Random Self Reducibility

This report was written as part of the 2025 edition of the self-study PhD course "Cryptographic Proof Techniques" by Prof. Elena Pagnin at Chalmers University of Technology, Göteborg, SE.

1 Random-self-reducible functions

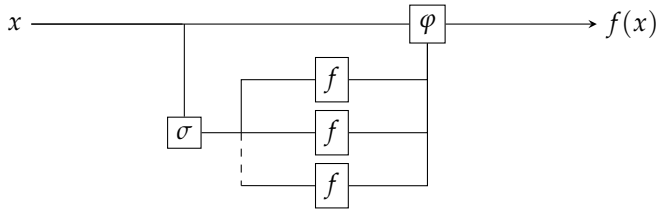
Without loss of generality, we can define a *decision problem* as a set D of binary strings where we are asked if a given $x \in \{0,1\}^*$ lies in D. Similarly, a *search problem* is a set S of binary strings (x, y) where given x we are asked to find an y such that $(x, y) \in \mathsf{S}$.

▶ **Definition 1** *A* poly-time reduction from problem P_1 to problem P_2 ($\mathsf{P}_1 \leq_p \mathsf{P}_2$) *is a poly-time algorithm transforming any instance x_1 of P_1 into an instance x_2 of P_2 such that $x_1 \in \mathsf{P}_1$ if and only if $x_2 \in \mathsf{P}_2$.*

Consider a function $f : \{0,1\}^* \to \{0,1\}^*$, whose inputs x have length $\mathsf{len}(x) = n$. We denote by r a sequence of fair coin tosses, with $\mathsf{len}(r) = \omega(n)$ for $\omega(n) = \mathsf{poly}(n)$.
We wish to study reductions running in probabilistic polynomial time ("poly-time"). They will perform $k(n) = \mathsf{poly}(n)$ random queries to f. We consider the following definitions, found for instance in Feigenbaum and Fortnow [1].

▶ **Definition 2** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is* nonadaptively $k(n)$-random-self-reducible *(nonadaptively k-rsr) if there are two polynomial-time computable functions σ and φ such that*

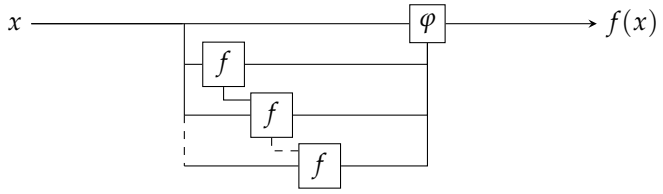
1. *for all n and all $x \in \{0,1\}^n$, for at least $3/4$ of all r in $\{0,1\}^{\omega(n)}$, we can compute $f(x)$ as*
$$f(x) = \varphi\big(x, r, f(\sigma(1,x,r)), \ldots, f(\sigma(k,x,r))\big)$$
2. *for all n, if r is an uniform random variable, then the random variables $\sigma(i, x_1, r)$ and $\sigma(i, x_2, r)$ are identically distributed for all $\{x_1, x_2\} \in \{0,1\}^n$ and all $i = 1, \ldots, k$.*



Intuitively, f is nonadaptively k-rsr if evaluating f on any size-n instance x can be reduced via φ to evaluating f on k size-n instances $\sigma(1,x,r), \ldots, \sigma(k,x,r)$. They are randomised by the same uniformly random variable r, which models $\omega(n)$ fair coin tosses.
Feigenbaum and Fortnow generalise this approach to multiround adaptive strategies.

▶ **Definition 3** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is* adaptively $k(n)$-random-self-reducible *(adaptively k-rsr) if there is a poly-time oracle machine φ that, on input x of $\mathsf{len}(x) = n$, produces $k(n)$ rounds of oracle f-queries $y_i(x, r)$ to f, where the i-th query y_i might depend on previous queries and answers. The reduction φ must be such that*

1. *for all n and all $x \in \{0,1\}^n$, for at least $3/4$ of all r in $\{0,1\}^{\omega(n)}$, it outputs the correct answer $f(x)$.*
2. *for all n, if r is chosen uniformly at random, then the random variables $y_i(x_1, r)$ and $y_i(x_2, r)$ are identically distributed for all $\{x_1, x_2\} \in \{0,1\}^n$ and all $i = 1, \ldots, k$, except if wrong answers were given on earlier rounds.*



The key difference lies in the generation of queries. Nonadaptive k-rsr requires the reduction φ to generate all oracle queries $\sigma(i, x, r)$ in parallel, solely based on the input x and the initial randomness r. Adaptive k-rsr allows it to proceed sequentially over k rounds, where the i-th query y_i may depend on $y_1, \ldots, y_{i-1}$. This latter approach is for instance better suited when the solution is found by narrowing the solution space.

