

An Introduction to Formal Methods for Cyber Security

Achim D. Brucker

a.brucker@exeter.ac.uk

<https://www.brucker.ch/>

Software Assurance & Security Research

Department of Computer Science, University of Exeter, Exeter, UK

<https://logicalhacking.com/>

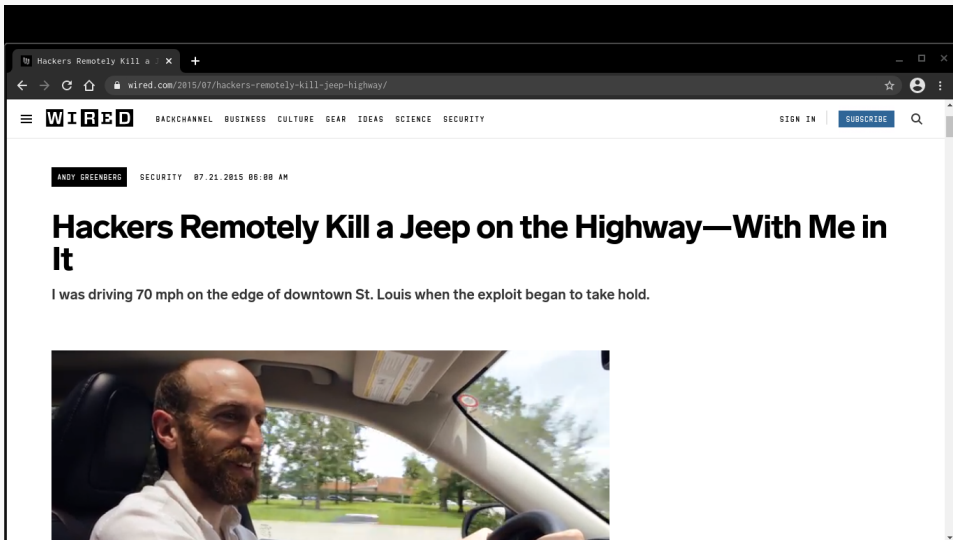
This is joint work with **Andreas Hess** (DTU, Denmark), **Sebastian Mödersheim** (DTU, Denmark),
and **Anders Schlichtkrull** (Aalborg University Copenhagen, Denmark).

September 9, 2023

Motivation







Where are security protocols used?

Securing your live

Hacking risk leads to recall

theguardian.com/technology/2017/aug/31/hacking-risk-recall-pacemakers-patient-death-fears-fda-firmware-update

News Opinion Sport Culture Lifestyle More

UK World Climate crisis Newsletters Football Coronavirus Business Environment UK politics Education Society Science Tech

Hacking

Hacking risk leads to recall of 500,000 pacemakers due to patient death fears

FDA overseeing crucial firmware update in US to patch security holes and prevent hijacking of pacemakers implanted in half a million people

Alex Hern
@alexhern
Thu 31 Aug 2017 13.23 BST

f t e



Abbott / St Jude Medical's Accent MRI pacemaker, one of the affected devices that had to be recalled. Photograph: Abbott / St Jude Medical


Where are security protocols used?

Secure communication on the internet

The screenshot shows a web browser window with the address bar displaying 'vle.exeter.ac.uk'. The page header features the University of Exeter logo and name. Below the header, there is a dark navigation bar with 'ELE' on the left and a menu icon on the right. A light blue banner contains text about UCU industrial action. To the right, there is a 'ELE Login' section with a 'Login' button. Below the banner, there are three images: a green abstract shape, a brick wall, and a group of people looking at a laptop.


Exeter Learning Environme x +


← → ↻ 🏠 🔒 vle.exeter.ac.uk 🔗 ☆ 🗄 👤 ⋮


 University of Exeter

ELE ☰

For information about the UCU industrial action, including FAQs on what to expect on a strike day, how to access teaching materials for cancelled sessions, and accessing support please [visit our website](#) ✕

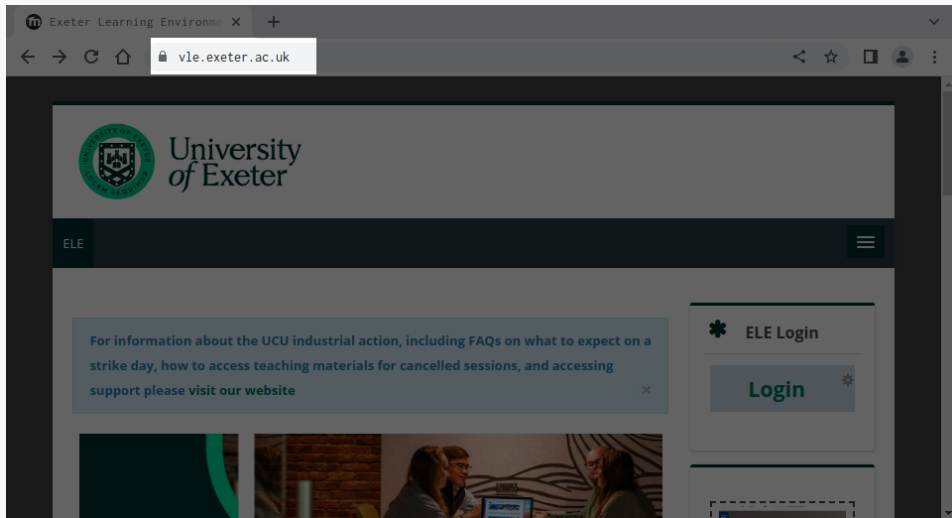
 ELE Login

 Login ⚙



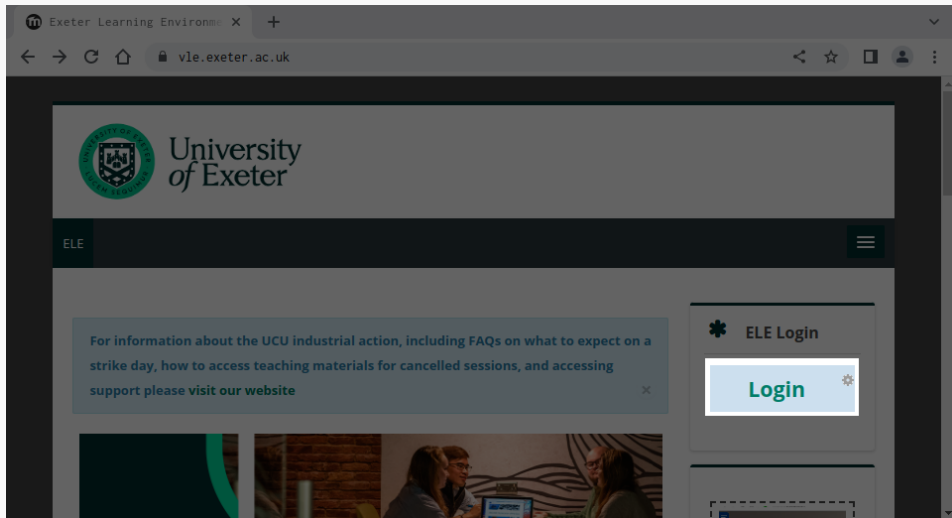
Where are security protocols used?

