Security Protocols
Foundations, Methods, and Tools

David Basin
Institute of Information Security, ETH Zurich

OAUTH Security Workshop
June 13th, 2017

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Welcome to ETH Zurich

We look forward to learning about your work

We will highlight some of our work that may be relevant

- Verification tools
- Identity management

Other things you can see while here

- Runtime monitoring (Dmitriy)
- Tamarin (Ralf, Cas, Lucca)
- Correct-by-construction development of protocols (Christoph))
- Verified Scion project, SD-WAN and more (Thilo, Christoph, Ralf)
- Particular protocols: voting, 5G (Ralf, Lucca)
- Access control, role & rule mining (Thilo)

A Typical Protocol

IKE, Phase 1, Main Mode, Digital Signatures, **Simplified**

$$
\begin{array}{lll}
(1) & I \rightarrow R: & C_I,\ ISA_I \\
(2) & R \rightarrow I: & C_I,\ C_R,\ ISA_R \\
(3) & I \rightarrow R: & C_I,\ C_R,\ g^x,\ N_I \\
(4) & R \rightarrow I: & C_I,\ C_R,\ g^y,\ N_R \\
(5) & I \rightarrow R: & C_I,\ C_R,\ \{ID_I,\ SIG_I\}_{SKEYID_e} \\
(6) & R \rightarrow I: & C_I,\ C_R,\ \{ID_R,\ SIG_R\}_{SKEYID_e}
\end{array}
$$

Properties?

$$
\begin{array}{lll}
SKEYID & = & h(\{N_I, N_R\}, g^{xy}) \\
SKEYID_d & = & h(SKEYID, \{g^{xy}, C_I, C_R, 0\}) \\
SKEYID_a & = & h(SKEYID, \{SKEYID_d, g^{xy}, C_I, C_R, 1\}) \\
SKEYID_e & = & h(SKEYID, \{SKEYID_a, g^{xy}, C_I, C_R, 2\}) \\
HASH_I & = & h(SKEYID_a, \{g^x, g^y, C_I, C_R, ISA_I, ID_I\}) \\
HASH_R & = & h(SKEYID_a, \{g^y, g^x, C_R, C_I, ISA_R, ID_R\}) \\
SIG_I & = & \{HASH_I\}_{K_I^{-1}} \\
SIG_R & = & \{HASH_R\}_{K_R^{-1}}
\end{array}
$$

Does argument order matter?

Why all the nested keyed hashes?

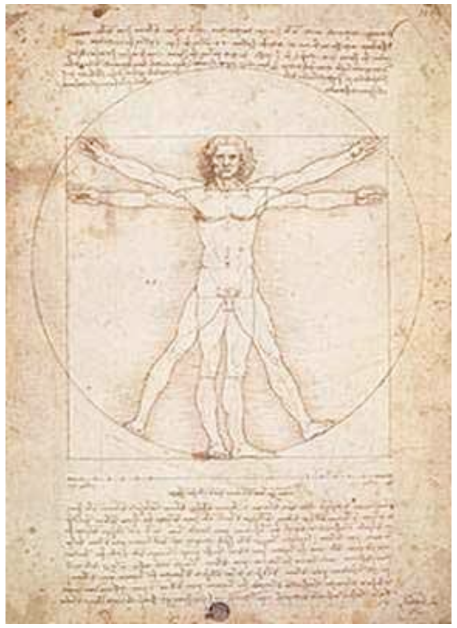
Protocol Design as an Art

Best practices, design by committee, reuse of previous protocols, ...

Whenever I made a roast, I always started off by cutting off the ends, just like my grandmother did. Someone once asked me why I did it, and I realized I had no idea. It had never occurred to me to wonder. It was just the way it was done. Eventually I asked my grandmother. “Why do you always cut off the ends of a roast?” She answered “Because my pan is small and otherwise the roasts would not fit.”

— *Anonymous*

Protocol Design as a Science



Methodology

- Can we soundly codify standard intuitions and best practices?
- Can the development be made systematic, incremental and scaleable?

Complexity

- What are the appropriate abstractions?
- How should we use these in the development?

Abstraction examples	
secrecy	encryption
authenticity	signatures, MACs
recentness	timestamps, nonces

Correctness

- Can development be combined with verification (correctness by construction)?
- Alternatively: can we take existing protocol (standards) and formally verify them?

Machine support

- Verification tools: OFMC, Tamarin, Scyther, Scyther Proof
- Testing tools: SecFuzz

