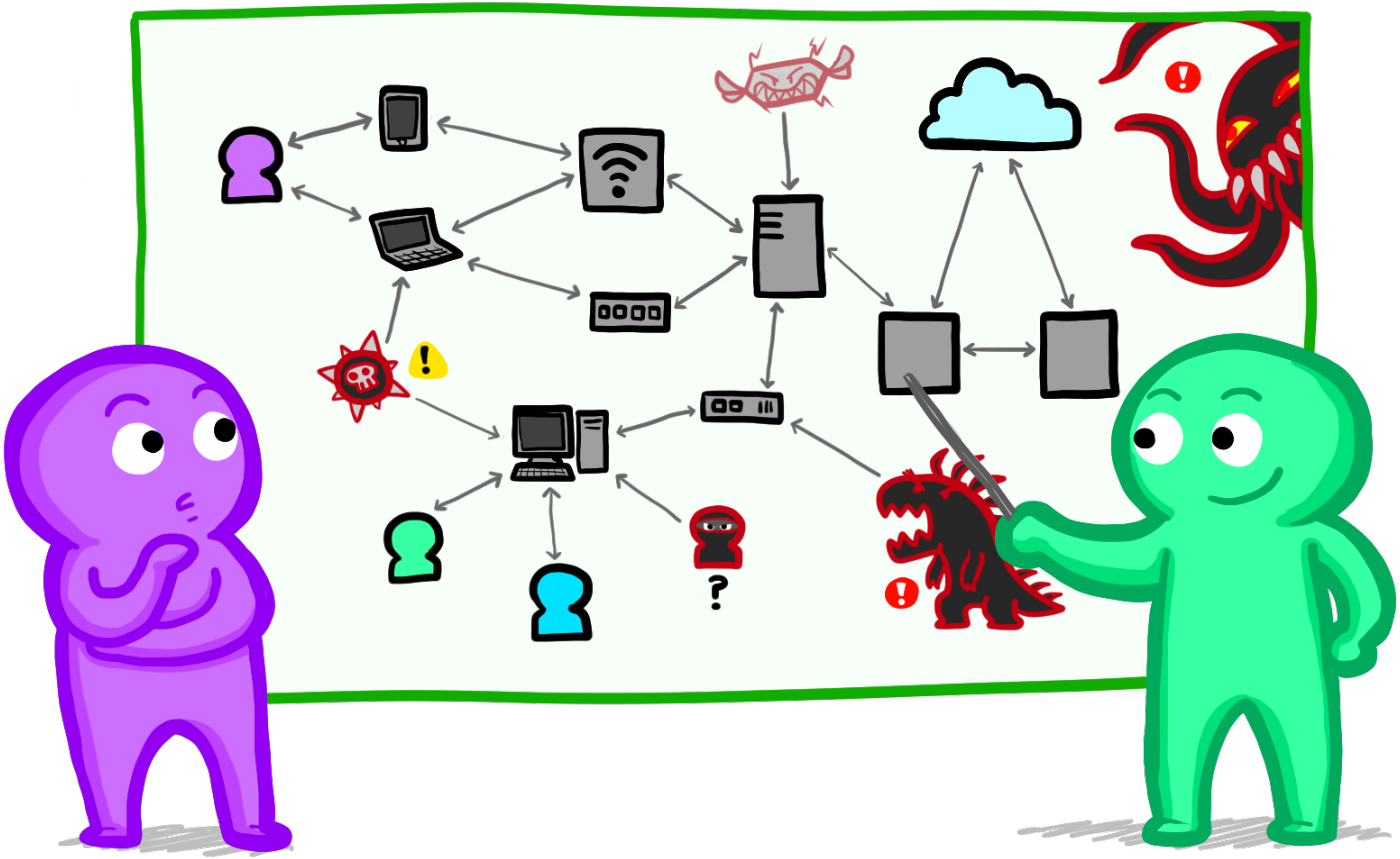
Evaluating the usability of the Ascon 1.2 suite

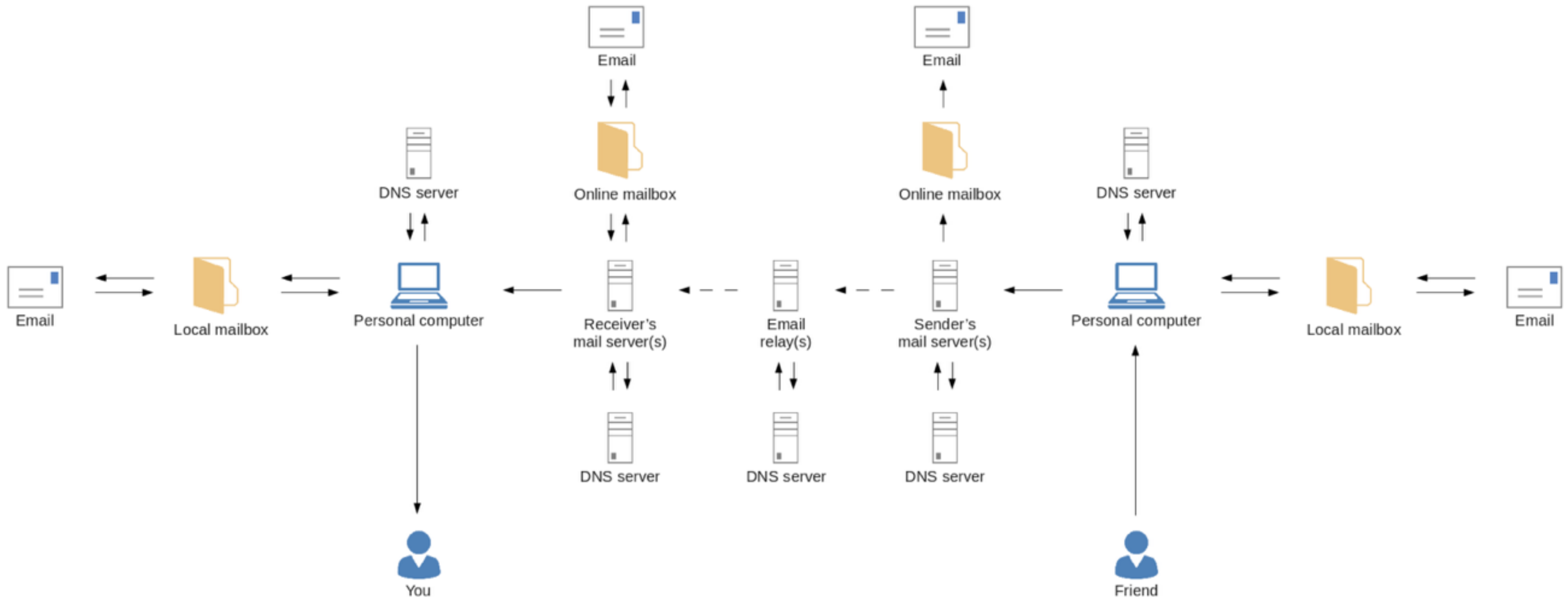
Arne Padmos

$ whoami



Ketrina Yim, 2019

Ridley & Lawler, 2014



Renaud et al., 2014

ARDUINO
UNO

Disclaimer:
all opinions are my own

Lightweight Cryptography Standardization Process: NIST Selects Ascon

February 07, 2023

The NIST Lightweight Cryptography Team has reviewed the finalists based on their submission packages, status updates, third-party security analysis papers, and implementation and benchmarking results, as well as the feedback received during workshops and through the lwc-forum. The decision was challenging since most of the finalists exhibited performance advantages over NIST standards on various target platforms without introducing security concerns.

The team has decided to standardize the **Ascon** family for lightweight cryptography applications as it meets the needs of most use cases where lightweight cryptography is required. Congratulations to the Ascon team! NIST thanks all of the finalist teams and the community members who provided feedback that contributed to the selection.

NIST's next steps will be to:

- Publish NIST IR 8454, which describes the details of the selection and the evaluation process
- Work with the Ascon designers to draft the new lightweight cryptography standard for public comments
- Host a virtual public workshop to further explain the selection process and to discuss various aspects of standardization (e.g., additional variants, functionalities, and parameter selections) as well as possible extensions to the scope of the lightweight cryptography project. The tentative dates for the workshop are June 21-22, 2023. More information will be provided in the upcoming weeks.

NIST Lightweight Cryptography Team

Also see the related NIST news article, *NIST Selects 'Lightweight Cryptography' Algorithms to Protect Small Devices*.

RELATED TOPICS

Security and Privacy: lightweight cryptography

Activities and Products: standards development

RELATED PAGES

News Item: Lightweight Cryptography Finalists Announced
Event: Lightweight Cryptography Workshop 2023

Focus on understanding real-world failure cases

Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

Explore secure channels and record protocols

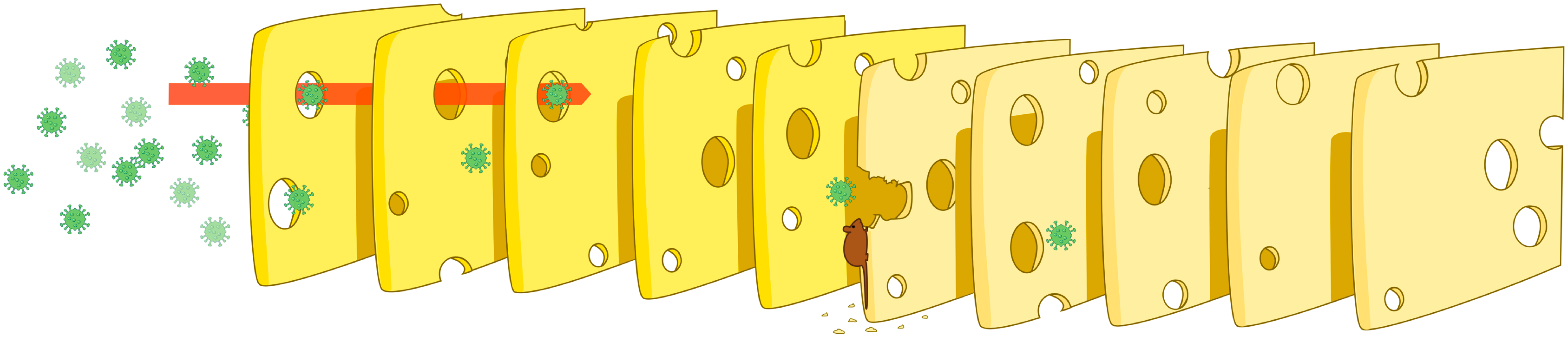
Focus on understanding real-world failure cases

Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

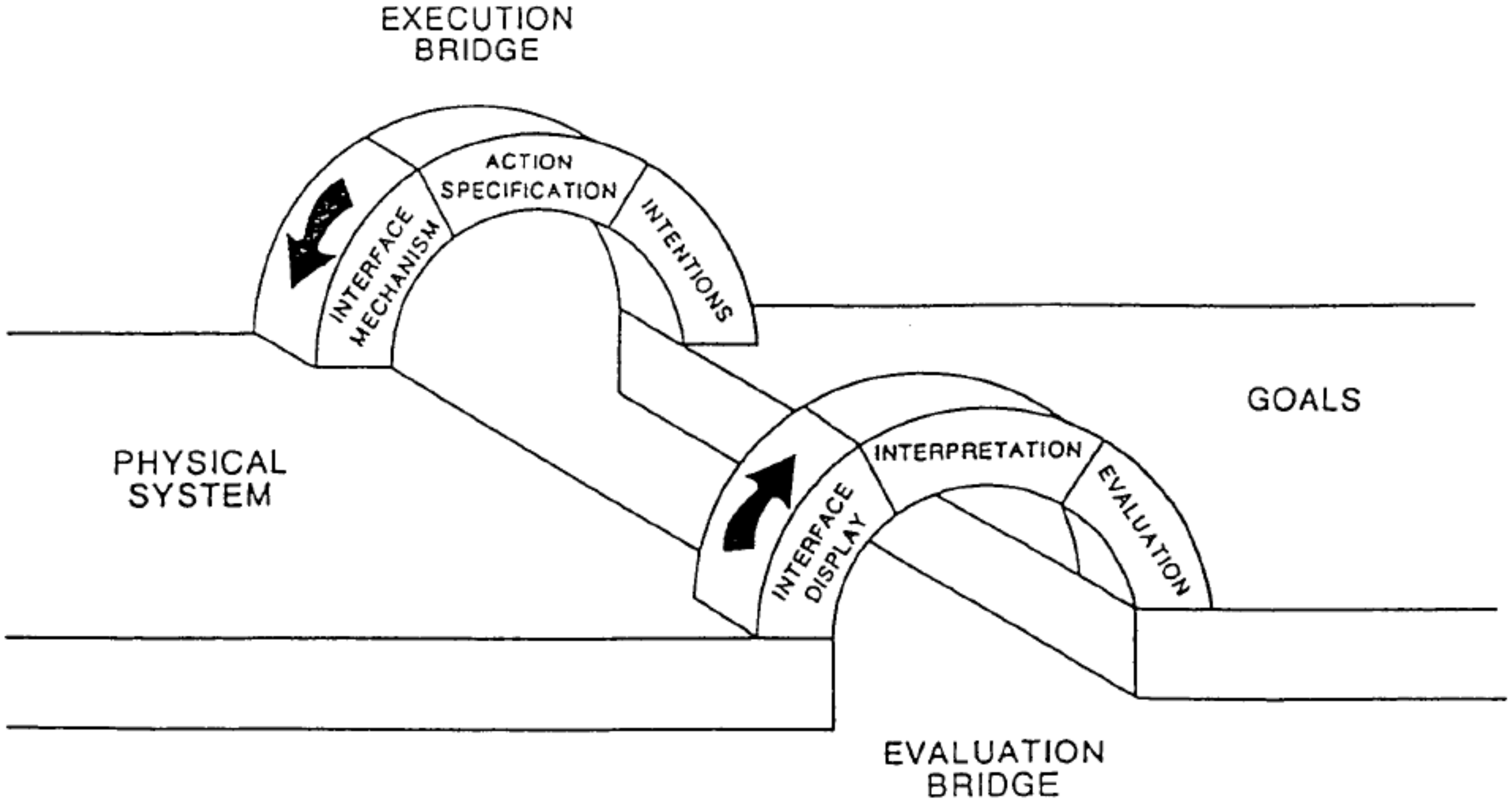
Explore secure channels and record protocols


