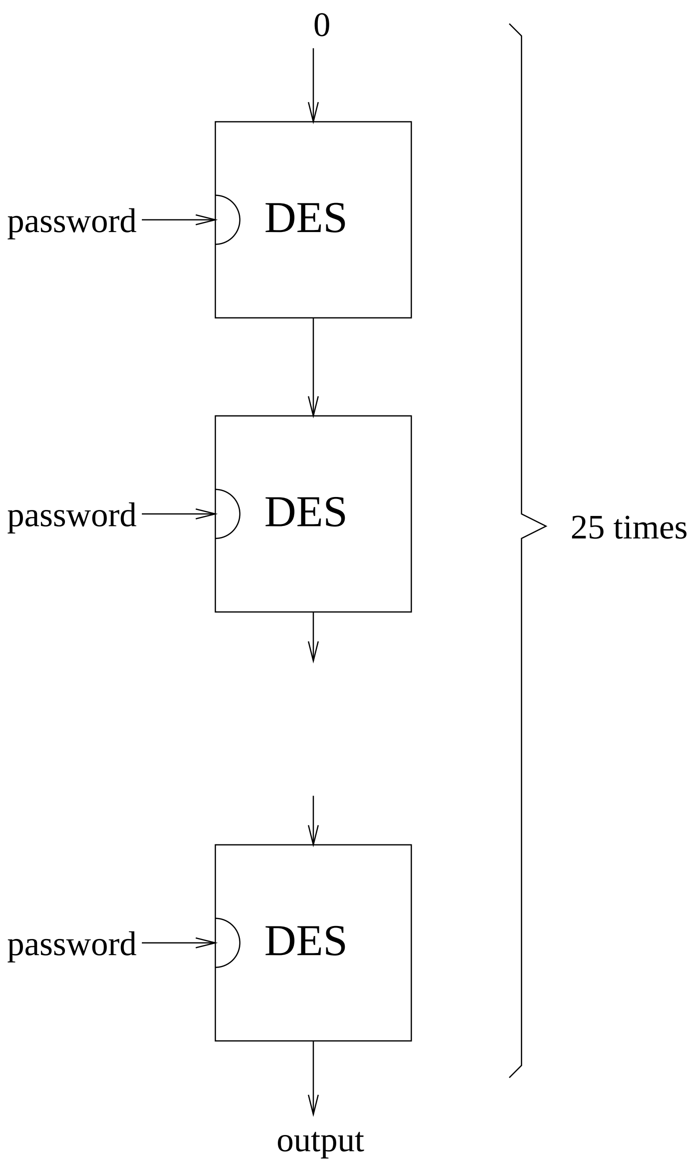
Practical proofs of security for the Unix password hash function

David Wagner and Ian Goldberg

University of California, Berkeley

The Unix hash function



Our approach

We want to prove something about the security of the Unix hash function.

We use a security reduction:

- Assume DES is secure.
- Prove that the Unix hash function is secure.