An adaptively $k(n)$-rsr function f is clearly also nonadaptively $k(n)$-rsr, but the converse was shown to be false by Feigenbaum et al. [2] with the construction of a set L'' such that $f = \chi_{L''}$ is adaptively k-rsr. A function f is called *poly-rsr* if it is adaptively or non-adaptively k-rsr for a polynomial number of queries $k = k(n)$.

In cryptography, we are interested in the case where f is a function solving a computational problem $\mathcal{P}$ whose hardness is a cryptographic security assumption. This means that finding $f(x)$ for an instance x of a problem $\mathcal{P}$, which depends on a set of parameters, is considered computationally challenging. Here $n = \mathsf{len}(x)$ can be treated as a security parameter.

2 Rsr problems for classical cryptography

2.1 The Discrete Logarithm problem

▶ **Problem 4 (DLOG)** *Consider a finite cyclic group $\mathbb{G} = \langle g \rangle$ of order n.*

- *Parameters: a description $\Gamma = (\mathbb{G}, g)$ of the group $\mathbb{G}$.*
- *Instance: an element $h \in \mathbb{G}$.*
- *Task: find the unique $e \in \mathbb{Z}_n$ such that $g^e = h$.*

For a nice analysis of the DLOG problem and its applications, see Joux et al [3]. The following lemma is often proven without most of the formalism in Theorem 2.

▶ **Lemma 5** *The $DLOG_{\mathbb{G},g}$ problem is poly-rsr.*

Proof. Given a fixed instance $x = h \in \mathbb{G}$, the discrete logarithm problem requires us to find $f(x) = \text{dlog}(x)$. Sample an uniformly random integer $r \in \mathbb{Z}_n$ of length $\omega(\mathsf{len}(x)) = \mathsf{len}(x)$, which is $\mathsf{poly}(\mathsf{len}(x))$, then $\sigma(1, x, r) = hg^r$ behaves as an uniformly random variable in $\mathbb{G}$. Given two instances $x = g^e$ and $\tilde{x} = g^{\tilde{e}}$, consider $y = \sigma(1, x, r)$ and $\tilde{y} = \sigma(1, \tilde{x}, r)$, then for some arbitrary $z = g^a \in \mathbb{G}$ and a uniformly random variable r

$$\begin{aligned}\mathbb{P}[y = z] = \mathbb{P}[xg^r = z] &= \mathbb{P}[g^{e+r} = g^a] \\ &= \mathbb{P}[r = a - e] = \frac{1}{n} \\ &= \mathbb{P}[r = a - \tilde{e}] = \mathbb{P}[\tilde{x}g^r = z] \\ &= \mathbb{P}[\tilde{y} = z]\end{aligned}$$

Finally, we exhibit the reduction $\varphi(x, r, f(\sigma(1, x, r))) = f(\sigma(1, x, r)) - r$, which is correct since

$$\begin{aligned}\varphi(x, r, f(\sigma(1, x, r))) &= \varphi(h, r, f(hg^r)) \\ &= f(hg^r) - r \\ &= (e + r) - r = f(x)\end{aligned}$$

This function satisfies the condition of 2, and the problem is nonadaptive k-rsr for $k = 1$. □

Remark how we exploited the omogeneous properties of $f = \text{dlog}$ in $f(xg^r) = f(x) + r$.
Formally, one should mind the distribution $\mathcal{D}$ used to sample instances and require that the $\sigma(i, x, r)$ are valid instances of the problem. This also implies verifying that they are random variables of distribution $\mathcal{D}$. We implicitly proved it by verifying $\mathcal{P}[y = z] = 1/n$.