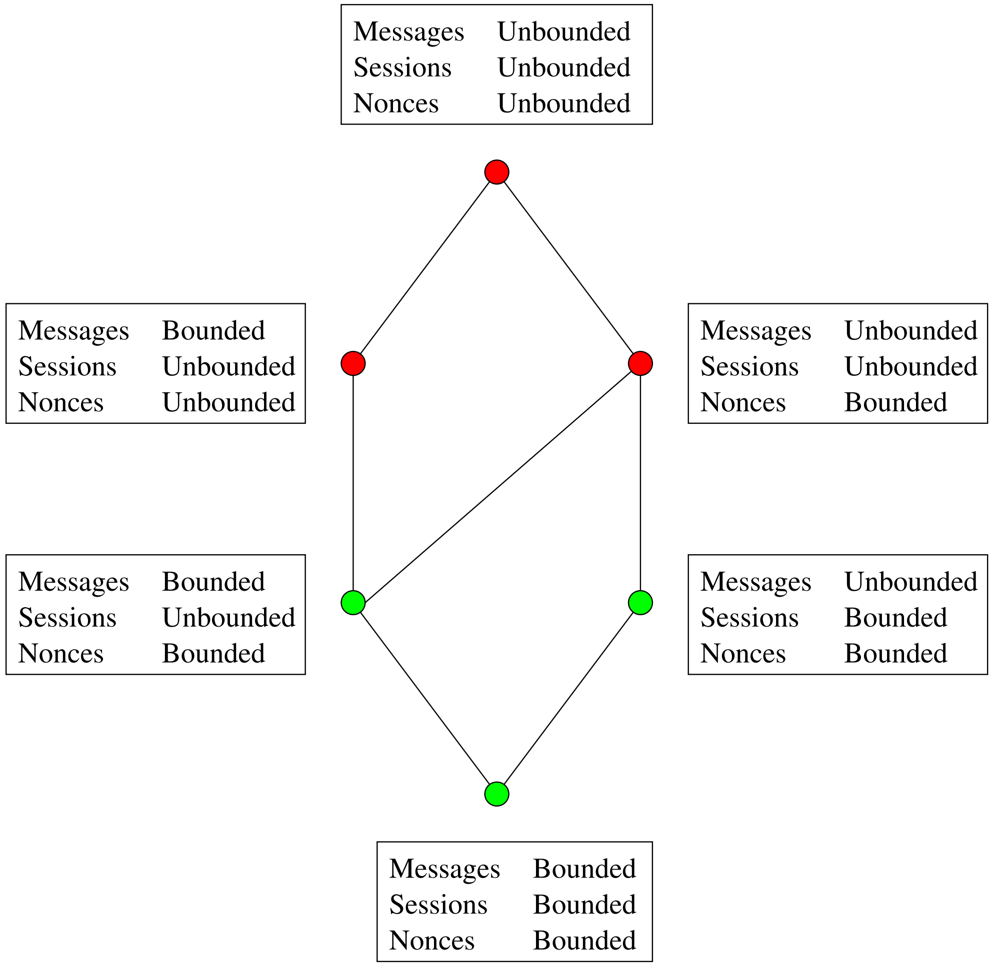
Security Protocol Verification and Develoment

Security Protocol Models

- Security protocols use cryptography to achieve their security goals (e.g., establish a secure channel, authentication, ...)
- Symbolic and computational models

Protocol Verification

- Secrecy problem is undecidable
- Problem caused by unboundedness of message size, # of sessions, # of nonces
- Decision procedures for restricted cases
- Unbounded verification (ProVerif, Scyther, Tamarin)



Protocol Development

- How to systematically develop protocols that are secure by construction?
- Has received less attention than post-hoc verification

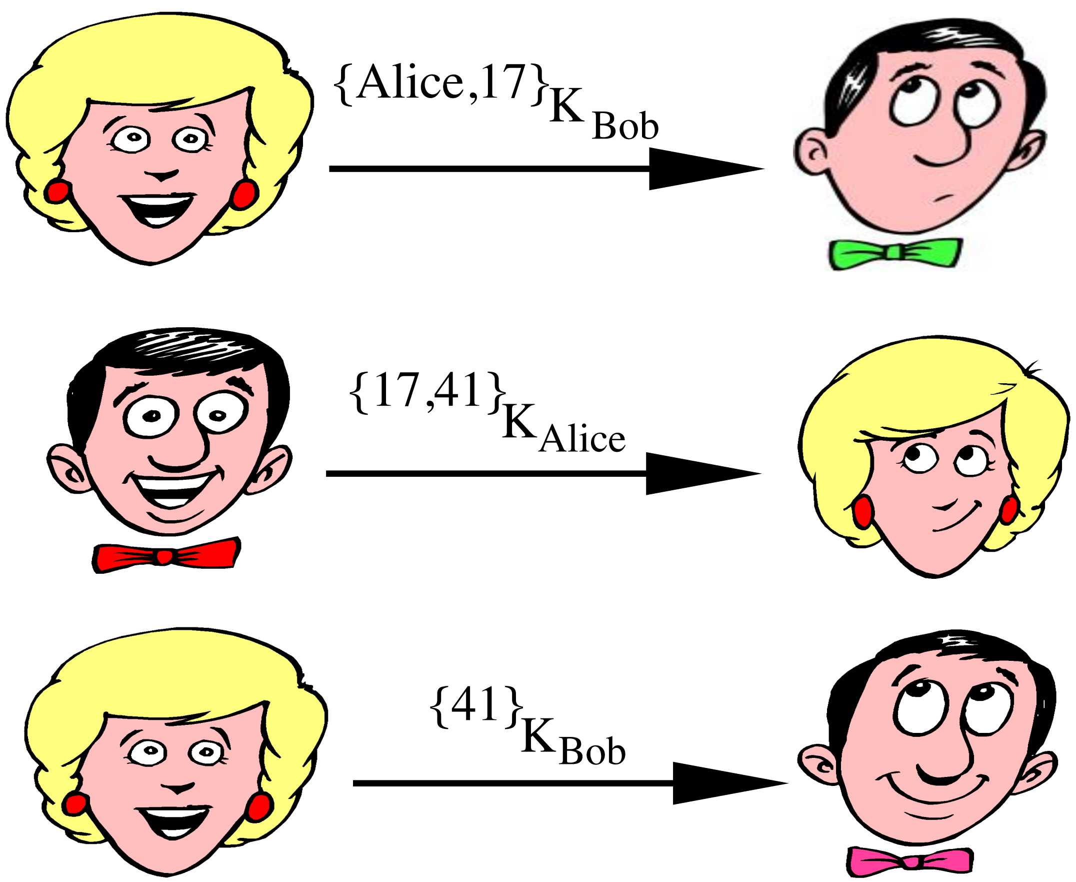
ETH Tools for Unbounded Protocol Analysis