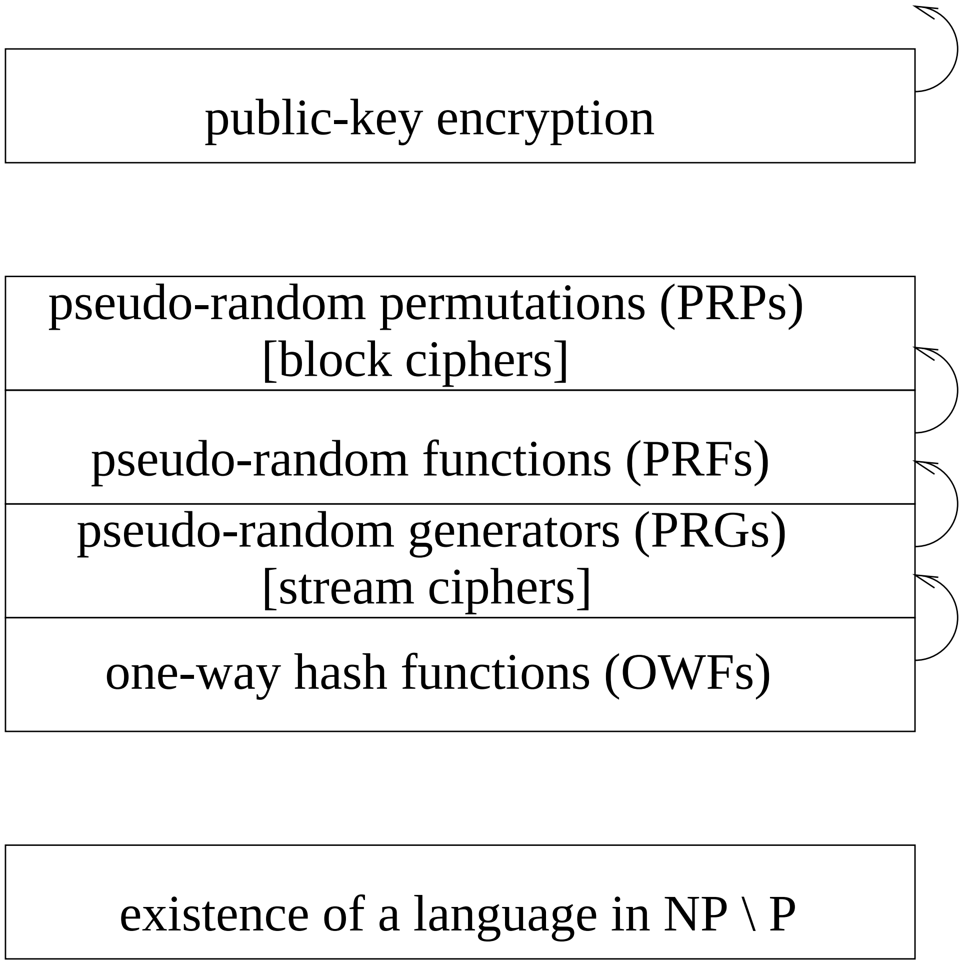
We focus on concrete security, i.e. a quantitative approach.

Classical crypto-complexity theory

Studies what complexity-theoretic assumptions you need for cryptography.
($P \neq NP$, etc.)

Tools: Asymptotic poly-time reductions, classical complexity theory.

The crypto complexity hierarchy



Concrete analysis

Criticisms of classical crypto theory:

- Its constructions are often neither fast nor secure, in practice.
  (Not all poly-time algorithms are efficient enough, and vice versa.)
- Doesn't help with real-world primitives, which are necessarily finite.
  (Asymptotic results only apply to infinite families of functions.)
- Reducing assumptions doesn't necessarily reduce the design effort.
  (Designing a secure block cipher is not much harder than designing a secure one-way function.)

But, the notion of a security reduction is very powerful.

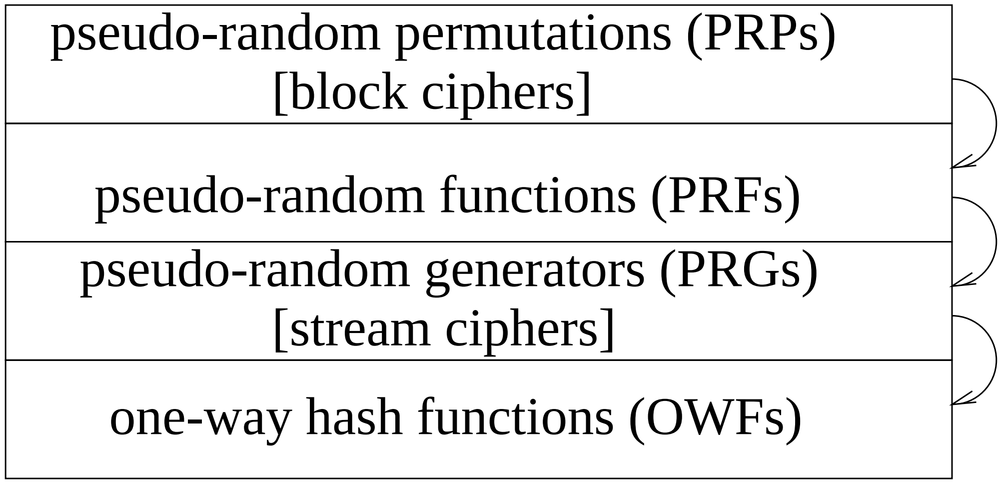
Concrete security = security reductions $-$ asymptotics $+$ practice-driven cryptographic assumptions.