Secure communication on the internet



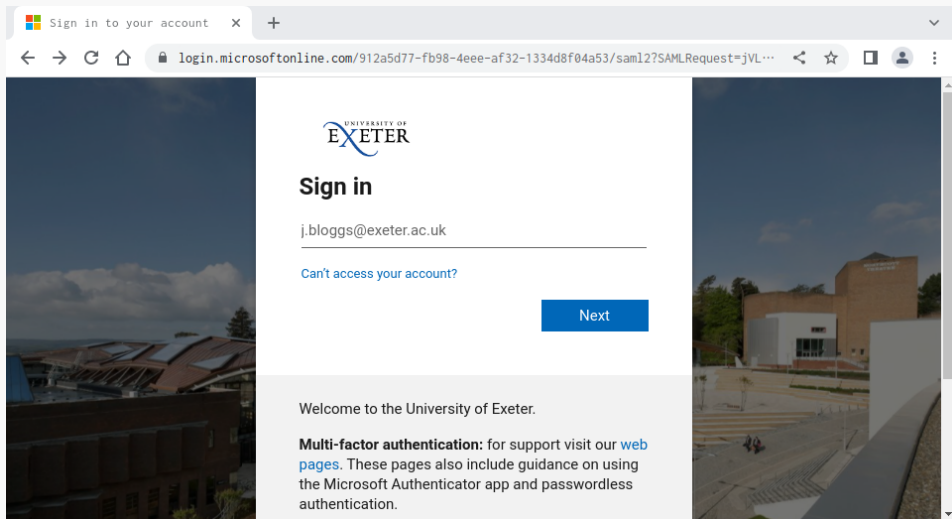
Where are security protocols used?

Secure communication on the internet



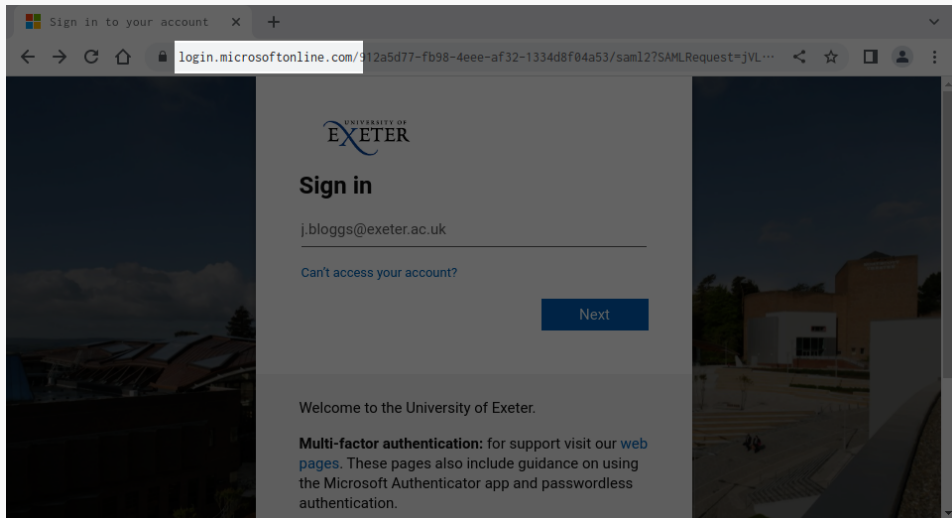
Where are security protocols used?

Secure communication on the internet



Where are security protocols used?

Secure communication on the internet



Where are security protocols used?

Secure communication on the internet

The screenshot shows a web browser window with the address bar displaying 'vle.exeter.ac.uk'. The page header features the University of Exeter logo and the name 'Achim D. Brucker' next to a profile picture. Below the header, there is a dark navigation bar with 'ELE' on the left and a menu icon on the right. The main content area includes a large image of three students studying at a table. To the right of this image, there is a section titled 'Exam & Revision Location Information' with a yellow background, stating: 'We are operating a separate version of ELE during the examinations period,'.



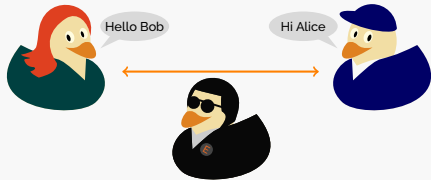
Definition (Protocol)

A **protocol** consists of a set of rules (conventions) that determine the exchange of messages between two or more principals.

In short, a **distributed algorithm** with emphasis on communication.

Definition (Security Protocol)

Security (or cryptographic) protocols use cryptographic mechanisms to achieve security objectives, e.g., entity or message authentication, confidentiality, key establishment, integrity, timeliness, fair exchange, non-repudiation.



Definition (Protocol)

A **protocol** consists of a set of rules (conventions) that determine the exchange of messages between two or more principals.

In short, a **distributed algorithm** with emphasis on communication.

Definition (Security Protocol)

Security (or cryptographic) protocols use cryptographic mechanisms to achieve security objectives, e.g., entity or message authentication, confidentiality, key establishment, integrity, timeliness, fair exchange, non-repudiation.

Motivation

The Needham-Schroeder Protocol for Public Keys (NSPK)

Problem: Alice wants to be sure that she talks to Bob (authenticity) and vice versa

Technologies available to us:

- ❑ Generation of secure (unpredictable!) random numbers.
- ❑ Perfect (unbreakable!) public-key cryptography.

Note:

- ❑ In public-key cryptography, each participant has a key pair:
 - ❑ A **public** key (e.g., K_B is the public key of Bob) used for encrypting messages.
 - ❑ A **private** key used for decrypting messages.

Problem: Alice wants to be sure that she talks to Bob (authenticity) and vice versa

Technologies available to us:

- ❑ Generation of secure (unpredictable!) random numbers.
- ❑ Perfect (unbreakable!) public-key cryptography.

Solution: Needham and Schroeder (1978) proposed the following protocol (NSPK):

Note:

- ❑ In public-key cryptography, each participant has a key pair:
 - ❑ A **public** key (e.g., K_B is the public key of Bob) used for encrypting messages.
 - ❑ A **private** key used for decrypting messages.

Problem: Alice wants to be sure that she talks to Bob (authenticity) and vice versa

Technologies available to us:

- ❑ Generation of secure (unpredictable!) random numbers.
- ❑ Perfect (unbreakable!) public-key cryptography.

Solution: Needham and Schroeder (1978) proposed the following protocol (NSPK):



Note:

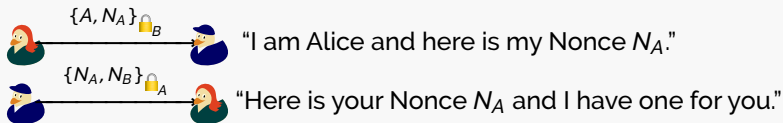
- ❑ In public-key cryptography, each participant has a key pair:
 - ❑ A **public** key (e.g., lock_B is the public key of Bob) used for encrypting messages.
 - ❑ A **private** key used for decrypting messages.

Problem: Alice wants to be sure that she talks to Bob (authenticity) and vice versa

Technologies available to us:

- ❑ Generation of secure (unpredictable!) random numbers.
- ❑ Perfect (unbreakable!) public-key cryptography.

