Weak Secrets and Computational Soundness

Martín Abadi

University of California

at Santa Cruz

Based on pieces of joint work with

Bruno Blanchet, Cédric Fournet,

Phillip Rogaway, and Bogdan Warinschi

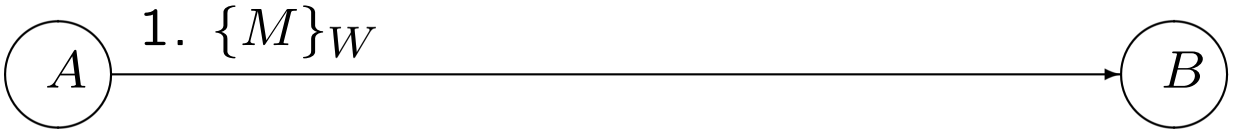
Secrets in security

Secrets play two roles in security.

- A lot of security policies are about protecting secrets, big and small.
- Many security mechanisms rely on secrets, from one-time pads and RSA keys to passwords.

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.

Simple encrypted communication



- A and B are two computers.
- 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 attack

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

– eavesdropping,

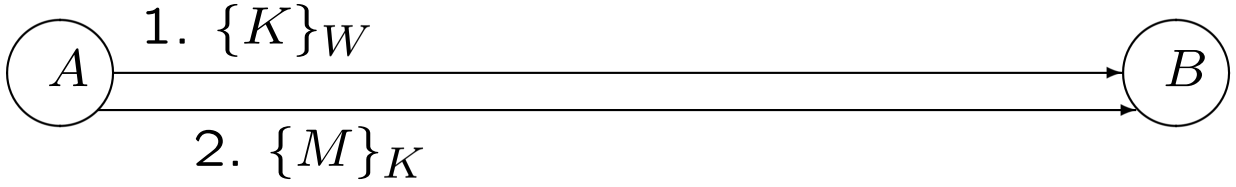
– decrypting $\{M\}_W$ and

– 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)

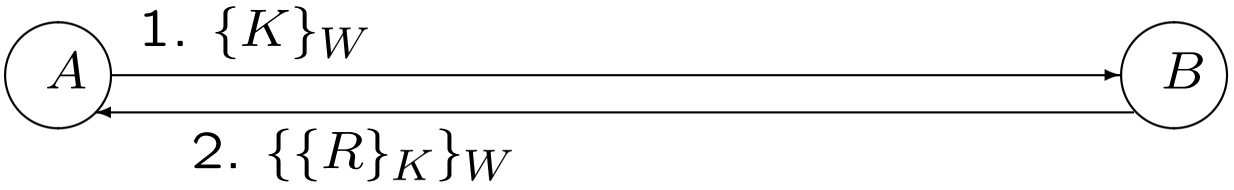
There are several clever protocols for "encrypted key exchange" (going back to Gong, Lomas, Needham, and Saltzer; Bellovin and Merritt).

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)



- 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.

Assuming that K looks random, an attacker does not get much.

EKE variants and alternatives

There are by now several variants of EKE,

- with different cryptosystems,
- so that B does not keep the password in clear,

  ⋮

There are also quite a few alternative protocols (e.g., SRP), and some proofs.

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 (see Lowe, Corin and Etalle, Delaune and Jacquemard, . . .).

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

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 $S \vdash M$:

$S \vdash 0$ $\quad$ $S \vdash 1$

$\vdots$

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

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

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

If $S \vdash M$ and $S \vdash K$ then $S \vdash \{M\}_K$.

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

$\vdots$

Message production, formally (cont.)

Sometimes an expression can be derived in more than one way, e.g., $\{0\}_W, W \vdash \{0\}_W$.

This coincidence indicates that a guess of W can be verified by someone who sees $\{0\}_W$.

Weak secrets in the applied pi calculus

(joint work with B. Blanchet and C. Fournet, in progress)

A secrecy property can be formulated as an observational equivalence between two versions of a process ($P[W/X]$ and $P[W'/X]$).

The protection of a weak secret against off-line attacks can also be formulated as a static equivalence (between frames φ and $\nu W.\varphi$).

Reasoning with weak secrets in ProVerif

ProVerif is a proof tool developed by B. Blanchet for the applied pi calculus.

We represent two versions $P[W/X]$ and $P[W'/X]$ of a process $P[X]$ by a single expression $P[\mathsf{choice}(W, W')/X]$.

This can be given a direct semantics, e.g.,

$$\overline{\mathsf{choice}(a_1, a_2)}\langle M\rangle.Q \mid \mathsf{choice}(a_1, a_2)(x).P \rightarrow Q \mid P\{M/x\}$$

This can also be translated to Horn clauses that model what the attacker observes at each step, and whether the attacker can see any difference between the two versions.

The formal view: strengths

(in general)

Logical methods and foundations

(in modal logics, process algebras, . . .).

Some simple, effective intuitions.

Easy human reasoning.

Techniques and tools for automated reasoning:

- decision procedures,
- model checkers,
- theorem provers.

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 strong keys).

Quantitative design and analysis
(e.g., recommending key strengths).

Weak secrets and the soundness problem

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?

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.

(See Yao, Blum, Goldwasser, Goldreich, Micali, Wigderson, Bellare, Rogaway, . . .)

The computational approach is

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

Bridging the gap

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.

Initial steps

One of the first steps in bridging the gap was a study of symmetric encryption with respect to passive attackers (joint work with Phil Rogaway).

There has been much further work in the last 4 years, by Jürjens; Micciancio and Warinschi; Laud; Horvitz and Gligor; Adao, Bana, and Scedrov; Herzog; Backes, Pfitzmann, and Waidner; Mitchell, Scedrov, et al.; Canetti; . . . .