H
C. Ross Greening



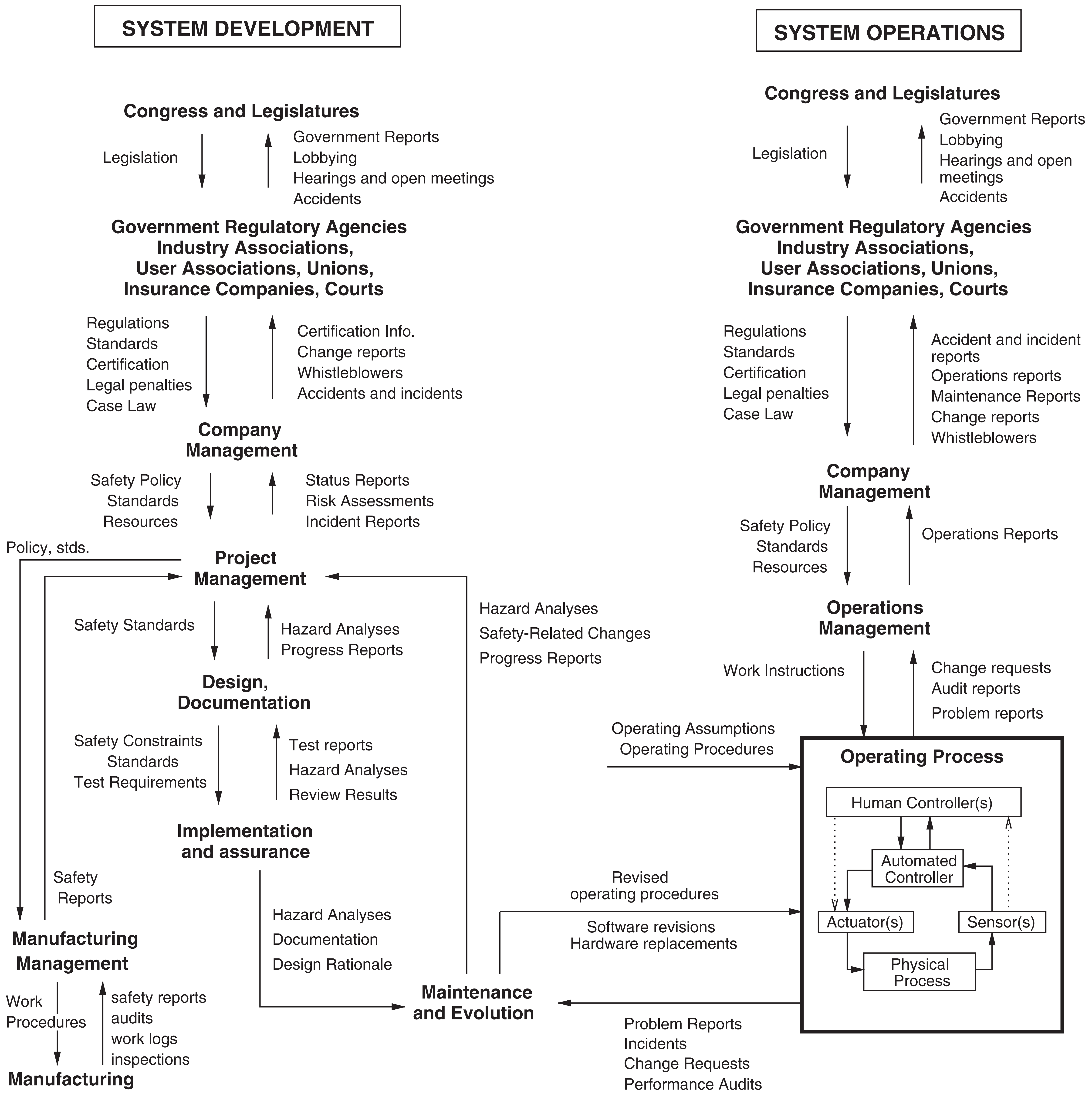
Ian Mackay, 2020


Postbus dec.rap:
Valideer:
Data vervang:
OpmPriklijst:
Klinische gegevens:
Materiaal selectie:
Afwijk. BTI :
Wijzig trial:
IJsselland
Ok
Exit
LAB
Geef patient identificatienummer
Lokaal intranet | Beveiligde modus: uitgeschakeld
100%
NL
10:21
30-10-2015
ThinkVision
bij noodgevallen
112
lenovo
lenovo

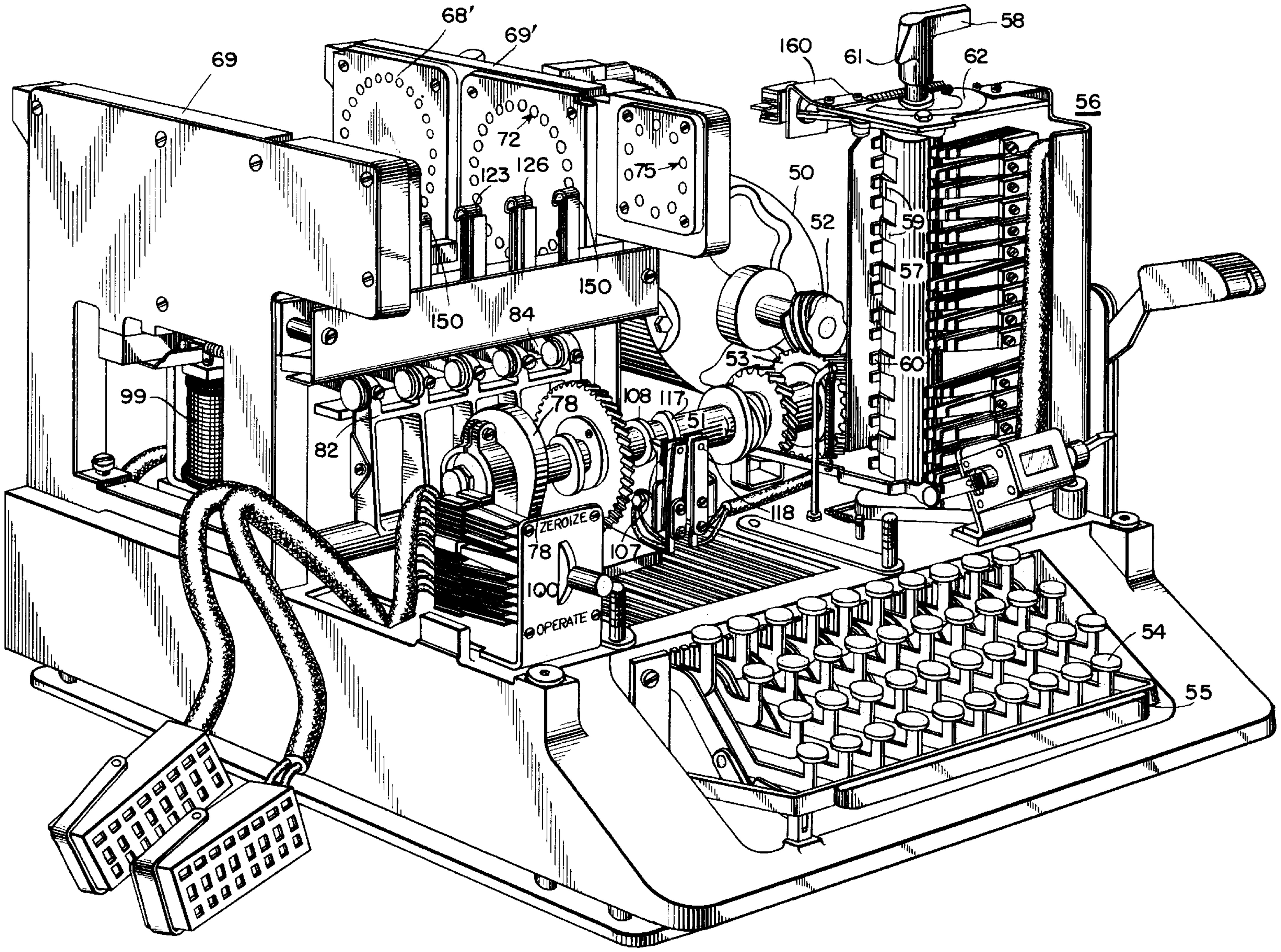


Don Norman, 1986


Kaptain Kobold, 2013



Nancy Leveson, 2012

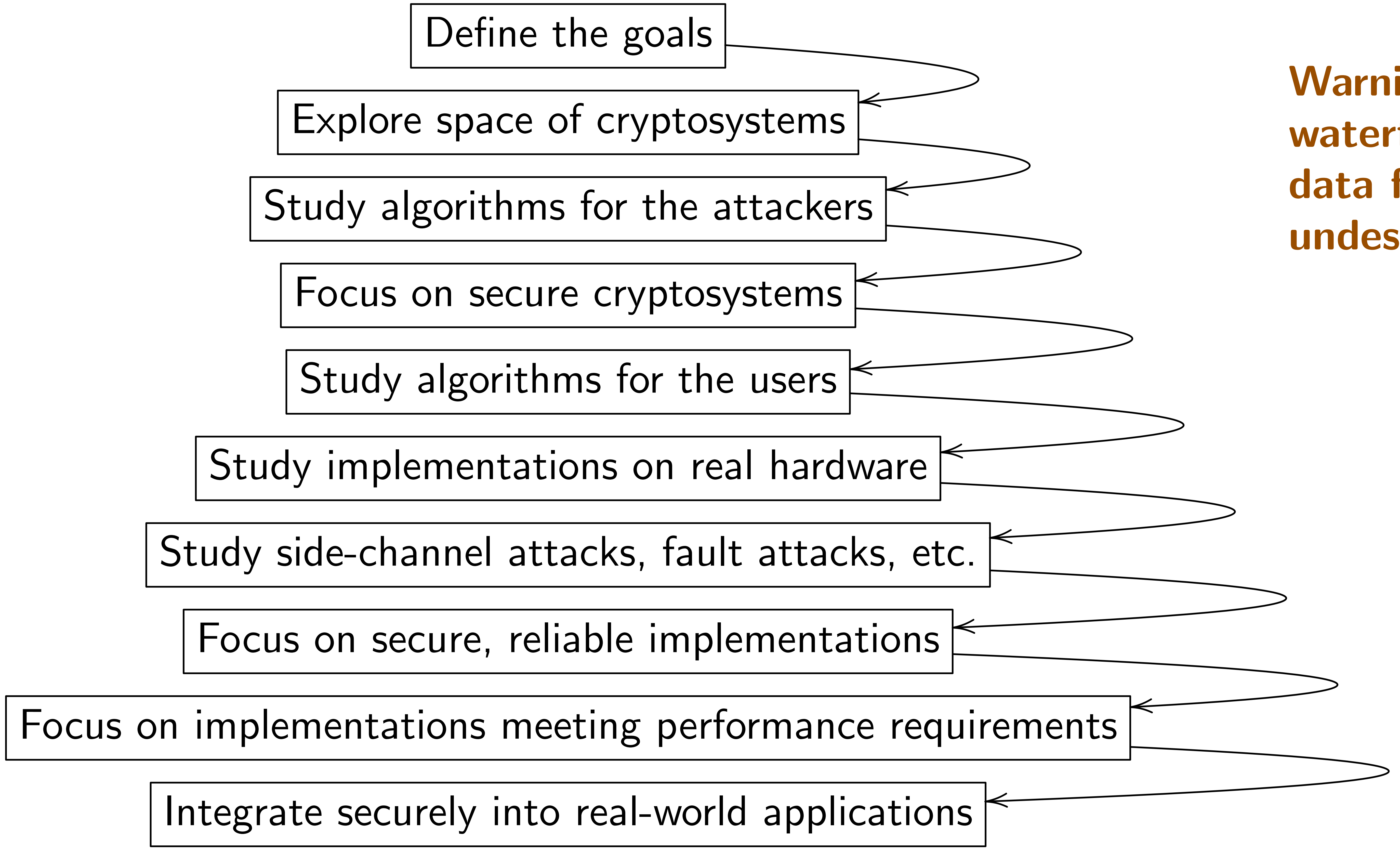


Safford & Seiler, 1944

Cryptographic standards and guidelines should be chosen to **minimise the demands on users** and implementers as well as the adverse consequences of human mistakes and equipment failures.

NISTIR 7977

Many stages of cryptographic research from design to deployment



Warning: waterfall data flow, undesirable.


Toho, 1954

Il est nécessaire, vu les circonstances
qui en commandent l'application,
que le système soit d'un usage facile.

Auguste Kerckhoffs, 1883

Why Cryptosystems Fail

Ross Anderson
University Computer Laboratory
Pembroke Street, Cambridge CB2 3QG
Email: rja14@cl.cam.ac.uk

Abstract

Designers of cryptographic systems are at a disadvantage to most other engineers, in that information on how their systems fail is hard to get: their major users have traditionally been government agencies, which are very secretive about their mistakes.

In this article, we present the results of a survey of the failure modes of retail banking systems, which constitute the next largest application of cryptology. It turns out that the threat model commonly used by cryptosystem designers was wrong: most frauds were not caused by cryptanalysis or other technical attacks, but by implementation errors and management failures. This suggests that a paradigm shift is overdue in computer security; we look at some of the alternatives, and see some signs that this shift may be getting under way.

quiries are conducted by experts from organisations with a wide range of interests - the carrier, the insurer, the manufacturer, the airline pilots' union, and the local aviation authority. Their findings are examined by journalists and politicians, discussed in pilots' messes, and passed on by flying instructors.

In short, the flying community has a strong and institutionalised learning mechanism. This is perhaps the main reason why, despite the inherent hazards of flying in large aircraft, which are maintained and piloted by fallible human beings, at hundreds of miles an hour through congested airspace, in bad weather and at night, the risk of being killed on an air journey is only about one in a million.

In the crypto community, on the other hand, there is no such learning mechanism. The history of the subject ([K1], [W1]) shows the same mistakes being made over and over again; in particular, poor management of codebooks

VII-26-X

A HISTORY OF U.S. COMMUNICATIONS SECURITY (U)
(The David G. Boak Lectures)

HANDLING INSTRUCTIONS

1. This publication consists of covers and numbered pages 1 to 101 inclusive. Verify presence of each page upon receipt.

2. Formal authorization for access to ~~SECRET~~ material is required for personnel to have access to this publication.

3. This publication will not be released outside government channels without approval of the Director, National Security Agency.

4. Extracts from this publication may be made for classroom or individual instruction purposes only. Such extracts will be classified ~~SECRET~~ NOFORN and accounted for locally until destroyed.

5. This publication will not be carried in aircraft for use therein.

NATIONAL SECURITY INFORMATION
Unauthorized Disclosure Subject to Criminal Sanctions

NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND 20755

Revised July 1973

Classified by Director, NSA, pursuant to NSA, Manual 123-2.
Exempt from General Declassification Schedule
of Executive Order 11652 Exempt Category 2.
Declassification date cannot be determined.

~~SECRET~~

ORIGINAL 1
Reverse (Page 2) Blank

DECLASSIFIED UNDER AUTHORITY OF THE
INTERAGENCY SECURITY CLASSIFICATION APPEALS PANEL,
E.O. 13526, SECTION 5.3(b)(3)

ISCAP APPEAL NO. 2009-049, document no. 1
DECLASSIFICATION DATE: October 14, 2015

~~SECRET~~

A HISTORY
OF
U.S. COMMUNICATIONS SECURITY (U)

THE DAVID G. BOAK LECTURES

VOLUME II

NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND 20755

The information contained in this publication will not be disclosed to foreign nationals or their representatives without express approval of the DIRECTOR, NATIONAL SECURITY AGENCY. Approval shall refer specifically to this publication or to specific information contained herein.

