Introduction to Computational Soundness

Martín Abadi

Security and simplification

Security is hard to define precisely.

"Security is fractal." (Butler Lampson)

Whatever our model, some attacker will probably ensure that it is incomplete or flawed.

Simplistic models are perhaps inevitable and dangerous, but they can be useful as

- metaphors,
- heuristics,
- abstractions.

Nevertheless, it is both satisfying and useful to justify simplifications when possible.

This lecture

Some background and motivation

A first computational justification of formal cryptography

Some variants and extensions (briefly)

This lecture (cont.)

Some variants and extensions (briefly):

- Secret sharing
- Exponentiation
- Weak secrets
- Adaptive adversaries

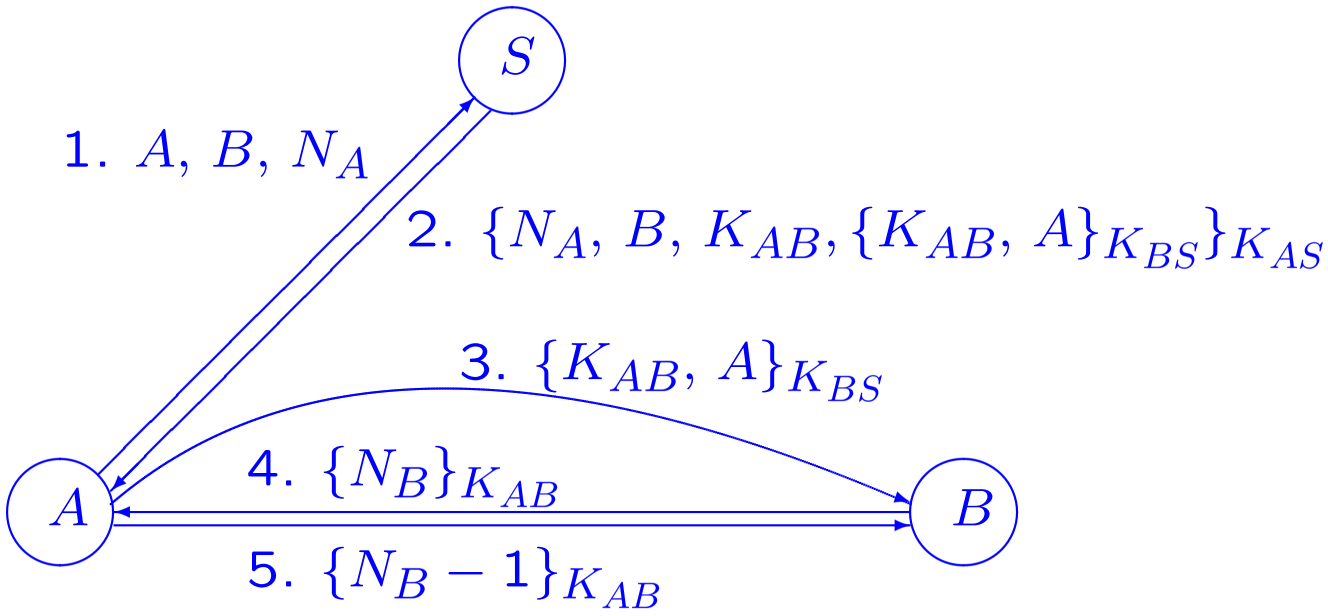
and

- XML access control
- Group key exchange
- Encrypted key exchange
- Multicast security

Not a complete survey!

Background and motivation

The Needham-Schroeder shared-key protocol



$\{X\}_K$: X encrypted with the shared key K
using a symmetric (shared-key) cryptosystem

N_A, N_B: nonces

K_{AB}: a session key for A and B,
for encrypting subsequent communications

A typical problem

Long after a run, an attacker may

– discover K_{AB},

– replay $\{K_{AB}, A\}_{K_{BS}}$ to B,

– conduct a handshake with B,

– send arbitrary data to B under K_{AB},
impersonating A. (Denning & Sacco)

Some observations

Most security protocols have subtleties and flaws.

Many of these have to do with cryptography.

Many of these don't have to do with the details of cryptography.

For design, implementation, and analysis, a fairly abstract view of cryptography is often practical.

The origin of formal cryptology

An attacker

- knows some data in advance,
- may intercept messages on the public network,
- may inject messages into the public network

What messages can the attacker produce?

- The attacker can be non-deterministic.
- But it cannot get too lucky in guessing keys.

