Simple composition theorems of one-way functions – proofs and presentations

Jaime Gaspar* Eerke Boiten†

December 18, 2014

Abstract

One-way functions are both central to cryptographic theory and a clear example of its complexity as a theory. From the aim to understand theories, proofs, and communicability of proofs in the area better, we study some small theorems on one-way functions, namely: composition theorems of one-way functions of the form "if f (or h) is well-behaved in some sense and g is a one-way function, then $f \circ g$ (respectively, $g \circ h$) is a one-way function".

We present two basic composition theorems, and generalisations of them which may well be folklore. Then we experiment with different proof presentations, including using the Coq theorem prover, using one of the theorems as a case study.

1 Composition theorems

1.1 Introduction

One-way functions are perhaps the most basic building blocks in cryptography. The basic building blocks are composed in order to produce more complicated structures. So composition theorems about one-way functions, that assert the behaviour of these basic building blocks under composition, seem fundamental to assess more complicated structures in cryptography. In this article we study two basic composition theorems, two generalisations of them, and then we use one of the generalisations as a case study in proof presentation.

Notation 1. Let us denote by $\mathcal{P}$ the set of all functions from $\{0,1\}^*$ to $\{0,1\}^*$ that are computable by a deterministic polynomial-time algorithm.

*School of Computing, University of Kent, Canterbury, Kent, CT2 7NF, United Kingdom. Centro de Matemática e Aplicações (CMA), FCT, UNL. mail@jaimegaspar.com / jg478@kent.ac.uk, www.jaimegaspar.com / www.cs.kent.ac.uk/people/rpg/jg478. Financially supported by a Research Postgraduate Scholarship from the Engineering and Physical Sciences Research Council / School of Computing, University of Kent.

†School of Computing & Centre for Cyber Security, University of Kent, Canterbury, Kent, CT2 7NF, United Kingdom. E.A.Boiten@kent.ac.uk, www.cs.kent.ac.uk/people/staff/eab2.

Informally, a one-way function g is a function that is easy to compute but difficult to invert: $x \mapsto g(x)$ is easy to compute but $x \hookleftarrow g(x)$ (or more precisely, $y \hookleftarrow g(x)$ such that $g(x) = g(y)$) is difficult to compute.

Definition 2 ([1, definition 2.2.1])**.** A function $g \in \mathcal{P}$ is *one-way* if and only if

$$\forall A, p, \exists N \colon \forall n > N,\ \Pr\left[g\big(A(g(U_n), 1^n)\big) = g(U_n)\right] < 1/p(n),$$

where

- A ranges over all probabilistic polynomial-time algorithms;
- p ranges over all positive polynomials with real coefficients;
- N and n range over $\mathbb{N}$;
- the internal coin tosses of A are uniformly distributed;
- U_n is a random variable uniformly distributed on $\{0,1\}^n$;
- Pr is taken over U_n and the internal coin tosses of A.

1.2 Specialised composition theorems

Theorem 3 (left composition, particular variant)**.** *If $f \in \mathcal{P}$ is an injective function and $g \in \mathcal{P}$ is a one-way function, then $f \circ g \in \mathcal{P}$ is a one-way function.*

Proof (sketch). We essentially need to show that if $\Pr\left[g\big(A(g(U_n), 1^n)\big) = g(U_n)\right] < 1/p(n)$, then $\Pr\left[(f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right] < 1/p(n)$, and this is true taking $A(x, y) = A'(f(x), y)$ and using $x = y \Leftrightarrow f(x) = f(y)$. □

Definition 4 ([1, definition 2.2.4])**.** A function $h \in \mathcal{P}$ is *length-preserving* if and only if $\forall x \in \{0,1\}^*$, $|h(x)| = |x|$.

Theorem 5 (right composition, particular variant)**.** *If $g \in \mathcal{P}$ is a one-way function and $h \in \mathcal{P}$ is a length-preserving injective function, then $g \circ h \in \mathcal{P}$ is a one-way function.*

Proof (sketch). We essentially need to show that if $\Pr\left[g\big(A(g(U_n), 1^n)\big) = g(U_n)\right] < 1/p(n)$, then $\Pr\left[(g \circ h)\big(A'((g \circ h)(U'_n), 1^n)\big) = (g \circ h)(U'_n)\right] < 1/p(n)$, and this is true taking $A(x, y) = h(A'(x, y))$ and making the change of variable $U_n = h(U'_n)$, which preserves probabilities because $h|_{\{0,1\}^n} \colon \{0,1\}^n \to \{0,1\}^n$ is injective and so bijective. □

Corollary 6 (left and right compositions, particular variant)**.** *If $f \in \mathcal{P}$ is an injective function, $g \in \mathcal{P}$ is a one-way function and $h \in \mathcal{P}$ is a length-preserving injective function, then $f \circ g \circ h \in \mathcal{P}$ is a one-way function.*

1.3 General composition theorems

A collision of a function f is a pair (x, y) such that $x \neq y$ and $f(x) = f(y)$. Informally, a collision-resistant function is a function such that it is difficult to find collisions. If we naively try to formalise this notion by

$$\begin{gathered}\forall A, p,\ \exists N\colon \forall n > N,\\ \Pr[A(0, 1^n) \neq A(1, 1^n) \wedge f(A(0, 1^n)) = f(A(1, 1^n))] < 1/p(n),\end{gathered} \tag{1}$$

then: f is collision-resistant if and only f is injective. Informally this is because the existential choice of algorithm A allows the collision to be picked directly. Let us sketch a proof:

> if f is not injective, then there is a collision (x, y), so the constant (on z) algorithm $A(0, z) = x$ and $A(1, z) = y$ falsifies (1); if f is injective, then there is no collision, so the probability $\Pr[\ldots]$ in (1) is 0, thus we have (1).