JULY 1981

CLASSIFIED BY NSA/CSSM 123-2
REVIEW ON 1 JULY 2001

DECLASSIFIED UNDER AUTHORITY OF THE
INTERAGENCY SECURITY CLASSIFICATION APPEALS PANEL,
E.O. 13526, SECTION 5.3(b)(3)

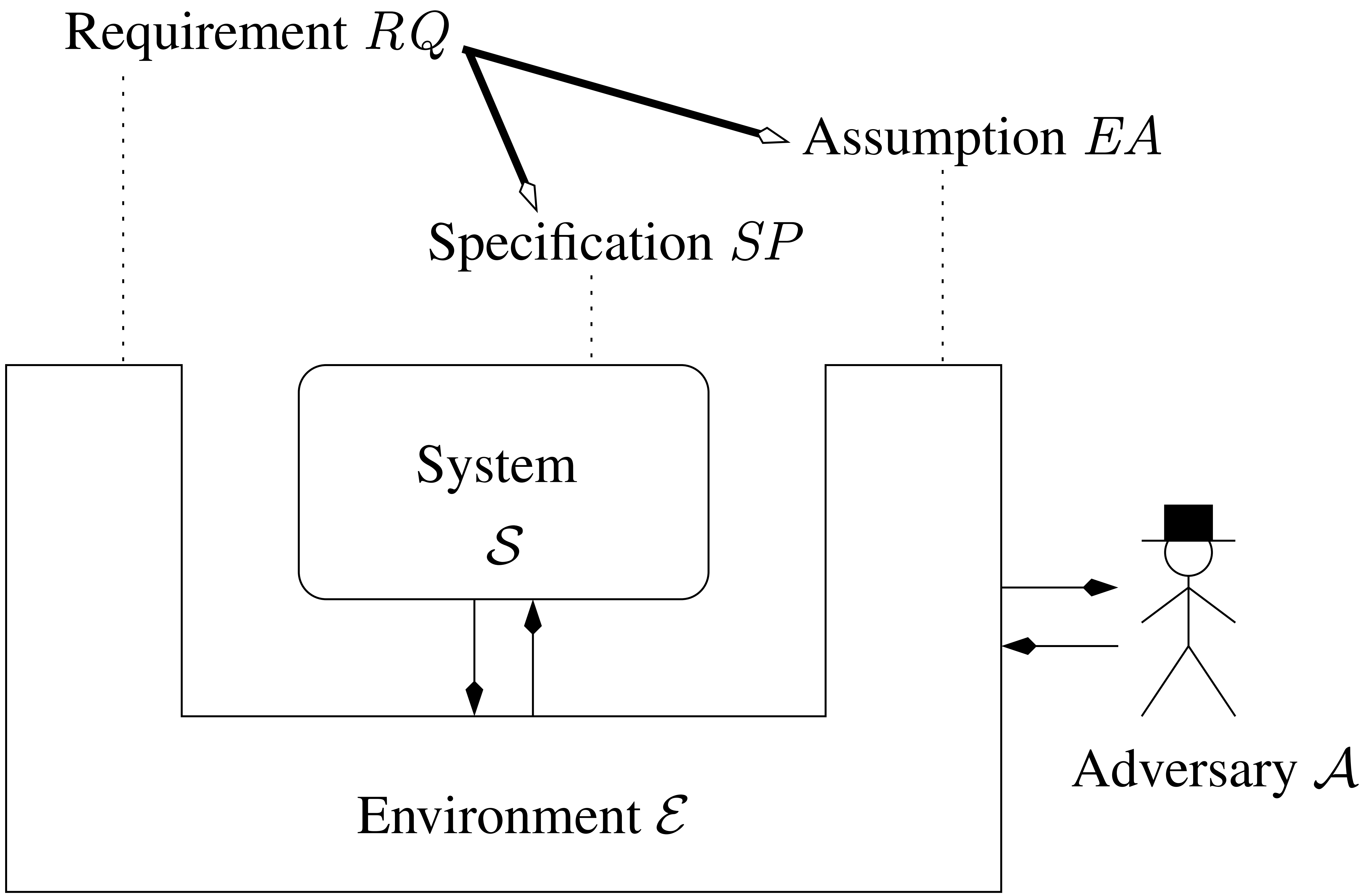
ISCAP APPEAL NO. 2009-049, document no. 2
DECLASSIFICATION DATE: October 14, 2015

NOT RELEASABLE TO FOREIGN NATIONALS

~~SECRET~~
HANDLE VIA COMINT CHANNELS ONLY

ORIGINAL
(Reverse Blank)

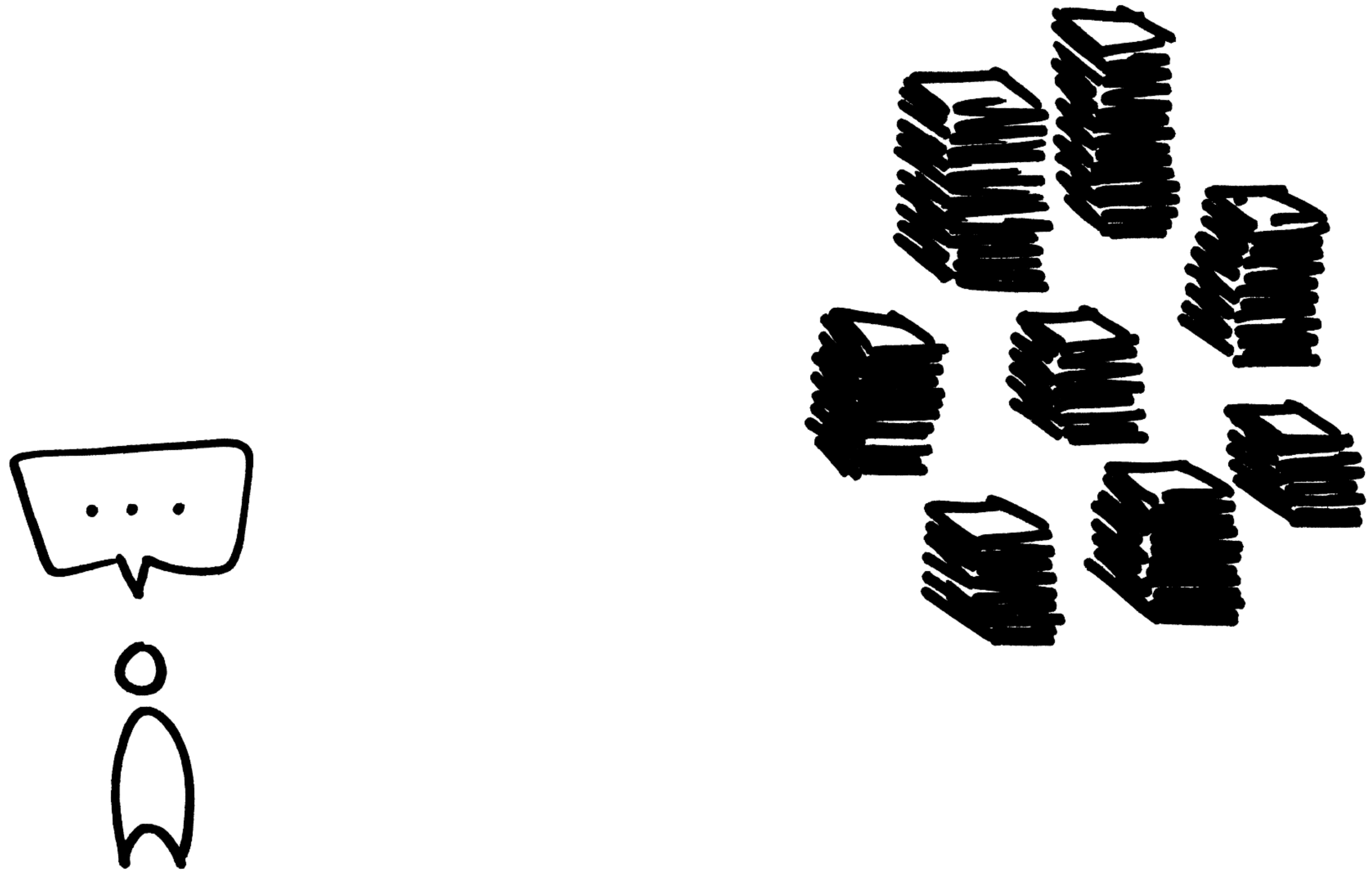
[redacted]



Dashti & Basin, 2016

Review of the December 2021 Log4j Event

Publication: July 11, 2022
Cyber Safety Review Board

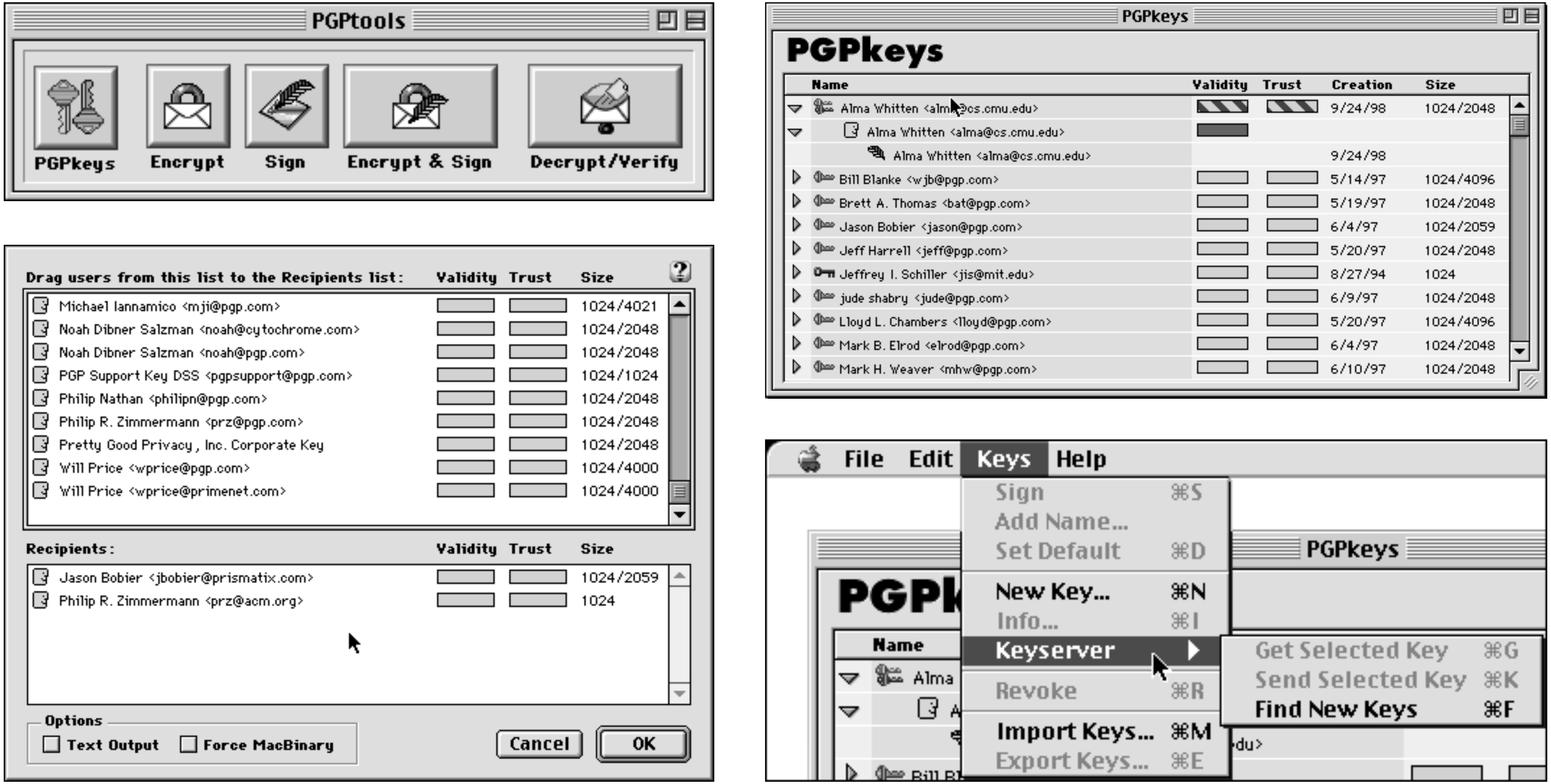


Focus on understanding real-world failure cases

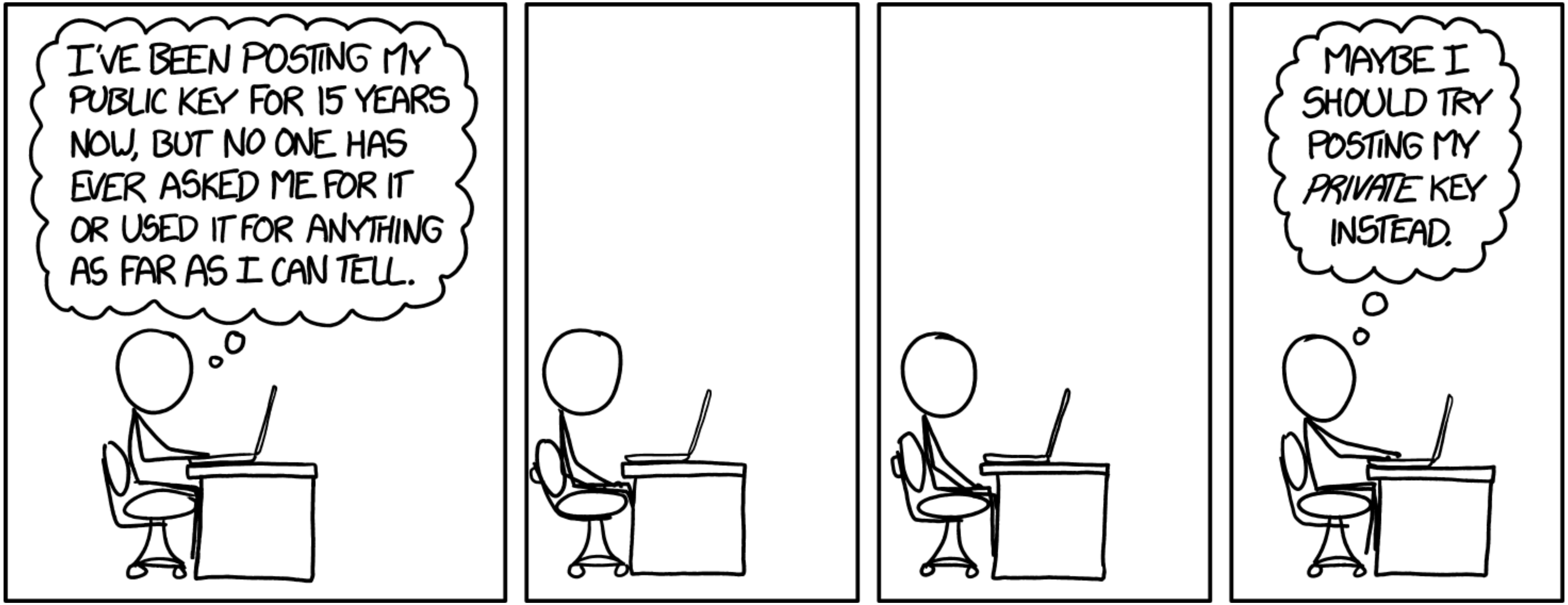
Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

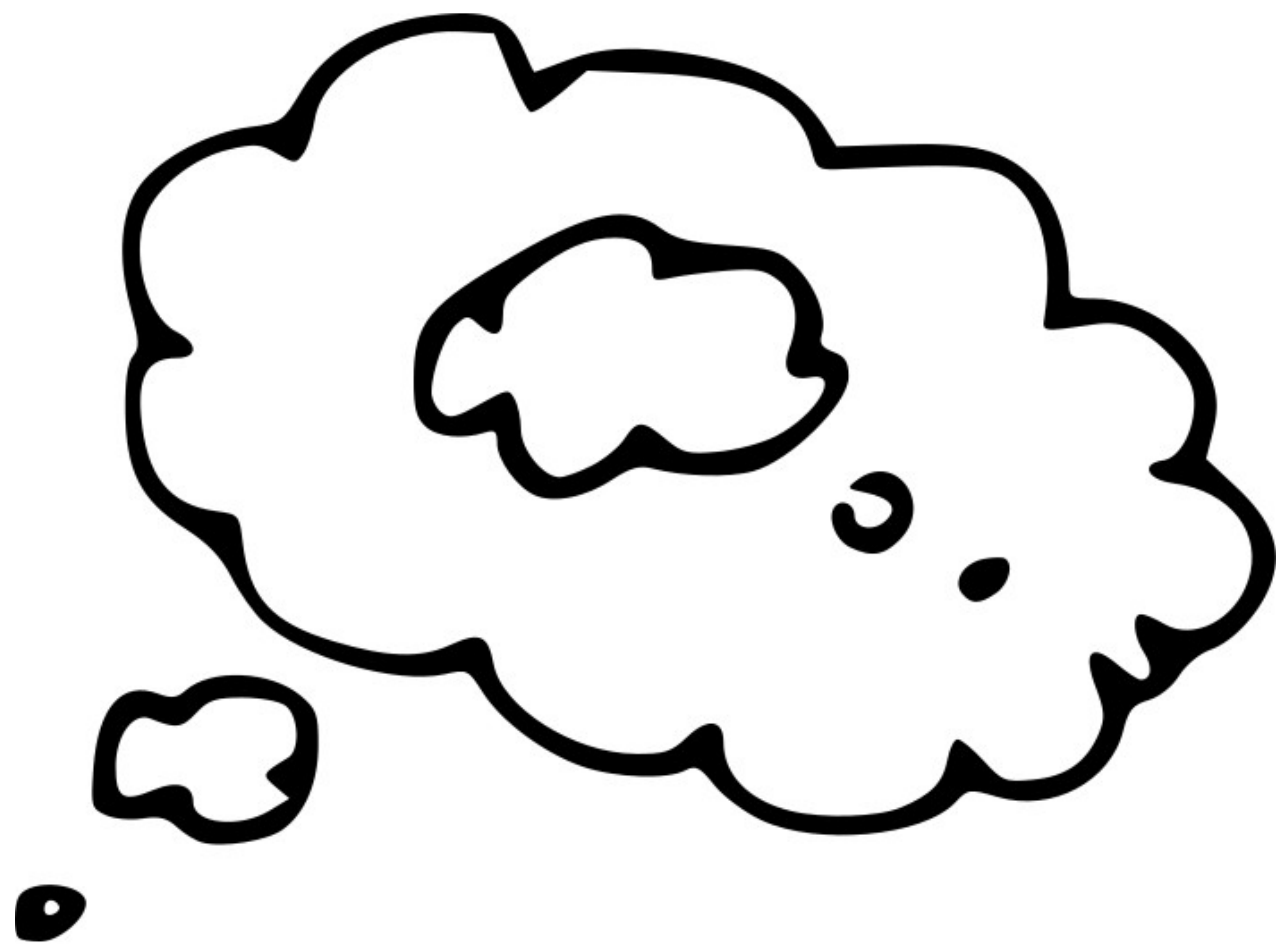
Explore secure channels and record protocols



Whitten & Tygar, 1999


I'VE BEEN POSTING MY PUBLIC KEY FOR 15 YEARS NOW, BUT NO ONE HAS EVER ASKED ME FOR IT OR USED IT FOR ANYTHING AS FAR AS I CAN TELL.
MAYBE I SHOULD TRY POSTING MY *PRIVATE* KEY INSTEAD.

xkcd, 2015



To strengthen systems across the board, security professionals must focus on creating developer-friendly approaches.