A simple solution:

- Key generation always yields a fresh key.
- Keys are not bitstrings.
- Cryptographic operations are treated symbolically.

The formal view: strengths

Some simple, effective intuitions.

Easy human reasoning.

Logical methods and foundations
(in modal logics, process algebras, . . .).

Techniques and tools for automated reasoning:

- Decision procedures (since Dolev & Yao).
- Model-checkers (FDR, Murϕ, . . .).
- Theorem provers (Coq, Isabelle, ProVerif, . . .).

The formal view: blind spots

Relevance to instantiations of protocols, where the cryptographic functions are not black boxes.

Details of assumptions, particularly on cryptography.

Probabilities and complexities.

Attacks that are assumed impossible, rather than actually proved impossible (e.g., guessing keys).

Quantitative design and analysis.

Some early concerns

What does formal analysis prove, if anything?

Even more dangerous to the protocol designer is the possibility that cryptographic operations may satisfy simple algebraic identities of which he is unaware [...] Such properties might allow an attacker to make more powerful inferences than the model reflects, defeating some protocols. However, once such identities are discovered, new algebras representing the cryptosystem can be formulated to represent these more powerful inferences.

Michael Merritt, *Cryptographic Protocols*, 1983

However, formal analysis was often more effective and useful than the alternatives.

The computational 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.

Comparison

The computational approach is

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

Towards a convergence (mid 1990s)

Computational cryptography became more relevant to the protocols that were the subject of formal analysis.

- See in particular the papers by Bellare and Rogaway on authentication and key distribution (1993–1998).

There were new efforts and successes in using formal methods for verification (not simply for finding bugs).

- With increasingly sophisticated tool support.
- Even for infinite-state systems.

The field as a whole grew.

Towards a convergence (cont.)

Concerted efforts to relate formal and computational models probably started around 1996 or 1997.

It was immediately clear that there is room for diverse approaches, and that the problem is much bigger than what a single paper could treat.

This workshop goes well beyond what we all envisioned at the start.

Computational soundness:

the approach and a result

Computational soundness

(starting in joint work with Phil Rogaway)

Soundness property (desired):

If a security property can be proved formally,
then it holds in the computational model.

The formal proof will

– not mention probabilities and complexities,

– consider attacks only in the formal model,

– establish an all-or-nothing statement.

Soundness means that the statement is true for all computationally reasonable attacks with high probability.

A simple case

Suppose that

$$(\text{new } K_1, \ldots, K_n)\,\text{send}\; M \approx (\text{new } K_1, \ldots, K_n)\,\text{send}\; N$$

in a process calculus (e.g., the spi calculus),

where $\approx$ is observational equivalence.

Can we justify this computationally?

For example, are M and N indistinguishable in polynomial time, provided that $K_1, \ldots, K_n$ are generated at random?

A term language

The set of expressions $\mathbf{Exp}$:

$M, N ::=$	expressions
i	bits (for $i \in \mathbf{Bool}$)
K	keys (for $K \in \mathbf{Keys}$)
(M, N)	pairs
$\{M\}_K$	symmetric encryptions

Patterns

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

$$
\begin{array}{lcll}
pattern(i) & = & i & (\text{for } i \in \mathbf{Bool}) \\
pattern(K) & = & K & (\text{for } K \in \mathbf{Keys}) \\
pattern((N_1, N_2)) & = & (pattern(N_1), pattern(N_2)) & \\
pattern(\{N\}_K) & = & \begin{cases} \{pattern(N)\}_K & \text{if } M \vdash K \\ \Box & \text{otherwise} \end{cases} &
\end{array}
$$

where

- $M \vdash M$.
- If $M \vdash (N_1, N_2)$ then $M \vdash N_1$ and $M \vdash N_2$.
- If $M \vdash \{N\}_K$ and $M \vdash K$ then $M \vdash N$.

Example: $pattern((\{\{0\}_{K_1}\}_{K_2}, K_2)) = (\{\Box\}_{K_2}, K_2)$

Equivalence ($\cong$)

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

Formally, two expressions are *equivalent* if they yield the same pattern (up to renaming).

Examples:

$0 \cong 0$

$0 \ncong 1$

$\{0\}_K \cong \{1\}_K$

$(K, \{0\}_K) \ncong (K, \{1\}_K)$

$(K, \{\{0\}_{K'}\}_K) \cong (K, \{\{1\}_{K'}\}_K)$

$(\{0\}_K, \{0\}_K) \cong (\{0\}_K, \{1\}_K)$

Finer points

Key equalities are concealed:

$$(\{0\}_K, \{1\}_K) \cong (\{0\}_K, \{1\}_{K'})$$

This is not standard in computational cryptography, but can be accommodated.

