The Beauty and the Beast

Eerke Boiten
Centre for Cyber Security
School of Computing
University of Kent, UK

FOSAD, August 2016

Objectives

- formal methods, proofs (etc.) matter
- crypto and security: harder
- risks in proofs
- a small contribution to easier proofs
- concrete stuff
- the F in FOSAD

Cryptography and Formal Methods

Formal methods use mathematics and logic to show that software (etc.) does what it is supposed to do.

State of the art: checking (on paper/w. tools) of critical components; possibly against single properties, not full functionality; automated checking of low level properties; support in animation, test generation.

Even UK parliament 2013!

Cryptography (superficially:)
Natural bedfellow: *high* effort for *small* components that require *high* confidence.

Overview of this talk:

- familiar example
- a dogmatic hierarchy
- state of the art?
- a small contribution

Example: RSA – intro

p, q large primes, $n = pq$, $\phi = (p-1)(q-1)$

(e, n) encryption key, (d, n) decryption key such that de **mod** $\phi = 1$.

This means $de = k(p-1)(q-1)+1$ for some k.

Plaintext encoded by number $T < n$
Encrypt: $C = T^e$ **mod** n
Decrypt: C^d **mod** $n \ \ ?{=}? \ T$

Fermat: x^{y-1} **mod** $y = 1$ (for suitable x, y)

RSA: A Beauty

Lemma: $(T^e)^d$ **mod** $p = T$ **mod** p

Proof (for $T \neq 0$; all top level "=" is **mod** p):

$(T^e)^d$

$=$

T^{ed}

$= \{ed = de = k(p-1)(q-1)+1\}$
$T^{k(p-1)(q-1)+1}$

$= \{$properties of exp. and mult.$\}$
$(T^{p-1})^{k(q-1)} \times T$

$= \{$**mod** distributes over exp. and mult.$\}$
$((T^{p-1}$ **mod** $p)^{k(q-1)}$ **mod** $p) \times (T$ **mod** $p)$

$= \{$Fermat$\}$
$(1^{k(q-1)}$ **mod** $p) \times (T$ **mod** $p)$

$=$

T

and similarly $(T^e)^d$ **mod** $q = T$ **mod** q

A Beauty, ctd

T^{ed} **mod** $p = T$ **mod** p
means
$(T^{ed} - T)$ **mod** $p = 0$, i.e., p divides $T^{ed} - T$.
And so does q.

Backwards from the desired conclusion:

$$
\begin{aligned}
& (T^e)^d \textbf{ mod } n = T \\
\equiv\ & \{n = pq, \text{arithmetic}\} \\
& ((T^e)^d - T) \textbf{ mod } pq = 0 \\
\equiv\ & \{p \text{ and } q \text{ are primes, dividing their product}\} \\
& (T^{ed} - T) \textbf{ mod } p = 0 \\
& \wedge\ (T^{ed} - T) \textbf{ mod } q = 0 \\
\equiv\ & \{\text{previous slide}\} \\
& \textbf{true}
\end{aligned}
$$

A Beast: RSA is secure

Show that if factoring is hard, so is breaking RSA.

I.e.,

Let $n = pq$, of size x; p and q distinct primes.
Given a probabilistic algorithm A such that
$Pr[A(n, e, t^e \textbf{ mod } n) = t] > 0.5$,
and A runs in time polynomial in x,
show an algorithm B using A as a subroutine
that runs in total time polynomial in x
such that $Pr[B(n) = (p, q)] > 0.5$.

(Unsolved, likely not true.
True if we force A to also compute the key d.)

The point: not such beautiful proofs in modern crypto, needing also complexity, probability, asymptotics.

More Beasts in Small Print

Two PhD students have worked on taming "The Beast". The first finished in 2008, and we quote from his thesis:

> Dan Grundy: Concepts and Calculation in Cryptography

(`www.cs.kent.ac.uk/~eab2/crypto/thesis.web.pdf`) where Dan in turn is mostly quoting

> Oded Goldreich: Foundations of Cryptography: Volume I: Basic Tools. Cambridge University Press, 2001.

Jaime Gaspar is the 2nd – started 2014.

Though it remains to decide what is meant by the terms "easy", "hard", and "a high level of probability", deciding which computations should be computationally easy and which should be hard is, as you may expect, a fundamental aspect of cryptography. As mentioned, RSA is a specific construction, and the requirement that d is hard to compute is specific to that construction; in general we'd like to build a more abstract theory around a concept that underlies many different cryptographic constructions.