So the formalisation (1) is uninteresting as it reduces to injectivity. The problem, as suggested by the proof, is that if there is a collision (x, y), then the constant algorithm $A(0, z) = x$ and $A(1, z) = y$ satisfies the condition inside the brackets in (1). So we need to change (1) so as to exclude the constant algorithm.

- The traditional way of changing (1) is to replace a single function f by a family $\{f_i\}_i$ of functions, randomly pick an f_i and require that the algorithm outputs a collision for this f_i; since f_i may change, the algorithm cannot be constant. But this way forces us not to speak of "a collision-resistant function f" but rather of "a collision-resistant family of functions $\{f_i\}_i$", which displeases us [2, page 137] [3, lecture 21] [4, section 2.2].
- We propose another way of changing (1) by demanding that the algorithm outputs arbitrary large collisions, or equivalently, infinitely many collision; then the algorithm cannot be constant. This way allows us to continue to speak of "a collision-resistant function f" instead of "a collision-resistant family of functions $\{f_i\}_i$", which pleases us.

We welcome comments on our proposed way of changing (1) resulting in the next definition.

Definition 7. A function $f \in \mathcal{P}$ is *collision-resistant* if and only if

$$\begin{gathered}\forall A, p,\ \exists N\colon \forall n > N,\ \Pr[(|A(0, 1^n)| \geq n \vee |A(1, 1^n)| \geq n) \wedge\\ A(0, 1^n) \neq A(1, 1^n) \wedge f(A(0, 1^n)) = f(A(1, 1^n))] < 1/p(n),\end{gathered}$$

where

- A ranges over all probabilistic polynomial-time algorithms;
- p ranges over all positive polynomials with real coefficients;
- N and n range over $\mathbb{N}$;

- the internal coin tosses of A are uniformly distributed;
- Pr is taken over the internal coin tosses of A.

Definition 8. A function $g \in \mathcal{P}$ is *length-nondecreasing* if and only if $\forall x \in \{0,1\}^*$, $|h(x)| \geq |x|$.

The following theorem almost generalises theorem 3 because collision-resistance generalises injectivity, but fails to truly generalise because it has the additional hypothesis that g is length-nondecreasing. We welcome suggestions on how to dismiss this additional hypothesis.

Theorem 9 (left composition, general variant). *If $f \in \mathcal{P}$ is a collision-resistant function and $g \in \mathcal{P}$ is a length-nondecreasing one-way function, then $f \circ g \in \mathcal{P}$ is a one-way function.*

Proof (sketch). We essentially need to show that if (a) $g\big(A(g(U_n), 1^n)\big) = g(U_n)$ has low probability, then (b) $(f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)$ has low probability, and this is true because

- (b) is the disjunction of

$$g\big(A'((f \circ g)(U_n)), 1^n\big) = g(U_n) \tag{c}$$

$$\vee$$

$$\begin{gathered} g\big(A'((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge \\ (f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n); \end{gathered} \tag{d}$$

- (c) has low probability because it is (a) with $A(x,y) = A'(f(x), y)$, and (d) has low probability because it is a collision of f (with $|g(U_n)| \geq n$). $\square$

An unusual way of defining injectiveness (one-to-oneness) of a function h is to say $\forall x \in \{0,1\}^*$, $|h^{-1}[x]| \leq 1$. In the next definition, we generalise this by allowing $q(|x|)$ (where q is polynomial) in place of 1.

Definition 10. A function $h \in \mathcal{P}$ is *polynomial-to-one* if and only if $\exists q$: $\forall x \in \{0,1\}^*$, $|f^{-1}[x]| \leq q(|x|)$, where q ranges over all positive polynomial with real coefficients.

The following theorem generalises theorem 5 because polynomial-to-oneness generalises injectivity.

Theorem 11 (right composition, general variant). *If $g \in \mathcal{P}$ is a one-way function and $h \in \mathcal{P}$ is a length-preserving polynomial-to-one function, then $g \circ h \in \mathcal{P}$ is a one-way function.*

Proof (sketch). We essentially need to show that if $\Pr[g(A(g(U_n), 1^n)) = g(U_n)] < 1/p(n)$, then $\Pr\big[(g \circ h)\big(A'((g \circ h)(U'_n), 1^n)\big) = (g \circ h)(U'_n)\big] < 1/p'(n)$, and this is true taking $A(x,y) = h(A'(x,y))$ and $p = p'q$, and making the change of variable $U_n = h(U'_n)$, which increases probabilities at most $q(n)$ times because

$$\{x \in \{0,1\}^n : P(h(x))\} = \bigcup_{y \in \{0,1\}^n : P(y)} h^{-1}[y] \;\Rightarrow$$
$$\Pr[P(h(x))] = \sum_{y \in \{0,1\}^n : P(y)} \Pr[h^{-1}[y]] = \sum_{y \in \{0,1\}^n : P(y)} |h^{-1}[y]|/2^n \leq$$
$$\sum_{y \in \{0,1\}^n : P(y)} q(n)/2^n = q(n) \sum_{y \in \{0,1\}^n : P(y)} 1/2^n = q(n) \Pr[P(y)].$$

where P is a predicate on $\{0,1\}^n$ and x and y are uniformly distributed on $\{0,1\}^n$. □

Corollary 12 (left and right compositions, general variant)**.** *If $f \in \mathcal{P}$ is a collision-resistant function, $g \in \mathcal{P}$ is a length-nondecreasing one-way function and $h \in \mathcal{P}$ is a length-preserving polynomial-to-one function, then $f \circ g \circ h \in \mathcal{P}$ is a one-way function.*

2 A case study in proof presentation

2.1 Introduction

Proofs in cryptography tend to be difficult to check due to the simultaneous use of four theories:

- probability theory (for example, when talking about the probability of certain events being low);
- computability theory (for example, when talking about certain problems being solved by probabilistic polynomial-time algorithms);
- asymptotic theory (for example, when talking about certain functions being negligible);
- cryptographic theory itself.

