Tamper-proof code

or

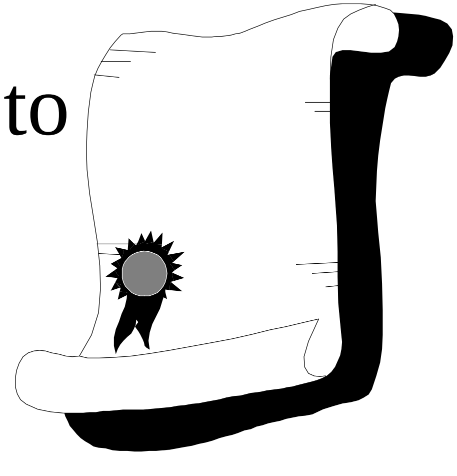
Cavemen use cryptography to

seal code, share keys,

and keep each other accountable.

Problem

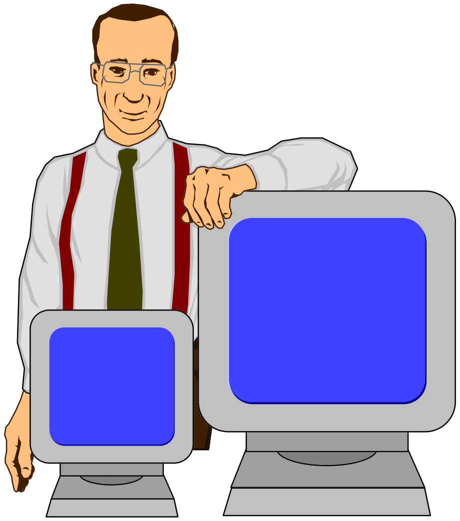
- We have an authority (a compiler instance) who is capable of certifying source code and producing vetted code.
- It needs to generate unforgeable safe certificates.
- The vetted code must be infeasible to change after certification.



Issues and non-issues.

- Do I trust my compiler ?
- Do I trust a compiler at Microsoft ?
- What if I change my mind about MS at some point in time ?

Can it be trusted ?



Rejected ideas

- Distribute source code.
  - Good: Trust is a local decision.
  - Bad: MS distributes code ? Can you compile it ?
- Vendor trusts compiler, distributes encrypted source code.
  - Good: Trust is a local decision.
  - Bad: MS trusts Joe Random's compiler ? Can you compile Word on a PC ?
- Central certifying authority.
  - Good and Bad: Trust is a centralized decision.
- Certify vetted code.

Key Idea.

- Trust is hard. Accountability is easy.
  - Keep people accountable.
  - Allow them to safely attach their names to code.
  - Allow them to revoke certifications.
  - Let everyone decide who they trust.

Solution Goals

- Certification: Attach a “seal of goodness.”
- Identification: Identify the certifying authority.
- Revocation: Allow the certifier to cope with security violations.
- Security: Make it impersonization infeasible.
- Simplicity: Keep trusted code base small.

SPIN Requirements

- Code cannot be distributed in the clear
  - individually checking all code is both impractical and undesirable.
- Disconnected operation must be possible.

Outline

- How to seal a piece of code.
- How to seal the sealer.
- How to transfer keys safely.
- What happens when a key is compromised.

Terminology

- Principle: A party to a cryptographic conversation.
- TCB: Trusted Computing Base.
- K, $K^{-1:}$ Keys.
- $\{M\}_x$:Message M encrypted with key X.
- T_A: A timestamp chosen by principle A.

Public-key symmetric cryptography

- A principal has two keys, K and K^{-1}.
- K^{-1} is secret, but K is available publicly. Neither is possible to discern from the other.
- $\{\{M\}_K\}_{K^{-1}} = \{\{M\}_{K^{-1}}\}_K = M$.

Message Digest Algorithms

- A one way hash function that computes a checksum over a bitstream.
- It is computationally infeasible to construct
  - two messages that have the same digest.
  - a message of a given prespecified digest.
- The output of digests exhibits randomness.
- MD5: 128-bit, fast digest algorithm.

How to seal a message.

- Meaning: “Principal A assures that at time T_A, message M produced the hash value H(M).”
- **Sealed Message**: A, T_A, M, $\{ H(M) \}_{K^{-1}_{A}}$.
- Operation:
  - Compute a one-way digest on M.
  - Sign the digest with A’s secret key at time T_A.
- M can be encrypted if necessary.

How to check a message for validity.

- Anyone can acquire A’s public keys.
- Suppose the recipient recevies: A, T_A, M”, { H(M) $\}_{K^{-1}_{A}}$.
- The recipient can calculate H(M”).
- The recipient can perform $\{\{ H(M) \}_{K^{-1}_{A}}\}_{K} =$ H(M).
- **Check** H(M”) =? H(M).

Key distribution

- A recipient R needs to build a database of public keys.
- A’s public key can be distributed via a private channel (floppy, cd-rom).
- R can acquire A’s key from S, whose public key she already knows.
- R can contact A, establish a secure channel and receive the key.

Protecting the certifier in storage

- No different from sealing any other code.
- This case is even cooler since the sealer can self-seal as part of its bootstrapping process.
- Reduces the window of vulnerability. (Brewer, et al.).

Protecting the certifier's execution

- Intractable problem: Make sure that the "certifier runs correctly."
  - Never decides to brand unsafe code safe.
  - Never gives out your secret keys to other parties.
- Simple solution: Create an environment in which a program can run without the possibility of being tampered with.
  - Single-user box in a sealed room off the net.
  - Hostile environment: special exec(2) call.

How to protect principals.

- Principals have a vested interest in keeping the secret key secret.
- If a key is compromised, revoke the public key of the compromised epoch, and supply clients with re-sealed software.

Assumptions about TCB

- The certification agent that checks the signature is trusted.
- The certification agent that attaches the signature is trusted.
- The compiler is trusted. There is no window of vulnerability between the compiler and CA.
- The kernel that runs the compiler is trusted to run the compiler free of tampering.

A concrete proposal

- Use MD5 for the digest.
- Principal names are arbitrary text strings, SPIN is one.
- Use a counter for time, incrementing the count at every compiler revision.
- Append the seal to the end of a COFF file. All utilities continue to work.
- Use RSA encryption for signatures. (legal snags).

Summary

- Cryptographic signatures, combined with message-digests solve the validity checking problem.
- Using a model based on accountability instead of formulating trust relationships allows us to be general.