2.2 The RSA Inversion problem

▶ **Problem 6 (RSA Inversion)** *Consider an RSA modulus $N = pq$ with distinct primes p, q, and a public exponent e such that $\gcd(e, \varphi(N)) = 1$.*

- *Parameters: the system description (N, e).*
- *Instance: an element $x \in \mathbb{Z}_N^*$.*
- *Task: find the unique $y \in \mathbb{Z}_N^*$ such that $y^e \equiv x \mod N$*

▶ **Lemma 7** *The $RSAI_{N,e}$ problem is poly-rsr.*

Proof. Given a fixed instance $x \in \mathbb{Z}_N^*$, the RSA Inversion problem requires us to find $f(x) = y$, the e-th root of x modulo N. Sample an uniformly random integer $r \in \mathbb{Z}_N$ of length $\omega(\mathsf{len}(x)) = \mathsf{len}(x)$, which is $\mathsf{poly}(\mathsf{len}(x))$. Define $\sigma(1, x, r) = xr^e$, this behaves as an uniformly random variable in $\mathbb{Z}_N$. Given two instances $x = y^e$ and $\tilde{x} = \tilde{y}^e$, define $z = \sigma(1, x, r)$ and $\tilde{z} = \sigma(1, \tilde{x}, r)$, then for some arbitrary instance $\overline{x} = \overline{y}^e \in \mathbb{Z}_N^*$ and a uniformly random variable r

$$\begin{aligned}\mathbb{P}[z = \overline{x}] = \mathbb{P}[xr^e = \overline{y}^e] &= \mathbb{P}[(yr)^e = \overline{y}^e] \\ &= \mathbb{P}[r = \overline{y}/y] = \frac{1}{\varphi(N)} \\ &= \mathbb{P}[r = \overline{y}/\tilde{y}] = \mathbb{P}[\tilde{x}r^e = \overline{y}^e] \\ &= \mathbb{P}[\tilde{z} = \overline{x}]\end{aligned}$$

We define the reduction $\varphi(x, r, f(\sigma(1, x, r))) = f(\sigma(1, x, r))/r$, which is correct since

$$\begin{aligned}\varphi(x, r, f(\sigma(1, x, r))) &= \varphi(x, r, f(xr^e)) \\ &= yr/r = f(x)\end{aligned}$$

therefore RSA inversion is nonadaptive k-rsr for $k = 1$. □

3 Rsr problems for post-quantum cryptography

3.1 The Endomorphism Ring problem

An *elliptic curve* is an algebraic variety

$$E = \left\{(x, y) \in \overline{\mathbb{F}}_{q^2} \times \overline{\mathbb{F}}_{q^2} \cup \{\mathcal{O}\} \mid y^2 = x^3 + Ax + B\right\}$$

where the discriminant $\Delta = -16(4A^3 + 27B^2)$ is non-zero. For theoretical reasons[1] we focus on elliptic curves over the field extension $\mathbb{F}_{q^2}$.
An *isogeny* is a surjective morphism $\psi : E \to E'$ such that for any $P, Q \in E$ we have $\psi(P + Q) = \psi(P) + \psi(Q)$. Its *degree* is the degree of ψ as a rational function, and its *representation* is any data that encodes the domain, the codomain, the degree, and an algorithm to efficiently evaluate the image of any point. Isogenies of degree n, also called n-isogenes, form form a graph whose vertices are elliptic curves and edges are isogenies. See Figure 1 for an example.

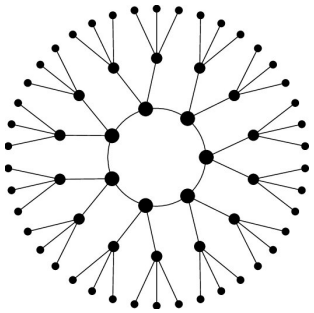


Figure 1: The 3-isogeny graph representing the isogeny class of the curves E with $j(E) = 607$ over F_{6007}. The different sizes of vertices represent different sizes of endomorphism rings. Michael Meyer, `https://doi.org/10.25972/OPUS-24682`, CC BY-SA 4.0.

The *endomorphisms* $\iota : E \to E$ of an elliptic curve E form a ring $\mathrm{End}(E)$. They include for instance the "trivial" endomorphisms $P \mapsto k \cdot P$, which act as scalar multiplication by a fixed k. In our case, $\mathrm{End}(E)$ is a four-dimensional $\mathbb{Z}$-module.

The book by Silverman [4] provides a more rigorous introduction to elliptic curves. For the applications to cryptography and efficient representations, see the survey by Robert [5].

[1] $\mathbb{F}_{q^2}$ is the smallest field extension of $\mathbb{F}_q$ where all the 2-torsion points (points P such that $2P = \mathcal{O}$) are defined.

▶ **Problem 8 (Endomorphism Ring)** *Consider the finite field* $\mathbb{F}_{q^2}$.