Solution: Needham and Schroeder (1978) proposed the following protocol (NSPK):



Note:

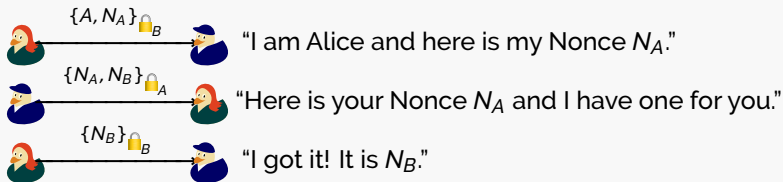
- ❑ In public-key cryptography, each participant has a key pair:
 - ❑ A **public** key (e.g., K_B is the public key of Bob) used for encrypting messages.
 - ❑ A **private** key used for decrypting messages.
- ❑ A Nonce (short for: *number used once*) is a fresh secret only known to the person generating it.

Problem: Alice wants to be sure that she talks to Bob (authenticity) and vice versa

Technologies available to us:

- ❑ Generation of secure (unpredictable!) random numbers.
- ❑ Perfect (unbreakable!) public-key cryptography.

Solution: Needham and Schroeder (1978) proposed the following protocol (NSPK):



Note:

- ❑ In public-key cryptography, each participant has a key pair:
 - ❑ A **public** key (e.g., lock_B is the public key of Bob) used for encrypting messages.
 - ❑ A **private** key used for decrypting messages.
- ❑ A Nonce (short for: *number used once*) is a fresh secret only known to the person generating it.

Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

Correctness (informal):

Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

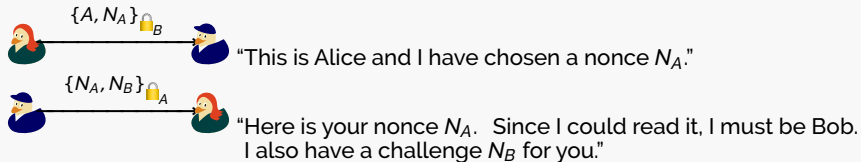
Correctness (informal):



Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

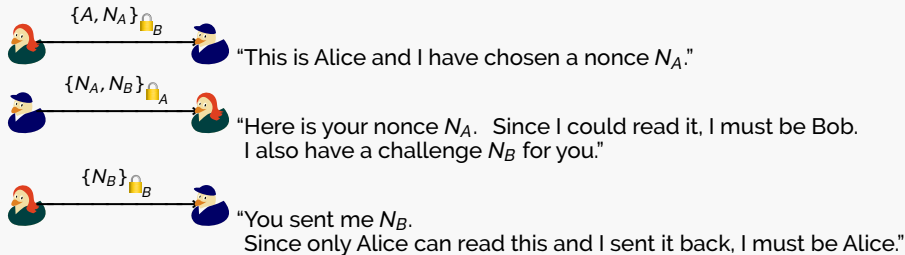
Correctness (informal):



Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

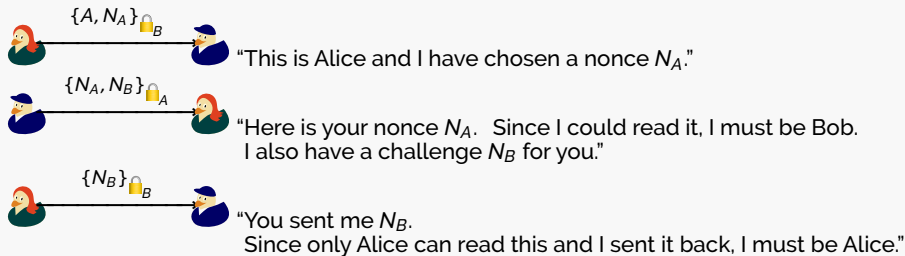
Correctness (informal):



Goal: After executing the protocol successfully,
Alice and Bob can be sure to talk to each other (and not to somebody else).

Note: This goal is called Mutual Authenticity (Two-way Authentication).

Correctness (informal):



Let's look at another scenario using NSPK ...