	Scyther	scyther-proof	Tamarin
Main reference	CCS'08, CAV'08	CSF 2010	CSF 2012 (extended version)
Example applications	Compromising adversaries, IKE, protocol security hierarchies	ISO/IEC 9798	Naxos, UM, Signed Diffie-Hellman
Unbounded verification	Yes	Yes	Yes
Attack finding and visualisation	Yes		Yes
Classical properties (secrecy, agreement, aliveness, synchronisation)	Yes	Yes	Yes
Complete characterization	Yes		Yes
Property specification using a guarded fragment of first-order logic			Yes
Protocol specification	Linear role scripts	Linear role scripts	Multiset Rewriting (branching, loops)
Cryptographic message model	Free term algebra	Free term algebra	Diffie-Hellman & user-defined subterm-convergent rewrite theory
Dynamic corruption	Yes	Yes	Yes
Compromising adversaries	Yes		Yes
User-specified adversaries			Yes (e.g., eCK, eCK-PFS)
Generating machine-checked proofs		Yes (via Isabelle/HOL)	
Generating protocol security hierarchies	Yes		
Has been used in teaching	Yes (exercises available)		Yes
Proof visualisation		Yes	Yes
Interactive proof construction and exploration			Yes

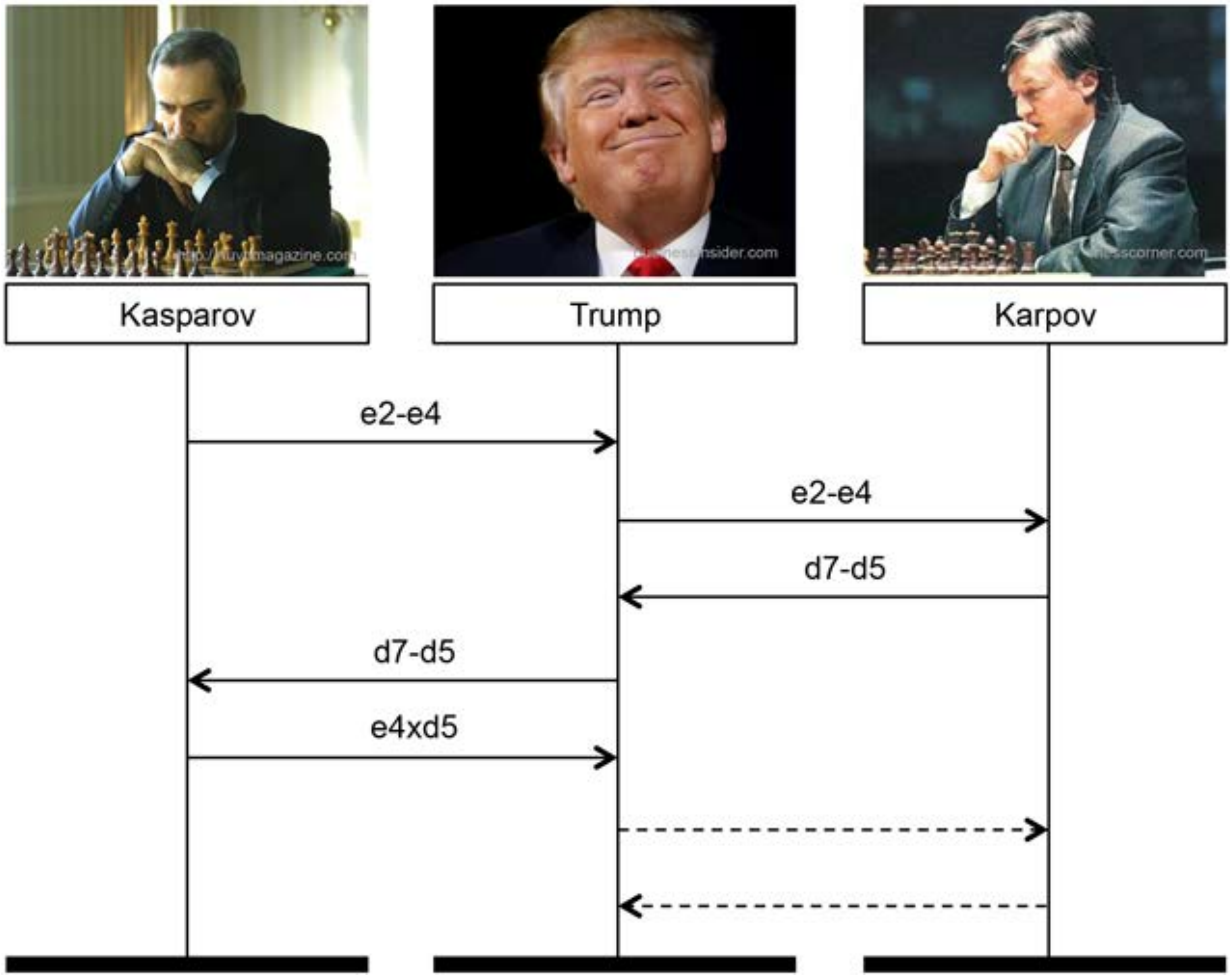
A Demo with Scyther

$$
\begin{array}{lll}
1. & A \to B: & \{A, N_A\}_{K_B} \\
2. & B \to A: & \{N_A, N_B\}_{K_A} \\
3. & A \to B: & \{N_B\}_{K_B}
\end{array}
$$