So proof presentation becomes especially important to facilitate to check proofs. In this section we show some possible proof presentations taking as a case study the proof of theorem 9.

2.2 Traditional proof

Proof.

1. We assume

$$\begin{aligned}\forall A, p,\ \exists N\colon \forall n > N,\ \Pr[(|A(0,1^n)| \geq n \vee |A(1,1^n)| \geq n) \wedge \\ A(0,1^n) \neq A(1,1^n) \wedge f(A(0,1^n)) = f(A(1,1^n))] < 1/p(n),\end{aligned} \tag{2}$$

$$\forall A', p',\ \exists N'\colon \forall n > N',\ \Pr\big[g\big(A'(g(U_n),1^n)\big) = g(U_n)\big] < 1/p'(n), \tag{3}$$

and we prove

$$\begin{gathered}\forall A'', p'',\ \exists N''\colon \forall n > N'', \\ \Pr\big[(f \circ g)\big(A''((f \circ g)(U_n),1^n)\big) = (f \circ g)(U_n)\big] < 1/p''(n).\end{gathered} \tag{4}$$

2. Let us take arbitrary A'' and p''. Taking

- $A(i,x)$ to be the algorithm that uniformly and randomly chooses $U_n \in \{0,1\}^n$ and outputs $\begin{cases} g\big(A''((f \circ g)(U_n),x)\big) & \text{if } i = 0 \\ g(U_n) & \text{if } i \neq 0 \end{cases}$ in (2);
- $p = 2p''$ in (2);
- $A'(x,y) = A''(f(x),y)$ in (3);
- $p' = 2p''$ in (3);

we get N and N' such that for all $n > N'' = \max(N, N')$ we have

$$\Pr[(|\overbrace{A(0,1^n)}^{=g(A''((f\circ g)(U_n),1^n))}| \geq n \vee \underbrace{|\overbrace{A(1,1^n)}^{=g(U_n)}| \geq n}_{\text{true}}) \wedge$$

$$\underbrace{A(0,1^n)}_{=g(A''((f\circ g)(U_n),1^n))} \neq \underbrace{A(1,1^n)}_{=g(U_n)} \wedge f(\underbrace{A(0,1^n)}_{=g(A''((f\circ g)(U_n),1^n)}) = f(\underbrace{A(1,1^n)}_{=g(U_n)})] < 1/\underbrace{p(n)}_{2p''(n)},$$

$$\Pr\big[\underbrace{g\big(A'(g(U_n),1^n)\big)}_{=g(A''((f\circ g)(U_n),1^n))} = g(U_n)\big] < 1/\underbrace{p'(n)}_{2p''(n)}.$$

3. The condition $(f \circ g)\big(A''((f \circ g)(U_n),1^n)\big) = (f \circ g)(U_n)$ in (4) is equivalent to the disjunction

$$\begin{gathered} g\big(A''((f \circ g)(U_n),1^n)\big) = g(U_n) \\ \vee \\ g\big(A''((f \circ g)(U_n),1^n)\big) \neq g(U_n) \wedge (f \circ g)\big(A''((f \circ g)(U_n),1^n)\big) = (f \circ g)(U_n) \end{gathered}$$

so

$$\begin{gathered} \Pr\big[(f \circ g)\big(A''((f \circ g)(U_n),1^n)\big) = (f \circ g)(U_n)\big] \leq \\ \Pr\big[g\big(A''((f \circ g)(U_n),1^n)\big) = g(U_n)\big] + \\ \Pr\big[g\big(A''((f \circ g)(U_n),1^n)\big) \neq g(U_n) \wedge \\ (f \circ g)\big(A''((f \circ g)(U_n),1^n)\big) = (f \circ g)(U_n)\big] < \\ 1/(2p''(n)) + 1/(2p''(n)) = 1/p''(n). \end{gathered}$$

thus we proved (4), as we wanted. □

2.3 Schematic proof

Proof. We essentially need to show that $(f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)$ has low probability, and we do this by showing that it is the disjunction of two conditions, each one with low probability:

$$
\begin{array}{c}
(f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) \\
= (f \circ g)(U_n)
\end{array}
\Leftrightarrow
\left\{
\begin{array}{c}
\overbrace{g\big(A'((f \circ g)(U_n), 1^n)\big) = g(U_n)}^{\text{low probability because } g \text{ is one-way}} \\
\vee \\
\underbrace{\begin{array}{c}
\Big(\big|g\big(A'((f \circ g)(U_n), 1^n)\big)\big| \geq n \vee \overbrace{|g(U_n)| \geq n}^{\text{true because } g \text{ is length-nondecreasing}}\Big) \wedge \\
g\big(A'((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge \\
(f \circ g)\big(A'((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)
\end{array}}_{\text{low probability because } g \text{ is collision-resistant}}
\end{array}
\right.
$$

□

2.4 Calculus proof

$$
\overbrace{\begin{gathered}\forall p, A,\ \exists N\colon \forall n > N,\ \Pr[(|A(0, 1^n)| \geq n \vee |A(1, 1^n) \geq n|) \wedge \\ A(0, 1^n) \neq A(1, 1^n) \wedge f(A(0, 1^n)) = f(A(1, 1^n))] < 1/p(n) \wedge\end{gathered}}^{f \text{ is collision-resistant}}
$$
$$
\underbrace{\forall p, A',\ \exists N'\colon \forall n > N',\ \Pr\left[g\big(A'(g(U_n), 1^n)\big) = g(U_n)\right] < 1/p(n)}_{g \text{ is one-way}}
$$

$$
\Big\Downarrow \quad \text{take } A(i, x) = \begin{cases} g(A((f \circ g)(U_n), 1^n)) & \text{if } i = 0 \\ g(U_n) & \text{if } i \neq 0 \end{cases}, \quad p = 2p \text{ and } A'(x, y) = A(f(x), y), \text{ and notice } |g(U_n)| \geq n
$$