A scenario with three participants



Alice



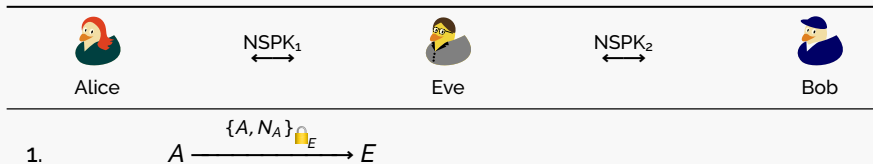
Eve



Bob

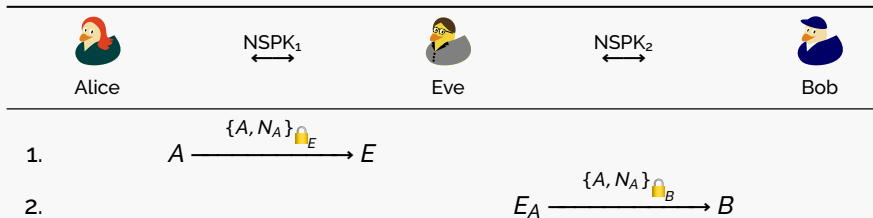
Two Concurrent Runs of NSPK

Protocols are typically *small* and *convincing*



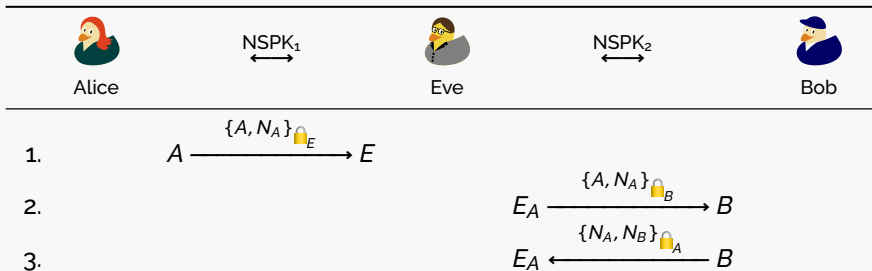
Two Concurrent Runs of NSPK

Protocols are typically *small* and *convincing*



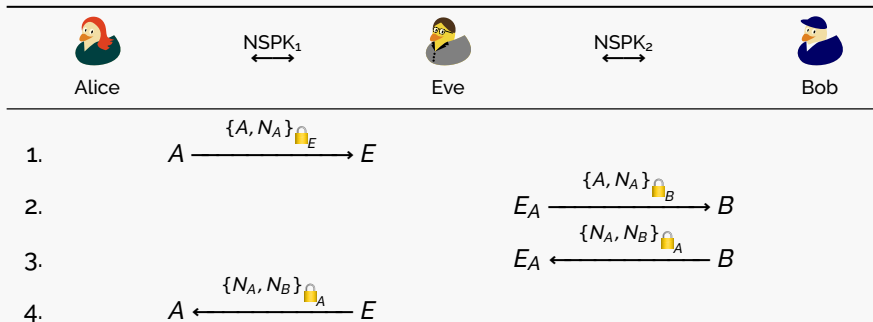
Two Concurrent Runs of NSPK

Protocols are typically *small* and *convincing*



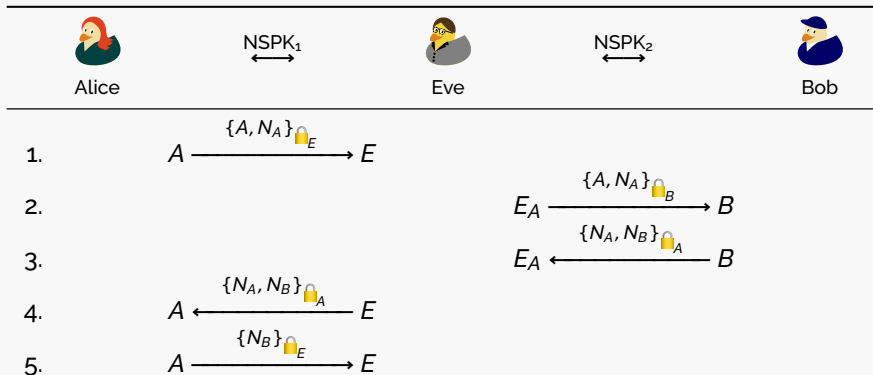
Two Concurrent Runs of NSPK

Protocols are typically *small* and *convincing*



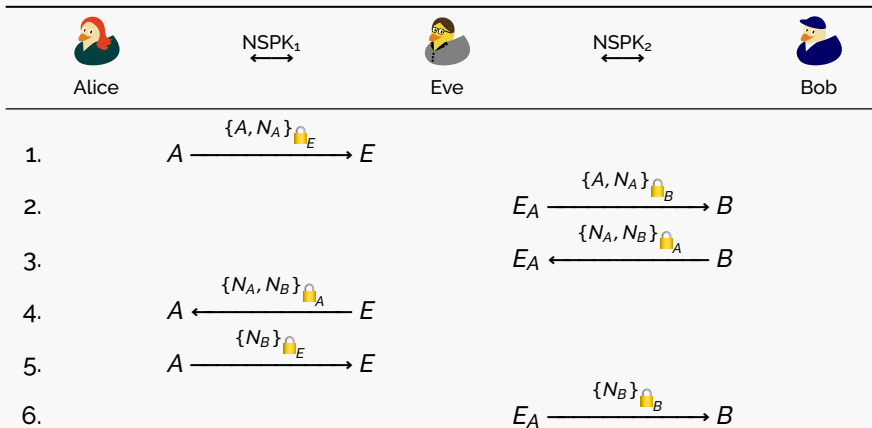
Two Concurrent Runs of NSPK

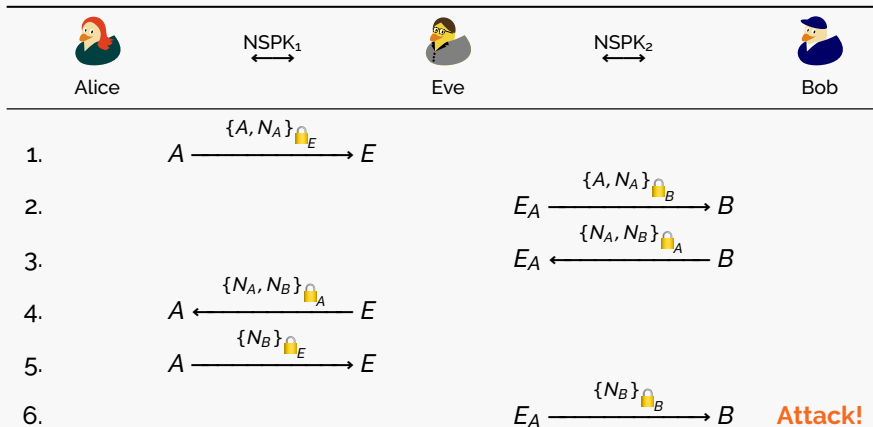
Protocols are typically *small* and *convincing*



Two Concurrent Runs of NSPK

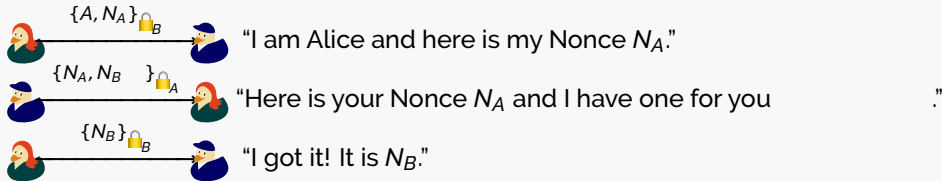
Protocols are typically *small* and *convincing*



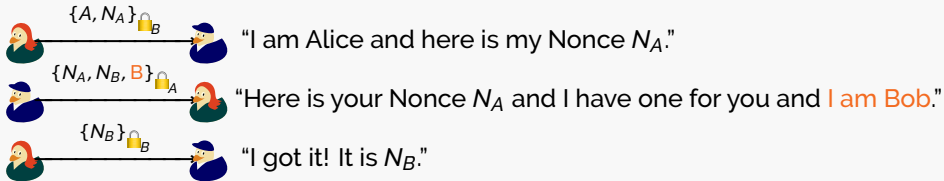


Bob **believes** he is speaking with Alice (but talks to Eve)!

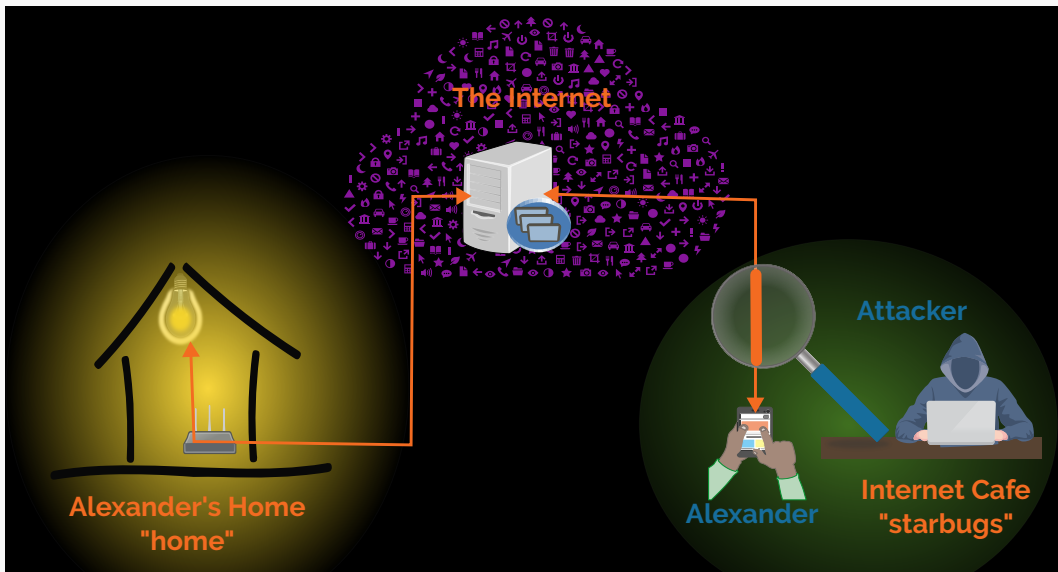
Needham-Schroeder



Needham-Schroeder with Lowe's fix:







Just because some systems uses cryptography (blockchain, 2-factor authentication, ...)¹
it does not mean, the system is secure!

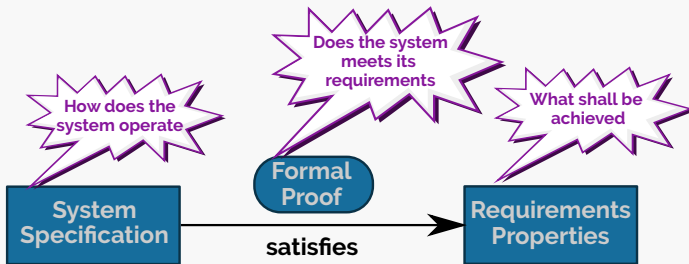


¹Insert your favorite security technology here.