- *Parameters: the field size* q.
- *Instance: a supersingular elliptic curve* E *over* $\mathbb{F}_{q^2}$.
- *Task: find the endomorphisms generating* $\mathrm{End}(E)$ *as a four-dimensional* $\mathbb{Z}$*-module.*

We will only sketch the proof of the following Lemma, which would require an heavy introduction to the theory of elliptic curves, isogenies, and efficient representations. Still, it is worth exploring why it requires relaxing the definition of nonadaptive k-rsr function. This proof is similar to that of Proposition 7.4 by Le Merdy and Wesolowski [6].

▶ **Lemma 9** *If we allow for the statistical indistinguishability of the distribution of the random variables* $\sigma(i, x, r)$ *in Theorem 2,* $\mathrm{EndRing}_q$ *is poly-rsr.*

Sketch of proof. Consider a fixed instance $x = E$ of $\mathrm{EndRing}_q$. The random variables $\sigma(i, x, r)$ sample an isogeny ψ_i as follows:

- Via Velu's formulae [4], we can uniquely characterise any isogeny ψ_i via its kernel $\ker(\psi_i)$. The degree $\deg(\psi_i)$ is the order of the points generating $\ker(\psi_i)$.
- Construct a degree-2^ℓ isogeny ψ_i. This is done via a computational trick, which is constructing a degree-2 isogeny and then composing it ℓ times:

$$\psi_i = \underbrace{\psi_i^{(2)} \circ \psi_i^{(2)} \circ \cdots \circ \psi_i^{(2)}}_{\ell \text{ times}}$$

- The degree-2 isogeny $\psi_i^{(2)}$, in turn, is constructed by sampling a basis P_1, P_2 of the 2-torsion subgroup $E[2]$, the set of points of order 2 in E.

The output ψ_i is a random walk on the 2-isogeny graph. We need to sample isogenies of degree 2^ℓ, where ℓ is actually $\ell > \log_2 p^2 + \epsilon$. The *Mixing Time Theorem* [7, Theorem 11] shows that doing this for a reasonable choice[2] of ε implies that the distribution of ψ_i in the graph is asymptotically indistinguishable from the stationary distribution: their statistical distance Δ converges to zero. Define the elliptic curve $E_i' = \sigma(i, x, r) = \psi_i(E)$. We have that E_i' is a new randomised instance of $\mathrm{EndRing}_q$. The reduction φ, on input the four generators $\iota_1', \iota_2', \iota_3', \iota_4'$ of $\mathrm{End}(E_i')$, returns

$$\iota_i = \psi_i \circ \iota_i' \circ \hat{\psi}_i$$

$$\begin{array}{ccc} E & \xrightarrow{\psi_i} & E_i' \\ {\scriptstyle \iota_i}\downarrow & & \downarrow{\scriptstyle \iota_i'} \\ E & \xleftarrow[\hat{\psi}_i]{} & E_i' \end{array}$$

[2] Define a function $\tau(\varepsilon, q) = O(\log(q) - \log(\varepsilon))$. We need to pick $\ell = \lceil \tau(1, 1/q) \rfloor$. See [6, Corollary 2.8, Proposition 7.4].

where $\hat{\psi}_i$ is the dual isogeny of ψ_i, computable in poly-time. For a non-trivial endomorphism ι'_i, we get a non-trivial ι_i.
This procedure has some failure rate, as we could end in a subring $R \subsetneq \mathrm{End}(E)$, but it can be proved it satisfies the bound in Theorem 2 for $k = 1$. □

Remark how we should also mind the distribution $\mathcal{D}$ sampling EndRing instances, and prove that the random variables $\sigma(i, x, r)$ are distributed accordingly. See [6] for more details.

3.2 The Group Action Inverse problem

Consider a multiplicative group $\mathbb{G}$ and a set X. Many of the post-quantum security assumptions in NIST standards and competitors can be modelled with the formalism of group actions.

▶ **Definition 10** *We say that $\mathbb{G}$ acts on X if there exists a map $\star : \mathbb{G} \times X \to X$ such that*

- *for every $x \in X$, we have $e \star x = x$, where e is the identity element in $\mathbb{G}$.*
- *for every $g, h \in \mathbb{G}$ and every $x \in X$, we have $(gh) \star x = g \star (h \star x)$.*

This map $\star$ is called group action.

A group action partitions X into *orbits*; the orbit associated to x is the set $\mathbb{G}_x = \{g \star x \mid g \in \mathbb{G}\}$.