Another step

(joint work with B. Warinschi, in progress)

We attempt to justify formal treatments of weak secrets, computationally.

We reuse prior work on computational soundness, and extend it.

A formal view of encryption

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

$M, N ::=$	expressions
0	bits
1	
(M, N)	pairs
K	ordinary keys
W	a weak key
$\{M\}_K$	encryptions
$\langle M \rangle_W$	

We can allow both symmetric and asymmetric encryption.

The notation $\langle M \rangle_W$ is meant to suggest that encryption may work differently depending on the type of key.

Patterns

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

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

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

Example with symmetric keys:

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

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.

Two expressions are equivalent *up to renaming* if they are equivalent after renaming of keys.

Examples

$0 \cong 0$

$0 \ncong 1$

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

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

$\{0\}_W \ncong \{1\}_W$

Further examples (finer points)

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

Encryption cycles do not leak data.

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

Ciphertexts do not reveal the size of plaintexts.

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

Plaintext equalities are concealed.

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

Key equalities are concealed.

All these have been subject of variations and refinements!

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 $\langle N \rangle_W$ where N is a key that does not appear elsewhere or a $\Box$.

Then we say that M hides W.

Examples

$\langle 0 \rangle_W$ does not hide W.

$\langle K \rangle_W$ hides W.

$(\langle K \rangle_W, \langle K \rangle_W)$ hides W.

$(\langle K \rangle_W, K)$ does not hide W.

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

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

$(\langle K \rangle_W, \{K'\}_K)$ does not hide W.

Semantics (idea)

We map:

$$\begin{array}{l} M \in \mathbf{Exp} \\ \eta \in \text{Parameter} \end{array} \quad \mapsto \quad \begin{array}{l} \text{a distribution on} \\ \text{bitstrings } [\![M]\!]_{\Pi[\eta]} \end{array}$$

and thereby

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

Semantics (idea, 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.

Then we set (roughly):

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

where we still need to define the function(s) $\mathcal{E}$...

Computational set-up
(for symmetric encryption)

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: a standard definition

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme,

let $\eta \in$ Parameter be a security parameter.

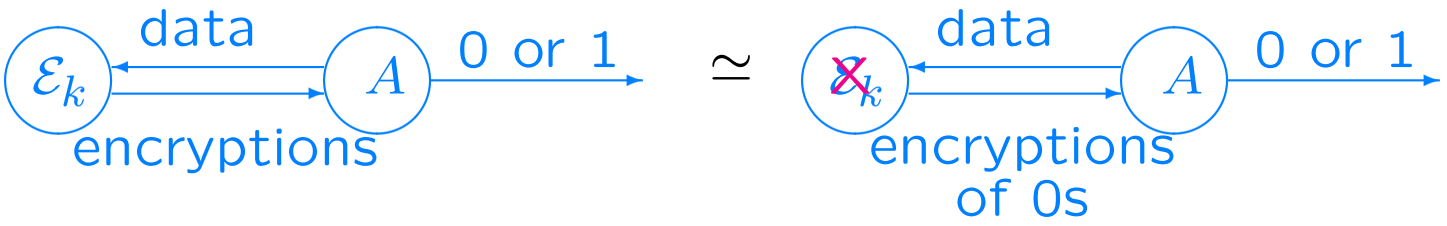
Let $A^{f(\cdot)}$ be A with access to oracle f.

Encryption scheme Π is secure if,

for every 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, no adversary can tell a true encryption oracle $\mathcal{E}_k(\cdot)$ from the fake one $\mathcal{E}_k(0^{|\cdot|})$.



(Goldwasser & Micali, Bellare et al.)

Varieties of secure encryption

According to the standard definition, does a secure encryption scheme conceal:

1) message equalities? Yes.

2) key equalities? Not necessarily.

3) message lengths? Not necessarily.

Fortunately, there are alternative definitions, in the style,

- with the missing properties (2) and (3),
- implementable with standard constructions.

Encryption cycles (e.g., $\{K\}_K$) remain a problem, and we rule them out.

Secure encryption: a variant

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme,

let $\eta \in$ Parameter be a security parameter.

Encryption scheme Π is secure if,

for every polynomial-time adversary A,

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

is negligible (that is, very small).

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 $(\langle K \rangle_W, \langle K \rangle_W)$ hides W).

Indistinguishable probability ensembles

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

D and D' are *indistinguishable* ($D \approx D'$) if,
for every polynomial-time adversary A,

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

is negligible.

Password-hiding password-based encryption

Let $(Dist_\eta)_\eta$ be a fixed distribution ensemble on strings.

A password-based encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ securely encrypts $(Dist_\eta)_\eta$ using passwords from Dictionary, if for any $w_0, w_1 \in$ Dictionary:

$$\left[x \xleftarrow{R} Dist_\eta : \mathcal{E}(w_0, x)\right] \approx \left[x \xleftarrow{R} Dist_\eta : \mathcal{E}(w_1, x)\right]$$

Theorem (Equivalence)

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

$$\text{If } M \cong N \text{ then } [\![M]\!]_{\Pi\{w\}} \approx [\![N]\!]_{\Pi\{w\}}.$$

That is, if M and N are formally equivalent then their meanings are computationally indistinguishable.

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 W then for all $w_0, w_1 \in$ Dictionary

$$[\![M]\!]_{\Pi\{w_0\}} \approx [\![M]\!]_{\Pi\{w_1\}}$$

Interpretation

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.

Neither abstraction nor simplicity is a substitute for getting it right.

B. Lampson

Hints for Computer System Design

But:

- Simple, abstract views of security are often useful.
- They can be right.