Related work

Luby and Rackoff gave strong evidence that the hash construction $H'(k) = E_k(0)$ is sound.
They showed that it is asymptotically secure against poly-time adversaries if E is.

But they didn't give an analysis of its concrete security, and they only analyzed a variant of the Unix hash function where the iteration is omitted.

A roadmap of our proof



Iterated PRP $\rightarrow$ Iterated PRF $\rightarrow$ PRF $\rightarrow$ PRG $\rightarrow$ OWF

Background: definitions

An attack R-breaks a crypto primitive if the algorithm succeeds in breaking the primitive with resources specified by R.

A crypto primitive is R-secure if there is no algorithm to R-break it.

The adversary's job is to distinguish between the pseudo-random primitive and a truly-random ideal version.

(Exception: for one-way functions, a successful adversary is one which can find a pre-image.)

Formal definitions: One-way functions

Adversary A attacking a OWF $h : \mathcal{K} \to \mathcal{Y}$:

Input: $y \in \mathcal{Y}$.
Output: Either $x \in \mathcal{K}$ with $h(x) = y$, or else $\perp$.
Success prob.: $p = \Pr[A(h(k)) \neq \perp]$, where k is uniformly distributed.
Resources: (t, p) means t hash-computations and success prob. p.

Notes: (1) the adversary only has to succeed on valid hash-outputs,
(2) h is only guaranteed one-way when the input is uniformly distributed.

Formal definitions: PRF

Adversary A attacking a PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$:

Input: An oracle $f : \mathcal{X} \to \mathcal{Y}$, chosen either as a F_k or as R.
($R : \mathcal{X} \to \mathcal{Y}$ is a random function.)
Output: 0 or 1, according to whether A guesses $f = R$ or $f = F_k$.
Measure of success: $\mathrm{Adv}\, A = \left| \Pr[A^{F_k} = 1] - \Pr[A^R = 1] \right|$.
Resources: (t, q, e) means t work, q oracle queries to f, and $e = \mathrm{Adv}\, A$.

Formal definitions: PRP

Similar to PRF: a PRP is indistinguishable from a random permutation.

Formal definitions: PRG

A PRG $g : \mathcal{K} \to \mathcal{Y}$ is secure if $g(k)$ is indistinguishable from a random bitstring when $k \in \mathcal{K}$ is uniformly distributed.

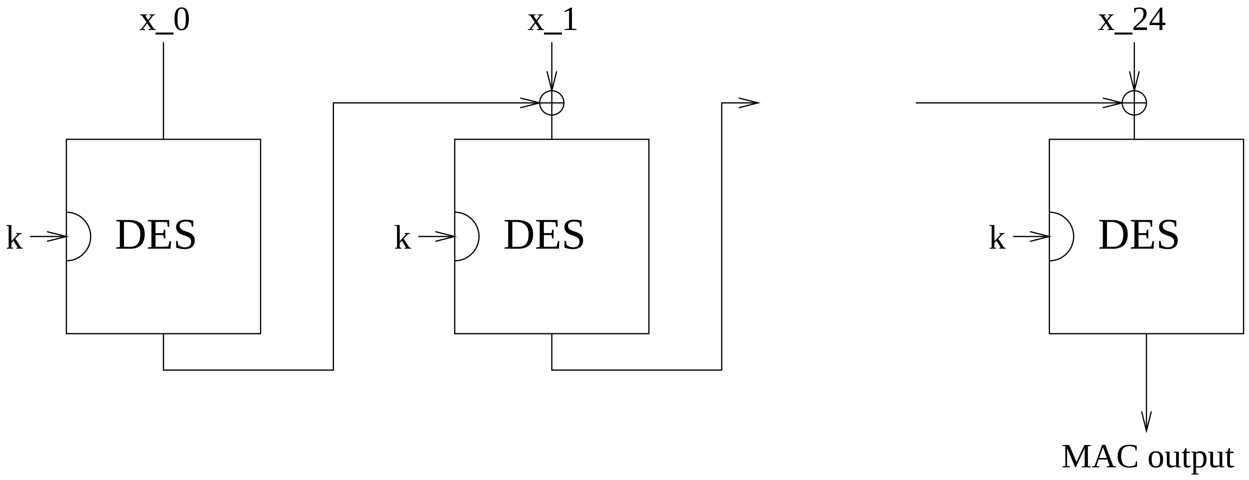
Easy: Iterated PRP $\rightarrow$ Iterated PRF