▶ **Definition 11** *A group action is transitive if and only if it only induces one orbit on X. In other words, for any given $x, y \in X$ there exists some $g \in \mathbb{G}$ such that $y = g \star x$.*

For a group action to be suitable for cryptographic use, we require $X, \mathbb{G}$ to be finite and the existence of probabilistic poly-time algorithms performing the following operations:

1. **Membership.** Given a string g, decide if it represents an element of $\mathbb{G}$. Given a string x, decide if it represents an element of X.
2. **Equality.** Given $g, h \in \mathbb{G}$, decide if $g = h$.
3. **Random sampling.** Sample an element in $\mathbb{G}$ with given distribution $\mathcal{D}_G$ on $\mathbb{G}$.
4. **Group operations.** Given $g, h \in \mathbb{G}$, compute g^{-1} and gh.
5. **Action.** Given $g \in \mathbb{G}$ and $x \in X$, compute $g \star x$.
6. **Representation.** Given an element $g \in \mathbb{G}$ or $x \in X$, return the (not necessarily unique) representation as string in $\{0,1\}^{\#\mathbb{G}}$.

▶ **Problem 12 (Group Action Inverse)** *Consider a set X and a group $\mathbb{G}$.*

- *Parameters: a group action $\star$ of $\mathbb{G}$ on X.*
- *Instance: two elements x, y of the set X.*
- *Task: find, if any, an element g such that $x = g \star y$.*

Denote by $\delta(y, x)$ the element(s) g such that $x = g \star y$. If the action is transitive, the solution of GAIP is the unique $\delta(y, x)$. The following was formalised by D'Alconzo [8].

▶ **Lemma 13** *If $\star$ is transitive, $GAIP_{\mathbb{G},X,\star}$ is poly-rsr.*

Proof. Consider the fixed instance (x, y) of GAIP. we are asked to find an $f(x, y)$ such that $x = f(x, y) \star y$. The function $\sigma(i, x, r)$ generates two uniformly random group elements $g_{i,x}$ and $g_{i,y}$, returning

$$\sigma(i, (x, y), r) = (g_{i,x} \star x, g_{i,y} \star y) = (x', y')$$

The outputs of σ are truly uniformly distributed: consider the random variables

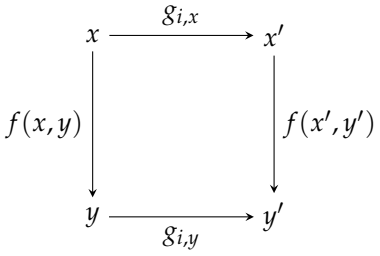
$$(x', y') = \sigma(i, (x, y), r)$$
$$(\tilde{x'}, \tilde{y'}) = \sigma(i, (\tilde{x}, \tilde{y}), r)$$

and two $z, w \in X$. If σ samples uniformly random group elements, then

$$\begin{aligned}\mathbb{P}[(x', y') = (z, w)] &= \mathbb{P}[(g_{i,x} \star x, g_{i,y} \star y) = (z, w)] \\ &= \mathbb{P}[g_{i,x} = \delta(x, z), g_{i,y} = \delta(y, w)] \\ &= \frac{i}{|\mathbb{G}|^2} = \mathbb{P}[g_{i,\tilde{x}} = \delta(\tilde{x}, z), g_{i,\tilde{y}} = \delta(\tilde{y}, w)] \\ &= \mathbb{P}[(\tilde{x'}, \tilde{y'}) = (z, w)]\end{aligned}$$

Finally, we define the function φ as

$$\begin{aligned}\varphi\big((x, y), r, f(\sigma(i, (x, y), r))\big) &= \varphi\big((x, y), r, f(x', y')\big) \\ &= g_{i,x}^{-1} \cdot f(x', y') \cdot g_{i,y}\end{aligned}$$



The reduction φ is correct with probability 1. In fact, with $k = 1$,

$$\begin{aligned}x &= g_{1,x}^{-1} \star x' = g_{1,x}^{-1} \star \big(f(x', y') \star y'\big) = g_{1,x}^{-1} \star \big(f(x', y') \star (g_{1,y} \star y)\big) \\ &= \big(g_{1,x}^{-1} \cdot f(x', y') \cdot g_{1,y}\big) \star y\end{aligned}$$

□

A first example is DLOG, which is the action of the group of exponents $\mathbb{G} = \mathbb{Z}_n$ on the multiplicative cyclic group $X = \langle g \rangle$. Since this action is transitive, we immediately get that DLOG is poly-rsr. A further and quantum-resistant example is Linear Code Equivalence, the computational problem behind the code-based signature scheme LESS [9].