Here is an instance (a protocol run):

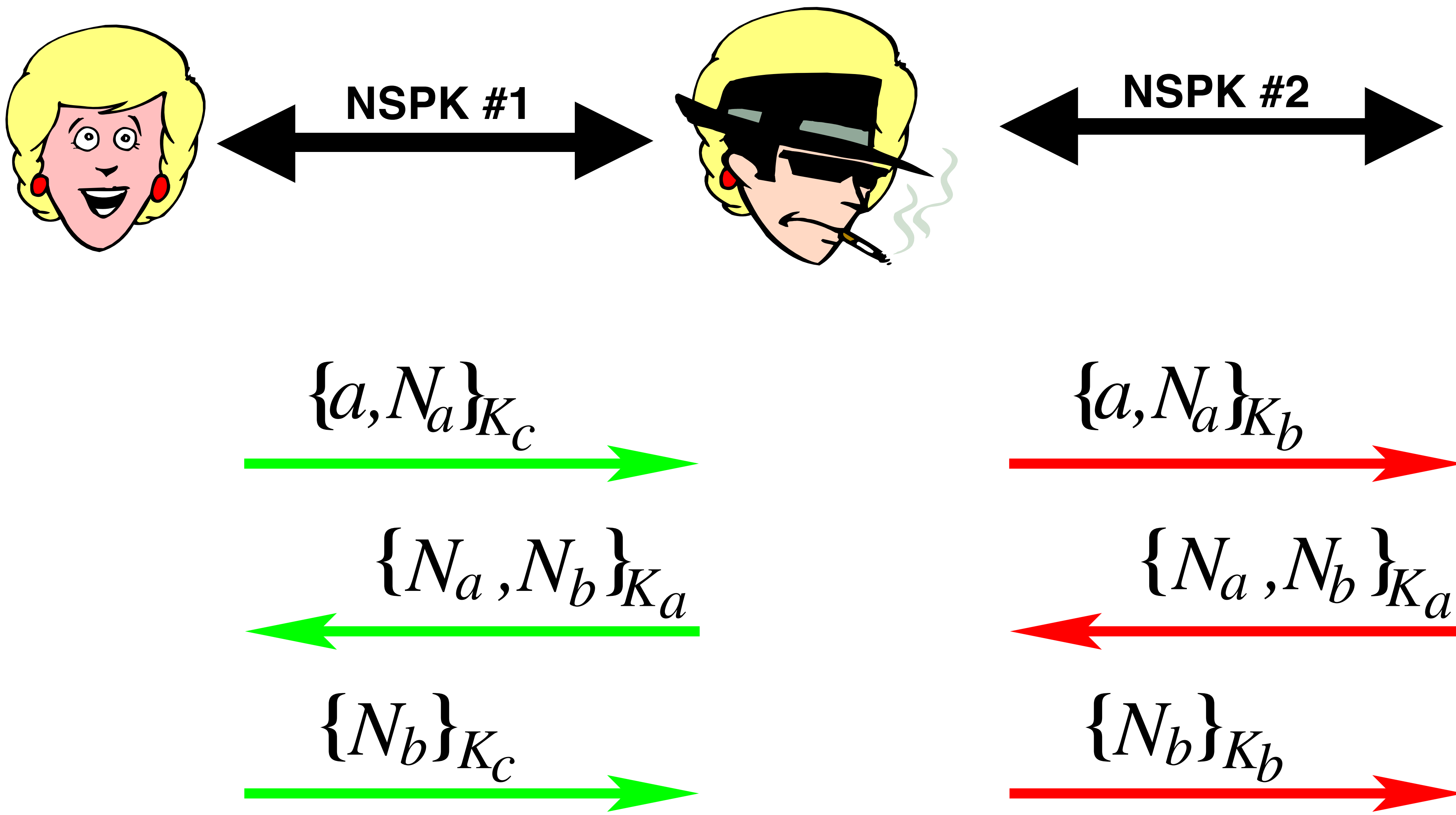


Even Trump can defeat Grandmasters



Attack on NSPK

1. $A \rightarrow B: \quad \{A, N_A\}_{K_B}$
2. $B \rightarrow A: \quad \{N_A, N_B\}_{K_A}$
3. $A \rightarrow B: \quad \{N_B\}_{K_B}$



$b(ob)$ believes he is speaking with $a(lice)$!

Focus: Provably Repairing the ISO/IEC 9798 Standard for Entity Authentication

Joint work with

Simon Meier

Cas Cremers

See: “Provably Repairing the ISO/IEC 9798 Standard for Entity Authentication”, Journal of Computer Security, 2013