Formal (Security) Verification

Motivation

It took 18 years to find the flaw in the Needham-Schroeder protocol.
Can we do better?



Goal: Formally model protocols and their properties and provide a mathematically sound means for reasoning about these models.

Basis: Suitable abstraction of protocols.

Analysis: Using formal methods based on mathematics and logic.

Formal (Security) Verification

Our Attacker Model

❖ Consider a public key system in which for every user X :

- ❖ There is a public encryption function E_X
 - ❖ every user can apply this function.
- ❖ There is a private decryption function D_X
 - ❖ only X can apply this function.

These functions have the property that

$$E_X D_X = D_X E_X = 1.$$

❖ The Dolev-Yao intruder:

- ❖ Controls the network (read, intercept, send).
- ❖ Is also a user, called Z .
- ❖ Can apply E_X for any X .
- ❖ Can apply D_Z .

158

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 30, NO. 2, MARCH 1983

Dolev, Dolev, and Yao, "On the Security of Public Key Protocols," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[1] G. O. Karac and J. K. K. "Reliability function of a discrete memoryless channel," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[2] G. O. Karac and J. K. K. "The security of discrete memoryless channels: A continuous approach," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[3] R. E. Blahut, "Information theory: A modern introduction," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[4] A. S. S. "An information-theoretic approach to the security of public key systems," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[5] J. C. C. "A security analysis of the discrete logarithm problem," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[6] G. O. Karac and J. K. K. "The security of discrete memoryless channels: A continuous approach," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

[7] R. E. Blahut, "Information theory: A modern introduction," *IEEE Trans. Inf. Theory*, vol. 30, no. 2, pp. 158-167, 1983.

On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure communications has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active intruder, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. INTRODUCTION

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [9], an improperly designed protocol could be vulnerable to an "active" intruder, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security

issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the intruder.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an encryption function E_X and a decryption function D_X , both are mappings from $\{0, 1\}^*$ (the set of all finite binary sequences) into $\{0, 1\}^*$. A secure public directory contains all the (X, E_X) pairs, while the decryption function D_X is known only to user X . The main requirements on E_X, D_X are:

- 1) $E_X D_X = D_X E_X = 1$, and
- 2) knowing $E_X(M)$ and the public directory does not reveal anything about the value M .

Thus everyone can send X a message $E_X(M)$, X will be able to decode it by forming $D_X(E_X(M)) = M$, but nobody other than X will be able to find M even if $E_X(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext M between two users. To give an idea of the way a intruder may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender's name, the receiver's name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender's name, text, receiver's name.

Example 1: Consider the following protocol for sending a plaintext M between A and B :

- a) A sends B the message $(A, E_B(M), B)$.
- b) B answers A with the message $(B, E_A(M), A)$.

Manuscript received July 15, 1981; revised August 3, 1982. This work was supported in part by DARPA under Grant N00014-81-1-0101 and by National Science Foundation under Grant MCS-77-08113-A01. This paper was partially presented at the 22nd Annual IEEE Symposium on Foundations of Computer Science, Nashville, TN, October 28-30, 1981.

D. Dolev was with the Computer Science Department, Stanford University, Stanford, CA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

A. C. Yao is with the Computer Science Department, Stanford University, Stanford, CA 94305.

Definition (Dolev-Yao Closure)

Given a set of terms M , we define $\mathcal{DY}(M)$ as the **least closure of M** under the following rules:

$$\begin{array}{c}
 \frac{}{m \in \mathcal{DY}(M)} \text{Axiom } (m \in M) \qquad \frac{s \in \mathcal{DY}(M)}{t \in \mathcal{DY}(M)} \text{Algebra } (s \approx t) \\
 \\
 \frac{t_1 \in \mathcal{DY}(M) \quad \dots \quad t_n \in \mathcal{DY}(M)}{f(t_1, \dots, t_n) \in \mathcal{DY}(M)} \text{Composition } (f \in \Sigma_p) \qquad \frac{\langle m_1, m_2 \rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)} \text{Proj}_i \\
 \\
 \frac{\{|m|\}_k \in \mathcal{DY}(M) \quad k \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)} \text{DecSym} \\
 \\
 \frac{\{m\}_k \in \mathcal{DY}(M) \quad \text{inv}(k) \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)} \text{DecAsym} \qquad \frac{\{m\}_{\text{inv}(k)} \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)} \text{OpenSig}
 \end{array}$$

There are many reasons for the state space being infinite, e.g.,

- ❏ **Messages:** The attacker can compose arbitrarily complex messages, ...
- ❏ **Sessions:** No bound on the number of executions of the protocol.
- ❏ **Nonces:** In an unbounded number of sessions, honest agents create an infinite number of fresh nonces.

For building a tool, we:

- ❏ May limit the number of sessions.
- ❏ May limit the number of agents.
- ❏ May limit the complexity of message (e.g., introducing types)

There are a lot successful automated tools for formal analysis of security protocols:

- ❖ OFMC, CL-AtSe, SATMC, ...: good for finding attacks, but only bounded verification!
 - ❖ H.530, Google's SAML SSO, Kerberos PKInit, ...
- ❖ SPASS/ProVerif, Scyther, AIF, ...: full verification, but no checkable proof.

Let's have a quick look at OFMC ...

VU#612636 - Google SAML

kb.cert.org/vuls/id/612636/

Search vulnerability notes

Carnegie Mellon University

Software Engineering Institute

CERT Coordination Center

Home

Notes

Search

Report a Vulnerability

Disclosure Guidance

VINCE

Home

Notes

VU#612636

Google SAML Single Sign on vulnerability

Vulnerability Note VU#612636

Original Release Date: 2008-09-02 | Last Revised: 2008-09-25

Print

Twitter

Facebook

Share

Overview

The SAML Single Sign-On (SSO) Service for Google Apps contained a vulnerability that could have allowed an attacker to gain access to a user's Google account.

Description

The Security Assertion Markup Language (SAML) is a standard for transmitting authentication data between two or more security domains. In SAML language, XML security packets are called

ABOUT VULNERABILITY NOTES

CONTACT US ABOUT THIS VULNERABILITY

PROVIDE A VENDOR STATEMENT

We have the problem
solved, haven't we??



Automated tools tend to

- ❑ focus on finding attacks, **not proving their absence**,
- ❑ use an axiomatic encoding of (logical) rules,
- ❑ use highly optimized and specialized inference algorithms.

How can we increase the trust in such tools?