$$
\begin{gathered}
\forall p, A,\ \exists N\colon \forall n > N,\ \Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge \right. \\
\left. (f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right] < 1/(2p(n)) \wedge \\
\forall p, A,\ \exists N'\colon \forall n > N',\ \Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) = g(U_n)\right] < 1/(2p(n))
\end{gathered}
$$

$$
\Big\Downarrow \ \text{take } N = \max(N, N')
$$

$$
\begin{gathered}
\forall p, A,\ \exists N\colon \forall n > N,\ \big(\Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge \right. \\
\left. (f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right] < 1/(2p(n)) \wedge \\
\Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) = g(U_n)\right] < 1/(2p(n))\big)
\end{gathered}
$$

$$
\Big\Downarrow \ \text{use } \Pr[P \vee Q] \leq \Pr[P] + \Pr[Q]
$$

$$
\begin{gathered}
\forall p, A,\ \exists N\colon \forall n > N,\ \Pr\left[\big(g\big(A((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge \right. \\
(f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\big) \vee \\
\left. g\big(A((f \circ g)(U_n), 1^n)\big) = g(U_n)\right] < 1/p(n)
\end{gathered}
$$

$$
\Big\Downarrow \ \text{use } (x \neq y \wedge f(x) = f(y)) \vee x = y \ \Leftrightarrow \ f(x) = f(y)
$$

$$
\begin{gathered}
\forall p, A,\ \exists N\colon \forall n > N, \\
\underbrace{\Pr\left[(f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right] < 1/p(n)}_{f \circ g \text{ is one-way}}
\end{gathered}
$$

2.5 Algebraic proof

Definition 13. A function $f\colon \mathbb{N} \to \mathbb{R}$ is *negligible* if and only if $\forall p,\ \exists N\colon \forall n > N,\ |f(n)| \leq 1/p(n)$, where

- p ranges over all positive polynomials with real coefficients;
- N and n range over $\mathbb{N}$.

We denote the set of all negligible functions by $\mathcal{N}$.

Proof. A function $f \in \mathcal{P}$ is one-way if and only if

$$
\begin{gathered}
\forall A,\ \bar{f}_A(n) = \Pr[(|A(0, 1^n)| \geq n \vee |A(1, 1^n)| \geq n) \wedge \\
A(0, 1^n) \neq A(1, 1^n) \wedge f(A(0, 1^n)) = f(A(1, 1^n))] \in \mathcal{N}.
\end{gathered}
$$

A function $g \in \mathcal{P}$ is one-way if and only if

$$\forall A,\ \bar{g}_A(n) = \Pr\left[g\big(A(g(U_n), 1^n)\big) = g(U_n)\right] \in \mathcal{N}.$$

A function $f \circ g \in \mathcal{P}$ is one-way if and only if

$$\forall A,\ \overline{fg}_A(n) = \Pr\left[(f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right] \in \mathcal{N}.$$

We have

$$\begin{gathered}(f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n) \ \Leftrightarrow\ g\big(A((f \circ g)(U_n), 1^n)\big) = g(U_n) \vee \\ \big(g\big(A((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge (f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\big)\end{gathered}$$

so

$$\begin{gathered}\underbrace{\Pr\left[(f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right]}_{\overline{fg}_A(n)} \leq \\ \underbrace{\Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) = g(U_n)\right]}_{\bar{g}_B(n)} + \\ \underbrace{\Pr\left[g\big(A((f \circ g)(U_n), 1^n)\big) \neq g(U_n) \wedge (f \circ g)\big(A((f \circ g)(U_n), 1^n)\big) = (f \circ g)(U_n)\right]}_{\bar{f}_C(n)}.\end{gathered}$$

where $B(x, y) = A(f(x), y)$ and $C(i, x) = \begin{cases} g\big(A((f \circ g)(U_n), x)\big) & \text{if } i = 0 \\ g(U_n) & \text{if } i \neq 0 \end{cases}$, and notice $|C(1, 1^n)| \geq n$.

We know $\overline{fg}_A \leq \bar{g}_B + \bar{f}_C$ where $\bar{g}_B, \bar{f}_C \in \mathcal{N}$, and we want to prove $\overline{fg}_A \in \mathcal{N}$. This follows from two "algebraic" facts about $\mathcal{N}$:

- $\mathcal{N}$ is closed under addition, that is, $\mathcal{N} + \mathcal{N} \subseteq \mathcal{N}$ (where $\mathcal{N} + \mathcal{N} = \{f + g : f, g \in \mathcal{N}\}$), or in other words, $\forall f, g \in \mathcal{N},\ f + g \in \mathcal{N}$;

- $\mathcal{N}$ is downwards closed, that is, $\forall f\colon \mathbb{N} \to \mathbb{R},\ \forall g \in \mathcal{N},\ (f \leq g \ \Rightarrow\ f \in \mathcal{N})$ (where $f \leq g$ means $\forall n \in \mathbb{N},\ f(n) \leq g(n)$). $\square$

2.6 Coq proof

Coq is a proof assistant: a software that helps us to write formal proofs and verifies the correction of the proofs. To write a formal proof in Coq, we need to tell Coq the following four items:

- the language of our theory;

- the axioms of our theory;

- the claim of our lemmas and theorem;

- the proofs of our lemmas and theorem.

The complete description of these four items takes 117 lines of code (see annex), so it would be tedious to go through the description line by line. For this reason, we focus only on a few representative lines of each item.

Language At first sight, it may look like that our theory talks only about a collision-resistant function f and a length-nondecreasing one-way function g. However, below the "bonnet", we also talk about probabilistic polynomial-time algorithms A, polynomials p, probabilities Pr, and even objects that we take for granted like natural numbers N and n and their order relation $>$. All this needs to be told to Coq. Actually, we could rely on the fact that Coq has libraries for dealing with some objects like N, n and $>$. However, since we need to introduce other objects like A, p and Pr from scratch, we may as well also do the same for N, n and $>$.