Green & Smith, 2016

Challenges in Authenticated Encryption

Editor
Daniel J. Bernstein

Contributors (alphabetical order; affiliations included for identification only)
Jean-Philippe Aumasson (Kudelski Security, Switzerland)
Steve Babbage (Vodafone, UK)
Daniel J. Bernstein (University of Illinois at Chicago, USA; Technische Universiteit Eindhoven, Netherlands)
Carlos Cid (Royal Holloway, University of London, UK)
Joan Daemen (STMicroelectronics, Belgium; Radboud Universiteit, Netherlands)
Orr Dunkelman (University of Haifa, Israel)
Kris Gaj (George Mason University, USA)
Shay Gueron (University of Haifa, Israel; Intel, Israel)
Pascal Junod (HEIG-VD, Switzerland)
Adam Langley (Google, USA)
David McGrew (Cisco, USA)
Kenny Paterson (Royal Holloway, University of London, UK)
Bart Preneel (KU Leuven, Belgium)
Christian Rechberger (Danmarks Tekniske Universitet, Denmark)
Vincent Rijmen (KU Leuven, Belgium)
Matt Robshaw (Impinj, USA)
Palash Sarkar (Indian Statistical Institute, Kolkata, India)
Patrick Schaumont (Virginia Tech, USA)
Adi Shamir (Weizmann Institute, Israel)
Ingrid Verbauwhede (KU Leuven, Belgium)

17. July 2015 (workshop) + 1. March 2017 (white paper)

Revision 1.05

The work described in this report has in part been supported by the Commission of the European Communities through the H2020-ICT program under contract H2020-ICT-2014 no. 645421.

Contents

ascon 0.0.9

```
pip install ascon
```

Latest version

Released: Mar 24, 2023

Lightweight authenticated encryption and hashing

Project links

Homepage

Statistics

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Project description

Python implementation of Ascon

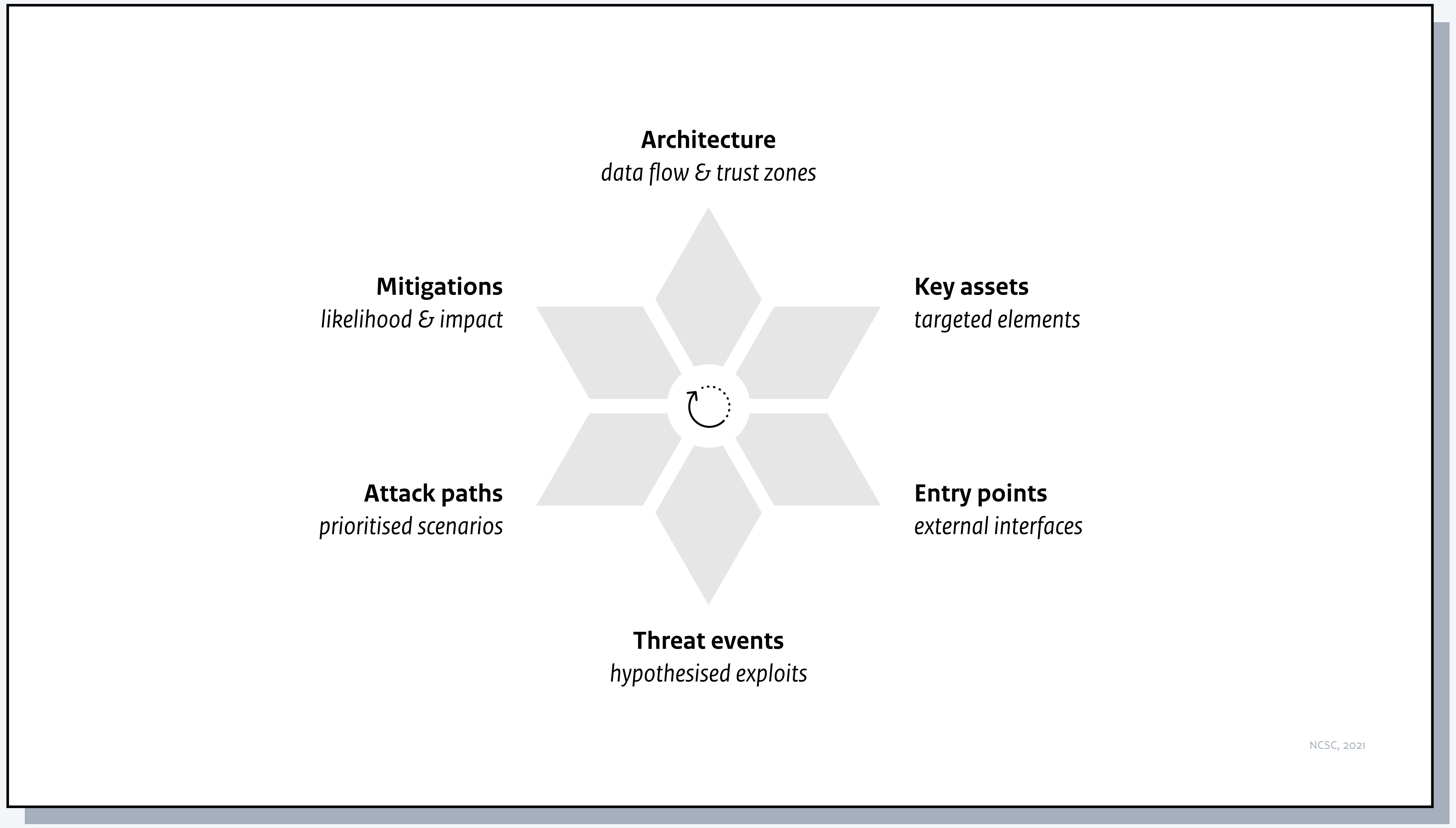
This is a Python3 implementation of Ascon v1.2, an authenticated cipher and hash function.

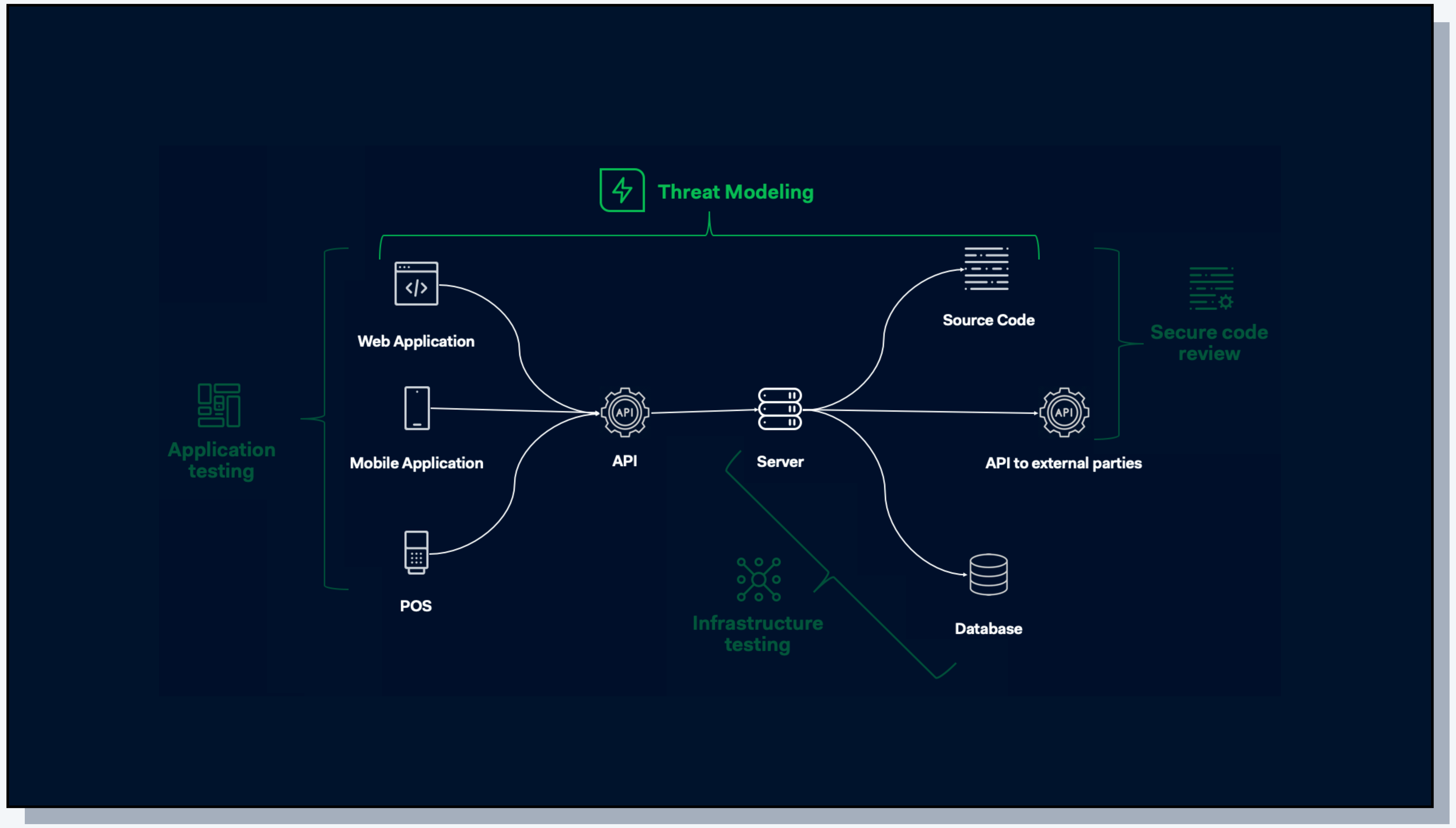
https://github.com/meichlseder/pyascon

Ascon

Ascon is a family of authenticated encryption (AEAD) and hashing algorithms designed to be lightweight and easy to implement, even with added countermeasures against side-channel attacks. It was designed by a team of cryptographers from Graz University of Technology, Infineon Technologies, and Radboud University: Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer.

Ascon has been selected as the standard for lightweight cryptography in the NIST Lightweight Cryptography competition (2019–2023) and as the primary choice for lightweight authenticated encryption in the final portfolio of the CAESAR competition (2014–2019).


Architecture
data flow & trust zones
Mitigations
likelihood & impact
Key assets
targeted elements
Attack paths
prioritised scenarios
Entry points
external interfaces
Threat events
hypothesised exploits
NCSC, 2021


Threat Modeling
Web Application
Source Code
Secure code review
Mobile Application
API
Server
API to external parties
Application testing
POS
Infrastructure testing
Database

Architectural diagrams
with security sauce

Processes
Where data will change from one form to another.

Data flows
Represents data moving from one part of the system to elsewhere.

Data stores
Indicates data at rest, i.e. a place for longer storage.

Terminators
Also called actors or external entities. These are the limits of analysis.

Trust zones
Can be drawn as trust boundaries, i.e. dotted lines between elements.

Confidentiality	Information disclosure
Integrity	Tampering
Availability	Denial of service
Authentication	Spoofing
Authorisation	Elevation of privilege
Accountability	Repudiation

6.3 SR-2: Threat model

6.3.1 Requirement

A process shall be employed to ensure that all products shall have a threat model specific to the current development scope of the product with the following characteristics (where applicable):

a) correct flow of categorized information throughout the system;

b) trust boundaries;

c) processes;

d) data stores;

e) interacting external entities;

f) internal and external communication protocols implemented in the product;

g) externally accessible physical ports including debug ports;

h) circuit board connections such as Joint Test Action Group (JTAG) connections or debug headers which might be used to attack the hardware;

i) potential attack vectors including attacks on the hardware, if applicable;

j) potential threats and their severity as defined by a vulnerability scoring system (for example, CVSS);

k) mitigations and/or dispositions for each threat;

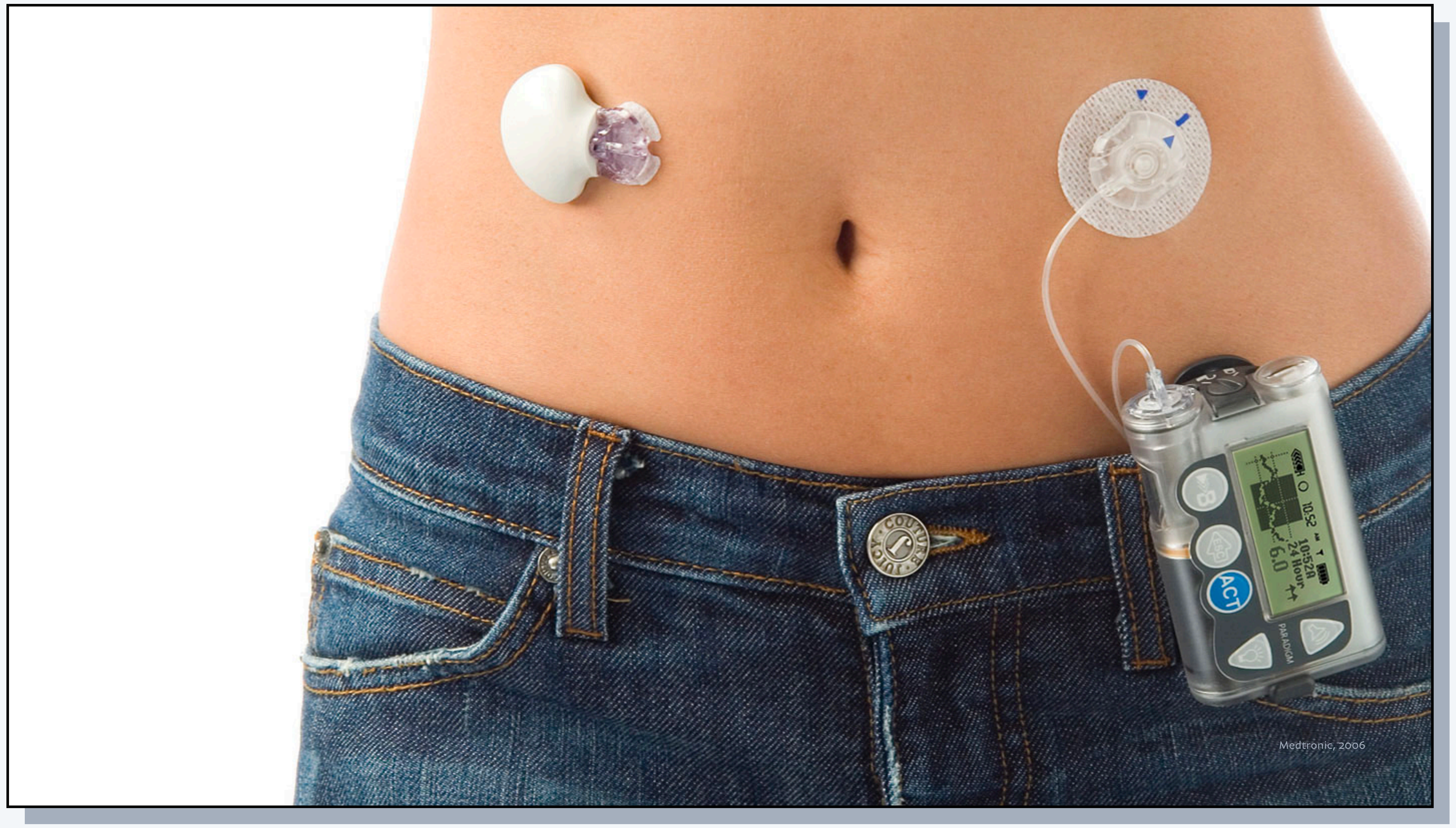
l) security-related issues identified; and

m) external dependencies in the form of drivers or third-party applications (code that is not developed by the supplier) that are linked into the application.