Observation:

Many interactive theorem provers (e.g., Coq, Isabelle) follow a different philosophy

- ❑ a small trusted kernel (often only a few hundred lines of code)
- ❑ extensions are developed as conservative embeddings (proven/derived rules instead of new axioms)

Furthermore, modern security protocols

- ❑ are not used in isolation, we need to be able to compose them
- ❑ might rely on a shared state (e.g., a database of revoked cryptographic keys)

A protocol is modelled by as inductive set with

- ❑ rules modelling the Dolev-Yao style attacker
- ❑ rules modelling the protocol traces
- ❑ rule(s) signalling security violations ("oops")

Example (NSL, role \mathfrak{A})

$$\frac{t \in \mathbb{T} \quad NA \notin \text{used}(t)}{\text{iknows } \{NA, A\}_{pk(B)} \# \text{state } \mathfrak{A} [A, B, NA] \# \text{secret } B \quad NA \# t \in \mathbb{T}}$$

$$\frac{t \in \mathbb{T} \quad \text{state } \mathfrak{A} [A, B, NA] \in [t] \quad \text{iknows } \{NA, NB, B\}_{pk(A)} \in [t]}{\text{iknows } \{NB\}_{pk(B)} \# t \in \mathbb{T}}$$

Remarks:

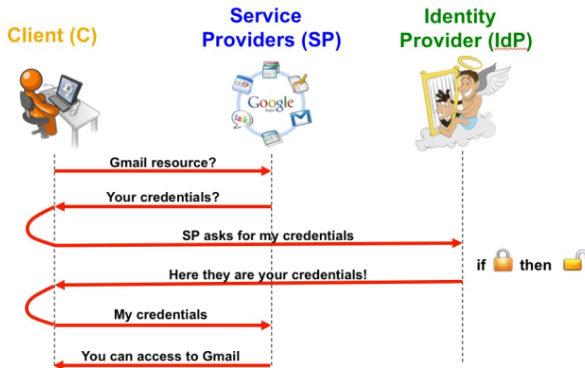
- ❑ the order of events in the traces never matters
- ❑ for a single message from B , Alice can react any number of times.
- ❑ the model is typed (i.e., an agent "knows" the purpose of message parts)

High trustworthiness of (a conservative construction in) Isabelle

Low degree of automation:

- ❑ security proof: "no oops-events in the inductively defined set"
- ❑ a lot of rules depend on the protocol and its security goals

- ❑ Abstract typed model vs. a (more) realistic concrete model
- ❑ Composition of different protocols
- ❑ Protocols with a shared state
- ❑ Better degree of automation



Leaving abstract two components here:

- ❑ The channels are running over TLS
- ❑ The client must in some way authenticate to the IdP

Knowledge: C: C,idp,SP,pk(idp);
idp: C,idp,pk(idp),inv(pk(idp));
SP: idp,SP,pk(idp)

Actions:

[C] *->* SP : C,SP,URI
SP *->* [C] : C,idp,SP,URI

C *->* idp : C,idp,SP,URI
idp *->* C : {C,idp}inv(pk(idp)),URI

[C] *->* SP : {C,idp}inv(pk(idp)),URI
SP *->* [C] : Data

Goals:

SP authenticates C on URI
C authenticates SP on Data
Data secret between SP,C

```

C->SP: C,NC,Sid,PC
SP->C: NSP,Sid,PSP,
      {SP,pk(SP)}inv(pk(s))
C->SP: {C,pk(C)}inv(pk(s)),
      {PMS}pk(SP),
      {hash(NSP,SP,PMS)}inv(pk(C)),
      {hash(prf(PMS,NC,NSP),C,SP,NC,NSP,Sid,PC,PSP,PMS)}
      clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {hash(prf(PMS,NC,NSP),C,SP,NC,NSP,Sid,PC,PSP,PMS)}
      serverK(NC,NSP,prf(PMS,NC,NSP))
C->SP: {|C,SP,URI|}clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {|C,idp,SP,URI|}serverK(NC,NSP,prf(PMS,NC,NSP))
C->idP: C,NC2,Sid2,PC
      idP->C: NidP,Sid2,PidP,
      {idP,pk(idP)}inv(pk(s))
C->idP: {C,pk(C)}inv(pk(s)),
      {PMS2}pk(idP),
      {hash(NidP,idP,PMS2)}inv(pk(C)),
      {hash(prf(PMS2,NC2,NidP),C,idP,NC2,NidP,Sid2,PC,PidP,PMS2)}
      clientK(NC2,NidP,prf(PMS,NC2,NidP))
idP->C: {hash(prf(PMS2,NC2,NidP),C,idP,NC2,NidP,Sid2,PC,PidP,PMS2)}
      serverK(NC2,NidP,prf(PMS2,NC2,NidP))
C->idP: {|C,idp,SP,URI,cky(C)|}clientK(NC2,NidP,prf(PMS,NC2,NidP))
idp->C: {|{C,idp}inv(pk(idp)),URI|}serverK(NC2,NidP,prf(PMS2,NC2,NidP))
C->SP: {|{C,idp}inv(pk(idp)),URI|}clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {|Data|}serverK(NC,NSP,prf(PMS,NC,NSP))
...

```



```

C->SP: C,NC,Sid,PC
SP->C: NSP,Sid,PSP,
      {SP,pk(SP)}inv(pk(s))
C->SP: {C,pk(C)}inv(pk(s)),
      {PMS}pk(SP),
      {hash(NSP,SP,PMS)}inv(pk(C)),
      {|hash(prf(PMS,NC,NSP),C,SP,NC,NSP,Sid,PC,PSP,PMS)|}
      clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {|hash(prf(PMS,NC,NSP),C,SP,NC,NSP,Sid,PC,PSP,PMS)|}
      serverK(NC,NSP,prf(PMS,NC,NSP))
C->SP: {|C,SP,URI|}clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {|C,idp,SP,URI|}serverK(NC,NSP,prf(PMS,NC,NSP))
C->idP: C,NC2,Sid2,PC
idP->C: NidP,Sid2,PidP,
      {idP,pk(idP)}inv(pk(s))
C->idP: {C,pk(C)}inv(pk(s)),
      {PMS2}pk(idP),
      {hash(NidP,idP,PMS2)}inv(pk(C)),
      {|hash(prf(PMS2,NC2,NidP),C,idP,NC2,NidP,Sid2,PC,PidP,PMS2)|}
      clientK(NC2,NidP,prf(PMS,NC2,NidP))
idP->C: {|hash(prf(PMS2,NC2,NidP),C,idP,NC2,NidP,Sid2,PC,PidP,PMS2)|}
      serverK(NC2,NidP,prf(PMS2,NC2,NidP))
C->idP: {|C,idp,SP,URI,cky(C)|}clientK(NC2,NidP,prf(PMS,NC2,NidP))
idP->C: {|{C,idp}inv(pk(idp)),URI|}serverK(NC2,NidP,prf(PMS2,NC2,NidP))
C->SP: {|{C,idp}inv(pk(idp)),URI|}clientK(NC,NSP,prf(PMS,NC,NSP))
SP->C: {|Data|}serverK(NC,NSP,prf(PMS,NC,NSP))

```

I think we can all agree: **This is too complicated**

Idea:

- ❑ Let's analyze TLS (in isolation)
- ❑ Let's analyze SAML (in isolation)

Is the composition secure, if the components are secure?

- ❑ Protocols with long-term state, e.g., a server maintaining a database
- ❑ General distributed systems: e.g. a server or device with API that do not necessarily have a simple linear order
- ❑ Complex shared messages, e.g., key certificates
- ❑ Desired interactions between protocols, e.g., one protocol generates a key, another protocol uses it.
- ❑ Support for modeling compromise and declassification of secrets.

Formal (Security) Verification

Our Results

Theorem (Typing Result)

Let P be a well-formed and type-flaw resistant protocol. If P is well-typed secure then P is also secure in the untyped model.