Let us see a representative example. In line 1 below we introduce a set $\mathbb{N}$ for the natural numbers, then in line 2 we introduce the relation "Greater Than" between natural numbers, and finally in line 3 we introduce the usual notation $>$ for that relation. In line 5 we introduce a set $\mathbb{T}$ for $\{0,1\}^*$, and in lines 6, 7 and 8 we introduce predicates $\mathbb{NP}$, `ℕℙ'` and `ℕℙ''` for probabilistic polynomial-time algorithms that differ in the number and type of their inputs.

```
1 Parameter ℕ : Set.
2 Parameter GT : ℕ -> ℕ -> Prop.
3 Notation "x '>' y" := (GT x y).
4
5 Parameter 𝕋 : Set.
6 Parameter ℕℙ : (𝕋 -> ℕ -> 𝕋) -> Prop.
7 Parameter ℕℙ' : (𝕋 -> 𝕋 -> ℕ -> 𝕋) -> Prop.
8 Parameter ℕℙ'' : (ℕ -> 𝕋 -> ℕ -> 𝕋) -> Prop.
```

Axioms Once we have introduced from scratch the language of our theory, we need to characterise the behaviour of the objects in that language by giving their axioms. We adopt a minimal approach in which we introduce only the axioms that we really use in the proof.

Let us see a representative example. In the code below, we introduce an axiom saying that for all probabilistic polynomial-time algorithm A and function g computable in polynomial time, the algorithm $B(i,x) = \begin{cases} A(x) & \text{if } i = 0 \\ g(x) & \text{if } i = 1 \end{cases}$ is a probabilistic polynomial-time algorithm (the extra input n in the code means that x ranges on $\{0,1\}^n$).

```
Axiom ℕℙ1 : forall (A : 𝕋 -> ℕ -> 𝕋) (g : 𝕋 -> 𝕋), (ℕℙ A
   -> ℙ g -> exists B : ℕ -> 𝕋 -> ℕ -> 𝕋, ℕℙ'' B /\
   forall (x : 𝕋) (n : ℕ), B _0 x n = A x n /\ B _1 x n
   = g x).
```

Lemmas and theorem To simplify the presentation of the proof, we split it in three parts: two lemmas and one theorem. In the code below, lemma L1 essentially says that the algorithm $B(i,x) = \begin{cases} g\big(A''((f \circ g)(x), x)\big) & \text{if } i = 0 \\ g(x) & \text{if } i \neq 0 \end{cases}$ is a probabilistic polynomial-time algorithm. Lemma L2 essentially says $P \Leftrightarrow Q \vee R$ where

$$
\begin{aligned}
P &= (f \circ g)(A((f \circ g)(x), 1^n)) = (f \circ g)(x),\\
Q &= g(A((f \circ g)(x), 1^n)) = g(x),\\
R &= \big(\big|g\big(A((f \circ g)(x), 1^n)\big)\big| \geq n \vee |g(x)| \geq n\big) \wedge\\
&\qquad g\big(A((f \circ g)(x), n)\big) \neq g(x) \wedge (f \circ g)\big(A((f \circ g)(x), 1^n)\big) = (f \circ g)(x),\\
R' &= (|B(0, r, 1^n)| \geq n \vee |B(1, r, 1^n)| \geq n) \wedge\\
&\qquad B(0, r, n) \neq B(1, r, n) \wedge f(B(0, r, n)) = f(B(1, r, n))
\end{aligned}
$$

(R' will appear later). Finally, theorem T1 is theorem 9. (The variable xr encodes the argument $x = xr_1$ of g and the input $r = xr_2$ on the random strip of the probabilistic Turing machine computing A, where $\cdot_1$ and $\cdot_2$ are respectively the first and second projections.)

```
Lemma L1 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋), ℕℙ' A ->
    exists (B : ℕ -> 𝕋 -> ℕ -> 𝕋), ℕℙ'' B /\ forall (xr :
    𝕋) (n : ℕ), B _0 xr n = g (A (f (g xr 1)) xr 2 n) /\ B
    _1 xr n = g xr 1.
Lemma L2 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋) (n : ℕ) (xr : 𝕋
    ), | xr 1 | ≥ n -> | xr 2 | ≥ n -> (f(g(A(f(g xr 1))
    xr 2 n)) = f(g xr 1) <-> g(A(f(g xr 1)) xr 2 n) = g xr 1
     \/ (| g(A(f(g xr 1)) xr 2 n) | ≥ n \/ | g xr 1 | ≥ n)
    /\ g(A(f(g xr 1)) xr 2 n) <> g xr 1 /\ f(g(A(f(g xr 1))
    xr 2 n)) = f(g xr 1)).
Theorem T1 : forall A : 𝕋 -> 𝕋 -> ℕ -> 𝕋, (ℕℙ' A ->
    forall p : ℕ -> ℕ, (⊓ p -> exists N : ℕ, forall n : ℕ
    , (n > N -> Pr n {xr : 𝕋 | f(g(A(f(g xr 1)) xr 2 n)) =
    f(g xr 1)} < (1 / p n)))).
```

Proof Let us focus on the last part of the proof (paragraph 3 of the traditional proof) because is the more difficult one. At this point of the proof

- we have N and N', and took $n > N'' = \max(N, N')$, which is such that $\Pr[Q] < 1/(2p(n))$ and $\Pr[R'] < 1/(2p(n))$ (paragraph 2 of the traditional proof);
- we are about to prove $\Pr[P] \leq \Pr[Q] + \Pr[R]$ and $\Pr[Q] + \Pr[R] < 1/p(n)$ (paragraph 3 of the traditional proof).