* *

One-way functions

The fundamental concept that underlies many cryptographic constructions is that of a so-called "one-way function". Roughly speaking, a one-way function is a function that is "easy" to compute, but "hard" on average to invert. One-way functions come in two flavours: "weak", and "strong", where strong one-way functions are "harder" to invert than weak one-way functions. The fundamental theorem that relates these concepts asserts that the existence of weak one-way functions equivales the existence of strong one-way functions, ie:

$$(5) \qquad \exists\text{weak one-way functions} \quad \equiv \quad \exists\text{strong one-way functions}$$

The following definitions of strong and weak one-way functions are taken verbatim from pages 33 and 35 of Goldreich's "Foundations of Cryptography" [28].

> A function $f : \{0,1\}^* \to \{0,1\}^*$ is called (strongly) one-way if the following two conditions hold:
>
> (1) Easy to compute: There exists a (deterministic) polynomial-time algorithm A such that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).
>
> (2) Hard to invert: For every probabilistic polynomial-time algorithm A', every positive polynomial $p(\cdot)$, and all sufficiently large n's,
>
> $$\Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

In the subsequent text we are told that:

> "U_n denotes a random variable distributed over $\{0,1\}^n$. Hence, the probability in the second condition is taken over all the possible values assigned to U_n and all possible internal coin tosses of A', with uniform probability distribution."

Weak one-way functions are defined as follows:

$$
\begin{aligned}
&\geq \Pr[B'(0^l U_{n-l}, f(U_n)) \in (0^l U_{n-l}, f^{-1}(f(U_n)))] \\
&= \Pr[B'(g(U_{2n})) \in g^{-1}(g(U_{2n}))|U_{2n} \in S_{2n}] \\
&\geq \frac{\Pr[B'(g(U_{2n})) \in g^{-1}(g(U_{2n}))] - \Pr[U_{2n} \notin S_{2n}]}{\Pr[U_{2n} \in S_{2n}]} \\
&= n \cdot \left(\Pr[B'(g(U_{2n})) \in g^{-1}(g(U_{2n}))] - \left(1 - \frac{1}{n}\right)\right) \\
&= 1 - n \cdot (1 - \Pr[B'(g(U_{2n}) \in g^{-1}(g(U_{2n}))])
\end{aligned}
$$

(For the second inequality, we used $\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}$ and $\Pr[A \cap B] \geq \Pr[A] - \Pr[\neg B]$.) It should not come as a surprise that the above expression is meaningful only in case $\Pr[B'(g(U_{2n})) \in g^{-1}(g(U_{2n}))] > 1 - \frac{1}{n}$.

It follows that for every polynomial $p(\cdot)$ and every integer n, if B' inverts g on $g(U_{2n})$ with probability greater than $1 - \frac{1}{p(2n)}$, then A' inverts f on $f(U_n)$ with probability greater than $1 - \frac{n}{p(2n)}$. Hence, if g is not weakly one-way (i.e., for every polynomial $p(\cdot)$ there exist infinitely many m's such that g can be inverted on $g(U_m)$ with probability $\geq 1 - 1/p(m)$), then also f is not weakly one-way (i.e., for every polynomial $q(\cdot)$ there exist infinitely many n's such that f can be inverted on $f(Un)$ with probability $\geq 1 - 1/q(n)$, where $q(n) = p(2n)/n$). This contradicts our hypothesis (that f is weakly one-way).

To summarize, given a probabilistic polynomial-time algorithm that inverts g on $g(U_{2n})$ with success probability $1 - \frac{1}{n} + \alpha(n)$, we obtain a probabilistic polynomial-time algorithm that inverts f on $f(U_n)$ with success probability $n \cdot \alpha(n)$. Thus, since f is (weakly) one-way, $n \cdot \alpha(n) < 1 - (1/q(n))$ must hold for some polynomial q, and so g must be weakly one-way (since each probabilistic polynomial-time algorithm trying to invert g on $g(U_{2n})$ must fail with probability at least $\frac{1}{n} - \alpha(n) > \frac{1}{n \cdot q(n)}$).