Outline

ISO/IEC 9898: Purpose and Content

Automatic analysis

Fixes and machine-checked correctness proof

Engineering principles

New version of standard & conclusions

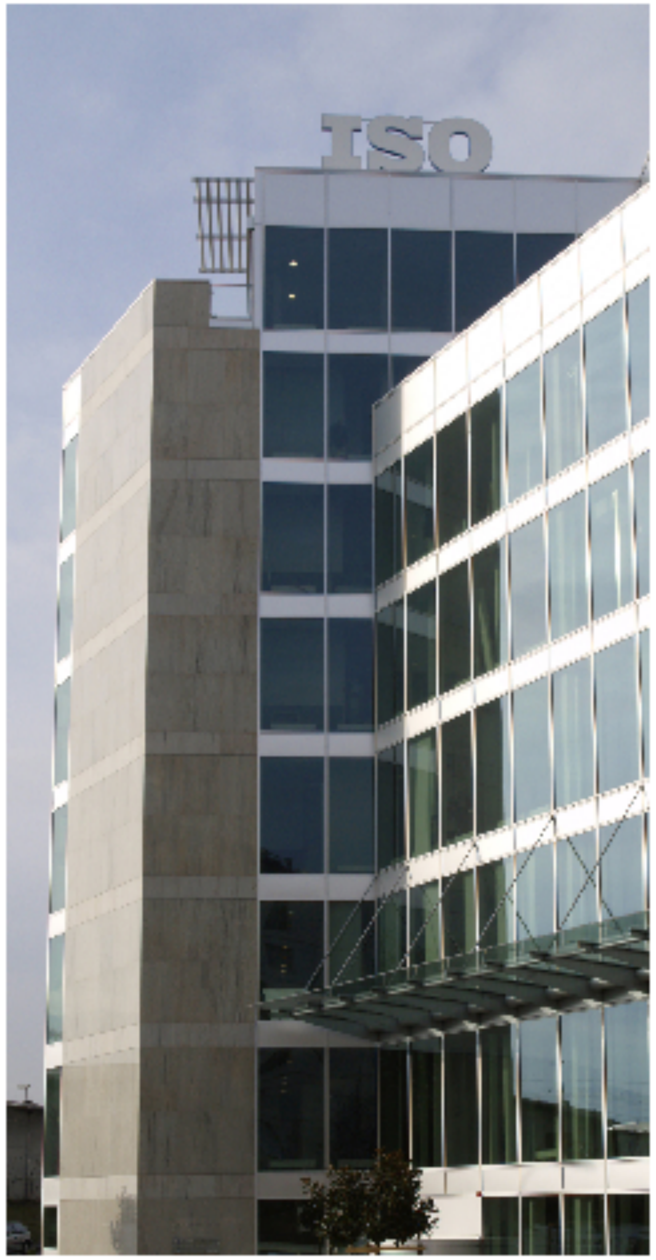
The ISO/IEC Standard

Entity Authentication Mechanism

17 base protocols

- Symmetric-key encryption, digital signatures, cryptographic check function
- Unilateral or mutual authentication
- Additional protocols with TTP

Further variants from optional fields



The ISO/IEC 9798 Standard

History

- Active development and updates since 1991
- Blueprints for protocol design
- Basis for ISO 11770 (Key Exchange) and NIST FIPS 196
- Mandated by other standards
  - e.g. European Banking Commission's smart card standards

International Organization for Standardization

Intended properties

- Entity authentication?
- E.g. resistant to reflection attacks
- Encrypted/signed payloads?

Standard issues: protocols and properties

Protocols

- 17 base protocols
- Optional text fields with application specific meaning
- Optional identifiers (can drop for efficiency?)

Protocol	Description
Part 2: Symmetric-key Cryptography	
9798-2-1	One-pass unilateral authentication
9798-2-2	Two-pass unilateral authentication
9798-2-3	Two-pass mutual authentication
9798-2-4	Three-pass mutual authentication
9798-2-5	Four-pass with TTP
9798-2-6	Five-pass with TTP
Part 3: Digital Signatures	
9798-3-1	One-pass unilateral authentication
9798-3-2	Two-pass unilateral authentication
9798-3-3	Two-pass mutual authentication
9798-3-4	Three-pass mutual authentication
9798-3-5	Two-pass parallel mutual authentication
9798-3-6	Five-pass mutual authentication with TTP, initiated by A
9798-3-7	Five-pass mutual authentication with TTP, initiated by B
Part 4: Cryptographic Check Functions	
9798-4-1	One-pass unilateral authentication
9798-4-2	Two-pass unilateral authentication
9798-4-3	Two-pass mutual authentication
9798-4-4	Three-pass mutual authentication