Finer points (cont.)

Ciphertexts do not reveal the size of plaintexts:

$$\{0\}_K \cong \{(((1,1),(1,1)),((1,1),(1,1)))\}_K$$

Encryption cycles do not leak data:

$$\{0\}_K \cong \{K\}_K$$

Both of these are problematic computationally.

Finer points (cont.)

Such discrepancies are not all bad.

In some cases they correspond to legitimate variations in definitions.

Research on computational soundness encouraged more research in cryptography, on key-dependent encryption.

One alternative formal definition

(and there have been several others!)

We redefine $\cong$ to rely on a new definition of patterns:

$$
\begin{array}{lcll}
pattern'(i) & = & i & (\text{for } i \in \mathbf{Bool}) \\
pattern'(K) & = & K & (\text{for } K \in \mathbf{Keys}) \\
pattern'((N_1, N_2)) & = & (pattern'(N_1), pattern'(N_2)) & \\
pattern'(\{N\}_K) & = & \begin{cases} \{pattern'(N)\}_K & \text{if } M \vdash K \\ \{struct(N)\}_K & \text{otherwise} \end{cases} &
\end{array}
$$

where K_0 is a fresh, fixed key and

$$
\begin{array}{lcll}
struct(i) & = & \text{bit} & (\text{for } i \in \mathbf{Bool}) \\
struct(K) & = & K_0 & (\text{for } K \in \mathbf{Keys}) \\
struct((N_1, N_2)) & = & (struct(N_1), struct(N_2)) & \\
struct(\{N\}_K) & = & \{struct(N)\}_{K_0} &
\end{array}
$$

The computational view

An encryption scheme consists of algorithms:

$$\begin{aligned} \mathcal{K} &: \text{Parameter} \times \text{Coins} \to \text{Key} \\ \mathcal{E} &: \text{Key} \times \text{String} \times \text{Coins} \to \text{Ciphertext} \\ \mathcal{D} &: \text{Key} \times \text{String} \to \text{Plaintext} \end{aligned}$$

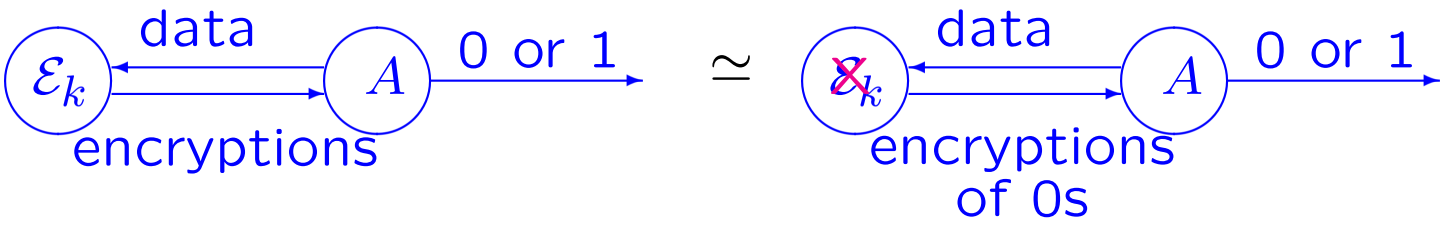
where Parameter $= 1^*$ (numbers in unary),
Key, Plaintext, Ciphertext $\subseteq$ String.

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

- if $m \in$ Plaintext then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$,
- if $m \notin$ Plaintext then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = \mathbf{0}$
  where $\mathbf{0} \in$ Plaintext (a fixed string).

Secure encryption

In a standard definition, an encryption scheme is secure if it is hard to distinguish a real encryption oracle from a fake one.



Encryption scheme Π is *secure* if,
for every probabilistic polynomial-time adversary A,

$$\mathsf{Adv}_{\Pi[\eta]}(A) \triangleq \Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) : A^{\mathcal{E}_k(\cdot)}(\eta) = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K}(\eta) : A^{\mathcal{E}_k(0^{|\cdot|})}(\eta) = 1\right]$$

is negligible (that is, "very small").