- ✦ Andreas V. Hess, Sebastian Mödersheim: A Typing Result for Stateful Protocols. CSF 2018: 374-388
- ✦ Andreas V. Hess, Sebastian Mödersheim, and Achim D. Brucker. Stateful Protocol Composition and Typing. In Archive of Formal Proofs, 2020.

Theorem (Stateful Protocol Composition (informal))

Given two protocols P_1 and P_2 where

- all access to shared sets is labeled (*)*
- $P_1 \parallel P_2$ is type-flaw resistant*
- the ground messages of P_1 and P_2 are disjoint*
- $P_1 \parallel P_2^*$ does not leak any classified secrets*
- $P_1^* \parallel P_2$ does not leak any classified secrets*

then $P_1 \parallel P_2$ is secure.

- Andreas V. Hess, Sebastian A. Mödersheim, and Achim D. Brucker. Stateful Protocol Composition in Isabelle/HOL. In ACM Transactions on Privacy and Security, 2023.
- Andreas V. Hess, Sebastian Mödersheim, and Achim D. Brucker. Stateful Protocol Composition and Typing. In Archive of Formal Proofs, 2020.

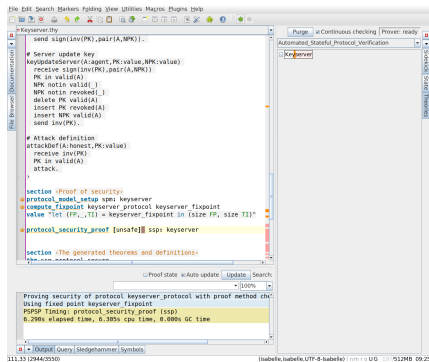
Isabelle/PSPSP:^a

- ❑ Stateful protocol (e.g., key revocation)
- ❑ Automated proof support for a large fragment of protocols.
(using an abstract fixed point computation)
- ❑ Horizontal and vertical composition.
- ❑ Formally proved equivalence (under certain conditions) of typed and untyped models.

Publications

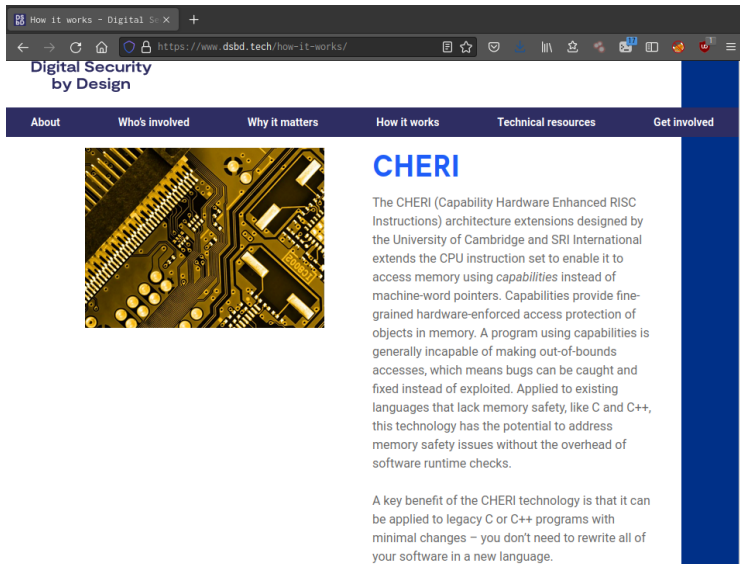
- ❑ Andreas V. Hess, Sebastian Mödersheim, Achim D. Brucker, and Anders Schlichtkrull. Performing Security Proofs of Stateful Protocols. In 34th CSF, 1, pages 143-158, IEEE, 2021.
- ❑ Andreas V. Hess, Sebastian Mödersheim, Achim D. Brucker, and Anders Schlichtkrull. Automated Stateful Protocol Verification. In Archive of Formal Proofs, 2020.

^aPerforming Security Proofs of Stateful Protocols



The screenshot displays the homepage of Logos Payment Solutions. The browser's address bar shows the URL `logos.dk`. The website's navigation menu includes links for [Cases](#), [Forside](#), [Job](#), [Kompetencer](#), [Kontakt](#), [Løsninger](#), [Nyheder](#), [Tjenester](#), and [Produkter](#). The main banner features the **LOGOS** logo and the text **Nem betaling overalt** (Easy payment everywhere), accompanied by a [Læs mere >](#) button. Below the banner is a large image of a building with the **LOGOS** sign. At the bottom, a cookie consent notice states: "Vi bruger cookies på vores hjemmeside for at give dig den mest relevante oplevelse ved at huske dine præferencer og gentage besøg. Ved at klikke på 'Accepter' giver du samtykke til brugen af ALLE cookies." The notice includes a [Cookie settings](#) link and an [Accepter](#) button.

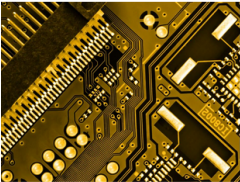
Further Application of Formal Methods for Security



The screenshot shows a web browser window with the URL <https://www.dsbd.tech/how-it-works/>. The page title is "Digital Security by Design". The navigation menu includes "About", "Who's involved", "Why it matters", "How it works" (which is the active page), "Technical resources", and "Get involved". The main content area features a large image of a circuit board on the left and the "CHERI" section on the right. The "CHERI" section has a blue heading and a paragraph explaining the technology. A vertical scrollbar is visible on the right side of the page.

Digital Security by Design

- About
- Who's involved
- Why it matters
- How it works
- Technical resources
- Get involved

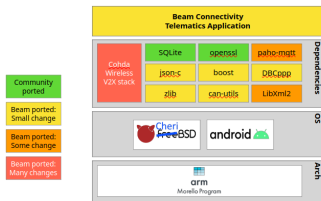


CHERI

The CHERI (Capability Hardware Enhanced RISC Instructions) architecture extensions designed by the University of Cambridge and SRI International extends the CPU instruction set to enable it to access memory using *capabilities* instead of machine-word pointers. Capabilities provide fine-grained hardware-enforced access protection of objects in memory. A program using capabilities is generally incapable of making out-of-bounds accesses, which means bugs can be caught and fixed instead of exploited. Applied to existing languages that lack memory safety, like C and C++, this technology has the potential to address memory safety issues without the overhead of software runtime checks.

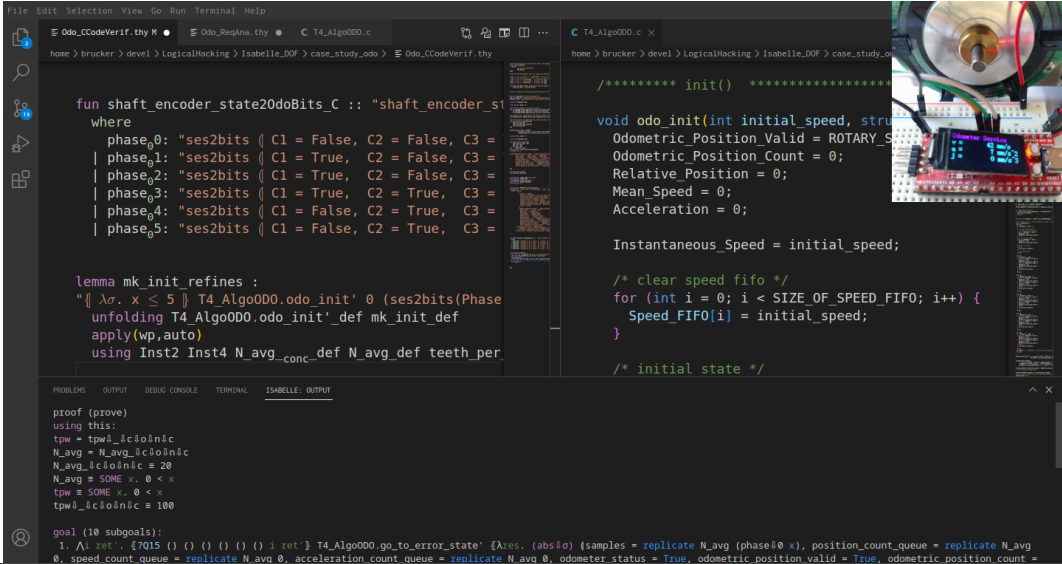
A key benefit of the CHERI technology is that it can be applied to legacy C or C++ programs with minimal changes – you don't need to rewrite all of your software in a new language.