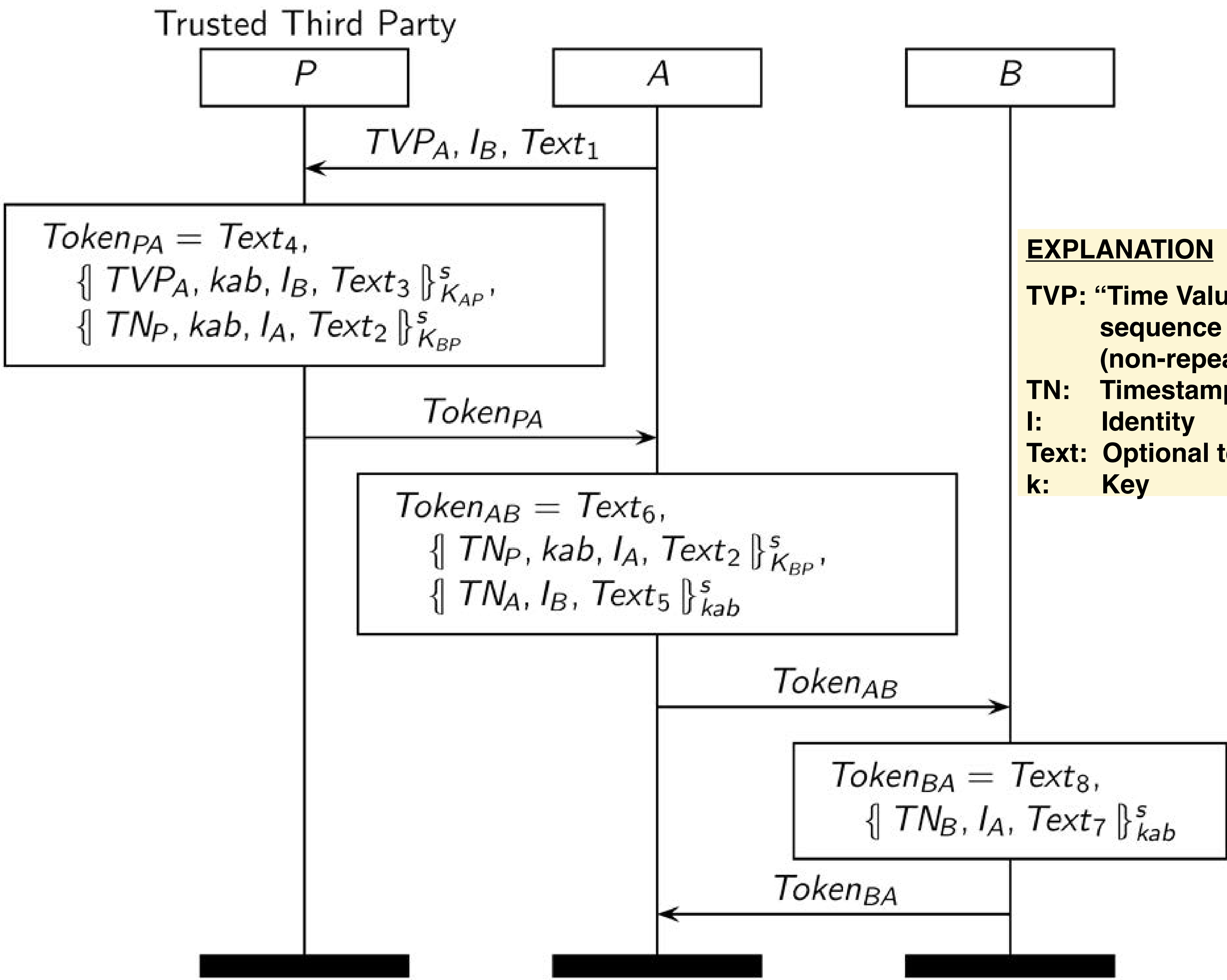
Properties

Table A. Typical interpretations of "a client C authenticated by a server S."

Variant	Entity authentication	Data agreement	Authenticated session key
Weaker	*Aliveness of C*: C has performed an action.	*Noninjective agreement on message m*: S has received the message *m* from C. C has sent *m* to S.	*Authenticated session key k*: session key *k* is a fresh session key, known only to C and S and possibly a trusted third party.
Stronger	*Recent aliveness of C*: C has performed an action (causally) after a specific action of S.	*Agreement on message m*: noninjective agreement on *m*, and S will not accept *m* if it is replayed by the adversary.	*Authenticated session key k with compromise resilience*: *k* is an authenticated session key, and compromise of an old session key does not lead to compromise of *k*.

ISO 9798-2-5

Symmetric key encryption with TTP



EXPLANATION

TVP: "Time Value Parameter" = sequence number, nonce, or timestamp (non-repeating)
TN: Timestamp or sequence number
I: Identity
Text: Optional text field
k: Key

Analysis

Request by CryptRec to evaluate standard

- Cryptography Research and Evaluation committees
- Funded by the Japanese government
- Part of long-running program to evaluate cryptographic mechanisms

Confirmation expected

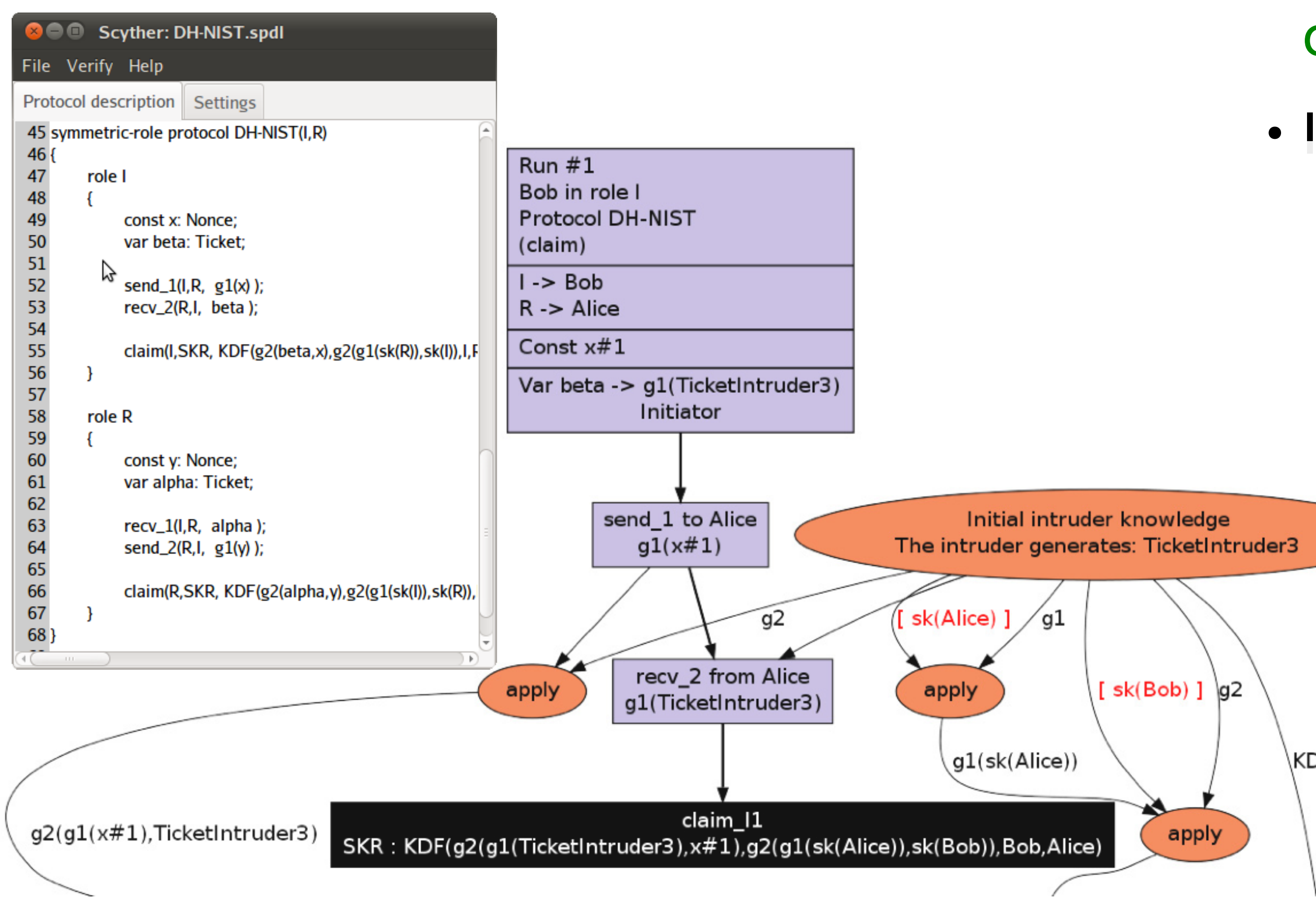
- Standard has been improved since 1994
- Substantial previous analysis (multiple rounds)