Semantics

We map:

$M \in \mathbf{Exp}$
$\eta \in$ Parameter
Π an encryption scheme
$\mapsto$
a distribution on bitstrings $[\![M]\!]_{\Pi[\eta]}$

and thereby

$$M \in \mathbf{Exp} \ \mapsto \ \text{an ensemble } [\![M]\!]_{\Pi}$$

(An *ensemble* is a collection of distributions on strings, one for each value of the security parameter.)

Semantics (cont.)

First, we map each key symbol K that occurs in M to a bitstring $\tau(K)$, using $\mathcal{K}(\eta)$.

Then we set (roughly):

$$
\begin{aligned}
[\![0]\!]_{\Pi[\eta]} &= 0 \\
[\![1]\!]_{\Pi[\eta]} &= 1 \\
[\![K]\!]_{\Pi[\eta]} &= \tau(K) \\
[\![(M,N)]\!]_{\Pi[\eta]} &= ([\![M]\!]_{\Pi[\eta]}, [\![N]\!]_{\Pi[\eta]}) \\
[\![\{M\}_K]\!]_{\Pi[\eta]} &= \mathcal{E}_{\tau(K)}([\![M]\!]_{\Pi[\eta]})
\end{aligned}
$$

We assume that lengths depend only on structure.

Indistinguishable probability ensembles

D and D' are *indistinguishable* ($D \approx D'$) if,

for every polynomial-time adversary A,

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

is negligible.

A computational soundness theorem

Let M and N be expressions without encryption cycles, and Π be a secure encryption scheme.

If $M \cong N$ then $[\![M]\!]_\Pi \approx [\![N]\!]_\Pi$.

There are many variants of this result.

Converses also hold (see Micciancio and Warinschi).

Closing the loop

(with M. Baudet and B. Warinschi)

We carefully pick a particular equational theory of messages and corresponding cryptographic assumptions.

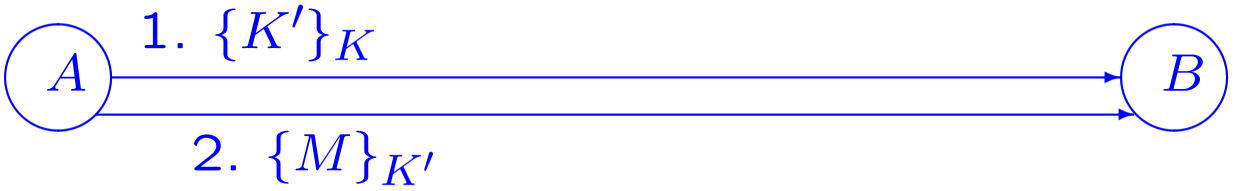
Suppose that

$$(\text{new } K_1, \ldots, K_n)\,\text{send}\, M \approx (\text{new } K_1, \ldots, K_n)\,\text{send}\, N$$

where $\approx$ is observational equivalence.

Then the interpretations of these messages are equivalent computationally.

Applications: A first, small example



- A and B share a long-term symmetric key K.
- A creates a new symmetric key K'.
- A sends K' to B encrypted under K.
- A and B can communicate under K'.

Formally, the transcript $(\{K'\}_K, \{M\}_{K'})$ is equivalent to any variant with a different payload $(\{K'\}_K, \{N\}_{K'})$.

A computational guarantee follows.

Interpretation / Benefits

Formal reasoning is sound.

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

We obtain simple intuitions and a proof method for the computational model.

Further developments

(the work of many people)

Other cryptographic primitives (e.g., hashing)

Applications

Adaptive attackers

Further developments (cont.)

Active attackers

Soundness results for particular proof methods:

- tools such as Casrul and ProVerif
- some type systems

Hybrid models

Direct computational proofs

Exponentiation and group key exchange

An example

(Bresson, Lakhnech, Mazaré, & Warinschi)

Three parties A_1, A_2, A_3 have secrets X_1, X_2, and X_3.

They send g^{X_1}, g^{X_2}, and g^{X_3}, respectively.

Then they send $g^{X_1X_2}$, $g^{X_2X_3}$, and $g^{X_3X_1}$.

They agree on the secret $h(g^{X_1X_2X_3})$.

Then A_1 generates a new key K and distributes it encrypted under $h(g^{X_1X_2X_3})$.

Finally, A_2 sends a secret expression F under K.

Is this secure?

An approach

(Bresson, Lakhnech, Mazaré, & Warinschi)

Extend the syntax of expressions with terms of the form g^{poly} and $\{M\}_{h(g^{\mathsf{poly}})}$.