* *

Goldreich's proof that

$\exists$weak one-way functions $\Rightarrow$ $\exists$strong one-way functions

*

Let f be a weak one-way function, and let p by the polynomial guaranteed by the definition of a weak one-way function. Namely, every probabilistic polynomial-time algorithm fails to invert f on $f(U_n)$ with probability at least $\frac{1}{p(n)}$. We assume for simplicity that f is length-preserving (i.e. $|f(x)| = |x|$ for all x's). This assumption, which is not really essential, is justified by Proposition 2.2.5. We define a function g as follows:

$$g(x_1, \ldots, x_{t(n)}) \stackrel{\text{def}}{=} f(x_1), \ldots, f(x_{t(n)}) \qquad (2.5)$$

where $|x_1| = \cdots = |x_{t(n)}| = n$ and $t(n) \stackrel{\text{def}}{=} n \cdot p(n)$. Namely, the $n^2p(n)$-bit-long input of g is partitioned into $t(n)$ blocks, each of length n, and f is applied to each block.

Clearly, g can be computed in polynomial-time (by an algorithm that breaks the input into blocks and applies f to each block). Furthermore, it is easy to see that inverting g on $g(x_1, \ldots, x_{t(n)})$ requires finding a pre-image to each $f(x_i)$. One may be tempted to deduce that it is also clear that g is a strongly one-way function. A naive argument might proceed by assuming implicitly (with no justification) that the inverting algorithm worked separately on each $f(x_i)$. If that were indeed the case, then the probability that an inverting algorithm could successfully invert all $f(x_i)$ would be at most $(1 - \frac{1}{p(n)})^{n \cdot p(n)} < 2^{-n}$ (which is negligible also as a function of $n^2p(n)$). However, the assumption that an algorithm trying to invert g works independently on each $f(x_i)$ cannot be justified. Hence, a more complex argument is required.

Following is an outline of our proof. The proof that g is strongly one-way proceeds by a contradiction argument. We assume, on the contrary, that g is not strongly one-way; namely, we assume that there exists a polynomial-time algorithm that inverts g with probability that is not negligible. We derive a contradiction by presenting a polynomial-time algorithm that, for infinitely many n's, inverts f on $f(U_n)$ with probability greater than $1 - \frac{1}{p(n)}$ (in contradiction to our hypothesis). The inverting algorithm for f uses the inverting algorithm for g as a subroutine (without assuming anything about the manner in which the latter algorithm operates). (We stress that we do not assume that the g-inverter works in a particular way, but rather use any g-inverter to construct, in a generic way, an f-inverter.) Details follow.

Suppose that g is not strongly one-way. By definition, it follows that there exists a probabilistic polynomial-time algorithm B' and a polynomial $q(\cdot)$ such that for infinitely many m's,

$$\Pr[B'(g(U_m)) \in g^{-1}(g(U_m))] > \frac{1}{q(m)} \tag{2.6}$$

Let us denote by M' the infinite set of integers for which this holds. Let N' denote the infinite set of n's for which $n^2 \cdot p(n) \in M'$ (note that all m's considered are of the form $n^2 \cdot p(n)$, for some integer n).

Using B', we now present a probabilistic polynomial-time algorithm A' for inverting f. On input y (supposedly in the range of f), algorithm A' proceeds by applying the following probabilistic procedure, denoted I, on input y for $a(|y|)$ times, where $a(\cdot)$ is a polynomial that depends on the polynomials p and q (specifically, we set $a(n) \stackrel{\text{def}}{=} 2n^2 \cdot p(n) \cdot q(n^2p(n))$).

Procedure I

Input: y (denote $n \stackrel{\text{def}}{=} |y|$).

For $i = 1$ to $t(n)$ do begin

1. Select uniformly and independently a sequence of strings $x_1, \ldots, x_{t(n)} \in \{0,1\}^n$.
2. Compute $(z_1, \ldots, z_{t(n)}) \leftarrow B'(f(x_1), \ldots, f(x_{i-1}), y, f(x_{i+1}), \ldots, f(x_{t(n)}))$.
3. If $f(z_i) = y$, then halt and output z_i.