The threat model shall be reviewed and verified by the development team to ensure that it is correct and understood.

The threat model shall be reviewed periodically (at least once a year) for released products and updated if required in response to the emergence of new threats to the product even if the design does not change.

Any issues identified in the threat model shall be addressed as defined in 10.4 and 10.5.



Components

Data flows

Crown jewels

Trust zones

Assumptions

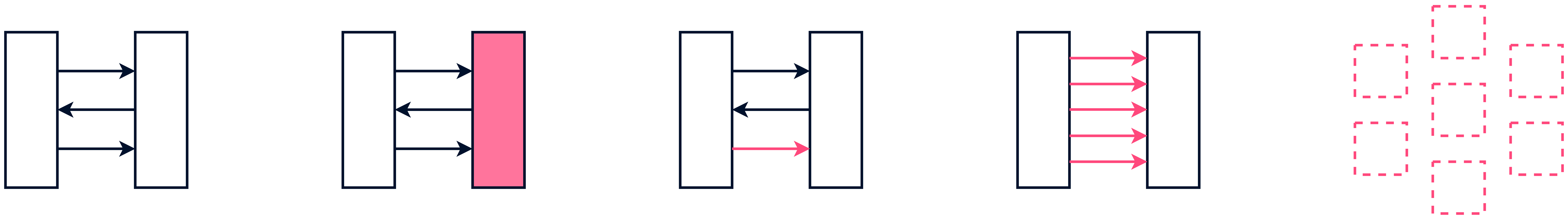
Threats (STRIDE)

Prioritisation

Countermeasures

Security testing

Follow-up



Silent 'pair programming'

— Don't want to break the flow
— Switch every five minutes
— Apply the refinement approach

10 min. Outline the program's structure as comments
What message(s) will you be sending/receiving?
Which algorithm(s) will you be using for this?

10 min. Write pseudocode to make your ideas tangible

20 min. Translate your pseudocode into Python code

```
https://pypi.org/p/ascon

$ pip install ascon

>>> import ascon
>>> ascon.[tab][tab]
>>> data = b"..."
>>> print(data.hex())
```

Mail your **commented code** to
ascon@arnepadmos.com

Phase 1 – Comments
Alignment of flows and our threat model

Phase 2 – Pseudocode
Match of structure to messages and threats

Phase 3 – Source code
Compare comments to the functions used

Exploratory initial qualitative observations:

— Zero, one, or just a couple of parameters passed
— Wrapper functions taking a message as input
— Hardcoded or empty nonce/key, e.g. in wrapper
— Parameters to library appearing out of thin air
— No key diversification, error handling, etc.

```python
#importing ascon
#create a string that contains the byte string
#sending encrypted data to the sensor
#decrypting the data

import ascon
def encrypt():
    key = b"SECRETSAREHIDDEN"
    message = b"hALLO DIT IS MIJN MESSAGE MET DE VOLGENDE WAARDE: "
    nonce = bytes(16)
    associateddata = b"RELATEDDATA"

    x = ascon.encrypt(key, nonce, associateddata, message, variant="Ascon-80pq")
    return x

#decrypting the data
def decrypt():
    key = b"SECRETSAREHIDDEN"
    nonce = bytes(16)
    associateddata = b"RELATEDDATA"
    y = ascon.decrypt(key, nonce, associateddata, x, variant="Ascon-80pq")
    return y
```

```python
import ascon                                       # Import the ASCON module

ascon = ascon.ASCON()                              # Create an ASCON object

data = b""                                         # Create an empty byte array

for i in range(0, 100):                            # Loop 100 times
    data += bytes([i])                             # Add the current value of i to the data array

def send_encrypted_message(message):               # Define a function to send an encrypted message
    ascon.send(ascon.encrypt(message))             # Encrypt the message and send it

def receive_encrypted_message():                   # Define a function to receive an encrypted message
    return ascon.decrypt(ascon.receive())          # Receive the message and decrypt it

def send_encrypted_ack():                          # Define a function to send an encrypted acknoledgement
    ascon.send(ascon.encrypt(b"\x06"))             # Encrypt the acknoledgement and send it

def receive_encrypted_ack():                       # Define a function to receive an encrypted acknoledgement
    return ascon.decrypt(ascon.receive())          # Receive the acknoledgement and decrypt it

print(data.hex())                                  # Print the data in hexadecimal format
```

```python
import ascon

def get_data():
    message = ascon.encrypt('give data')
    sensor = 'XX-XX-XX-XX-XX-XX'

    data = ascon.decrypt(send(message,sensor)) # send message to mac sensor and encrypt + [...]

    if data != NULL:
        # data is present so we send the data back
        message = 'ack'
        return data
    elif data = NULL:
        # if no resonse is given, try again
        get_data()

def processdata():
    data = get_data()
    if data < 4.0:
        ins_pump() # send prompt for pump to pump insulin
    elif data > 7.0:
        alert_message() # alert on screen that glucose is too high
```

```
#   messages    :
#data is gatherd from sensor and send to the pump
#data is encrypted (Ascon128)
#pump send a request
#get and ack from the pump
#message auth before sending the data with(ascon-Mac)

#   who :
#data is gatherd on the sensor
#data is sent to the pump
#ack from pump

#   processing  :
#gathering of data from the sensor
#sending of data from sensor to pump
#decryptin of data on the pump
#sending of ack from the pump to the sensor

#pseudo:

#def message_auth(mac-sender, mac-reciever)

    #send auth message from mac-sender to mac reciever
    #if mac_reciever == mac_reciever :
```

```python
# PHASE 1

# Sensor:
    # send sugarlevels, authentication, checksum
    # send ack received, authentication, checksum
    # send battery level // if battery is low send alert
    # log battery level // if abnormal send alert
    # log connection // if connection behaviour is abnormal drop connection for 10 min.

# Pump:
    # send ack, authentication, checksum
    # log insulin injection

# PHASE 2

# check authentication by checking authentication message
# check integrity by checking the checksum
# check elevation of privilege by checking the log of the battery

# PHASE 3

import ascon
from time import sleep
from time import perf_counter
```

```python
def authentication():
    id = 'apparaatnaam'
    hash = 'hash'
    encrypted_auth = ascon.encrypt(id + hash)
    return encrypted_auth

def send_ack():
    encrypted_auth = authentication()
    ack = 'message_received'
    return encrypted_auth, ack

def day_log_battery():
    log = []
    x = True
    tic = perf_counter()
    while x == True:
        battery_level = check_battery_level(sensorid)
        log.append(battery_level)
        sleep(300) # 300 seconds = 5 min
        toc = perf_counter()
        if toc - tic == 86400: # 86400 seconds = 1 day
            x = False
    return log
```

```python
# There is a sensor and a pump, which is sending data from sensor to pump.
# There will be a acknowledgement from pump to sensor.
# The data will be send in integers.
# Spoofing = act as an pump.
# Tampering = interrupt data.
# Information disclosure = intercept and capture sensor data.
# DOS = battery drainage and send garbage.

#Psuedocode
#[...]

def data_encrypt(key, nonce, associateddata, plaintext, data):

    ascon.encrypt(key, nonce, associateddata, plaintext, variant="Ascon-128")
    data = b"blahblahblah"
    print(data.hex())
    return data.hex

def data_decyrpt(key, nonce, associateddata, plaintext, data):

    ascon.decrypt(key, nonce, associateddata, plaintext, variant="Ascon-128a")
    data = b"blahblahblah"
    print(data.hex())
    return data.hex
```

```python
#Sensor:
#   measure blood
#   create uid for message
#   encrypt message + uid
#   Send ecnrypted message to pump
#   if ack with uid not received in less than 10 seconds, send message again.
#   after ack: uid + 1

#Pump:
#   receive data
#   decrypt data
#   send ack to sensor with uid
#   send insuline
#   if uid is lower than or equal to last_uid, drop package
#   otherwise: send insuline and set last_uid to current uid

uid = 1

def sensor_send():
    last_five = []
    measurement = random.choice([1, 2, 3])
    message = str(uid) + ':' + str(measurement)
    b = message.encode('utf-8')
    message = message.hex()
```

Random ideas for future work:

- Use of ‘AEAD’ and ‘XOF’, not ‘MAC’ or ‘hash’
- Define standard serialisation, e.g. AD | n | C | t
- Appropriate parameter ordering for functions
- Creation of a compatible user-friendly wrapper
- Impact of programming paradigm on output

RC 9265 (#40713) 2/17/82
Computer Science 6 pages

Using the "Thinking-aloud" Method in Cognitive Interface Design

Clayton Lewis

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

Abstract: "Thinking-aloud" is a method for studying mental processes in which participants are asked to make spoken comments as they work on a task. The method is appropriate for studying the cognitive problems that people have in learning to use a computer system. This note discusses the strengths and weaknesses of the method, and gives some suggestions about its use based on laboratory experience at Yorktown.

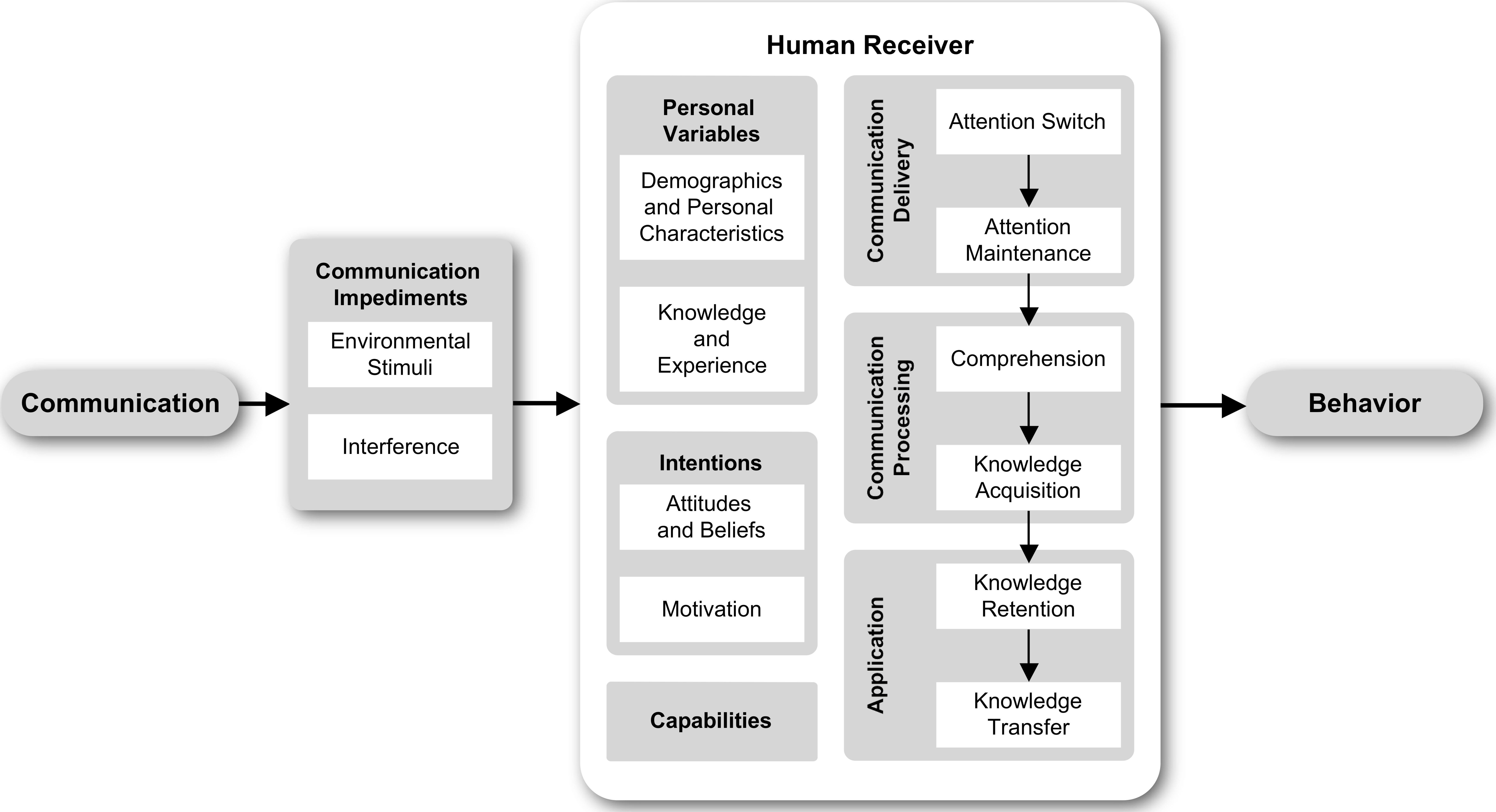
IDEO.org, 2015

SCA Evaluation & Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process

Jens-Peter Kaps
& Kris Gaj

CERG
Cryptographic Engineering Research Group
cryptography.gmu.edu

GEORGE MASON UNIVERSITY



Lorrie Cranor, 2008

How might we
integrate usability
into our process?

How would you
like your designs
to be evaluated?

Focus on understanding real-world failure cases

Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

Explore secure channels and record protocols

NIST Internal Report
NIST IR 8454

Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process

Meltem Sönmez Turan
Kerry McKay
Donghoon Chang
Lawrence E. Bassham

Software performance

- Ascon
- GIFT-COFB *
- SPARKLE
- Xoodyak
- TinyJAMBU

Hardware performance

- Ascon
- Xoodyak
- TinyJAMBU

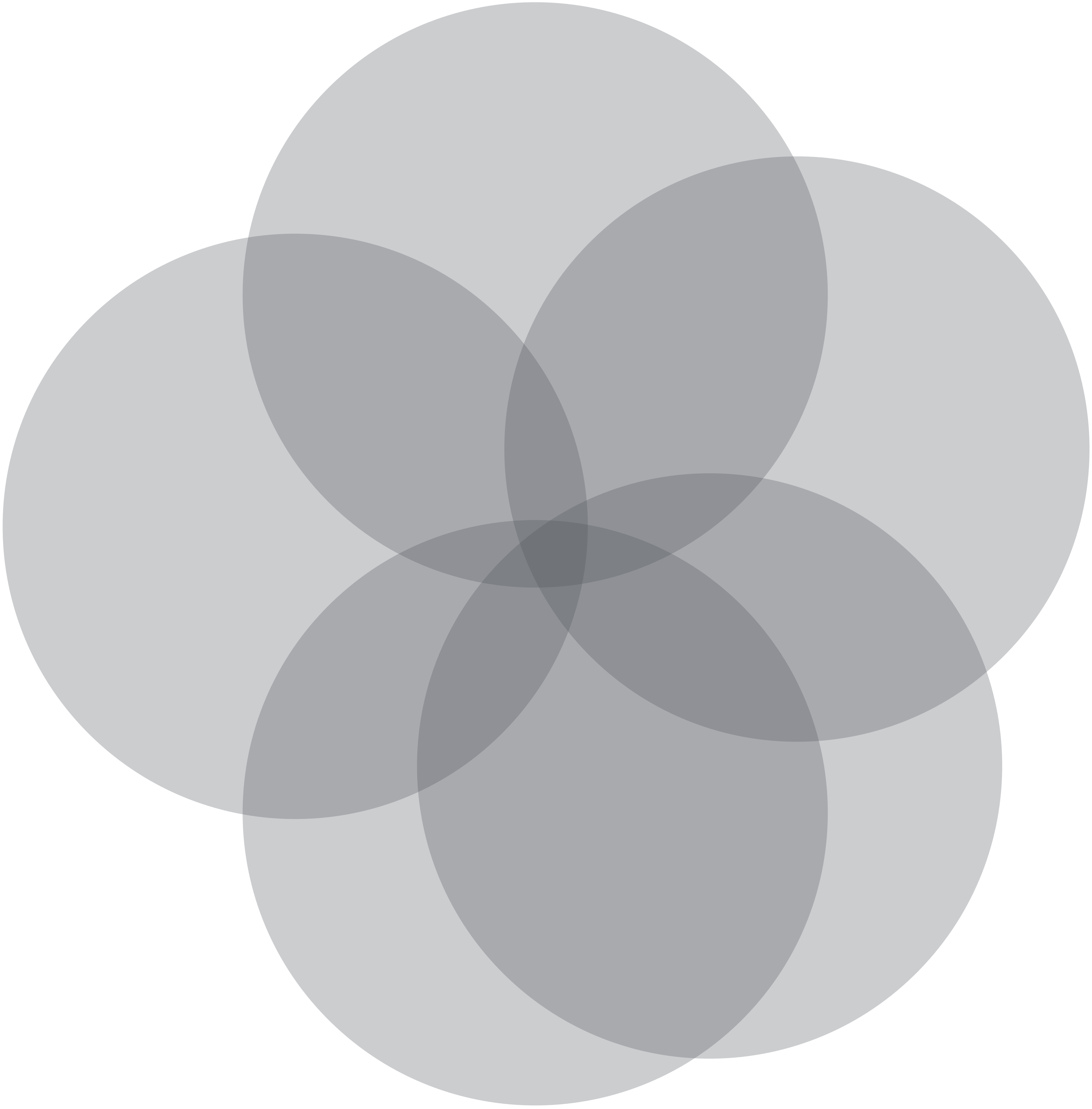
Protected performance

- Ascon
- ISAP
- Xoodyak
- TinyJAMBU

Modified after external analysis

- Grain-128AEAD
- TinyJAMBU

** might be vulnerable to a cache-timing attack*



Ascon

Xoodyak

There are some additional considerations, such as nonce-misuse security, releasing unverified plaintext security, the impact of state recovery, and post-quantum security of the candidates.