Our work:

- ❖ Code review of TCU implementation: memory overflows are not always the biggest issues.
- ❖ Review of security architecture (theoretical assessment of compartmentalization).
- ❖ Comparison of CHERI C, Misra C, and formal verification.



The screenshot displays a development environment with two main code editors. The left editor shows a Coq proof script for a shaft encoder, and the right editor shows a C implementation of an odometer. A small image of the hardware is visible in the top right corner.

Left Editor (Odo_CCodeVerif.thy):

```

fun shaft_encoder_state20doBits_C :: "shaft_encoder_st
  where
    phase_0: "ses2bits ( C1 = False, C2 = False, C3 =
    | phase_1: "ses2bits ( C1 = True, C2 = False, C3 =
    | phase_2: "ses2bits ( C1 = True, C2 = False, C3 =
    | phase_3: "ses2bits ( C1 = True, C2 = True, C3 =
    | phase_4: "ses2bits ( C1 = False, C2 = True, C3 =
    | phase_5: "ses2bits ( C1 = False, C2 = True, C3 =

lemma mk_init_refines :
  "λ σ. x ≤ 5 ⟹ T4_AlgoOD0.odo_init' 0 (ses2bits(Phase
    unfolding T4_AlgoOD0.odo_init'_def mk_init_def
    apply(wp,auto)
    using Inst2 Inst4 N_avg_conc_def N_avg_def teeth_per

proof (prove)
  using this:
    tpw = tpw @ c @ o @ n @ c
    N_avg = N_avg @ c @ o @ n @ c
    N_avg @ c @ o @ n @ c = 20
    N_avg = SOME x. 0 < x
    tpw = SOME x. 0 < x
    tpw @ c @ o @ n @ c = 100

goal (10 subgoals):
  1. λ i ret'. {Q15 (i) () () () () i ret'} T4_AlgoOD0.go_to_error_state' {λ res. (abs 0) {samples = replicate N_avg (phase 0 x), position_count_queue = replicate N_avg
    0, speed_count queue = replicate N_avg 0, acceleration count queue = replicate N_avg 0, odometer status = True, odometric position valid = True, odometric position count =
  
```

Right Editor (T4_AlgoOD0.c):

```

/***** init() *****/

void odo_init(int initial_speed, struct
  Odometric_Position_Valid = ROTARY_S
  Odometric_Position_Count = 0;
  Relative_Position = 0;
  Mean_Speed = 0;
  Acceleration = 0;

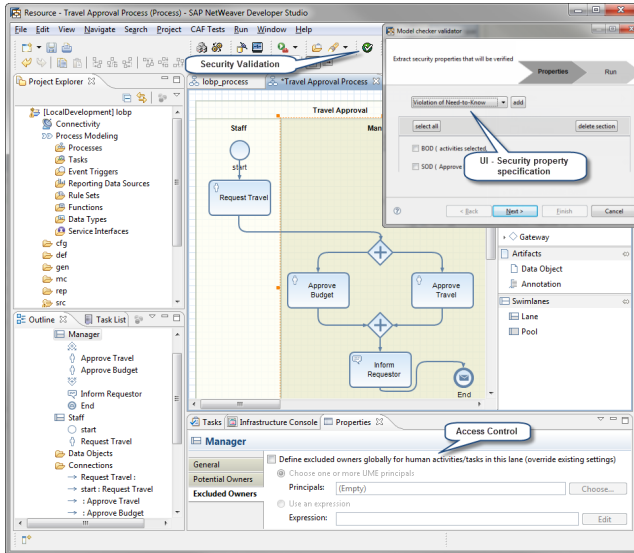
  Instantaneous_Speed = initial_speed;

  /* clear speed fifo */
  for (int i = 0; i < SIZE_OF_SPEED_FIFO; i++) {
    Speed_FIFO[i] = initial_speed;
  }

  /* initial state */
  
```

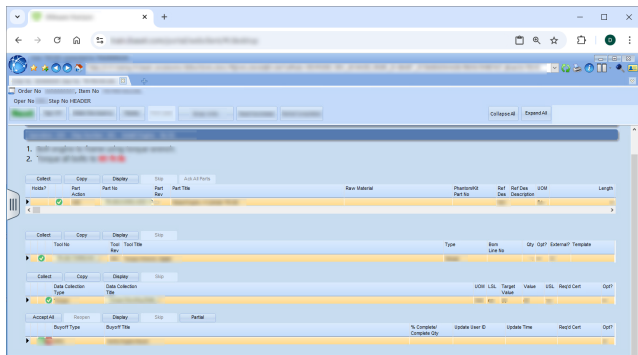
Hardware Image: A small image in the top right corner shows a physical implementation of the system, featuring a shaft encoder mounted on a PCB with various electronic components.

How it started: formal validation of business process models at SAP



ASLan

```
hc rbac_ac(Subject, Role, Task)
    := CanDo(Subject, Role, Task)
    :- user_to_role(Subject, Role),
       poto(Role, Task)
hc poto_T6 :=
    poto(Agent, ApproveBudget)
hc poto_T7 :=
    poto(Agent, ApproveTravel)
```



Focus on logic vulnerabilities and insider attacks:

- ❖ Forcing production stops.
- ❖ Use of non-approved parts.
- ❖ Approvals by non-authorized persons.
- ❖ "Shortcutting" process steps.
- ❖ Forcing inconsistencies of digital twins.
- ❖ ...



Key Take-Aways

For security experts:

- ❖ A system can only be secure wrt a threat model (attacker)
- ❖ Composing secure parts is not a guarantee for a secure overall system

For formal methods experts:

- ❖ We can formalize large aspects of security protocols in Isabelle
- ❖ We can have both: automation and the high assurance of Isabelle

Contact:



Prof. Dr. Achim D. Brucker
Department of Computer Science
University of Exeter
Streatham Campus
Exeter, EX4 4QF, UK

✉ a.brucker@exeter.ac.uk
✉ @adbrucker
in <https://de.linkedin.com/in/adbrucker/>
🌐 <https://www.brucker.ch/>
📖 <https://logicalhacking.com/blog/>

© 2025 LogicalHacking.com, A.D. Brucker.

- ❑ This presentation is classified as *Public (CC BY-NC-ND 4.0)*:
Except where otherwise noted, this presentation is licensed under a Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International Public License (CC BY-NC-ND 4.0).