(This is considered a *success*).

end

Using Eq. (2.6), we now present a lower bound on the success probability of algorithm A'. To this end we define a set, denoted S_n, that contains all n-bit strings on which the procedure I succeeds with non-negligible probability (specifically, greater than $\frac{n}{a(n)}$). (The probability is taken only over the coin tosses of procedure I.) Namely,

$$S_n \stackrel{\text{def}}{=} \left\{ x : \mathsf{Pr}[I(f(x)) \in f^{-1}(f(x))] > \frac{n}{a(n)} \right\}$$

In the next two claims we shall show that S_n contains all but at most a $\frac{1}{2p(n)}$ fraction of the strings of length $n \in N'$ and that for each string $x \in S_n$ the algorithm A' inverts f on $f(x)$ with probability exponentially close to 1. It will follow that A' inverts f on $f(U_n)$, for $n \in N'$, with probability greater than $1 - \frac{1}{p(n)}$, in contradiction to our hypothesis.

Claim 2.3.2.1: For every $x \in S_n$,

$$\mathsf{Pr}[A(f(x)) \in f^{-1}(f(x))] > 1 - \frac{1}{2^n}$$

Proof: By definition of the set S_n, the procedure I inverts $f(x)$ with probability at least $\frac{n}{a(n)}$. Algorithm A' merely repeats I for $a(n)$ times, and hence

$$\mathsf{Pr}[A'(f(x)) \notin f^{-1}(f(x))] < \left(1 - \frac{n}{a(n)}\right)^{a(n)} < \frac{1}{2^n}$$

The claim follows. □

Claim 2.3.2.2: For every $n \in N'$,

$$|S_n| > \left(1 - \frac{1}{2p(n)}\right) \cdot 2^n$$

Proof: We assume to the contrary, that $|S_n| \leq (1 - \frac{1}{2p(n)}) \cdot 2^n$. We shall reach a contradiction to Eq. (2.6) (i.e., our hypothesis concerning the success probability of B'). Recall that by this hypothesis (for $n \in N0$),

$$s(n) \stackrel{\text{def}}{=} \mathsf{Pr}[B'(g(U_{n^2p(n)})) \in g^{-1}(g(U_{n^2p(n)}))] > \frac{1}{q(n^2p(n))} \tag{2.7}$$

Let $U_n^{(1)}, \ldots, U_n^{(n \cdot p(n))}$ denote the n-bit-long blocks in the random variable $U_{n^2p(n)}$ (i.e., these $U_n^{(i)}$'s are independent random variables each uniformly distributed in $\{0,1\}^n$). We partition the

Formal methods dogmatism: a hierarchy of approaches

How to relate a solution and its mathematical properties? In order of preference . . .

0. construct solution from properties ("derivation");

1. construct proof that solution satisfies properties ("post-hoc verification");

2. be able to verify any proof that solution satisfies properties ("proof checking");

3. having such proofs.

All vs. *Some* properties . . .

In Crypto: 3. Having Proofs

State of the art: yes. Conference culture, so quick checks by few.

2. Proof Checking

State of the art: some support for game-hopping proofs

1. Verification

State of the art: fully verified SSL (MS/Inria), game-hopping proofs (CryptoVerif), process algebra based checking of high level properties (ProVerif, CSP) – abstraction risks

0. Derivation

State of the art: Little done (Carroll Morgan et al. closest)

The logic of large enough

(Boiten and Grundy, MPC 2010).

Asymptotic reasoning, complexity theory: properties hold "for large enough" numbers.

$$\langle \exists X :: \langle \forall x : x > X : P \rangle \rangle$$

Assumption throughout: x and X are natural numbers.

Name this quantifier!
"Almost all", "all but finitely many"

$$\langle \diamondsuit x :: P \rangle \quad \equiv \quad \langle \exists X :: \langle \forall x : x > X : P \rangle \rangle$$

Intuition/notation: $\mathbb{N}$ as a timeline . . .

Known since 1970s, modal quantifiers since 1950s. (Emphasis: expressiveness, general class.)

Why useful?

- often left implicit in context – "not done", real risks (order?);
- where it *is* explicit: *two* dummies;
- its negation also interesting;
- basis for more notations (eliminate more dummies).