▶ **Problem 14 (Linear Code Equivalence)** *Consider linear codes defined over* $\mathbb{F}_q$.

– *Parameters: code parameters* n, k, q.

– *Instance: the generator matrices G, G' of two (n,k)-codes C, C'.*
– *Task: find, if any, $S \in GL_k(\mathbb{F}_q)$ and $Q \in M_n(\mathbb{F}_q)$ such that $G' = SGQ$.*

This is an instance of Group Action Inversion, where the action

$$\star : \mathbb{G} \times X \longrightarrow X$$
$$((S;(\alpha,Q)),M) \longmapsto S \cdot \alpha(MQ)$$

is induced on the set of full-rank $k \times n$ matrices by the group $\mathbb{G} = GL_k(\mathbb{F}_q) \times (\mathrm{Aut}(\mathbb{F}_q) \rtimes M_n(\mathbb{F}_q))$, whose elements are a triple $g = (S;\alpha,Q)$ of an invertible matrix S, a field automorphism α, and a monomial matrix Q. The operation $g \star M$ mixes and scales the columns of M (multiplication by Q), applies an automorphism (α) and performs a change of basis (multiplication by S). The non-commutative group operation on $\mathbb{G}$ is

$$g_1 \cdot g_2 = (S_1;(\alpha_1,Q_1)) \cdot (S_2;(\alpha_2,Q_2)) = (S_1S_2;(\alpha_1 \circ \alpha_2, Q_1Q_2))$$

However, we have the following.

▶ **Corollary 15** *LCE is not poly-rsr.*

Proof. The action is not transitive, as two codes with the same parameters n,k but two different invariants cannot be mapped one into the other. Consider for instance the two generator matrices

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \qquad G' = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

over $\mathbb{F}_2$. They generate two $(4,2)$-linear codes, namely

$$C = \{(1,0,1,0),(0,1,0,1),(1,1,1,1),(0,0,0,0)\}$$
$$C' = \{(1,0,1,1),(0,1,1,0),(1,1,0,1),(0,0,0,0)\}$$

Their minimum distances are $d(C) = 2$ and $d(C') = 3$. The action of an element g such that $C' = g \star C = \phi(C)$ would be an isometry, meaning that $d(C') = d(\phi(C))$, which is not the case. □

4 Consequences

4.1 One-way functions

Angluin and Lichtenstein in [10] remark how most classical security assumptions are based on random self-reducible problems: quadratic residuals, RSA inversion, discrete logarithm. Some we saw in Section 2, and post-quantum ones in Section 3. A suggested reason is that the class of poly-rsr functions seemingly gives rise to one-way functions.

Abadi et al. [11] consider *instance hiding schemes*, where a secret instance x is efficiently transformed into a uniformly random instance y such that knowing $f(y)$ permits the recovery of $f(x)$. This allows users to rely on possibly malicious programs.
For example, in the Discrete Logarithm Problem, finding x in $h = g^x$ can be transformed by choosing a random r and querying the solution for $h' = h \cdot g^r = g^{x+r}$. The resulting solution x' then yields $x = x' - r$. This mechanism enables verifiable yet private computation of f, as the program only solves a random-looking problem instance, preserving the confidentiality of x.

▶ **Corollary 16** *Since DLOG is nonadaptively* 1*-rsr, then it is a* 1*-ihs (instance hiding scheme), meaning that an instance of DLOG can be hidden by one other instance.*

This can be generalised to nonadaptively $k(n)$-rsr functions using the same φ and σ of Theorem 2.

Further, random-self-reducibility allows reliable use of a flawed program P to evaluate f. This is similar to an instance hiding scheme, but we are unsure on the correctness of the output of P. Consider a program $\mathcal{P}$ evaluating f that is correct on at least $1/2$ of inputs. Compute i randomised instances $y_i = \sigma(i, x, r)$ from the correct input x. Once the flawed output is obtained, invert the reduction with φ. By repeating this process multiple times, the correct answer $f(x)$ is likely the majority result, effectively mitigating the error rate of $\mathcal{P}$. The fault tolerance of this process depends on that of φ in Theorem 2. The computational overhead mainly depends on k and σ.
Blum et al. [12] define this property *self-correction*.

