proving protocols safe from guessing

ernie cohen

August 19, 2002

guessing attacks

- problem: users choose weak secrets (passwords)
- adversary can alternate classic Dolev-Yao attacks with offline dictionary attacks
- example: ISO one-pass symmetric key protocol:

$$A \to S \text{: } \{Na, T\}_{k(A)}$$

  this can be attacked if

  - T or Na is recognizable (e.g., a text or timestamp)
  - the same T or Na appears in two different messages

LGSN timestamp protocol

Na1,Na2,Nb1,Nb2 are weak nonces
Ca,Cb are strong nonces
Tsa,Tsb are timestamps
Pa and Pb are weak passwords (shared with S)
PKs is the public key of S

$$A \rightarrow S: \{A, B, Na1, Na2, Ca, \{TSa\}_{Pa}\}_{PKs}$$
$$S \rightarrow B: A, B$$
$$B \rightarrow S: \{B, A, Nb1, Nb2, Cb, \{TSb\}_{Pb}\}_{PKs}$$
$$S \rightarrow A: \{Na1, Na2 \oplus Kab\}_{Pa}$$
$$S \rightarrow B: \{Nb1, Nb2 \oplus Kab\}_{Pb}$$

what is a Dolev-Yao guessing attack?

- obvious model: attack = a predicate built from published messages, D-Y operators, equality (and perhaps other predicates) that distinguishes a correct guess from some incorrect guess

- ex: $A \to S : \{Na, T\}_{k(A)}$

  - T a recognizable text:

    $$test(G) \Leftrightarrow isText(cdr(dec(\{Na, T\}_{k(A)}, G)))$$

  - same T sent twice:

    $$test(G) \Leftrightarrow cdr(dec(\{Na_0, T\}_{k(A)}, G)) = cdr(dec(\{Na_1, T\}_{k(A)}, G))$$

  - M sent by two users:

    $$test(G_0, G_1) \Leftrightarrow$$
    $$cdr(dec(\{Na_0, T\}_{k(A_0)}, G_0)) = cdr(dec(\{Na_1, T\}_{k(A_1)}, G_1))$$

a more tractable model (Gong et al, Lowe)

guessing attack = using a guess to generate a verifier (a message that is published, recognizable, or generated in two different ways)

(equivalent to previous definition, with some technical provisos)

- $\{Na, T\}_{k(A)} \xrightarrow[k(A)]{dec} \{Na, T\} \xrightarrow{cdr} \boxed{\mathsf{T}}$
- $\{Na_0, T\}_{k(A)} \xrightarrow[k(A)]{dec} \{Na_0, T\} \xrightarrow{cdr} \boxed{\mathsf{T}} \xleftarrow{cdr} \{Na_1, T\} \xleftarrow[k(A)]{dec} \{Na_1, T\}_{k(A)}$
- $\{Na_0, T\}_{k(A_0)} \xrightarrow[k(A_0)]{dec} \{Na_0, T\} \xrightarrow{cdr} \boxed{\mathsf{T}} \xleftarrow{cdr} \{Na_1, T\} \xleftarrow[k(A_1)]{dec} \{Na_1, T\}_{k(A_1)}$

important to disallow steps that undo other steps, e.g.

$\xrightarrow{guess} G \xrightarrow[X]{enc} G_X \xrightarrow[Y]{dec} \boxed{\mathsf{G}} \xleftarrow{guess}$ or $\xrightarrow{guess} G \xrightarrow[X]{cons} \{G, X\} \xrightarrow{car} \boxed{\mathsf{G}} \xleftarrow{guess}$

proving guessing attacks impotent

- a set of steps S *saturates* a set of messages M iff every step s whose inputs are a subset of $out(S) \cup M$ is either (1) inside M, or (2) undoes a step of S
- theorem: there is an effective attack on M iff every saturation of M has an M-verifier.
- goal: find a saturation of M for which it is easy to prove the absence of verifiers
- idea: structure the saturation like the TAPS secrecy invariant

TAPS - a first-order crypto verifier

- proves arbitrary first-order safety properties
- method: guess invariants, verify with resolution prover
- fast (usually $\leq$ 1 sec; quadratic growth)
- usually automatic, but user can provide hints
- n-party, repeated, recursive, certificate chains, ...
- allows arbitrary axioms about the state space, keying functions, etc.
- untyped, unlimited message size / principals / runs
- verified about 100 protocols, including all but one Isabelle example, and all but three of the Clark-Jacob survey
- has found bugs missed in published model-checking analyses

the TAPS secrecy invariant

$$pub(X) \Rightarrow ok(X)$$

where ok is the strongest state predicate such that

$$\begin{array}{rcl} prime(X) & \Rightarrow & ok(X) \\ pub(X) \wedge pub(Y) & \Rightarrow & ok(\{X, Y\}) \\ pub(X) \wedge pub(Y) & \Rightarrow & ok(X_Y) \end{array}$$

