Bridging the Gap: Formal vs. Complexity-Theoretical Reasoning about Cryptography

Jan Jürjens*

Computing Laboratory, University of Oxford

Joint work with

Martín Abadi

Bell Labs Research, Lucent

* http://www.jurjens.de/jan

Security Protocols

Cryptographic Protocol:

Exchange of messages for distributing session keys, authenticating principals etc. using cryptographic algorithms

Security Protocols

Cryptographic Protocol:

Exchange of messages for distributing session keys, authenticating principals etc. using cryptographic algorithms

Goals

- Session Keys: Establish or distribute keys for secure communication sessions between principals
- Authentication: Assurance of who you are talking to
- more specialised goals: Fair Exchange, Nonrepudiation, Anonymity,. . .

Security Protocols

Cryptographic Protocol:

Exchange of messages for distributing session keys, authenticating principals etc. using cryptographic algorithms

Goals

- Session Keys: Establish or distribute keys for secure communication sessions between principals
- Authentication: Assurance of who you are talking to
- more specialised goals: Fair Exchange, Nonrepudiation, Anonymity,. . .

Small, security-critical parts of a system

Security Protocols: Problems

Many protocols have vulnerabilities or subtleties for various reasons

- weak cryptography
- core message exchange
- interfaces, prologues, epilogues
- deployment
- implementation bugs

Security Protocols: Problems

Many protocols have vulnerabilities or subtleties for various reasons

- weak cryptography
- core message exchange
- interfaces, prologues, epilogues
- deployment
- implementation bugs

Message exchange: Problems

Classical example:

Needham-Schroeder (1978):

Authentication Protocols (shared/public key)

Message exchange: Problems

Classical example:

Needham-Schroeder (1978):

Authentication Protocols (shared/public key)

$\leadsto$ attack by Denning-Sacco (1981)

Message exchange: Problems

Classical example:

Needham-Schroeder (1978):

Authentication Protocols (shared/public key)

$\leadsto$ attack by Denning-Sacco (1981)

(btw: also more recent protocols)

Abstract protocol descriptions

Specify protocol participants as processes using formal methods (following Dolev, Yao 1982).

Abstract protocol descriptions

Specify protocol participants as processes using formal methods (following Dolev, Yao 1982).

In addition to expected participants, we model an attacker, who:

- may participate in some protocol runs,
- knows some data in advance,
- may intercept messages on the public network,
- injects messages that it can produce into the public network

The formal view: strengths and blind spots

Easy human reasoning (e.g., “BAN logic” – Burrows, Abadi, Needham 1989).

Techniques and tools for automated reasoning:

- Model-checkers
- Theorem provers

The formal view: strengths and blind spots

Easy human reasoning (e.g., "BAN logic" – Burrows, Abadi, Needham 1989).

Techniques and tools for automated reasoning:

- Model-checkers
- Theorem provers

But: does not consider attacks that rely e.g. on guessing values (such as keys).

The complexity-theoretical view

Keys and messages are bitstrings, not formal expressions.

A good protocol is one that resists computationally reasonable attacks with high probability.

This computational view leads to another rigorous approach for reasoning about protocols.

(See Yao, Blum, Goldwasser, Goldreich, Micali, Wigderson, Pfitzmann, Bellare, Rogaway, . . . 1982 $\leadsto$)

Formal vs. complexity-theoretical

In our opinion, the complexity-theoretical approach is

- foundationally more satisfying,
- more complete

Formal vs. complexity-theoretical

In our opinion, the complexity-theoretical approach is

- foundationally more satisfying,
- more complete,
- further from general methods for reasoning about reactive systems,
- harder to apply,
- sometimes overkill,
- harder to link to naive intuitions.

Model vs. Reality

Relevance of formal models to running of program only as good as faithfulness to which models represent real executions of the program (Manna, Pnueli 1991)

Deployment of security protocols: D. Gollmann

Here: formal modelling of cryptography

Bridging the gap

Soundness property (desired):

> If a security property can be proved formally,
> then it holds in the computational model (with high probability, for all computationally reasonable attacks).

The formal proof will

- not mention probabilities and complexities,
- consider attacks only in the formal model,
- establish an all-or-nothing statement.

Abadi, Rogaway 2000: Soundness of single messages

This work: Soundness of protocol runs with passive adversary.

A formal view of security protocols

System: collection of synchronously executed programs, communicating asynchronously through "channels".
Used to specify a protocol.