Basic properties (unsurprising)

If x does not occur free in P then

$$\langle \Diamond x :: P \rangle \equiv P$$

Proof:

$\langle \Diamond x :: P \rangle$

$\equiv$ { definition of $\Diamond$ }

$\langle \exists X :: \langle \forall x : x > X : P \rangle \rangle$

$\equiv$ { $\forall$ not trivialised: non-empty range }

$\langle \exists X :: P \rangle$

$\equiv$ { $\exists$ not trivialised: non-empty range }

P

If x does not occur free in E then

$$\langle \Diamond x :: x > E \rangle \equiv \textbf{true}$$

From monotonicity of $\forall$ and $\exists$ we get:

$$\langle \forall\, x :: P \Rightarrow Q \rangle \;\Rightarrow\; (\langle \diamondsuit x :: P \rangle \Rightarrow \langle \diamondsuit x :: Q \rangle) \quad (0)$$

Intuitively

$$\langle \forall\, x :: P \rangle \quad \Rightarrow \quad \langle \diamondsuit x :: P \rangle$$

proved using (0):

$\langle \forall\, x :: P \rangle$

$\equiv$ { left identity of $\Rightarrow$, heading for (0) }

$\langle \forall\, x :: \mathbf{true} \Rightarrow P \rangle$

$\Rightarrow$ { (0) with $P, Q := \mathbf{true}, P$ }

$\langle \diamondsuit x :: \mathbf{true} \rangle \;\Rightarrow\; \langle \diamondsuit x :: P \rangle$

$\equiv$ { x doesn't occur in **true** }

$\mathbf{true} \;\Rightarrow\; \langle \diamondsuit x :: P \rangle$

$\equiv$ { left identity of $\Rightarrow$ }

$\langle \diamondsuit x :: P \rangle$

Conjunction

Conjunction distributes over $\diamondsuit$:

$$\langle \diamondsuit x :: P \rangle \ \wedge \ \langle \diamondsuit x :: Q \rangle \quad \equiv \quad \langle \diamondsuit x :: P \ \wedge \ Q \rangle$$

Proof of $\Leftarrow$ from weakening;
$\Rightarrow$ starting with witnesses X_0 and X_1:

$\langle \forall\, x : x > X_0 : P \rangle \ \wedge \ \langle \forall\, x : x > X_1 : Q \rangle$

$\quad \Rightarrow \ \{ \ \text{arithmetic} \ \}$

$\langle \forall\, x : x > X_0 \uparrow X_1 :: P \rangle \ \wedge \ \langle \forall\, x : x > X_0 \uparrow X_1 : Q \rangle$

$\quad \equiv \ \{ \ \text{distributivity} \ \}$

$\langle \forall\, x : x > X_0 \uparrow X_1 :: P \ \wedge \ Q \rangle$

$\quad \Rightarrow \ \{ \ \exists\text{-introduction, with} \ \ X := X_0 \uparrow X_1 \ \}$

$\langle \exists\, X :: \langle \forall\, x : x > X : P \ \wedge \ Q \rangle \rangle$

$\quad \equiv \ \{ \ \text{definition of} \ \diamondsuit \ \}$

$\langle \diamondsuit x :: P \ \wedge \ Q \rangle$

So in every proof about "large enough" you have this implicit "big X" which you need to find a value for. Here, we're avoiding having to mention this all the time – we need to refer to it once in a proof of a property of $\Diamond\!\!\!\!\Box$, and then never again.

From binary to general conjunction

Weakening also accounts for

$$\langle \forall\, y :: \langle \diamondsuit\!\!\!\!\diamond\, x :: P \rangle \rangle \;\Leftarrow\; \langle \diamondsuit\!\!\!\!\diamond\, x :: \langle \forall\, y :: P \rangle \rangle$$

but $\Rightarrow$ proof doesn't carry across: would need maximum of possibly infinite set. Indeed it doesn't hold: consider $x \geq y$ for P .

Consequence: *having universally* and *large-enough quantified variables implicit in context at the same time is ambiguous!*

Multiple large enough

Definition for two simultaneous dummies:

$$\langle \diamondsuit\!\!\!\!\square\, x, y :: P \rangle \quad \equiv \quad \langle \exists\, X, Y :: \langle \forall\, x, y : x > X \wedge y > Y : P \rangle \rangle$$

Equivalently:

$$\langle \diamondsuit\!\!\!\!\square\, x, y :: P \rangle \;\equiv\; \langle \exists\, Z :: \langle \forall\, x, y : x > Z \wedge y > Z : P \rangle \rangle$$

Nesting property:

$$\langle \diamondsuit\!\!\!\!\square\, x, y :: P \rangle \quad \Rightarrow \quad \langle \diamondsuit\!\!\!\!\square\, x :: \langle \diamondsuit\!\!\!\!\square\, y :: P \rangle \rangle$$