4.2 Weak instances

A cryptographic protocol is secure only if the underlying mathematical problem is hard to solve for the vast majority of instances, but average-case hardness can hard to prove for general problems. However, for a poly-rsr f, it is sufficient to prove worst-case hardness.

▶ **Corollary 17** *Consider a poly-rsr function f. If an algorithm $\mathcal{A}$ correctly solves $f(x)$ for a non-negligible fraction $\rho(n) > 0$ of uniformly random length-n inputs x, then we can construct a randomized algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine to correctly evaluate $f(x)$ for all inputs x with high probability.*

Intuitively, if an efficient algorithm exists for a non-negligible fraction of inputs, then it is easy for all inputs. We are interested in the contrapositive: if the function f is *worst-case hard* (i.e., no polynomial-time algorithm solves it for every input), then it must also be *average-case hard*.

The set of weak instances S_{weak} consists of all x such that $f(x)$ can be computed faster than the average case. If f is not poly-rsr, an attacker could identify and exploit a small subset of weak keys to break the system. Having a poly-rsr f ensures that the difficulty of solving a randomly chosen instance x is on average equivalent to the difficulty of the hardest instance.

(DLOG) This, together with its commutativity ($g^a \cdot g^b = g^b \cdot g^a$), is perhaps behind the wide number of applications of DLOG [3].

(RSAI) The great discourse on weak RSA instances mostly regards the hardness of integer factorisation, not of the RSA Inversion problem. Also, attacks work on a negligible set of parameters, which do not affect the hardness of the average instance.

4.3 Computational comlexity & polynomial hierarchy

We can classify decision problems in complexity classes:

- P, all decision problems D that can be solved by a deterministic algorithm in poly-time.
- NP, all decision problems D that can be solved by a non-deterministic algorithm in poly-time.
- coNP, all decision problems D whose complement $\overline{\mathsf{D}} = \{x \mid x \notin \mathsf{D}\}$ is in NP.

It is known that $\mathsf{P} \subseteq \mathsf{NP}$ and $\mathsf{P} \subseteq \mathsf{coNP}$. The question of whether $\mathsf{P} = \mathsf{NP}$ remains the central open problem in complexity theory.

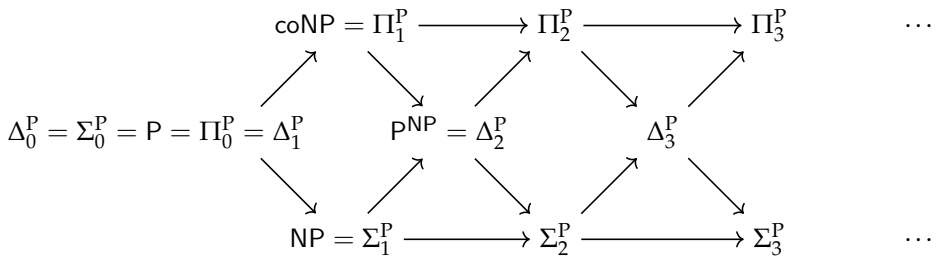
More generally, for any complexity class C, a problem D is C-*hard* if every problem in C can be reduced to D in polynomial time, and if it also lies in C we call it C-*complete*. This allows to construct the *Polynomial Hierarchy* (PH), a chain of complexity classes Σ_i^P and Π_i^P defined as

$$\begin{aligned} \Sigma_0^P &= \mathsf{P} & \Pi_0^P &= \mathsf{P} & \Delta_0^P &= \mathsf{P} \\ \Sigma_1^P &= \mathsf{NP} & \Pi_1^P &= \mathsf{coNP} & \Delta_1^P &= \mathsf{P}^{\mathsf{NP}} \\ \Sigma_{i+1}^P &= \mathsf{NP}^{\Sigma_i^P} & \Pi_{i+1}^P &= \mathsf{coNP}^{\Sigma_i^P} & \Delta_{i+1}^P &= \mathsf{P}^{\Sigma_i^P} \end{aligned}$$

Here notation such as $\mathsf{NP}^{\Sigma_i^P}$ represents oracle access to Σ_i^P. The entire hierarchy is defined as the union of all levels:

$$\mathsf{PH} = \bigcup_{i \geq 0} \Sigma_i^P = \bigcup_{i \geq 0} \Pi_i^P$$

It is known that each level is contained in the next, $\Sigma_i^P \subseteq \Sigma_{i+1}^P$. If there exists an integer i such that $\Sigma_i^P = \Sigma_{i+1}^P$, the Polynomial Hierarchy is said to collapse at the i-th level, implying $\mathrm{PH} = \Sigma_i^P$. It is conjectured that PH does not collapse at any level.