Channel $A \to B$ from A to B:

- A sends messages out to the network, with intended recipient B
- B receives message from network, apparently from A

Formal view: Systems

For each channel c the system contains exactly one program P_c that outputs on c.

P_c takes input from any channel at time n and computes output on the channel c for time $n+1$ (local state via (unobservable) feedback channels; gives iteration).

Formal view: Expressions

Sent messages: expressions

$M, N ::=$	expressions
$\quad \varepsilon$	null value
$\quad K$	key ($K \in \mathbf{Keys}$)
$\quad b$	bit ($b \in \mathbf{Bool}$)
$\quad a$	input on channel ($a \in \mathbf{Channels}$)
$\quad x$	variable ($x \in \mathbf{Var}$)
$\quad (M, N)$	pair
$\quad \{M\}_P^r$	encryption (P program, $r \in \mathbf{Coins}$)

Here: symmetric encryption

Formal view: Programs

System nodes are specified by programs:

$P, Q ::=$	programs
E	expression
$(\nu n)P$	"new" $(n \in \mathbf{Keys} \cup \mathbf{Coins})$
$\mathit{if}\ \ P = Q\ \ \mathit{then}\ \ P'\ \ \mathit{else}\ \ Q'$	conditional
$\mathit{case}\ \ P\ \ \mathit{of}\ \ (x, y)\ \ \mathit{do}\ \ Q\ \ \mathit{else}\ \ R$	case: pair $(x, y \in \mathbf{Var})$
$\mathit{case}\ \ P\ \ \mathit{of}\ \ \{x\}_{P'}\ \ \mathit{do}\ \ Q\ \ \mathit{else}\ \ R$	case: encryption $(x \in \mathbf{Var})$

Formal view: Interpretation

For program P and valuation $\vec{M} : \mathbf{Channels} \cup \mathbf{Var} \to \mathbf{Exp}$, define $[\![P]\!](\vec{M})$ (result from one iteration of P given $\vec{M}$). E.g.

$$
\begin{array}{ll}
[\![K]\!](\vec{M}) = K & (K \in \mathbf{Keys}) \\
[\![u]\!](\vec{M}) = \vec{M}_u & (u \in \mathbf{Channels} \cup \mathbf{Var}) \\
[\![(\nu v)P]\!](\vec{M}) = [\![P]\!](\vec{M}) & (v \in \mathbf{Keys} \cup \mathbf{Coins}) \\
[\![\{P\}^r_Q]\!](\vec{M}) = \{[\![P]\!](\vec{M})\}^r_{[\![Q]\!](\vec{M})} & \text{if } [\![P]\!](\vec{M}) \in \mathbf{Plain} \text{ and } [\![Q]\!](\vec{M}) \in \mathbf{Keys} \\
[\![\{P\}^r_Q]\!](\vec{M}) = \{0\}^r_{[\![Q]\!](\vec{M})} & \text{if } [\![P]\!](\vec{M}) \notin \mathbf{Plain} \text{ and } [\![Q]\!](\vec{M}) \in \mathbf{Keys} \\
[\![\{P\}^r_Q]\!](\vec{M}) = \perp & \text{if } [\![Q]\!](\vec{M}) \notin \mathbf{Keys}
\end{array}
$$

Extend to sequences of inputs and systems of programs.

Equivalence

Informally, two expressions are equivalent if they look the same to an attacker.

Formally, define equivalence using patterns where undecryptable subexpressions are substituted with the symbol $\Box_r$ ($r \in \mathbf{Coins}$).

Example:

$(\{\{0\}_{K_1}\}_{K_2}, K_2) \cong (\{\{1\}_{K_1}\}_{K_2}, K_2)$

since both expression yield $(\{\Box\}_{K_2}, K_2)$.

Systems are equivalent if they yield equivalent sequences of expressions.

Secrecy as formal equivalence

We express the protocol by a parameterized process, $P(x)$, where x is the data to be transmitted.

Two processes are equivalent if no environment can distinguish them.

(This is observational or testing equivalence.)

The secrecy of the data can be expressed as:

$P(M)$ and $P(N)$ are equivalent, for any M and N.