- each prime case represents a message sent by a protocol action followed by adversarial destruction
- forward chaining through the secrecy invariant (= backward search) leads only to $prime$ cases or smaller messages
- (establishing the secrecy invariant usually requries substantial authentication reasoning, to prove closure under destructors)

example

$A \rightarrow S$: $\{Na, M\}_{k(A)}$

$$
\begin{aligned}
p0(A, Na, T) &\Rightarrow prime(\{Na, T\}_{k(A)}) \\
p0(A, Na, T) \wedge pub(k(A)) &\Rightarrow prime(\{Na, T\}) \\
p0(A, Na, T) \wedge pub(k(A)) &\Rightarrow prime(Na) \\
p0(A, Na, T) \wedge pub(k(A)) &\Rightarrow prime(T)
\end{aligned}
$$

assuming all passwords are private, the last three cases go away

generating a saturation

- goal: construct a first-order saturation of the ok messages
- choose a predicate giving the messages available for the attack
  - (like a secrecy invariant, but with guessing steps)
- choose a predicate giving the core steps
  - obligation: includes any destructor step outside ok whose inputs are available and whose main input is $prime$ or the output of a core step (requires authentication proofs)
- a destructor step is in the saturation iff it is a core step
- a constructor step is in the saturation if its inputs are available and it doesn't undo a core step

example saturation

$$A \rightarrow S\text{: } \{Na, T\}_{k(A)}$$

- core steps:

$$\{Na, T\}_{k(A)} \xrightarrow[k(A)]{dec} \{Na, T\} \quad \text{where} \quad p0(A, Na, T) \land ano(k(A))$$
$$\{Na, T\} \xrightarrow{car} Na \quad \text{where} \quad p0(A, Na, T) \land ano(k(A))$$
$$\{Na, T\} \xrightarrow{cdr} T \quad \text{where} \quad p0(A, Na, T) \land ano(k(A))$$

- obligations:

$$p0(A, Na, T) \land ano(k(A)) \land \neg ok(Na) \;\Rightarrow\; \neg encp(Na) \land \neg consp(Na)$$
$$p0(A, Na, T) \land ano(k(A)) \land \neg ok(T) \;\Rightarrow\; \neg encp(T) \land \neg consp(T)$$

checking for verifiers

- $cons$ steps and symmetric encryption steps can't produce verifiers!
- an asymmetric encryption step can produce a verifier only if its output is $prime$ or the output of a core step (in each case, without an available decryption key)
- remaining possibilities are collisions between core steps
- many of the proofs require authentication reasoning (but TAPS does those automatically)
- worst case: number of proof obligations grows quadratically with the size of the core

verifier obligations example

$$
\begin{aligned}
p0 \wedge ano(k(A)) &\Rightarrow \neg okg(Na) \\
p0 \wedge ano(k(A)) &\Rightarrow \neg okg(T) \\
p0 \wedge ano(k(A)) &\Rightarrow Na \neq T \\
p0 \wedge p0' \wedge ano(k(A)) \wedge ano(k(A')) & \\
\wedge \{Na, T\}_{k(A)} \neq \{Na', T'\}_{k(A')} &\Rightarrow Na \neq Na' \\
& \quad \wedge T \neq T' \\
& \quad \wedge Na \neq T' \\
p0 \wedge p0' \wedge ano(k(A)) \wedge ano(k(A')) &\Rightarrow Na' \neq \{Na, T\} \\
p0 \wedge p0' \wedge ano(k(A)) \wedge ano(k(A')) &\Rightarrow T' \neq \{Na, T\} \\
p0 \wedge p0' \wedge ano(k(A')) &\Rightarrow Na' \neq \{Na, T\}_{k(A)} \\
p0 \wedge p0' \wedge ano(k(A')) &\Rightarrow T' \neq \{Na, T\}_{k(A)}
\end{aligned}
$$

EKE

Kab is a weak symmetric key shared between A and B
Ka is a freshly generated public key
R is a freshly generated symmetric key

$$A \rightarrow B\text{: } Ka_{Kab}$$
$$B \rightarrow A\text{: } (R_{Ka})_{Kab}$$
$$A \rightarrow B\text{: } Na_R$$
$$B \rightarrow A\text{: } \{Na, Nb\}_R$$
$$A \rightarrow B\text{: } Nb_R$$

- authentication + secrecy: under 1 second
- guessing obligations: 15 seconds (57 obligations)
- still lots of room for optimization

what about exclusive or?

- most protocols for weak secrets use Vernam encryption
- TAPS doesn't handle it yet (neither does any other verifier)
- solution: construct a (protocol- and state-specific) basis for xor
  - all encryptions produce unit length output
  - each point is an xor of atoms and encryptions
- new secrecy invariant: a message is ok iff it is a tuple of a disjoint sum of published points
- additional proof obligation: every sum of two points is a disjoint sum of points