Figure 1 helps to follow the reasoning. In the code below, in line 1 we replace the goal $\Pr[P] < 1/p(n)$ by the two goals (a) $\Pr[P] \leq \Pr[Q] + \Pr[R]$ and (b) $\Pr[Q] + \Pr[R] < 1/p(n)$. In line 2 we replace (a) by the goal $P \Leftrightarrow Q \vee R$, which is true by lemma L1. In line 3 we replace (b) by the two goals (c) $\Pr[Q] < 1/(2p(n))$ and (d) $\Pr[R] < 1/(2p(n))$. In line 4 we prove (c). In line 5 we replace (d) by the two goals (e) $\Pr[R'] < 1/(2p(n))$ and (f) $\Pr[R] = \Pr[R']$. In line 6 we prove (e). Finally, in line 7 we prove (f).

```
1 apply LET1 with (x := (Pr n {xr : 𝕋 | f (g (A (f (g xr_1
     )) xr_2 n)) = f (g xr_1)})) (y := Pr n {xr : 𝕋 | g (A
     (f (g xr_1)) xr_2 n) = g xr_1} + Pr n { xr : 𝕋 | (| g
     (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g
     (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f (g xr_1
     )) xr_2 n)) = f (g xr_1)}) (z := 1 / p n).
2 apply Pr1 with (n := n) (P := fun xr : 𝕋 => f (g (A (f
     (g xr_1)) xr_2 n)) = f (g xr_1)) (Q := fun xr : 𝕋 => g
     (A (f (g xr_1)) xr_2 n) = g xr_1) (R := fun xr : 𝕋 =>
     (| g (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n)
     /\ g (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f
     (g xr_1)) xr_2 n)) = f (g xr_1)). apply L2.
3 apply S1.
4 apply H6. apply max2 with (i := n) (j := N) (k := N').
     exact H7.
5 replace (Pr n {xr : 𝕋 | (|g (A (f (g xr_1)) xr_2 n) | ≥ n
     \/ | g xr_1 | ≥ n) /\ g (A (f (g xr_1)) xr_2 n) <> g
     xr_1 /\ f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)})
     with (Pr n {r : 𝕋 | ((|B _0 r n |) ≥ n \/ (|B _1 r n
     |) ≥ n) /\ B _0 r n <> B _1 r n /\ f (B _0 r n) = f
     (B _1 r n)}).
6 apply H5. apply max1 with (i := n) (j := N) (k := N').
     exact H7.
7 apply Pr2 with (A := A) (B := B) (n := n). apply H4.
```

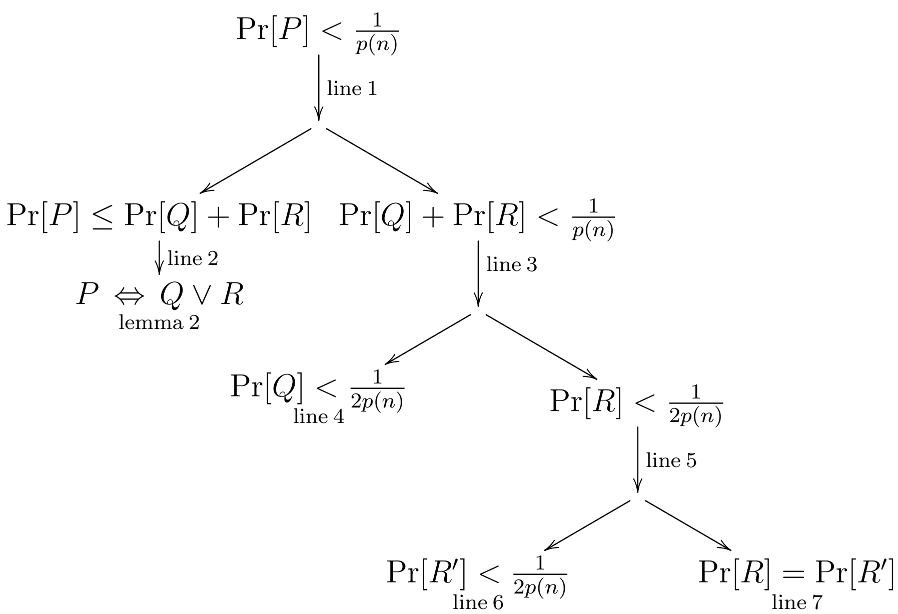

Figure 1: schematic representation of the last part of the proof.