Tools used

Scyther

Symbolic analysis of security protocols

- Falsification (attack finding)
- Unbounded verification



Scyther-proof

- Embedding of protocol semantics and protocol-independent invariant in the ISABELLE/HOL theorem prover
- Algorithm similar to Scyther that outputs proof script for Isabelle/HOL
- Independent verifiability

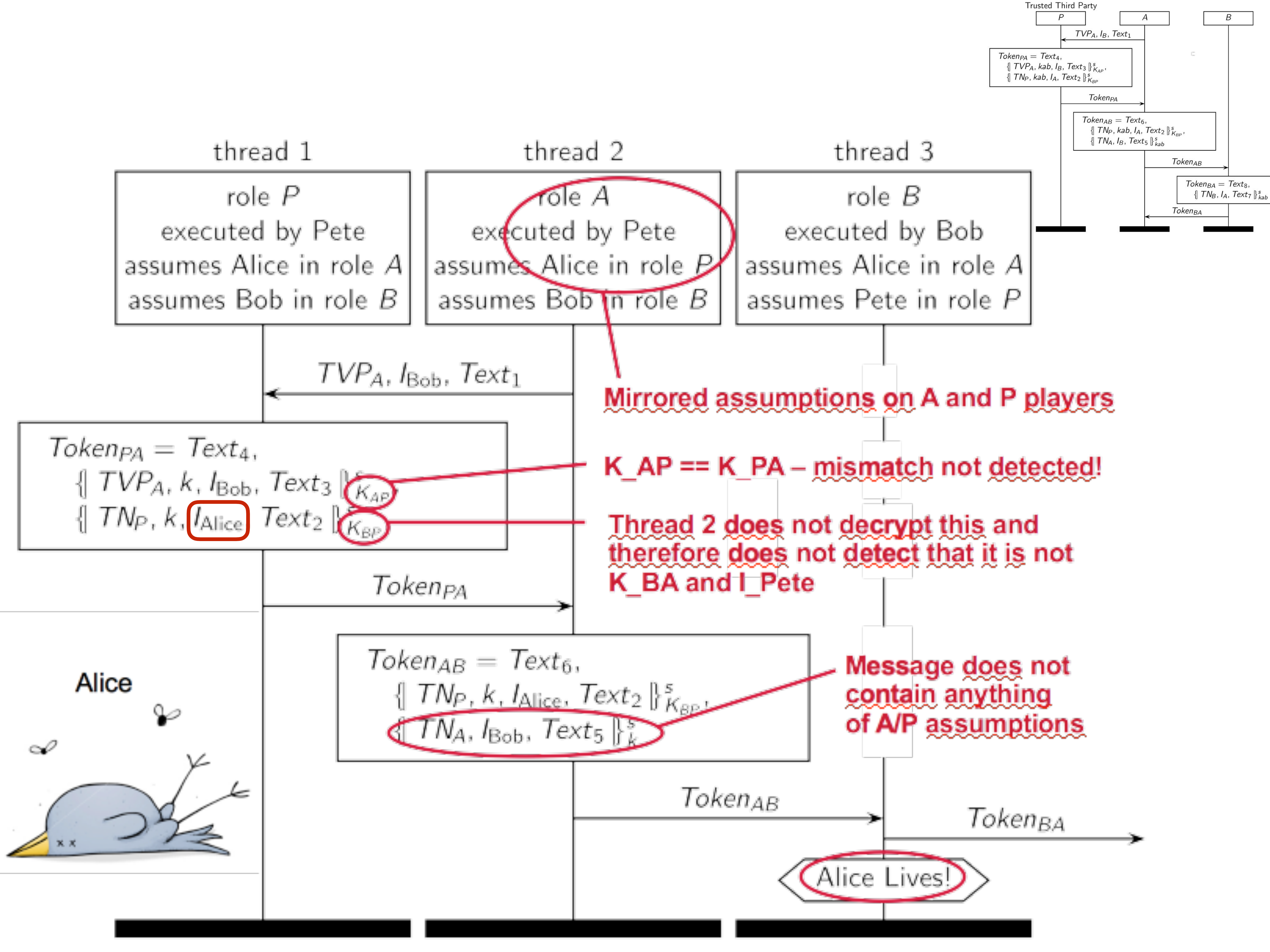
Results

No strong authentication properties

Aliveness < Agreement < Synchronisation

Under some conditions no authentication (weakest violated property listed)