Theorem 1 *A* (t, q, e)*-secure PRP* $E : \mathcal{K} \times \mathcal{Y} \rightarrow \mathcal{Y}$ *is a* $(t, q, e + q^2/2|\mathcal{Y}|)$*-secure PRF.*

DES-CBC-MAC

Define

DES-CBC-MAC$_k(x) = \mathrm{DES}_k(x_{24} \oplus \mathrm{DES}_k(x_{23} \oplus \ldots \mathrm{DES}_k(x_1 \oplus \mathrm{DES}_k(x_0)) \ldots))$.



Bellare, Kilian, and Rogaway showed that DES-CBC-MAC is a secure PRF if DES is.

Analyzing iteration

A key trick:

$$f^n(x) = f\text{-CBC-MAC}(x, 0, 0, \ldots, 0).$$

Therefore,

$$\boxed{\text{Unix-hash}(k) = \text{DES-CBC-MAC}_k(0).}$$

The trick: Iterated PRF $\to$ PRF

Theorem 2 *If* $F : \mathcal{K} \times \mathcal{Y} \to \mathcal{Y}$ *is a* (t, q, e)*-secure PRF, then* F^n *is a* $(t', q/n, e')$*-secure PRF, where* $t' = t - q \log_2 |\mathcal{Y}|$ *and* $e' = e + 1.5q^2/|\mathcal{Y}|$.

Proof. Note that $F_k^n(x) = F_k(\cdots(F_k(x))\cdots)$ is a F_k-CBC-MAC on $(x, 0, 0, \ldots, 0)$. Now invoke Bellare, Kilian, and Rogaway (Lemma 4.1). $\square$

Easy: PRF $\rightarrow$ PRG

Theorem 3 *If F is a $(t, 1, e)$-secure PRF, then $g(k) = F_k(0)$ is a (t, e)-secure PRG.*

Corollary 1 *If DES is a $(t, 25, e)$-PRP, then $H(k) = DES_k^{25}(0)$ is a $(t, e + 25^2/2^{63})$-PRG. In other words, the Unix hashing algorithm H is a secure pseudo-random generator if DES is a secure block cipher.*

The meat: PRG $\rightarrow$ OWF

The main technical result:

Theorem 4 *Let $g : \mathcal{K} \rightarrow \mathcal{Y}$ be a (t, e)-secure PRG, with $|\mathcal{Y}| > |\mathcal{K}|$. Then g is a (t, p)-secure one-way function, where $p = e/(1 - |\mathcal{K}|/|\mathcal{Y}|)$.*