Define formal equivalence.

Prove computational soundness under DDH
(via a generalization of DDH).

An approach (cont.)

Write the expression $E(F)$ that represents the transcript of the protocol:

$$(g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_3X_1}, \{K\}_{h(g^{X_1X_2X_3})}, \{F\}_K)$$

Compare this with $E(0)$.

They are formally equivalent!

So they are also computationally equivalent,
via computational soundness.

Adaptive security and multicast

Adaptive security

(Micciancio & Panjwani)

So far, we were concerned with the equivalence of two expressions M and N.

More generally, we may consider sequences of expressions ($M_0, M_1, \ldots$ and $N_0, N_1, \ldots$).

- The adversary produces the sequences.
- They are evaluated under a fixed key assignment τ.

This is a step towards general active adversaries.

Adaptive security (cont.)

The sequences may be produced adaptively:
each expression may depend on previous interactions.

We need to assume that these expressions do not contain cycles and do not reveal previously secret keys.

Under this assumption, the soundness theorem says:

- The adversary picks two equivalent sequences.
- Then the adversary cannot distinguish the bitstrings that correspond to the two sequences, computationally.

An extension applies when some expressions can represent pseudo-random generators ($G_i(K)$).

Motivation (partial)

Micciancio and Panjwani reviewed a dozen multicast protocols from the literature.

- 9 of these were published with only symbolic proofs of security.

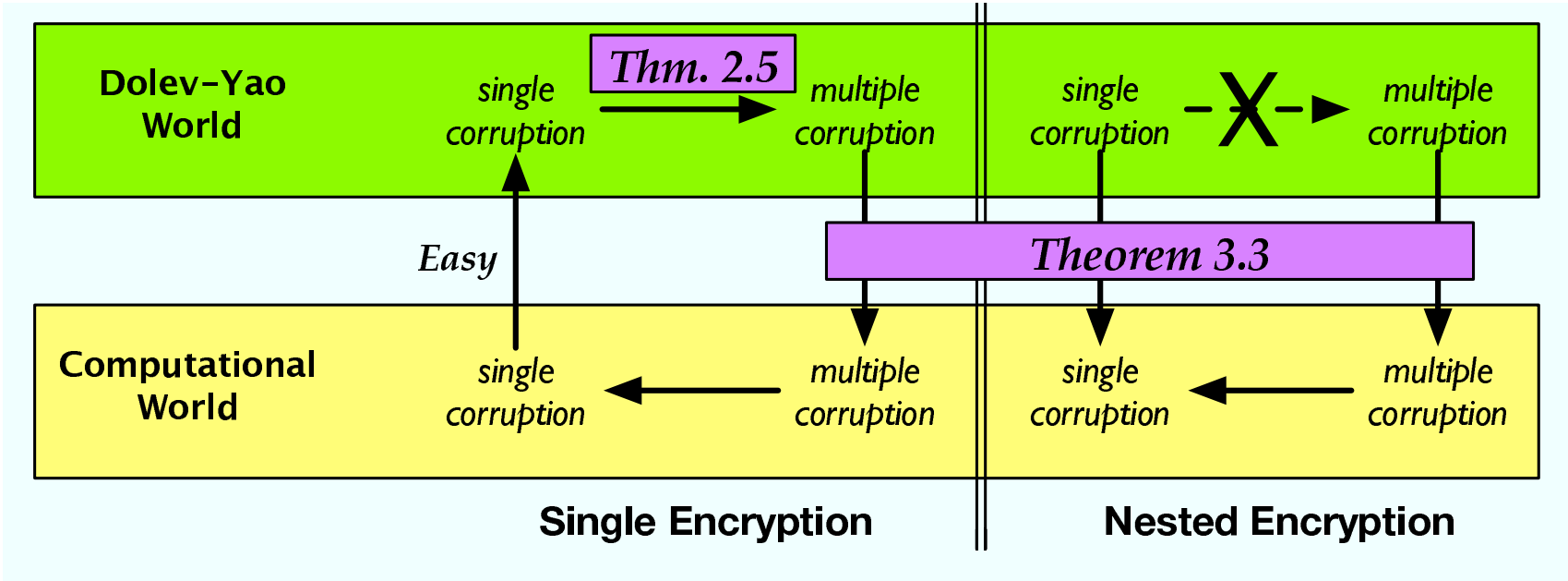
- 7 of these have weaknesses.

The formal definition of the adaptive model and its computational soundness allow them to do better.