The computational view

An encryption scheme consists of algorithms:

$$
\begin{aligned}
\mathcal{K} &: \mathsf{Parameter} \times \mathsf{Coins} \to \mathsf{Key} \\
\mathcal{E} &: \mathsf{Key} \times \mathsf{String} \times \mathsf{Coins} \to \mathsf{Cipher} \\
\mathcal{D} &: \mathsf{Key} \times \mathsf{String} \to \mathsf{Plain}
\end{aligned}
$$

where $\mathsf{Parameter} = 1^*$ (numbers in unary),
$\mathsf{Key}, \mathsf{Plain}, \mathsf{Cipher} \subseteq \mathsf{String}$.

For all $\eta \in \mathsf{Parameter}$, $k \in \mathcal{K}(\eta)$, and $r \in \mathsf{Coins}$,

- if $m \in \mathsf{Plain}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$,
- if $m \notin \mathsf{Plain}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = \perp$ (error message)

Indistinguishable ensembles

A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for all $c > 0$ there exists N such that $\epsilon(\eta) \leq \eta^{-c}$ for all $\eta \geq N$.

An ensemble (or *probability ensemble*) is a collection of distributions on strings, $D = \{D_\eta\}$, one for each η.

We say that D and D' are indistinguishable and write $D \approx D'$, if

$$\begin{aligned}\epsilon(\eta) \triangleq\ & \Pr[x \stackrel{R}{\leftarrow} D_\eta :\ A(\eta, x) = 1] - \\ & \Pr[x \stackrel{R}{\leftarrow} D'_\eta :\ A(\eta, x) = 1]\end{aligned}$$

is negligible for all polynomial-time adversaries A.

(Variant of) Secure encryption

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme,
let $\eta \in \mathsf{Parameter}$ be a security parameter.

Define

$$\begin{aligned} \mathsf{Adv}_{\Pi,\eta}(A) \;\triangleq\;& \Pr\left[k, k' \xleftarrow{R} \mathcal{K}(\eta) : \; A^{\mathcal{E}_k(\cdot),\, \mathcal{E}_{k'}(\cdot)}(\eta) = 1\right] - \\ & \Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) : \; A^{\mathcal{E}_k(\mathbf{0}),\, \mathcal{E}_k(\mathbf{0})}(\eta) = 1\right] \end{aligned}$$

Encryption scheme Π is secure if $\mathsf{Adv}_{\Pi,\eta}(A)$ is negligible for all polynomial-time adversaries A.

(Goldwasser & Micali, Bellare et al.; Abadi & Rogaway)

Repetition concealing, message-length concealing, which-key concealing

Wrong key ?

In formal models, decrypting a message with the "wrong" key is noticeable error. Computational counterpart:

Encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is confusion-free if for all $m \in \mathsf{String}$ the probability $\Pr[k, k' \xleftarrow{R} \mathcal{K}(\eta), x \xleftarrow{R} \mathcal{E}_k(m) : \mathcal{D}_{k'}(x) \neq \perp]$ is negligible.

"Encryption without integrity-checking useless" (Bellovin)

Related: committing encryption (M. Fischlin)

(Simplified) Computational Interpretation

To any set $\mathcal{P}$ of interacting programs we assign a distribution $[\![\mathcal{P}]\!]_{\Pi,\eta}$ on traces (given an encryption scheme Π and a security parameter η):

Map each key symbol K to a bitstring $\tau(K)$, using $\mathcal{K}(\eta)$, and each coin symbol r to a bitstring $\tau(r)$. Then for expressions

$$
\begin{aligned}
[\![b]\!]_{\Pi,\eta} &= b \\
[\![K]\!]_{\Pi,\eta} &= \tau(K) \\
[\![(M,N)]\!]_{\Pi,\eta} &= ([\![M]\!]_{\Pi,\eta}, [\![N]\!]_{\Pi,\eta}) \\
[\![\{M\}_K^r]\!]_{\Pi,\eta} &= \mathcal{E}_{\tau(K)}^{\tau(r)}([\![M]\!]_{\Pi,\eta})
\end{aligned}
$$

and extend to programs.

Protocol Executions

Three of the subtleties:

- $\mathit{if}\ K = K'\ \mathit{then}$ publish secret $\mathit{else}\ \mathit{nil}$ is secure formally, but only "mostly secure" computationally
- message repetition
- recognize decryption with wrong key

Main theorem:
Computational soundness of formal encryption

Let $\mathcal{P}$ and $\mathcal{P}'$ be sets of programs that do not generate encryption cycles and Π be a secure and confusion-free encryption scheme.

If $[\![\mathcal{P}]\!] \cong [\![\mathcal{P}']\!]$ then $[\![\mathcal{P}]\!]_\Pi \approx [\![\mathcal{P}]\!]_\Pi$.