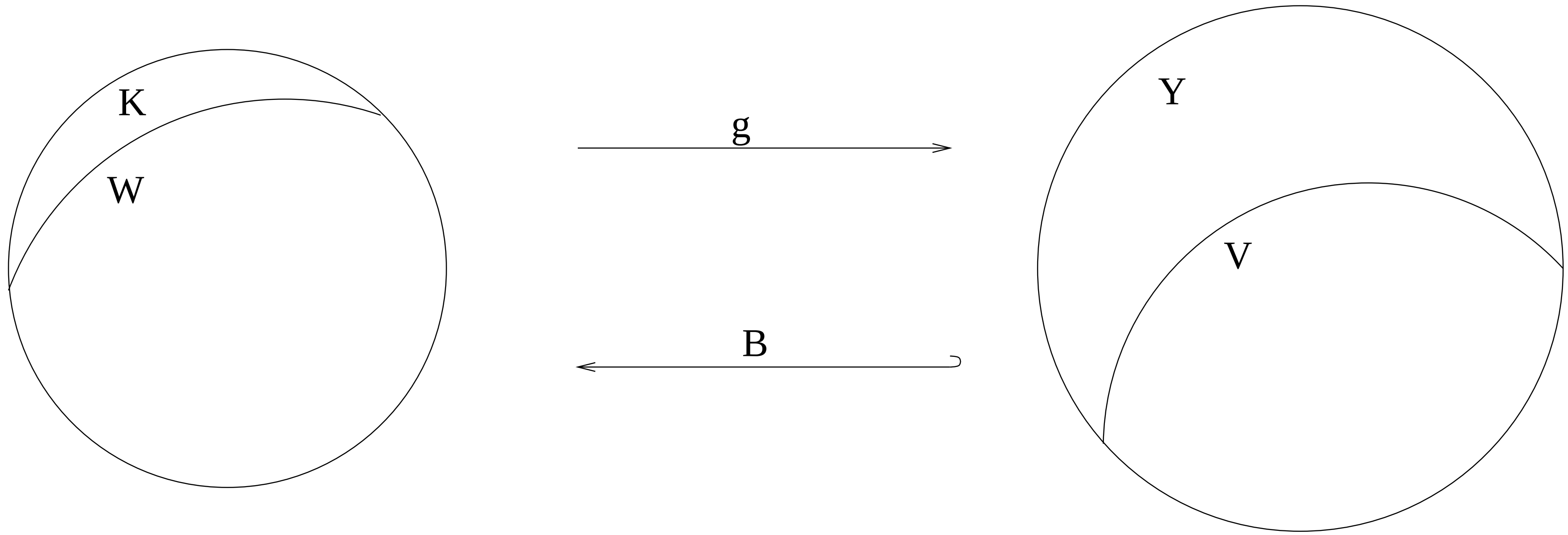
Proof of Theorem 4

Proof. We show the contrapositive.
Suppose adversary B (t, p)-breaks g (as a one-way function).
The claim is that the adversary A defined below (t, e)-breaks g (as a PRG).

$$A(y) = \begin{cases} 1 & \text{if } B(y) \neq \perp \\ 0 & \text{otherwise.} \end{cases}$$

The rest of the argument is a proof of this claim using a counting argument.

Pictorial view of the counting argument



$V = \{y : B(y) \neq \perp\}$ is the set of g-outputs where B succeeds.

$W = \{k : g(k) \in V\}$ is the set of g-inputs which are not secure against B.

The counting argument

Lemma 1 *$|V| \leq |W|$.*

Proof. The restriction of B to V may be viewed as a function with signature $V \to W$.

(Proof: If $v \in V$, then $g(B(v)) = v \in V$, so $B(v) \in W$.)

Furthermore, this function is one-to-one.

(Proof: If $v, v' \in V$, then $v = g(B(v)) = g(B(v')) = v'$, so $v = v'$.)

$\square$

Calculating the advantage

First, $\mathsf{Pr}[A(g(k)) = 1] = \mathsf{Pr}[k \in W] = p$.

Also $|W| = |\mathcal{K}| \cdot p$, so by Lemma 1,

$$\mathsf{Pr}[A(u_Y) = 1] = \mathsf{Pr}[u_Y \in V] = |V|/|\mathcal{Y}| \leq |W|/|\mathcal{Y}| = |\mathcal{K}|/|\mathcal{Y}| \cdot p.$$

Plugging into the definition of $\mathrm{Adv}\, A$ gives

$$\mathrm{Adv}\, A \geq |p - |\mathcal{K}|/|\mathcal{Y}| \cdot p| = (1 - |\mathcal{K}|/|\mathcal{Y}|) \cdot p = e.$$

□

Implications for the Unix hash

Corollary 2 *If DES is a* $(t', 25, e')$*-secure PRP, then the Unix hash function is a* (t, e)*-secure OWF, where* $t \approx t'$ *and* $e \approx (1 + 1/255)e'$*.*

Suppose that, if you're only allowed at most 25 chosen plaintext queries, there are no good attacks on DES. In this case, the Unix hash is secure.

Non-uniform distributions

What we're missing is a proof of security in the case that the password k is non-uniformly distributed.

Open questions for research:

- Given a cryptographic primitive good for uniformly distributed keys, can we construct a (perhaps probabilistic) variant that is provably secure for any distribution on the key?

Summary

- Near-optimal lower bounds for the Unix hash function.
  The construction preserves the security of DES extremely well.
- Possibly of independent interest: (1) very fast, security-preserving construction of one-way functions from block/stream ciphers; (2) a trick for handling for iterated encryption.
- Research challenge: analysis of non-uniform keying?