NISTIR 8454

Ascon

Xoodyak

Xoodyak is a permutation-based AEAD and hashing scheme. Xoodyak is built from a fixed 384-bit permutation (called Xoodoo) operated in Cyclist mode. The design approach of Xoodoo is closely related to that of the KECCAK permutation. The 384-bit state is arranged in a three-dimensional array of $3 \times 4 \times 32$ bits, nonlinearity is provided by simple operations on 3-bit columns, linear mixing is provided by mixing between sheets and moving the bits within the sheets around, and a constant addition ensures that there is some difference between rounds. Cyclist mode takes a fixed permutation and provides the functionality of both hashing (sponge mode) and AEAD (duplex mode) along with some new functionality, including tuple hashing, XOFs, and the generation of rolling subkeys.

Submission updates. In the final round, the key and nonce are processed together in a single call instead of separately in two calls, resulting in 12 fewer rounds needed to compute, which leads to fast processing of short messages.

Variants. The variants of the Xoodyak family are listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Rounds*
Xoodyak	128	128	128	192	192	12

Hash variants	*Digest size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Rounds*
Xoodyak	256	130	254	12
XOF variants				

submitters identify all known intellectual property that could be infringed by implementing their candidate algorithm. Among the finalists, applicable patents were only identified for PHOTON-Beetle [31]. After the review process was completed, intellectual property considerations did not factor into decisions made during the selection process.

2.2.1. Selection of ASCON

After evaluating the finalists according to the criteria presented above, NIST has selected the ASCON family for standardization.

The ASCON family includes AEAD and hash functions, as well as additional XOFs. This allows it to satisfy a wide range of application needs and there is low additional cost to implement additional functionalities thanks to its permutation-based design.

ASCON is the most mature of the finalists in terms of security. While some of the other finalists were not published prior to the lightweight standardization process, the AEAD variants of the ASCON family had already been presented and analyzed as part of the CAESAR competition.[2] Three profiles were created during the competition, including one for lightweight authenticated encryption. Ultimately, the AEAD variants of ASCON were selected as the primary choice for lightweight applications in the final CAESAR portfolio. ASCON's maturity can also be seen in the tweaks for the final round, where there were additional variants added but none of the second-round variants were modified. This is in contrast to some other finalists that included design tweaks to address attacks.

With ASCON's long history comes a wealth of analyses. It was the submission with the most third-party analysis and implementations. Despite the head-start on cryptanalytical attacks, ASCON has remained strong. AEAD variants of the ASCON family provide a high

5. Next Steps

In June 2023, NIST will host the Sixth Lightweight Cryptography Workshop to further explain the selection process and to discuss various aspects of standardization. Among the topics of interest are additional variants, functionalities, and parameter selection. There has been public interest in possible extensions to the scope of the lightweight cryptography project. In particular, the community has expressed interest in the development of MAC and deterministic random bit generator standards based on the ASCON permutation.

NIST will work with the ASCON designers to draft the new lightweight cryptography standard. There will be a public comment period of at least 45 days during which NIST will solicit public feedback on the draft and publish the comments that were received. NIST will address each of the comments by making minor edits to the document or noting issues raised for future consideration.

The final version of NIST's ASCON standard will be published shortly after public comments have been resolved. At this time, the validation tests and procedures will be developed, followed by inclusion in validation processes under the cryptographic algorithm validation program and cryptographic module validation program.


AES
CAST-256
CRYPTON
DEAL
DFC
E2
LOKI97
MAGENTA
MARS
RC6
RIJNDAEL
SAFER+
SERPENT
TWOFISH
AES Efficiency Testing
EFFICIENCY
ALGORITHM

NIST, 1999

Volume 126, Article No. 126024 (2021) https://doi.org/10.6028/jres.126.024
Journal of Research of the National Institute of Standards and Technology

Development of the Advanced Encryption Standard

Miles E. Smid

Formerly: Computer Security Division,
National Institute of Standards and Technology,
Gaithersburg, MD 20899, USA

mesmid@verizon.net

Strong cryptographic algorithms are essential for the protection of stored and transmitted data throughout the world. This publication discusses the development of Federal Information Processing Standards Publication (FIPS) 197, which specifies a cryptographic algorithm known as the Advanced Encryption Standard (AES). The AES was the result of a cooperative multiyear effort involving the U.S. government, industry, and the academic community. Several difficult problems that had to be resolved during the standard's development are discussed, and the eventual solutions are presented. The author writes from his viewpoint as former leader of the Security Technology Group and later as acting director of the Computer Security Division at the National Institute of Standards and Technology, where he was responsible for the AES development.

Key words: Advanced Encryption Standard (AES); consensus process; cryptography; Data Encryption Standard (DES); security requirements, SKIPJACK.

Accepted: June 18, 2021

Published: August 16, 2021; **Current Version:** August 23, 2021

This article was sponsored by James Foti, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology (NIST). The views expressed represent those of the author and not necessarily those of NIST.

https://doi.org/10.6028/jres.126.024

1. Introduction

In the late 1990s, the National Institute of Standards and Technology (NIST) was about to decide if it was going to specify a new cryptographic algorithm standard for the protection of U.S. government and commercial data. The current standard was showing signs of age and would not be up to the task of providing strong security much longer. NIST could step aside and let some other entity manage the development of new cryptographic standards, it could propose a short-term fix with a limited lifetime, or it could establish a procedure to develop a completely new algorithm. In January 1997, NIST decided to move forward with a proposal for developing an Advanced Encryption Standard (AES), which would be secure enough to last well into the next millennium. In December of 2001, after five years of effort, the finished standard was approved and published. The journey from initial concept to final standard was not straightforward. This paper covers the motivation for the development of the AES, the process that was followed, and the problems that were encountered and solved along the way. It documents a significant milestone in the history of NIST's computer security program, which will be celebrating its 50th anniversary in 2022.

1

How to cite this article:
Smid ME (2021) Development of the Advanced Encryption Standard.
J Res Natl Inst Stan 126:126024. https://doi.org/10.6028/jres.126.024

NISTIR 8319

Review of the Advanced Encryption Standard

Nicky Mouha

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8319

National Institute of Standards and Technology
U.S. Department of Commerce

Guidelines for Submissions of Modes of Operation

Submissions should specify a mode of operation for a symmetric (secret) key block cipher algorithm. At a minimum, the mode should support underlying block ciphers with key-block combinations of 128-128, 192-128, and 256-128 bits. However, the specification should be generic – i.e., written to handle other key-block combinations, if they can be supported. Example modes include, but are not limited to, techniques for performing encryption, message authentication, hashing, and random bit generation. It will be helpful to receive variations of Counter mode arising from alternative methods/guidelines for prescribing the generation of counters.

NIST requests that submissions of modes of operation include the following six items:

- cover sheet
- mode specification
- summary of properties
- test vectors
- performance estimates
- intellectual property statements/agreements/disclosures.

These items are discussed below.

Cover Sheet

The cover sheet shall contain the following information:

- name of submitted mode of operation;
- principal submitter's name, telephone, fax, organization, postal address, e-mail address;
- name(s) of auxiliary submitter(s);
- name of mode's inventor(s)/developer(s);
- name of owner, if any, of the mode (typically, the owner will be the same as the submitter).

Mode Specification

A complete written specification of the mode of operation should be provided, including all mathematical equations, tables, diagrams, and parameters that are needed to implement the mode. NIST encourages submitters to elaborate on the intended use(s) of the mode, the design rationale, the relevant properties, proofs (if any), the comparison with other modes, and the mode's overall advantages/disadvantages.

Summary of Properties

To assist NIST and the public to draw comparisons and contrasts between the various candidate modes, the submissions should include a table or outline that identifies the following characteristics:

4 JULY 2001

Dual Counter Mode

MIKE BOYLE

CHRIS SALTER

INTRODUCTION

For the past 18 months, the NSA has been developing a high-speed encryption mode for IP packets. The mode that we designed is identical in many aspects to Jutla's Integrity Aware Parallelizable Mode (IAPM). There is one important difference in our proposal. In the IP world, a large number of packets might arrive out of order. Integrity Aware Parallelizable Mode (IAPM) and the proposed variations incur a large overhead for out of order packets[JU 01]. Each packet requires at least the time to perform a full decryption to obtain an IV before decryption of the cipher can begin. This note describes our solution to this problem.

First, we describe the basic mode and its features. We then describe how to implement this mode for IPSec.

DUAL COUNTER MODE

Dual counter mode is a hybrid of ECB mode and counter mode. Let E represent encryption by a codebook of width W. Let P_1, P_2, ..., P_j be j blocks of plaintext and let C_1, C_2, ..., C_j be the corresponding ciphertext. Let f be a polynomial of degree W for a primitive linear feedback shift register. Also, let $\{x_i\}$ be the sequence of fills generated by this polynomial. The first fill, x_0, is a secret shared between the two peers. This initial fill is most easily derived from the key exchange[1]. Dual counter mode can be described as follows:

j = # of datablocks

For i = 1, ..., j

$x_i = f(x_{i-1})$

$C_i = E(P_i \oplus x_i) \oplus x_i$

Quite likely the cipherblocks will travel in packets. If the packets arrive in order, the receiver does not lose track of the fill needed to decrypt the cipher.

TWO IMPLEMENTATION MODES

We knew that many implementers would want to verify the data integrity of packets. This mode has the property that any change to a ciphertext block causes the decrypted plaintext to be garbled. Thus it is easy to add a checksum to verify data integrity.

[1] Of course, care should be taken in producing this value. For example, the designers of the key exchange for IPsec used secure hashes such as SHA-1 to isolate keying material.

A Note on NSA's Dual Counter Mode of Encryption

Pompiliu Donescu * pompiliu@eng.umd.edu
Virgil D. Gligor ** gligor@eng.umd.edu
David Wagner * * * daw@cs.berkeley.edu

September 28, 2001

Abstract. We show that both variants of the Dual Counter Mode of encryption (DCM) submitted for consideration as an AES mode of operation to NIST by M. Boyle and C. Salter of the NSA are insecure with respect to both secrecy and integrity in the face of chosen-plaintext attacks. We argue that DCM cannot be easily changed to satisfy its stated performance goal and be secure. Hence repairing DCM does not appear worthwhile.

1 Introduction

On August 1, 2001, M. Boyle and C. Salter of the NSA submitted two variants of the Dual Counter Mode (DCM) of encryption [1] for consideration as an AES mode of operation to NIST. The DCM goals are: (1) to protect both the secrecy and integrity of IP packets (as this mode is intended to satisfy the security goals of Jutla's IAPM mode [4]), and (2) to avoid the delay required before commencing the decryption of out-of-order IP packets, thereby decreasing the decryption latency of IAPM. DCM is also intended to allow high rates of encryption.

The authors argue that DCM satisfies the first goal because "an error in a cipher block causes all data in the packet to fail the integrity check". DCM appears to satisfy the second goal because it maintains a "shared secret negotiated during the key exchange," which avoids the delay inherent to the decryption of a secret IV before the first out-of-order packet arrival can be decrypted. The authors note correctly that Jutla's IAPM mode does not satisfy their second goal.

In this note, we show that both variants of DCM are insecure with respect to both secrecy and integrity in the face of chosen-plaintext attacks. Further, we argue that DCM cannot be easily changed to satisfy its stated performance goal for the decryption of out-of-order packets *and* be secure. We conclude since other proposed AES modes satisfy the proposed goals for DCM, even if repairing DCM is possible, which we doubt, such an exercise does not appear to be worthwhile.

1 VDG Inc., 6009 Brookside Drive, Chevy Chase, MD 20815.
2 Electrical and Computer Engineering Department, University of Maryland, College Park, Maryland 20742.
3 Computer Science Division, EECS Department, University of California Berkeley, Berkeley, CA. 94720.

Cryptanalysis of OCB2

Akiko Inoue and Kazuhiko Minematsu

NEC Corporation, Japan
a-inoue@cj.jp.nec.com, k-minematsu@ah.jp.nec.com

Abstract. We present practical attacks against OCB2, an ISO-standard authenticated encryption (AE) scheme. OCB2 is a highly-efficient blockcipher mode of operation. It has been extensively studied and widely believed to be secure thanks to the provable security proofs. Our attacks allow the adversary to create forgeries with single encryption query of almost-known plaintext. This attack can be further extended to powerful almost-universal and universal forgeries using more queries. The source of our attacks is the way OCB2 implements AE using a tweakable blockcipher, called XEX*. We have verified our attacks using a reference code of OCB2. Our attacks do not break the privacy of OCB2, and are not applicable to the others, including OCB1 and OCB3.

Keywords: OCB, Authenticated Encryption, Cryptanalysis, Forgery, XEX

1 Introduction

Authenticated encryption (AE) is a form of symmetric-key encryption that provides both confidentiality and authenticity of messages. Now it is widely accepted

Damien Newman, 2002

Research

Assurance

Cryptographic competitions

Daniel J. Bernstein[1,2]

[1] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607–7045, USA
[2] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
djb@cr.yp.to

Abstract. Competitions are widely viewed as the safest way to select cryptographic algorithms. This paper surveys procedures that have been used in cryptographic competitions, and analyzes the extent to which those procedures reduce security risks.

Keywords: cryptography, competitions, DES, AES, eSTREAM, SHA-3, CAESAR, NISTPQC, NISTLWC

1 Introduction

> *The CoV individual reports point out several shortcomings and procedural weaknesses that led to the inclusion of the Dual EC DRBG algorithm in SP 800-90 and propose several steps to remedy them. . . . The VCAT strongly encourages standard development through open*

github.com/arnepadmos/hackathon

Making and breaking IoT protocol development and evaluation processes

An exploration into the influence of competition structure on the assurance provided by an adversarial process. Participants will propose competition blueprints and identify problems related to incentives (mis)alignment, exploring how to shape the dynamics of adversarial engineering design competitions such that they incentivise and align security engineering and assurance efforts in the IoT domain.

