PECDSA
How to build a DL-based digital signature scheme with the best proven security

Louis Granboulan[*]

École Normale Supérieure
Louis.Granboulan@ens.fr

Abstract. Many variants of the ElGamal signature scheme have been proposed. The most famous is the DSA standard. If computing discrete logarithms is hard, then some of these schemes have been proven secure in an idealized model, either the random oracle or the generic group. We propose a generic but simple presentation of signature schemes with security based on the discrete logarithm. We show how they can be proven secure in idealized model, under which conditions. We conclude that none of the previously proposed digital signature schemes has optimal properties and we propose a scheme named PECDSA.

Keywords: digital signature, DSA variants, idealized model, proven security.

1 Introduction

1.1 Motivations

Very often, cryptographic primitives are based on some core technique, but the details of the design are subject to many variants. The goal of this paper is to find the best design for digital signature schemes based on the intractability of the discrete logarithm problem.

Many variants of the original ElGamal [11] signature scheme have been described [1, 2, 17–19, 22–25, 29–31, 34, 35]. We want to compare them and to find the best variant.

[*] Part of this work has been supported by the Commission of the European Communities through the IST Programme under Contract IST-1999-12324 (NESSIE). This paper is NESSIE document **NES/DOC/ENS/WP5/022/1**. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

1.2 Overview of the results

We show how to build many DL-based signature schemes by mixing four components which we name: a group, a hash function, a projection and a category. All these components can take many possible values, and we describe some of them. We show that most published DL-based signature schemes can be described that way.

We show that the random oracle model, which usually is used to idealize collision-resistant hash functions, can also be used to idealize collision-resistant almost invertible functions.

Then we give three security proofs for DL-based signature schemes, in different idealized worlds, and we explain which are the requirements that a signature scheme needs to meet to be secure in these idealized worlds.

We show that none of the previously published signature schemes has been proven in the three idealized worlds. We describe PECDSA, which has all the required properties. We also describe a variant of PECDSA with partial message recovery and the same security properties.

1.3 Related work

Some effort has been done towards an global description of DL-based schemes. Horster, Michels and Petersen [13–15] defined the Meta-ElGamal signature schemes. Brickell, Pointcheval, Vaudenay and Yung [6] defined the Trusted ElGamal-type signature schemes of type I and II.

The security proofs in this paper are directly inspired from security proofs that were published for some DL-based signature schemes [6–8, 39].

2 Preliminary definitions

2.1 Digital signature schemes

A signature scheme is built on three algorithms: a key generation algorithm, a signing algorithm and a verification algorithm.

The key generation algorithm generates a random public key and secret key pair $(\mathsf{pk}, \mathsf{sk})$.

Let $\mathcal{M}$ be the set of the possible messages and $\mathcal{A}$ be the set of possible appendices. The (randomized) signing algorithm takes a message $m \in \mathcal{M}$ and the secret key sk and generates the appendix $a \in \mathcal{A}$. The signed message is (m, a). The (randomized) verification algorithm takes a signed

message $(m, a) \in \mathcal{M} \times \mathcal{A}$ and answers a boolean value, whether the signed message is valid or not.
All signed messages generated by the signing algorithm should be detected as valid by the verification algorithm and it should be difficult to generate a valid signed message without the secret key.

Signature scheme with partial message recovery. Let $\mathcal{M} = \hat{\mathcal{M}} \times \bar{\mathcal{M}}$ be the set of the possible messages and $\mathcal{A}$ be the set of possible appendices. The (randomized) signing algorithm takes a message $(\hat{m}, \bar{m}) \in \mathcal{M}$ and the secret key sk and generates the appendix $a \in \mathcal{A}$. The signed message is $(\hat{m}, a)$. The (randomized) verification algorithm takes a signed message $(\hat{m}, a) \in \hat{\mathcal{M}} \times \mathcal{A}$ and answers a recovered message $\bar{m} \in \bar{\mathcal{M}}$ and a boolean value, whether the signed message is valid or not.
All signed messages generated by the signing algorithm should be detected as valid by the verification algorithm and it should be difficult to generate a valid signed message without the secret key. The verification algorithm should also return the value $\bar{m}$ that was used in the signing algorithm.

2.2 Proven security

We say that a signature scheme is secure if it is existentially unforgeable against adaptive chosen message attacks, which means that no forger exists. A forger is an algorithm that receives a public key, is allowed to ask signature queries for this public key, and returns a forgery. The forger succeeds if the forgery is a valid signed message for a message that was not the input of a signature query. Strong unforgeability (also known as non-malleability) authorizes any forgery that was not the answer of a signature query, even if the message was the input of some signature query.
A proof of security makes some assumptions (hardness of discrete logarithm, collision-resistance of a given function, ...) and shows that if there exists a forger then one of these assumptions does not hold. It explains how to build an algorithm that breaks one of these assumptions if given access to a forger. This algorithm is called *simulator* because it answers the signature queries made by the forger, and therefore simulates the signing algorithm. This simulation should be indistinguishable from the actual signing algorithm.
The *efficiency* of the proof is measured by the gap between the success probability of the forger and the success probability of the simulator,

and the comparison of their running times. In tight proofs the success probabilities and the running times are similar. In loose proofs the simulator succeeds with much lower probability than the forger, and security parameters need to be increased for the proof to be really meaningful [4].