They also play a role in proofs of lower bounds.

One set of results:

corrupting one vs. corrupting many



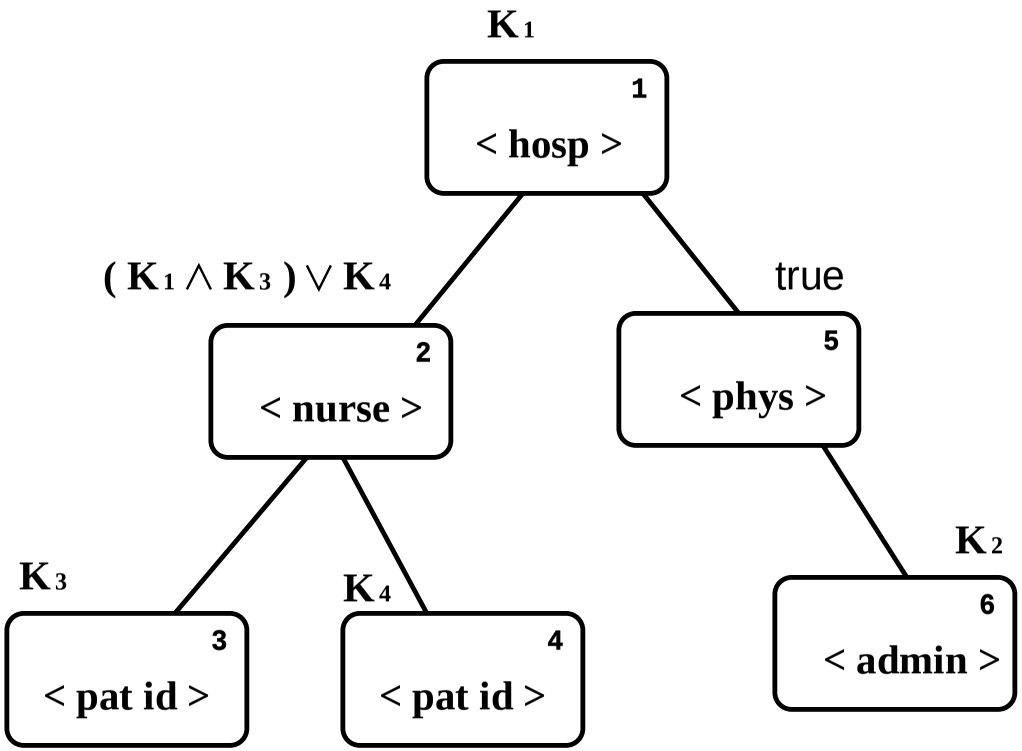
Results are proved first in the Dolev-Yao model then interpreted in the computational model by proving soundness of the Dolev-Yao definitions (Thm. 3.3).

Cryptographic access control for XML

Cryptographic access control for XML

(Miklau & Suciu)

A protection is an XML tree in which nodes are guarded by positive boolean formulas over a set of symbols $K_1, K_2, \ldots$ that stand for cryptographic keys.



Analysis of cryptographic access control

(with Bogdan Warinschi)

Some results:

- We found some flaws (related to weak secrets).
- We defined assumptions and goals computationally.
- We proved a soundness theorem (without weak secrets).

Weak secrets

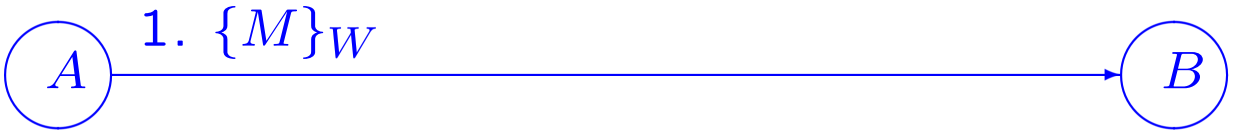
Weak secrets

Secrets drawn from a small space are weak secrets (e.g., most passwords).

They are a little easier to protect but much harder to use.

Therefore, the design and analysis of systems that rely on weak secrets is particularly delicate.

Simple encrypted communication



- W is a secret shared by A and B.
- M is a message (with timestamps, etc.).
- $\{M\}_W$ is the message encrypted under W.