QR

Project description

The precise project description, including the format and detailed content, will be determined based on input gathered before the start of the hackathon to ensure that it aligns with the interest of potential participants. Key aspects of the hackathon structure that have already been

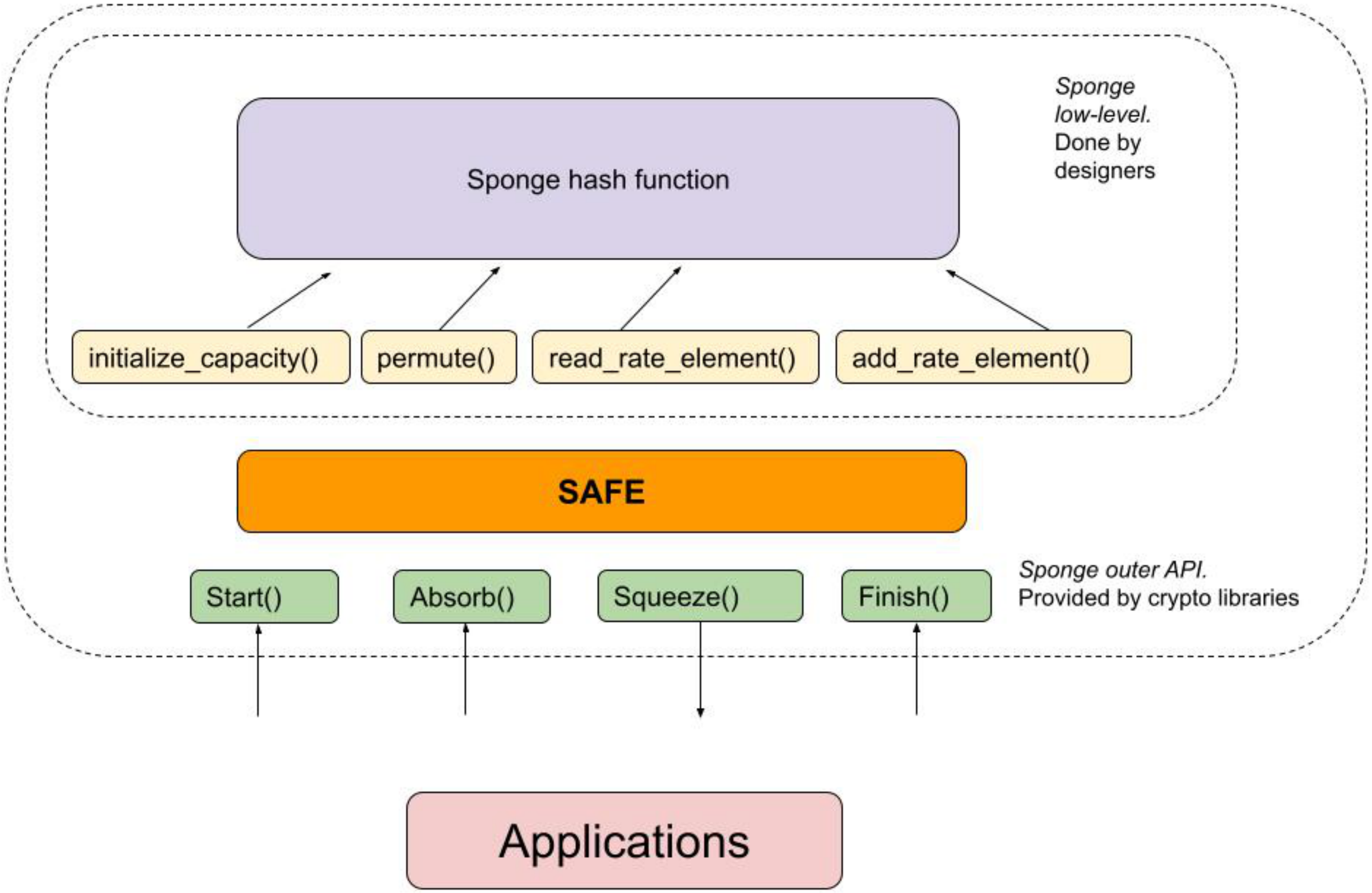
Focus on understanding real-world failure cases

Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

Explore secure channels and record protocols

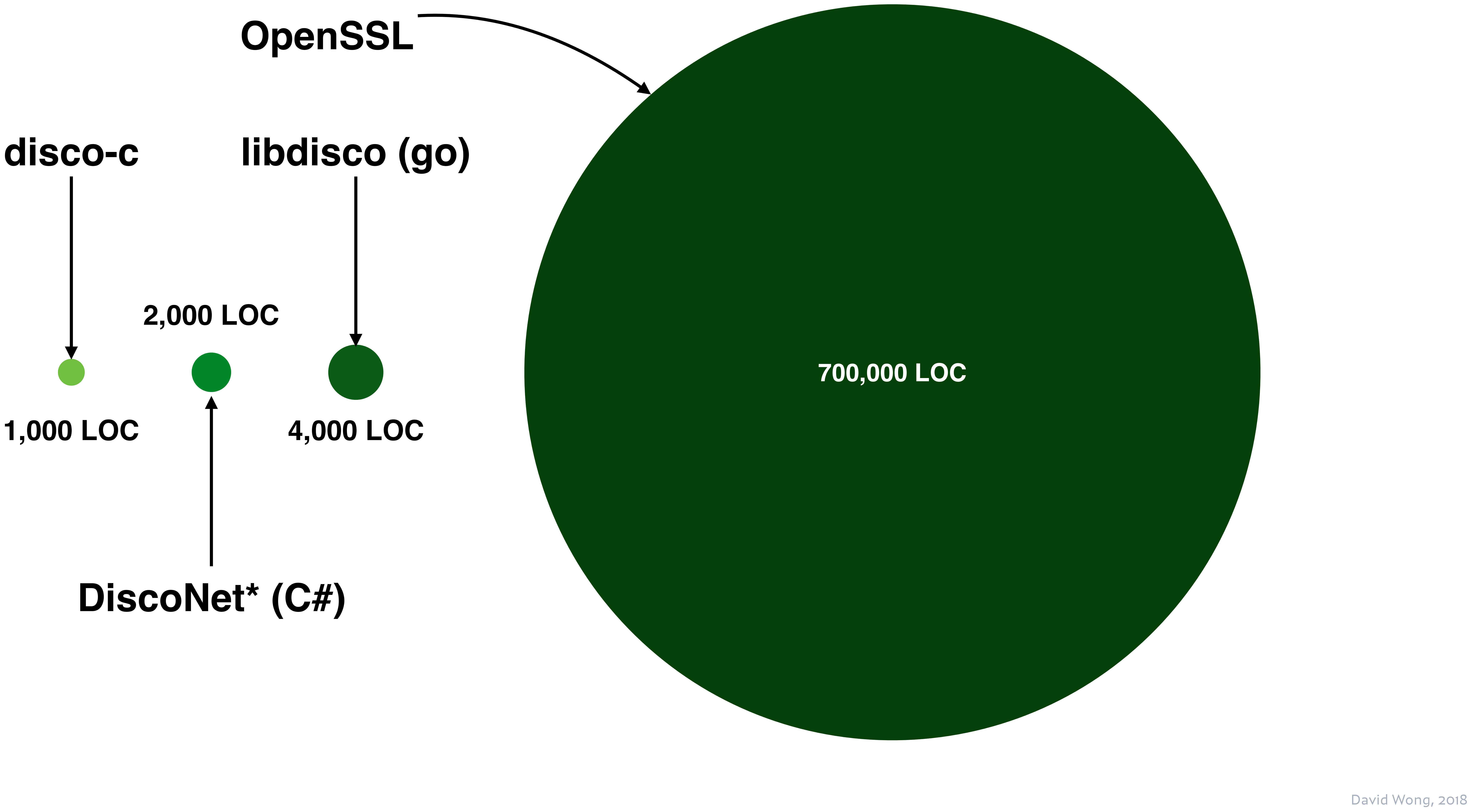
As illustrated by BLINKER, Strobe, SHOE, and Cyclist, sponges can be the basis for simple, lightweight two-party half-duplex record protocols.



Aumasson et al., 2023

The complete symmetric crypto API

clone()	absorb()	squeeze()	squeeze_tag()
squeeze_key()	max_tag_len()	encrypt()	encrypt_detached()
decrypt()	decrypt_detached()	ratchet()	tag_len()
tag_pull()	tag_verify()	close()	reset()


OpenSSL
disco-c
libdisco (go)
2,000 LOC
700,000 LOC
1,000 LOC
4,000 LOC
DiscoNet* (C#)
David Wong, 2018

Motivation for BLINKER

Legacy protocols are unsuited for ultra-lightweight applications.

Academic research has focused on lightweight **primitives**, and suitable lightweight, **general purpose communications protocols** have not been proposed.

We need a generic **short-distance lightweight link layer** security provider that can function independently from upper layer application functions.

- Design with mathematical and legal **provability** in mind.
- Aim at simplicity and small footprint: use a **single** sponge permutation for key derivation, confidentiality, integrity, etc. (Instead of distinct algorithms.)
- Use a **single state variable** in both directions, instead of 8+ cryptovariables.
- Ideally this protocol would be realizable with semi-autonomous integrated hardware, without much CPU or MCU involvement.

Security Goals

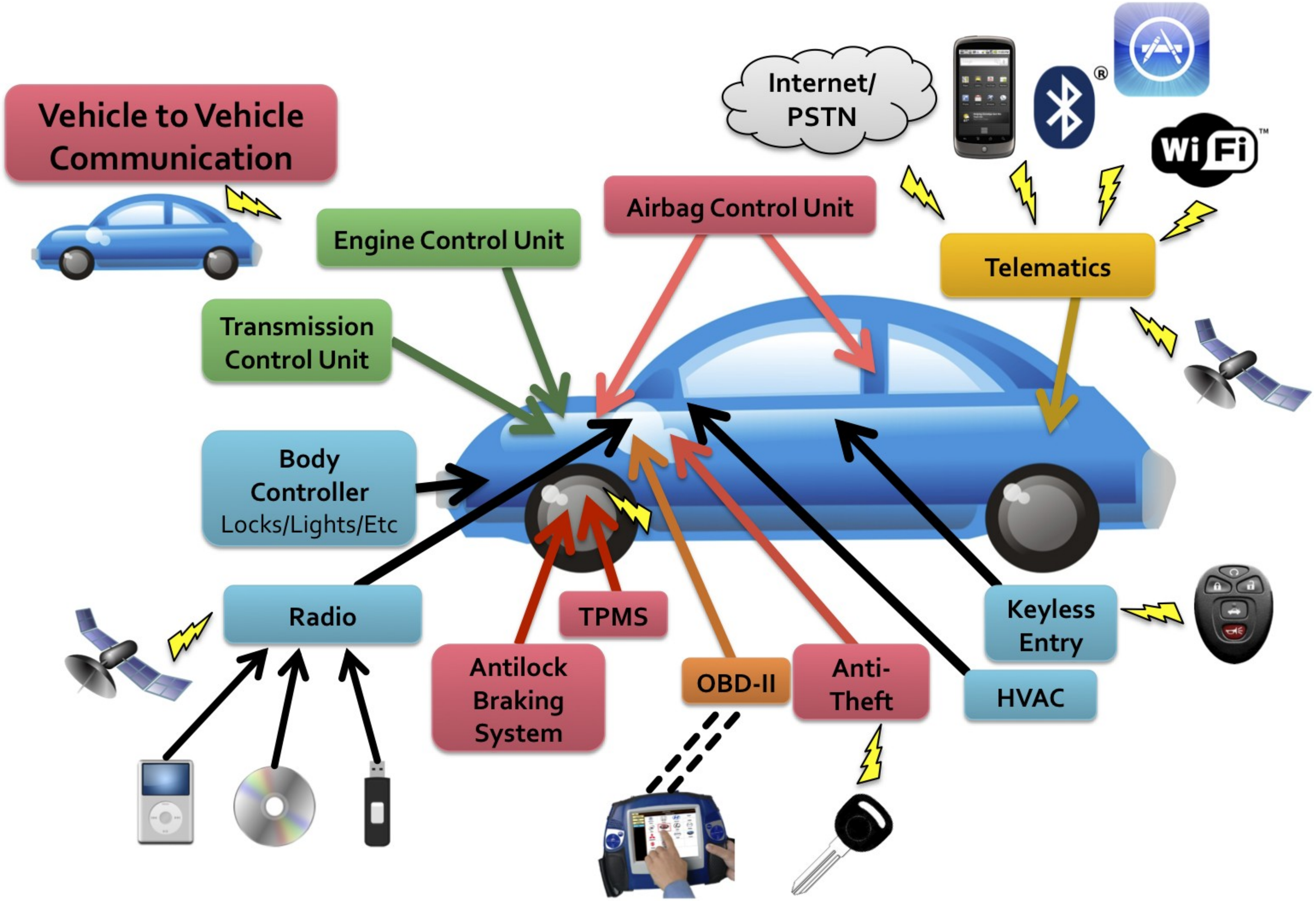
Protocol designers should have provable bounds on these three goals:

priv The ciphertext result C of enc(S, P, pad) must be indistinguishable from random when S is random and P may be chosen by the attacker.

auth The probability of an adversary of choosing a message C that does not result in a FAIL in dec(S, C, pad) without knowledge of S is bound by a function of the authentication tag size t and number of trials.

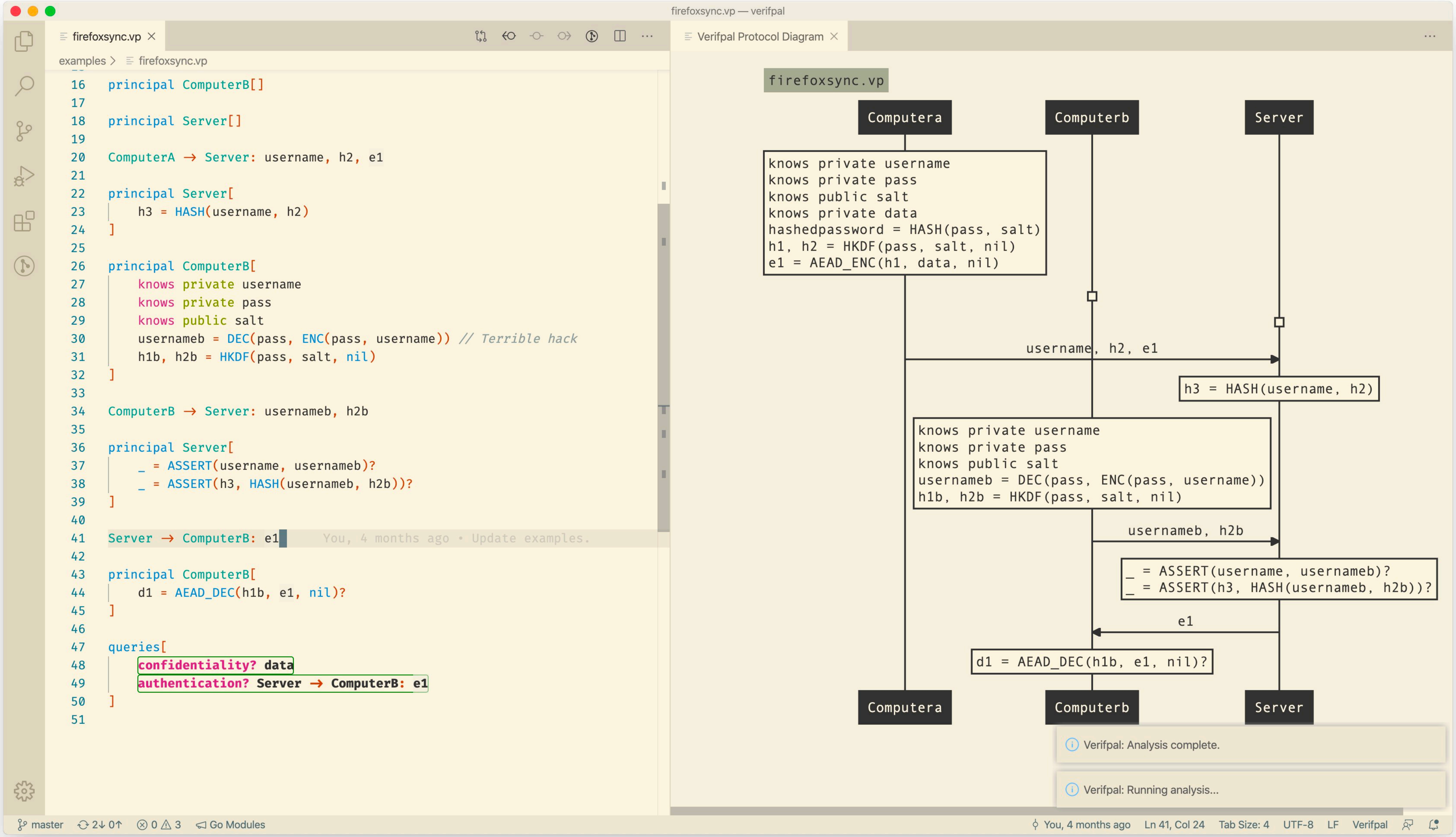
sync Each party can verify that all previous messages of the session have been correctly received and the absolute order in which messages were sent.