Idealized models. For a security proof in an idealized model, some components of the scheme are replaced with an ideal function, and the simulator has the control of this ideal function. The forger needs to make queries to the simulator to compute these functions.
For example, in the random oracle model [3], a hash function is replaced by an oracle that returns a random value for each new query. In the generic group model [36, 7], group operation are given random answers, provided that they agree with the group laws.
Both models may give a security proof for a scheme that is insecure when the idealized component is replaced by any efficient concrete implementation [9, 10]. However, not being able to prove the security of a scheme even in an idealized model is a security concern.

We consider that proofs in an idealized model as a validation for the design. Different proofs where different components are idealized give more confidence in a design.

3 The toolbox

3.1 Construction of the DL-based signature schemes

The signature scheme is built on

- A **DL-group** $\langle G \rangle$ of order q where computing discrete logarithms is hard.
  The set of possible private keys is $\mathcal{V} \subset \mathbb{Z}_q$. If the private key of the signer is $v \in \mathcal{V}$, the corresponding public key is the element $V = G^v$.
  Depending on the category, we may need $\mathcal{V} = \mathbb{Z}_q$ or $\mathcal{V} = \mathbb{Z}_q^\times$.
  All groups will be denoted multiplicatively, even in the case of elliptic curves.
- A **projection**. It is a function $\mathsf{p} : \langle G \rangle \to \mathcal{R}$.
- A **hash function** that makes a digest of the message. It is a function $\mathsf{H} : \mathcal{M} \times \mathcal{R} \to \mathcal{H}$ where $\mathcal{M}$ is the set of possible messages.
- A **category** that defines the formulas for signature and verification. The category defines two functions ϕ and $\psi : \mathcal{H} \times \mathcal{R} \times \mathcal{S} \to \mathbb{Z}_q$ and a function $\sigma : \mathcal{I} \to \mathcal{S}$ where $\mathcal{I} \subset \mathcal{H} \times \mathcal{R} \times \mathcal{V} \times \mathcal{K}$ with $\mathcal{S} = \mathbb{Z}_q$ or $\mathbb{Z}_q^\times$ and $\mathcal{K} = \mathbb{Z}_q$ or $\mathbb{Z}_q^\times$.

The digital signature scheme works as follows:

- **Verification.** The verification of $(m, r, s) \in \mathcal{M} \times \mathcal{R} \times \mathcal{S}$ computes $h = \mathsf{H}(m, r)$, $\alpha = \phi(h, r, s)$, $\beta = \psi(h, r, s)$, $R = G^{\alpha} V^{\beta}$ and checks if $R \in G^{\mathcal{K}}$ and if $r \overset{?}{=} \mathsf{p}(R)$.
- **Signature.** To sign the message m one takes a random $k \in \mathcal{K}$, computes $R = G^k$, $r = \mathsf{p}(R)$, $h = \mathsf{H}(m, r)$ until $(h, r, v, k) \in \mathcal{I}$ and $s = \sigma(h, r, v, k)$. The signed message is (m, r, s).
- **Partial message recovery.** For some schemes the p function is designed to allow partial message recovery. The verification $r \overset{?}{=} \mathsf{p}(R)$ also extracts the recovered message $\bar{m}$.

3.2 DL-groups.

DL-based signature schemes do computations in a cyclic group $\langle G \rangle$ of known order q and known generator G. Adding or taking inverses of elements of the group should be easy, yet elements of $\langle G \rangle$ can be indistinguishable from elements of a larger set.
Usually $\langle G \rangle$ is a cyclic subgroup of the multiplicative group $\mathbb{Z}_p^{\times}$ of integers modulo p, or an elliptic curve subgroup, and q is a prime number.

The exponentiation is a bijection from $\mathbb{Z}_q$ to $\langle G \rangle$ defined by $k \mapsto G^k$ and the discrete logarithm is the inverse of that bijection. By definition, computing the discrete logarithm in a DL-group is intractable.

$\mathbb{Z}_q^{\times}$ is the set of invertible elements of $\mathbb{Z}_q$ and for any $k \in \mathbb{Z}_q^{\times}$ the group element G^k is a generator of $\langle G \rangle$. One must know the factorization of q to compute inverses in $\mathbb{Z}_q^{\times}$.