An attacker that guesses W can confirm the guess, by:

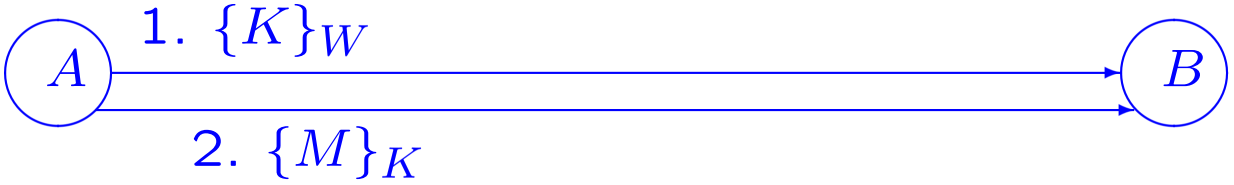
– eavesdropping and decrypting $\{M\}_W$,

– seeing if the result makes sense.

The attacker can guess many times without detection.

$\Rightarrow$ This protocol is not adequate when W is weak.

Encrypted communication (Take 2)



- A creates a random, strong, symmetric key K.
- A sends K to B encrypted under W.

The previous attack no longer works.

But a variant of the attack still works.

An attacker who intercepts $\{K\}_W$ and $\{M\}_K$ can confirm a guess of W.

(Cf. Kerberos.)

Encrypted communication (Take 3)

There are several clever protocols for "encrypted key exchange".

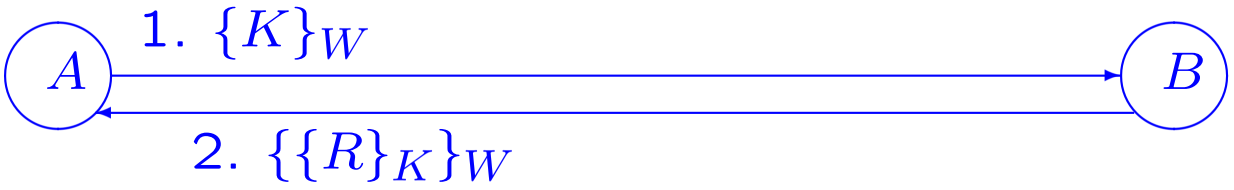
These protocols allow the exchange of a key despite the weakness of a password.

But:

- They are rather subtle and complex.
- Some of them have flaws.

EKE—sketch of one basic version

(Bellovin & Merritt)



- A generates a public-key pair and sends the public key K encrypted under the password W to B.
- B obtains K, generates a fresh secret R, and sends R encrypted under K then W to A.
- Afterwards A and B both know R.

Formal reasoning with weak secrets

At least since Gong's "Verifiable-Text Attacks in Cryptographic Protocols" (1990) there have been several attempts to systematize the proper use of weak secrets.

Some of these attempts have led to formal definitions and to their use in algorithms and tools.

These are based on a symbolic treatment of keys and cryptographic operations.

Weak secrets and the soundness problem

(with M. Baudet and B. Warinschi)

The formal definitions look reasonable enough,
but are they?

There are some differences across them.

When the formal definitions indicate that there is an attack, there often is.

Do they in fact detect all attacks against weak secrets?

A term language, extended

The set of expressions $\mathbf{Exp}$:

$M, N ::=$		expressions
	i	bit (for $i \in \mathbf{Bool}$)
	(M, N)	pairs
	K	keys (symmetric and asymmetric)
	$\{M\}_K$	encryption (symmetric and asymmetric)
	W	a distinct, single weak key
	$\{M\}_W$	encryption under the weak key

Message production, formally

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

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

If $M \in S$ then $S \vdash M$.

If $S \vdash (M, N)$ then $S \vdash M$ and $S \vdash N$.

If $S \vdash \{M\}_K$ and $S \vdash K$ then $S \vdash M$
(for symmetric encryption).

If $S \vdash \{M\}_K$ and $S \vdash K'$ then $S \vdash M$
where K' is the inverse of K
(for asymmetric encryption).

If $S \vdash \{M\}_W$ then $S \vdash M$.

$\vdots$

Patterns, extended

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

$$
\begin{array}{lcll}
pattern(i) & = & i & (\text{for } i \in \mathbf{Bool}) \\
pattern((N_1, N_2)) & = & (pattern(N_1), pattern(N_2)) & \\
pattern(K) & = & K & \\
pattern(\{N\}_K) & = & \begin{cases} \{pattern(N)\}_K & \text{if } M \vdash \text{inverse of } K \\ \Box & \text{otherwise} \end{cases} & \\
pattern(W) & = & W & \\
pattern(\{N\}_W) & = & \{pattern(N)\}_W &
\end{array}
$$

Hiding in expressions

Consider an expression M, and calculate its pattern M'.