Protocol	Violated property	Assumptions
9798-2-3	A Agreement(B,TNB,Text3)	
9798-2-3	B Agreement(A,TNA,Text1)	
9798-2-3-udkey	A Agreement(B,TNB,Text3)	
9798-2-3-udkey	B Agreement(A,TNA,Text1)	
9798-2-5	A Alive	Alice-talks-to-Alice
9798-2-5	B Alive	
9798-2-6	A Alive	
9798-2-6	B Alive	
9798-3-3	A Agreement(B,TNB,Text3)	
9798-3-3	B Agreement(A,TNA,Text1)	
9798-3-7-1	A Agreement(B,Ra,Rb,Text8)	Type-flaw
9798-4-3	A Agreement(B,TNb,Text3)	
9798-4-3	B Agreement(A,TNa,Text1)	
9798-4-3-udkey	A Agreement(B,TNb,Text3)	
9798-4-3-udkey	B Agreement(A,TNa,Text1)	


Trusted Third Party
P
A
B
TVP_A, I_B, Text_1
Token_PA = Text_4, {| TVP_A, kab, I_B, Text_3 |}^s_K_AP, {| TN_P, kab, I_A, Text_2 |}^s_K_BP
Token_PA
Token_AB = Text_6, {| TN_P, kab, I_A, Text_2 |}^s_K_BP, {| TN_A, I_B, Text_5 |}^s_kab
Token_AB
Token_BA = Text_8, {| TN_B, I_A, Text_7 |}^s_kab
Token_BA
thread 1
thread 2
thread 3
role P
executed by Pete
assumes Alice in role A
assumes Bob in role B
role A
executed by Pete
assumes Alice in role P
assumes Bob in role B
role B
executed by Bob
assumes Alice in role A
assumes Pete in role P
TVP_A, I_Bob, Text_1
Mirrored assumptions on A and P players
Token_PA = Text_4,
{| TVP_A, k, I_Bob, Text_3 |}^s_K_AP
{| TN_P, k, I_Alice, Text_2 |}^s_K_BP
K_AP == K_PA – mismatch not detected!
Thread 2 does not decrypt this and
therefore does not detect that it is not
K_BA and I_Pete
Token_PA
Alice
Token_AB = Text_6,
{| TN_P, k, I_Alice, Text_2 |}^s_K_BP,
{| TN_A, I_Bob, Text_5 |}^s_k
Message does not
contain anything
of A/P assumptions
Token_AB
Token_BA
Alice Lives!

Root causes of the problems

Message format is consistent and minimal

- Good design individually, but leads to possible confusion between different messages

No type information for fields

- Combined with above, can lead to type flaw attacks

Identity of one agent always included to break symmetry of shared keys

- Great, but doesn't work for three parties

Prudent engineering

Original rules [Abadi and Needham, 1995] insufficient

- **Principle 1**
  - Every message should say what it means: the interpretation of the message should depend only on its content. it should be possible to write down a straightforward English sentence describing the content — though if there is a suitable formalism available that is good too."

- **Principle 3**
  - "If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message."

New principles

Positional tagging

"Cryptographic message components should contain information that uniquely identifies their origin. In particular, the information should identify the protocol, the protocol variant, the message number, and the particular position within the message, from which the component was sent."

Example: message with fields omitted should contain information to determine this.

Inclusion of identities and their roles

"Each cryptographic message component should include information about the identities of all the agents involved in the protocol run and their roles unless there is a compelling reason to do otherwise." (Possible compelling reason: identity protection)

Example: include ordered sequence of identities involved for each role.

Repairing ISO/IEC 9798

We proposed fixes and machine-checked correctness proofs

- Fixes do not require additional cryptography
- Fixes follow new principles

Scyther-proof generates proof scripts for Isabelle-HOL

Proofs even guarantee correctness when executing all the protocols in parallel

- Excludes multi-protocol attacks

Effort

Modeling effort: a couple of weeks

- Abstraction level of standard close to formal models
- Some iteration inevitable after initial analysis with Scyther

Generating proof scripts using Scyther-proof

- 20 seconds

Checking correctness of scripts in Isabelle/HOL

- 3 hours (correctness for all protocols in parallel)

Experience similar on other projects

- and also with proprietary designs

Conclusion

Improving the ISO/IEC 9798 standard

- Old version: only weak authentication, sometimes none
- Successful interaction between researchers and standardization committee
- New version of the standard has been released which guarantees strong authentication (synchronization)
- Machine-checked symbolic proofs of standard

International Organization for Standardization

Future standardization efforts should take note

- Automated formal analysis is feasible and useful
- Current work: more complex protocols
  - Rekeying, databases, complex control flow
  - 5G protocols
  - Also identity management