(Proof next slide)

Proof:

$\langle \diamondsuit\!\!\!\!\diamondsuit\, x, y :: P \rangle$

$\quad \equiv \quad \{ \quad$ definition of $\diamondsuit\!\!\!\!\diamondsuit \quad \}$

$\langle \exists X, Y :: \langle \forall x, y : x > X \wedge y > Y : P \rangle \rangle$

$\quad \equiv \quad \{ \quad$ nesting of $\forall$ and of $\exists \quad \}$

$\langle \exists X :: \langle \exists Y :: \langle \forall x : x > X : \langle \forall y : y > Y : P \rangle \rangle \rangle \rangle$

$\quad \Rightarrow \quad \{ \quad \exists \forall \Rightarrow \forall \exists \quad \}$

$\langle \exists X :: \langle \forall x : x > X : \langle \exists Y : \langle \forall y : y > Y : P \rangle \rangle \rangle \rangle$

$\quad \equiv \quad \{ \quad$ definition of $\diamondsuit\!\!\!\!\diamondsuit$, twice $\quad \}$

$\langle \diamondsuit\!\!\!\!\diamondsuit\, x :: \langle \diamondsuit\!\!\!\!\diamondsuit\, y :: P \rangle \rangle$

$\Leftarrow$ does not hold: $y > x$ for P . (Same counterexample stops swapping $\diamondsuit\!\!\!\!\diamondsuit\, x$ and $\diamondsuit\!\!\!\!\diamondsuit\, y$.)

Consequence: *if multiple* $\diamondsuit\!\!\!\!\diamondsuit$ *quantified variables in context, have to be explicit about their order (or lack of).*

Infinitely many

The dual of $\langle\!\boxed{\diamond}\!\rangle$:

$$\langle\!\boxed{\diamond}\, x :: P\rangle \quad \equiv \quad \neg\langle\!\langle\diamond\!\rangle\!\rangle\, x :: \neg P\rangle$$

Consequently,

$$\langle\!\boxed{\diamond}\, x :: P\rangle \quad \equiv \quad \langle \forall\, X :: \langle \exists\, x : x > X : P\rangle\rangle$$

and also

$$\langle\!\boxed{\diamond}\, x :: P\rangle \quad \equiv \quad \langle x : P : x\rangle \text{ is infinite}$$
$$\langle\!\langle\diamond\!\rangle\!\rangle\, x :: P\rangle \quad \equiv \quad \langle x : \neg P : x\rangle \text{ is finite}$$

(Contradiction proofs of "large enough" via "infinitely many"!)

Towards calculational asymptotics

Asymptotics: "at extremes things get very close".
$\diamondsuit\!\!\!\!\Box$ eliminates quantification of "how extreme" but not of "how close" (ϵ).

Asymptotic comparisons (based on absolute differences):

$$f \leftrightarrow g \quad \equiv \quad \langle \forall\, \epsilon : \epsilon > 0 : \langle \diamondsuit\!\!\!\!\Box\, x :: |f.x - g.x| < \epsilon \rangle \rangle$$

$$f \triangleleft g \quad \equiv \quad \langle \forall\, \epsilon : \epsilon > 0 : \langle \diamondsuit\!\!\!\!\Box\, x :: 0 \leq g.x - f.x < \epsilon \rangle \rangle$$

Properties . . .

Asymptotic comparisons (relative differences):

$$
\begin{array}{lcl}
f \prec g & \equiv & f/g \leftrightarrow 0 \\
f \ll g & \equiv & \langle \exists\, C :: \langle \diamondsuit\!\!\!\!\cdot\, x :: |f.x| \leq C \cdot |g.x| \rangle \rangle \\
f \asymp g & \equiv & f \ll g \ \wedge \ g \ll f \\
f \sim g & \equiv & f/g \leftrightarrow 1
\end{array}
$$

Properties: order-like, and

$$
\begin{array}{rcll}
f \leftrightarrow g \ \wedge \ \neg(f \leftrightarrow 0) & \Rightarrow & f \sim g \\
\sim & \subseteq & \asymp \\
\asymp & \subseteq & \ll \\
\asymp \ ; \ \prec & \subseteq & \prec \\
\prec \ ; \ \asymp & \subseteq & \prec
\end{array}
$$

Final comment

Proofs in crypto are hard and confusing. Maybe
 helps a little.

Design of notation in general.