That is, if $\mathcal{P}$ and $\mathcal{P}'$ are formally equivalent then their meanings are computationally indistinguishable.

Interpretation

Identified conditions when formal reasoning is sound.

Interpretation

Identified conditions when formal reasoning is sound.

If there is an attack in the computational model, then there is also an attack in the formal model.

Interpretation

Identified conditions when formal reasoning is sound.

If there is an attack in the computational model, then there is also an attack in the formal model.

We have a simple method for proving some indistinguishability results.

Related work

Pfitzmann, Schunter, Waidner:

Cryptographic Security of Reactive Systems

Lincoln, Mitchell, Mitchell, Scedrov:

formal model including probabilities and complexities.

Conclusion

Foundations and validation of formal cryptology.

Proof methods for computational cryptology.

Links with intuitions about cryptography.

Further Work

Abstracting from failure probabilities in reasoning about dependable systems

Future Work

(Completeness)

Active adversary

Other cryptographic primitives

Protocol interactions vs. cryptographic equations

Interpretation II

$$[\![if\ \ P = Q\ \ then\ \ P'\ \ else\ \ Q']\!](\vec{M}) = [\![P']\!](\vec{M}) \qquad \text{if } [\![P]\!](\vec{M}) = [\![Q]\!](\vec{M})$$

$$[\![if\ \ P = Q\ \ then\ \ P'\ \ else\ \ Q']\!](\vec{M}) = [\![Q']\!](\vec{M}) \qquad \text{if } [\![P]\!](\vec{M}) \neq [\![Q]\!](\vec{M})$$

$$[\![case\ \ P\ \ of\ \ (x, y)\ \ do\ \ Q\ \ else\ \ R]\!](\vec{M}) = [\![Q]\!](\vec{M}[x \to M_1, y \to M_2])$$

$$\text{if } [\![P]\!](\vec{M}) = (M_1, M_2)$$

$$[\![case\ \ P\ \ of\ \ (x, y)\ \ do\ \ Q\ \ else\ \ R]\!](\vec{M}) = [\![R]\!](\vec{M}) \quad \text{if } [\![P]\!](\vec{M}) \text{ is not a pair}$$

$$[\![case\ \ P\ \ of\ \ \{x\}_{P'}\ \ do\ \ Q\ \ else\ \ R]\!](\vec{M}) = [\![Q]\!](\vec{M}[x \to M])$$

if $[\![P]\!](\vec{M}) = \{M\}^r_{[\![P']\!](\vec{M})}$ for $M \in \mathbf{Plain}$ and $r \in \mathbf{Coins}$ and if $[\![P']\!](\vec{M}) \in \mathbf{Keys}$

$$[\![case\ \ P\ \ of\ \ \{x\}_{P'}\ \ do\ \ Q\ \ else\ \ R]\!](\vec{M}) = [\![R]\!](\vec{M}) \quad \text{otherwise}$$

Message production, formally

Suppose that S is a set of messages that the attacker knows in advance, invents, or receives.

Then it can produce M if $S \vdash M$:

- $\mathcal{M} \vdash M$ for any $M \in \mathcal{M}$,
- if $\mathcal{M} \vdash (N_1, N_2)$ then $\mathcal{M} \vdash N_1$ and $\mathcal{M} \vdash N_2$,
- if $\mathcal{M} \vdash \{N\}_K^r$ (for any $r \in \mathbf{Coins}$) and $\mathcal{M} \vdash K$ then $\mathcal{M} \vdash N$.

Patterns

We map each expression to a *pattern* that the attacker can see with no a priori knowledge:

$$
\begin{array}{lcll}
pattern(K) & = & K & \text{(for } K \in \mathbf{Keys}) \\
pattern(b) & = & b & \text{(for } b \in \mathbf{Bool}) \\
pattern((M,N)) & = & (pattern(M), pattern(N)) & \\
pattern(\{M\}_K^r) & = & \begin{cases} \{pattern(M)\}_K^r & \text{if } K \in T \\ \Box_r & \text{otherwise} \end{cases} &
\end{array}
$$

where $T = \{K \in \mathbf{Keys} \mid M \vdash K\}$.

Example: $pattern((\{\{0\}_{K_1}^r\}_{K_2}^{r'}, K_2)) \;=\; (\{\Box_r\}_{K_2}^{r'}, K_2)$