First two are standard Authentication Encryption requirements, the **last one is new**.



Checkoway et al., 2011

Nadim Kobeissi, 2019



Nadim Kobeissi, 2020

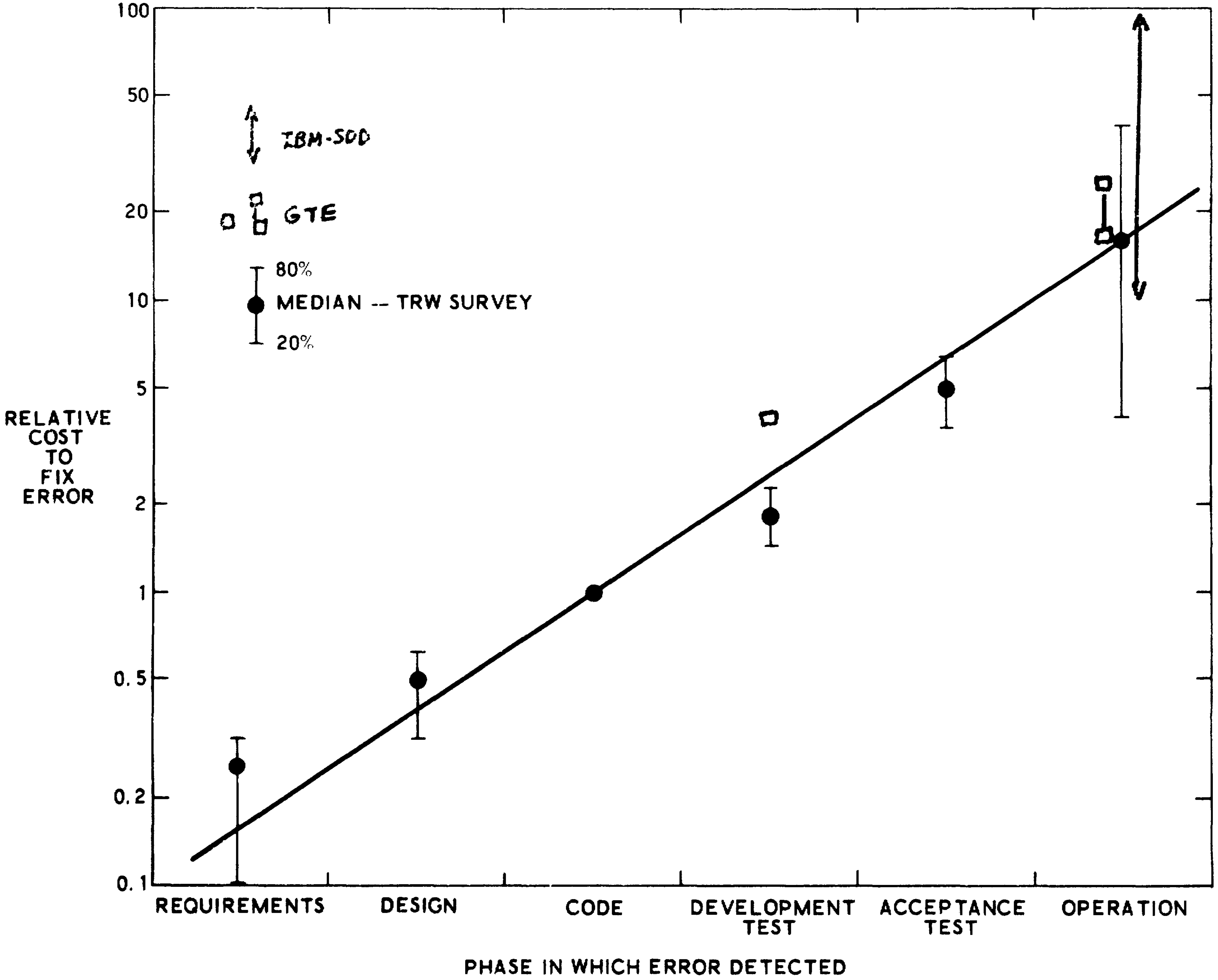
Examples of protocol-related problems:

— Field, content, channel, domain, key confusion
— Key extraction allows decrypting prior traffic
— Drop, delay, preplay, reflect, reorder, replay, etc.
— Recovery of a wrapped key due to nonce reuse
— Not using safer features because of higher cost

Things that could use further study:

— Tuples and types, plus making these efficient
— Reuse of internal state for KDF and/or chaining
— Relevance of Ascon modes to [Turbo]SHAKE
— Support for sessions, including key ratcheting
— Multi-key attacks and key-reuse resistance

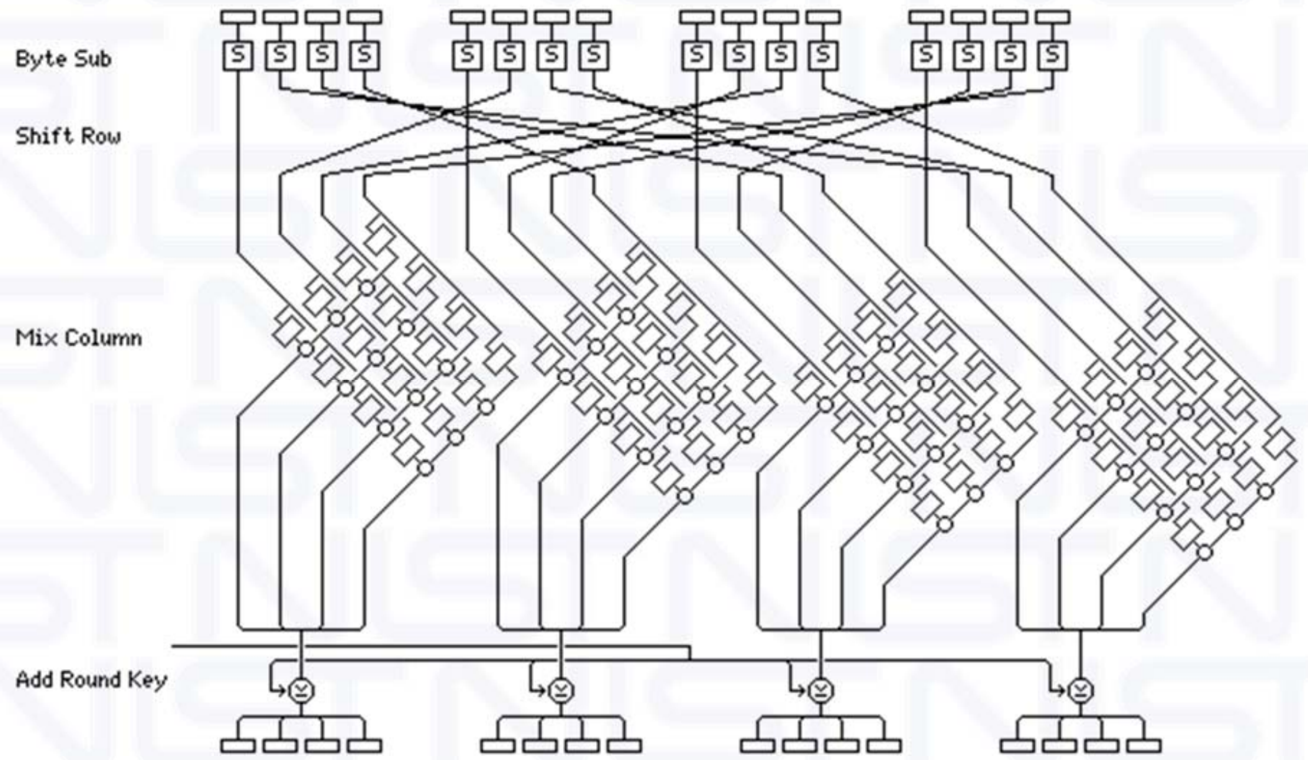

Joi Ito, 2007



Barry Boehm, 1976

NIST GCR 18-017

The Economic Impacts of the Advanced Encryption Standard, 1996 - 2017



David P. Leech

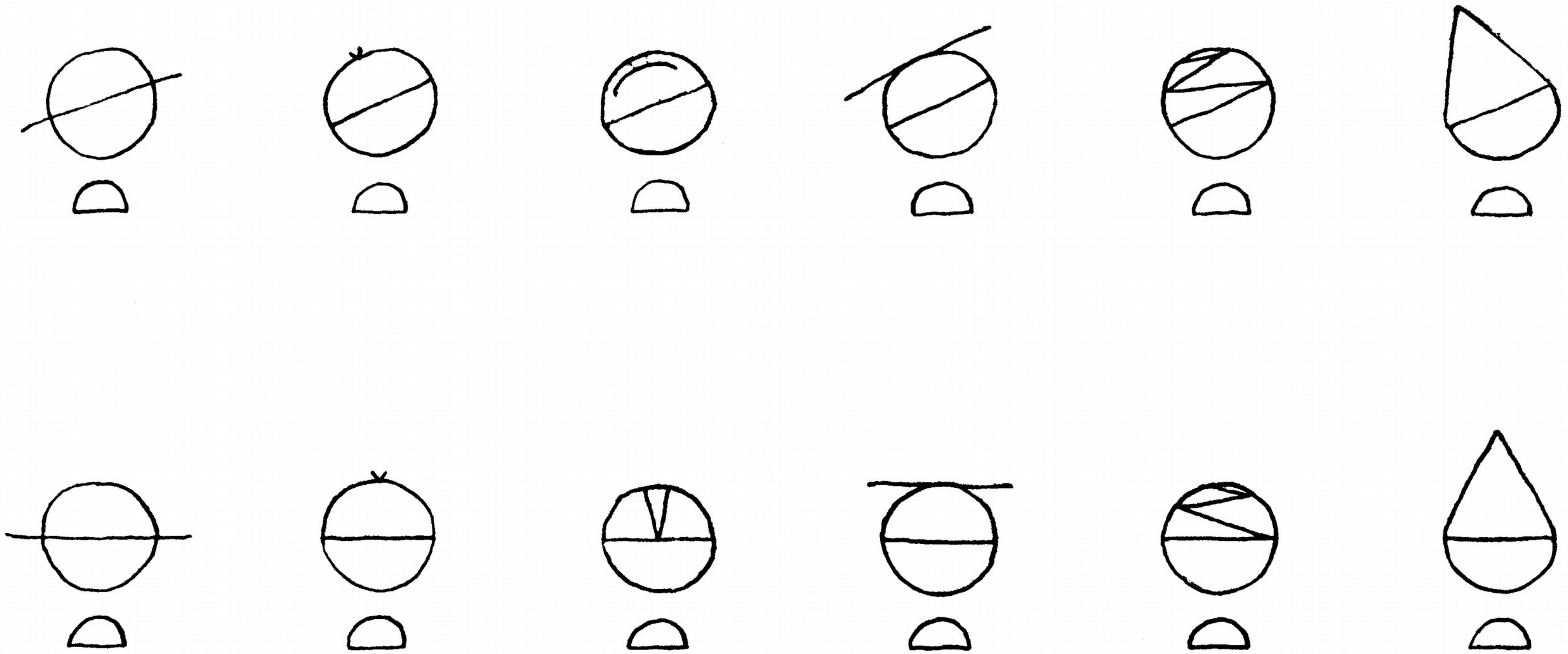
Stacey Ferris, CPA

John T. Scott, Ph.D.

September 2018

National Institute of Standards and Technology
U.S. Department of Commerce

This publication is available free of charge from: https://doi.org/10.6028/NIST.GCR.18-017

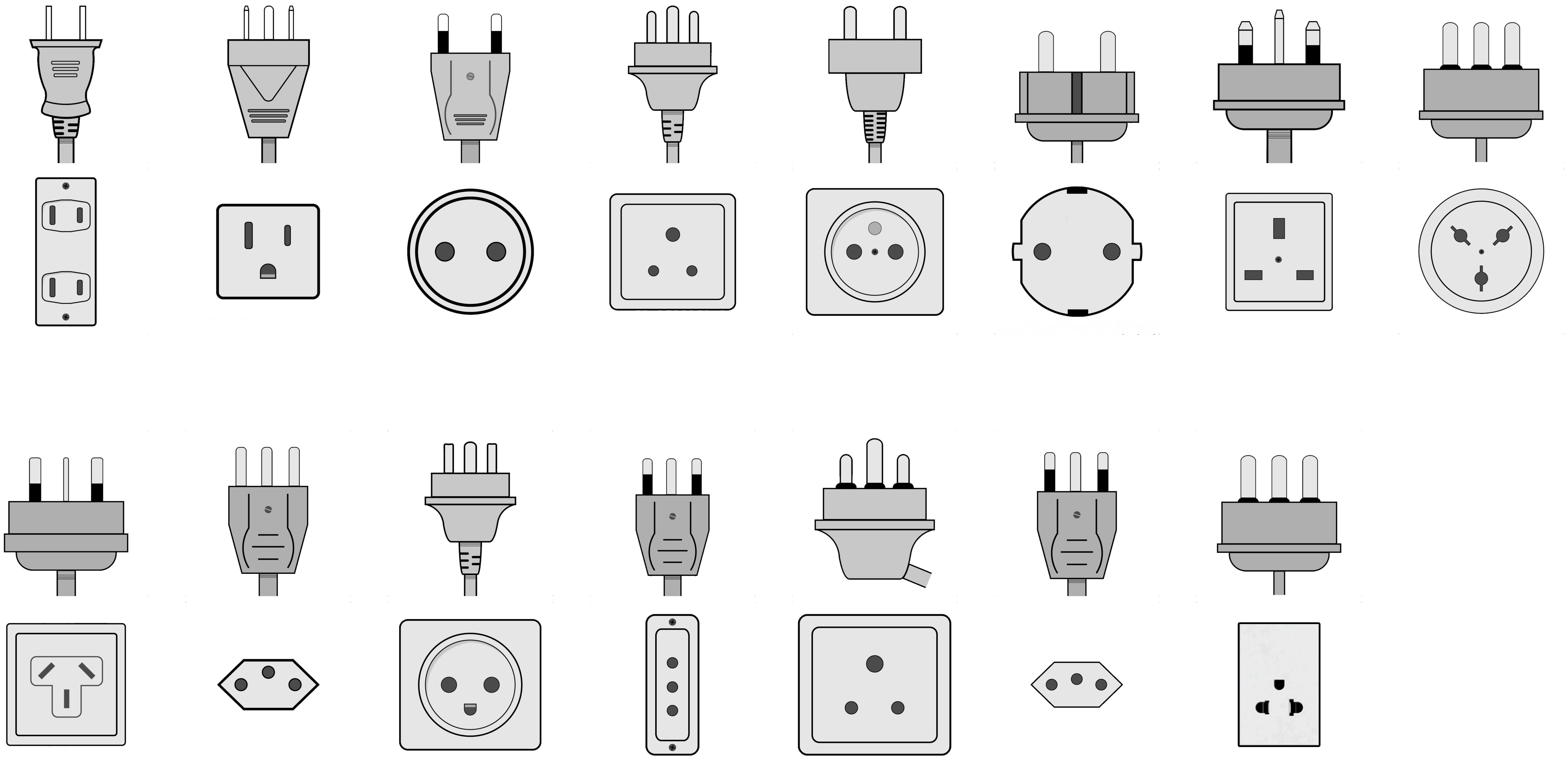


Focus on understanding real-world failure cases

Test the toolbox with end-users for footguns

Provide structures and incentives for assurance

Explore secure channels and record protocols



IEC, 2021

Linux Nordwalde, 2007

hello@arnepadmos.com