Suppose that, in M', all occurrences of W are in subexpressions of the form $\{N\}_W$ where N is a key that does not appear elsewhere or a $\Box$.

Then we say that M hides W.

Examples

$\{0\}_W$ does not hide W.

$\{K\}_W$ hides W.

$(\{K\}_W, \{K\}_W)$ hides W.

$(\{K\}_W, K)$ does not hide W.

$(\{K\}_W, \{K\}_{K'})$ hides W.

$(\{K\}_W, \{\{R\}_K\}_W)$ hides W, for K a public key
(as in EKE).

Semantics, extended

We map:

$M \in \mathbf{Exp}$
$\eta \in$ Parameter
Π an encryption suite
D a dictionary

$\mapsto$ a distribution on bitstrings $[\![M]\!]_{\Pi[\eta]}\mathsf{D}$

and thereby

$M \in \mathbf{Exp}$
Π an encryption suite
D a dictionary

$\mapsto$ an ensemble $[\![M]\!]_{\Pi}\mathsf{D}$

(A *dictionary* is a collection of possible values for weak secrets.)

Semantics, extended (cont.)

First, we map each key symbol K that occurs in M to a bitstring $\tau(K)$, using a key generator $\mathcal{K}(\eta)$, and W to a given w picked from a dictionary D.

Then we set (roughly):

$$
\begin{aligned}
[\![0]\!]_{\Pi[\eta]}\mathsf{D} &= 0 \\
[\![1]\!]_{\Pi[\eta]}\mathsf{D} &= 1 \\
[\![(M,N)]\!]_{\Pi[\eta]}\mathsf{D} &= ([\![M]\!]_{\Pi[\eta]}\mathsf{D}, [\![N]\!]_{\Pi[\eta]}\mathsf{D}) \\
[\![K]\!]_{\Pi[\eta]}\mathsf{D} &= \tau(K) \\
[\![\{M\}_K]\!]_{\Pi[\eta]}\mathsf{D} &= \mathcal{E}_{\tau(K)}([\![M]\!]_{\Pi[\eta]}\mathsf{D}) \\
[\![W]\!]_{\Pi[\eta]}\mathsf{D} &= w \\
[\![\{M\}_W]\!]_{\Pi[\eta]}\mathsf{D} &= \mathcal{E}_w([\![M]\!]_{\Pi[\eta]}\mathsf{D})
\end{aligned}
$$

where we still need to describe the functions $\mathcal{K}$, $\mathcal{E}$, ...

What about encryption under a weak secret?

There is not even a well established notion for this!

For the present purposes, we do not need this encryption to hide the underlying plaintext.

We do want it to hide the key.

It need not be probabilistic, and in fact it is probably better if it is deterministic (so $(\{K\}_W, \{K\}_W)$ hides W).

Password-hiding password-based encryption

Let $(Dist_\eta)_\eta$ be an ensemble.

A password-based encryption function $\mathcal{E}$ *securely encrypts* $(Dist_\eta)_\eta$ using passwords from D,
if for any probabilistic polynomial-time adversary A, and any singleton subdictionaries D^0 and D^1 of D,

$$\Pr\left[w_0 \xleftarrow{R} \mathsf{D}^0_\eta, w_1 \xleftarrow{R} \mathsf{D}^1_\eta : \ A^{\mathcal{E}_{w_1}(Dist_\eta)}(\eta, w_0, w_1) = 1\right] -$$
$$\Pr\left[w_0 \xleftarrow{R} \mathsf{D}^0_\eta, w_1 \xleftarrow{R} \mathsf{D}^1_\eta : \ A^{\mathcal{E}_{w_0}(Dist_\eta)}(\eta, w_0, w_1) = 1\right]$$

is negligible.

(A singleton dictionary is one with one value of the weak secret for each value of η.)

Theorem (Hiding passwords)

Let M be an expression without encryption cycles, and Π be a secure encryption scheme, with a password-based encryption scheme that securely encrypts keys and ciphertexts.

If M hides passwords in D formally,
then for all singleton dictionaries D^0 and D^1,
$[\![M]\!]_\Pi \mathsf{D}^0 \approx [\![M]\!]_\Pi \mathsf{D}^1$.

Conclusion

Outlook

It is now clear that formal proofs (even automated ones) can be interpreted computationally.

The work is not finished.

Computational proofs are progressing too.