Annex

```
(* LANGUAGE *)
Parameter ℝ: Set. (* Real numbers *)
Parameter ℕ : Set. (* Natural numbers *)
Parameter 𝕋 : Set. (* Two-star, that is, 2* = {0,1}* *)
Parameter ℙ : (𝕋 -> 𝕋) -> Prop. (* Computable in polynomial time (on the length of the input) *)
Parameter ℕℙ : (𝕋 -> ℕ -> 𝕋) -> Prop. (* Randomised polynomial time algorithms (on the length of the input) *)
Parameter ℕℙ' : (𝕋 -> 𝕋 -> ℕ -> 𝕋) -> Prop. (* Randomised polynomial time algorithms (on the length of the input) *)
Parameter ℕℙ'' : (ℕ -> 𝕋 -> ℕ -> 𝕋) -> Prop. (* Randomised polynomial time algorithms (on the length of the input) *)
Parameter Pr : ℕ -> Set -> ℝ. (* Probability predicate *)
Parameter ℿ : (ℕ -> ℕ) -> Prop. (* Positive polynomials *)
Parameter GT : ℕ -> ℕ -> Prop. (* "Greater than" between natural numbers *)
Notation "x '>' y" := (GT x y). (* ">" for "greater than" *)
Parameter GET : ℕ -> ℕ -> Prop. (* "Greater than or equal to" between natural numbers *)
Notation "x '≥' y" := (GET x y) (at level 70). (* "≥" for "greater than or equal to" *)
Parameter LT : ℝ -> ℝ -> Prop. (* "Less than" between real numbers *)
Notation "x '<' y" := (LT x y) (at level 70). (* "<" for "less than" *)
Parameter LET : ℝ -> ℝ -> Prop. (* "Less than or equal to" between real numbers *)
Notation "x '≤' y" := (LET x y) (at level 70). (* "≤" for "less than or equal to" *)
Parameter I : ℕ -> ℝ. (* "Inversion of" a natural number *)
Notation "1 / x" := (I x) (at level 60). (* "1 /" for "inverse of" *)
Parameter S : ℝ -> ℝ -> ℝ. (* "sum of" real numbers *)
Notation "x + y" := (S x y). (* "+" for "sum of" *)
Parameter D : ℕ -> ℕ. (* "Double of" a natural number *)
Notation "2 x" := (D x) (at level 70). (* "2" for "double of" *)
```

```
Parameter max : ℕ -> ℕ -> ℕ. (* "Maximum of" natural numbers *)
Parameter p_1 : 𝕋 -> 𝕋. (* First and second projects of a binary string regarded as an... *)
Parameter p_2 : 𝕋 -> 𝕋. (* ...ordered pair of binary strings under some suitable encoding *)
Notation "x ₁" := (p_1 x) (at level 1). (* "₁" for "first projection" *)
Notation "x ₂" := (p_2 x) (at level 1). (* "₂" for "first projection" *)
Parameter L : 𝕋 -> ℕ. (* "Length of" a string *)
Notation "| x |" := (L x) (at level 70). (* "| |" for "length of" *)
Parameter _0 _1 : ℕ. (* Natural numbers zero and one *)
Parameter f g : 𝕋 -> 𝕋. (* Functions f and g *)

(* AXIOMS *)
Require Import Coq.Logic.Classical_Prop. (* Classical logic *)
Axiom ℙ1 : forall g : 𝕋 -> 𝕋, ℙ g -> ℙ (fun xr : 𝕋 => g xr ₁). (* g(xr₁) is computable in polynomial time *)
Axiom ℕℙ1 : forall (A : 𝕋 -> ℕ -> 𝕋) (g : 𝕋 -> 𝕋), (ℕℙ A -> ℙ g -> exists B : ℕ -> 𝕋 -> ℕ -> 𝕋, ℕℙ'' B /\ forall (x : 𝕋) (n : ℕ), B _0 x n = A x n /\ B _1 x n = g x). (* Algorithm definition by cases B(0,x) = A(x), B(1,x) = g(x) *)
Axiom ℕℙ2 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋) (f g : 𝕋 -> 𝕋), ℕℙ' A -> ℙ f -> ℙ g -> ℕℙ (fun (xr : 𝕋) (n : ℕ) => g (A (f (g xr ₁)) xr ₂ n)). (* g(A(f(g(xr₁)),xr₂,n)) is a randomised polynomial time algorithm *)
Axiom ℕℙ3 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋) (f : 𝕋 -> 𝕋), ℕℙ' A -> ℙ f -> ℕℙ' (fun (x y : 𝕋) (n : ℕ) => A (f x) y n). (* A(f(x),y,n) is a randomised polynomial time algorithm *)
Axiom ℿ1 : forall p : ℕ -> ℕ, (ℿ p -> ℿ (fun n : ℕ => 2 p n)). (* The double of a positive polynomial is a positive polynomial *)
Axiom max1 : forall i j k : ℕ, i > max j k -> i > j. (* i > max(j,k) -> i > j *)
Axiom max2 : forall i j k : ℕ, i > max j k -> i > k. (* i > max(j,k) -> i > k *)
Axiom S1 : forall (n : ℕ) (p : ℕ -> ℕ) (X Y : ℝ), X < 1 / (2 p n) -> Y < 1 / (2 p n) -> (X + Y) < 1 / p n. (* Essentially x,y < 1/2 -> x + y < 1 *)
Axiom GET1 : forall x y z : ℕ, x ≥ y -> y ≥ z -> x ≥ z. (* Transitivity of ≥ *)
```

```
Axiom LET1 : forall x y z : ℝ, x ≤ y -> y < z -> x < z. (* Mixed transitivity of ≤ and < *)
Axiom Pr1 : forall (n : ℕ) (P Q R : 𝕋 -> Prop), (forall x : 𝕋, | x_1 | ≥ n -> | x_2 | ≥ n -> (P x <-> Q x \/ R x)) -> Pr n {x : 𝕋 | P x} ≤ Pr n {x : 𝕋 | Q x} + Pr n {x : 𝕋 | R x}. (* Essentially Pr[A U B] ≤ Pr A + Pr B *)
Axiom Pr2 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋) (B : ℕ -> 𝕋 -> ℕ -> 𝕋) (n : ℕ), (forall (x : 𝕋) (n : ℕ), B _0 x n = g (A (f (g x_1)) x_2 n) /\ B _1 x n = g x_1) -> Pr n {r : 𝕋 | (| B _0 r n | ≥ n \/ | B _1 r n | ≥ n) /\ B _0 r n <> B _1 r n /\ f (B _0 r n) = f (B _1 r n)} = (Pr n {xr : 𝕋 | (| g (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)}). (* Essentially A = B -> Pr A = Pr B *)