However, there is a catch.

▶ **Theorem 18 ([1])** *If an* NP*-complete function* f *is adaptively* $O(\log n)$*-rsr, then the polynomial hierarchy collapses at the third level.*

Most conjecture that NP-complete problems cannot be poly-rsr, as this would imply a significant collapse in the polynomial hierarchy. Problems such as (the decisional variant of) DLOG lie in $\mathsf{NP} \cap \mathsf{coNP}$ and are believed to not be NP-complete. This also holds for the other problems we saw in Sections 2.1, 2.2, 3.1 and 3.2

Sources and research material

This work is based on the following publications:

[1] *Nonadaptive and adaptive rsr.*
Joan Feigenbaum and Lance Fortnow. "Random-self-reducibility of complete sets". In: *SIAM Journal on Computing* 22.5 (1993), pp. 994–1005.

[2] *In-depth analysis of adaptive k-rsr functions. For an adaptive k-rsr which is not nonadaptive k-rsr, see Section 3.*
Joan Feigenbaum, Lance Fortnow, Carsten Lundj, and Daniel Spielman. "The Power of Adaptiveness and Additional Queries in Random-Self-Reduct ions". In: *IEEE* (1992).

[3] *A compendium of properties of the discrete logarithm problem.*
Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. "The past, evolving present, and future of the discrete logarithm". In: *Open Problems in Mathematics and Computational Science*. Springer, 2014, pp. 5–36. URL: `https://www-users.cse.umn.edu/~odlyzko/doc/discretelogs2014.pdf`.

[4] *See Chapter 3 for isogenies and endomorphisms.*
Joseph H Silverman. *The arithmetic of elliptic curves*. Vol. 106. Springer, 2009.

[5] *Theory and applications of isogenies to cryptography.*
Damien Robert. *On the efficient representation of isogenies (a survey)*. Cryptology ePrint Archive, Paper 2024/1071. 2024. URL: `https://eprint.iacr.org/2024/1071`.

[6] *They show that the OneEnd problem (finding a non-trivial endomorphism) is poly-rsr. This is similar to our proof that EndRing (finding the generators of all endomorphisms) is poly-rsr. The two problems are actually reducible to each other in poly-time.*
Arthur Herlédan Le Merdy and Benjamin Wesolowski. *Unconditional foundations for supersingular isogeny-based cryptography*. Cryptology ePrint Archive, Paper 2025/271. 2025. URL: `https://eprint.iacr.org/2025/271`.

[7] *See Theorem 11 for the result about stationary distributions in the supersingular graph.*
Andrea Basso, Giulio Codogni, Deirdre Connolly, Luca De Feo, Tako Boris Fouotsa, Guido Maria Lido, Travis Morrison, Lorenz Panny, Sikhar Patranabis, and Benjamin Wesolowski. *Supersingular Curves You Can Trust*. Cryptology ePrint Archive, Paper 2022/1469. 2022. URL: `https://eprint.iacr.org/2022/1469`.

[8] *The theory of rsr problems applied to group actions. The author claims an error in the proof of Theorem 15, point 3.*
Giuseppe D'Alconzo. "A note on the hardness of problems from cryptographic group actions". In: *arXiv preprint arXiv:2202.13810* (2022).

[9] *The post-quantum signature scheme LESS, a NIST candidate whose security assumption is based on the hardness of a poly-rsr problem.*
The LESS team. *LESS 2.0*. Tech. rep. 2025. URL: `https://www.less-project.com/resources/`.

[10] *See Section 2.3 for an analysis of rsr for classical problems.*
Dana Angluin and David Lichtenstein. "Provable security of cryptosysterns: A survey". In: *Yale Univ., New Haven, CT, USA, Tech. Rep. TR-288* (1983).

[11] *The first approach to instance-hiding schemes.*
Martin Abadi, Joan Feigenbaum, and Joe Kilian. "On hiding information from an oracle". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 195–203.

[12] *Self-correcting schemes.*
Manuel Blum, Michael Luby, and Ronitt Rubinfeld. "Self-testing/correcting with applications to numerical problems". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 73–83.