Axiom F1 : ℙ f. (* f is... *)
Axiom F2 : forall A : ℕ -> 𝕋 -> ℕ -> 𝕋, (ℕℙ'' A -> forall p : ℕ -> ℕ, (Π p -> exists N : ℕ, forall n : ℕ, (n > N -> Pr n {r : 𝕋 | (| A _0 r n | ≥ n \/ | A _1 r n | ≥ n) /\ A _0 r n <> A _1 r n /\ f (A _0 r n) = f (A _1 r n)} < (1 / p n)))). (* ...collision-resistant *)
Axiom G1 : ℙ g. (* g is... *)
Axiom G2 : forall A : 𝕋 -> 𝕋 -> ℕ -> 𝕋, (ℕℙ' A -> forall p : ℕ -> ℕ, (Π p -> exists N : ℕ, forall n : ℕ, (n > N -> Pr n {xr : 𝕋 | g (A (g xr_1) xr_2 n) = g xr_1} < (1 / p n)))). (* ...one-way *)
Axiom G3 : forall x : 𝕋, | g x | ≥ | x |. (* g is length-nondecreasing *)

(* LEMMAS, THEOREM AND PROOFS *)
Lemma L1 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋), ℕℙ' A -> exists (B : ℕ -> 𝕋 -> ℕ -> 𝕋), ℕℙ'' B /\ forall (xr : 𝕋) (n : ℕ), B _0 xr n = g (A (f (g xr_1)) xr_2 n) /\ B _1 xr n = g xr_1.

Proof.
  intros A H1.
  destruct ℕℙ1 with (A := fun (xr : 𝕋) (n : ℕ) => g (A (f (g xr_1)) xr_2 n)) (g := fun (xr : 𝕋) => g xr_1) as [B [H2 H3]].
    apply ℕℙ2 with (A := A). exact H1.
    exact F1.
    exact G1.
    apply ℙ1 with (g := g). exact G1.
```

```
    exists B.
    split.
      exact H2.
      exact H3.
Qed.

Lemma L2 : forall (A : 𝕋 -> 𝕋 -> ℕ -> 𝕋) (n : ℕ) (xr : 𝕋), | xr_1 | ≥ n -> | xr_2 | ≥ n -> (f(g(A(f(g xr_1)) xr_2 n)) = f(g xr_1) <-> g(A(f(g xr_1)) xr_2 n) = g xr_1 \/ (| g(A(f(g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g(A(f(g xr_1)) xr_2 n) <> g xr_1 /\ f(g(A(f(g xr_1)) xr_2 n)) = f(g xr_1)).

Proof.
  intros A n xr H1 H2.
  split.
    intro H3.
    destruct classic with (P := g (A (f (g xr_1)) xr_2 n) = g xr_1) as [H4 | H5].
      left. exact H4.
      right. split.
        right. apply GET1 with (x := | g xr_1 |) (y := | xr_1 |) (z := n).
          apply G3.
          exact H1.
          split.
            exact H5.
            exact H3.
    intros [H6 | [H7 [H8 H9]]].
    replace (g (A (f (g xr_1)) xr_2 n)) with (g xr_1). reflexivity.
    exact H9.
Qed.

Theorem T1 : forall A : 𝕋 -> 𝕋 -> ℕ -> 𝕋, (ℕℙ' A -> forall p : ℕ -> ℕ, (ℙ p -> exists N : ℕ, forall n : ℕ, (n > N -> Pr n {xr : 𝕋 | f(g(A(f(g xr_1)) xr_2 n)) = f(g xr_1)} < (1 / p n)))).

Proof.
  intros A H1 p H2.
  (* Introducing B such that B(0,xr,n) = g(A(f(g(xr_1))),xr_2,n) and B(1,xr,n) = g(xr_1) *)
  destruct L1 with (A := A) as [B [H3 H4]].
    exact H1.
  (* Getting N1 from collision-resistance of f *)
```

```
  destruct F2 with (A := B) (p := fun n : ℕ => 2 p n) as [N H5].
    exact H3.
    apply ℿ1 with (p := p). exact H2.
  (* Getting N2 from one-wayness of g *)
  destruct G2 with (A := fun (x y : 𝕋) (n : ℕ) => A (f x) y n) (p := fun n : ℕ => 2 p n) as [N' H6].
    apply ℕℙ3 with (A := A). exact H1. exact F1.
    apply ℿ1 with (p := p). exact H2.
  (* Taking N = max(N1,N2) *)
  exists (max N N').
  intros n H7.
  (* Spliting a probability into the sum of two probabilities *)
  apply LET1 with (x := (Pr n {xr : 𝕋 | f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)})) (y := Pr n {xr : 𝕋 | g (A (f (g xr_1)) xr_2 n) = g xr_1} + Pr n { xr : 𝕋 | (| g (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)}) (z := 1 / p n).
    apply Pr1 with (n := n) (P := fun xr : 𝕋 => f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)) (Q := fun xr : 𝕋 => g (A (f (g xr_1)) xr_2 n) = g xr_1) (R := fun xr : 𝕋 => (| g (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)).
      apply L2.
    apply S1.
      apply H6. apply max2 with (i := n) (j := N) (k := N'). exact H7.
      replace (Pr n {xr : 𝕋 | (|g (A (f (g xr_1)) xr_2 n) | ≥ n \/ | g xr_1 | ≥ n) /\ g (A (f (g xr_1)) xr_2 n) <> g xr_1 /\ f (g (A (f (g xr_1)) xr_2 n)) = f (g xr_1)}) with (Pr n {r : 𝕋 | (|B _0 r n | ≥ n \/ |B _1 r n | ≥ n) /\ B _0 r n <> B _1 r n /\ f (B _0 r n) = f (B _1 r n)}).
        apply H5. apply max1 with (i := n) (j := N) (k := N'). exact H7.
        apply Pr2 with (A := A) (B := B) (n := n). apply H4.
Qed.
```

References

[1] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2004.

[2] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. http://cseweb.ucsd.edu/ mihir/papers/gb.pdf.

[3] Rafael Pass and Chin Isradisaikul. Cryptography. https://www.cs.cornell.edu/courses/cs6830/2009fa/.

[4] Bart Preneel. The state of cryptographic hash functions. In Ivan Damgård, editor, *Lectures on Data Security – Modern Cryptology in Theory and Practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 158–182. Springer, 1999.