Let $\#q$ be an integer smaller than $\log_2 q$ and let $[\mathbb{Z}_q]_{\#}$ be the subset of $\mathbb{Z}_q$ that contains the integers smaller than $2^{\#q}$. The group operation $\oplus$ in $[\mathbb{Z}_q]_{\#}$ corresponds to the XOR of bits strings of length $\#q$.

3.3 Hash function

The function $\mathsf{H} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{H}$ should be easy to compute, has uniform random output for random m and may have some of the following properties.

- H is *Type I* if $\forall m \in \mathcal{M}, r, r' \in \mathcal{R}$, $\mathsf{H}(m, r) = \mathsf{H}(m, r')$. This common value is $\mathsf{H}(m)$.
  H is *Type II* if it is not *Type I*.

- H with Type I is *collision-resistant* if it is hard to find distinct inputs $m \neq m'$ such that $\mathsf{H}(m) = \mathsf{H}(m')$.
- H with Type II is *collision-resistant* if it is hard to find distinct inputs $(m, r) \neq (m', r')$ such that $\mathsf{H}(m, r) = \mathsf{H}(m', r')$.
- H with $\mathcal{H} \subset [\mathbb{Z}_q]_\#$ and $\mathcal{R} \subset [\mathbb{Z}_q]_\#$ is *xor-collision-resistant* if given random r, r' it is hard to find m, m' such that $\mathsf{H}(m, r) \oplus r = \mathsf{H}(m', r') \oplus r'$.
- H with $\mathcal{H} \subset \mathbb{Z}_q$ and $\mathcal{R} \subset \mathbb{Z}_q$ is *add-collision-resistant* if given random r, r' it is hard to find m, m' such that $\mathsf{H}(m, r) + r = \mathsf{H}(m', r') + r'$.
- H with $\mathcal{H} \subset \mathbb{Z}_q$ and $\mathcal{R} \subset \mathbb{Z}_q^\times$ is *div-collision-resistant* if given random r, r' it is hard to find m, m' such that $\mathsf{H}(m, r)/r = \mathsf{H}(m', r')/r'$.
- H is *suitable as a random oracle* if the knowledge of many input-output pairs does not restrict substantially the possible images space for a distinct input.

3.4 Projections

The function $\mathsf{p} : \langle G \rangle \to \mathcal{R}$ is easy to compute by the signer, and the verifier should be able to test if $r \stackrel{?}{=} \mathsf{p}(R)$ for $R \in \langle G \rangle$ and $r \in \mathcal{R}$. NB: if elements of $\langle G \rangle$ are indistinguishable from elements of a larger set, then p is defined on the whole larger set. The projection may have some of the following properties.

- p is ε-*almost uniform* if $\forall r \in \mathcal{R}$, $\Pr_{R \in \langle G \rangle}[\mathsf{p}(R) = r] \geq \varepsilon$.
- p is ε-*almost invertible* if there exists an efficient algorithm to compute the function $\mathsf{p}^{-1} : \mathcal{R} \to \langle G \rangle$ such that
  - $\forall R \in \mathsf{p}^{-1}(r), \mathsf{p}(R) = r$
  - At least a proportion ε of the sets $\mathsf{p}^{-1}(r)$ is non empty.
  - Elements randomly taken from random sets $\mathsf{p}^{-1}(r)$ are indistinguishable from elements randomly taken from $\langle G \rangle$.
- p is $\ell + 1$-*collision-resistant* for $\ell \geq 1$ if it is hard to find distinct $R_0, ..., R_\ell$ such that $\mathsf{p}(R_0) = ... = \mathsf{p}(R_\ell)$.
- p is *suitable as a random oracle* if the knowledge of many input-output pairs does not restrict substantially the possible images space for a distinct input. NB: an almost invertible function can be suitable as a random oracle (see also section 4.1).

Examples of projections.

- **ElGamal projection.** For $\langle G \rangle \subset \mathbb{Z}_p^\times$ and $\mathcal{R} = \mathbb{Z}_p$ it is $\mathsf{p}(R) = R$.

This function is almost uniform and collision-resistant. It is not almost-invertible if appartenance to $\langle G\rangle$ is hard to check. It is not suitable as a random oracle.

- **DSA projection.** For $\langle G\rangle \subset \mathbb{Z}_p^\times$ and $\mathcal{R} = \mathbb{Z}_q$ it is $\mathsf{p}(R) = R \bmod q$.
  This function is almost uniform and probably $\log q$-collision-resistant [6]. It is not almost-invertible if appartenance to $\langle G\rangle$ is hard to check. It is not suitable as a random oracle.
- **EC projections.** If $\langle G\rangle$ is a subgroup of an elliptic curve defined over some finite field $\mathbb{F}$, let (R_x, R_y) be the coordinates of the point $R \in \langle G\rangle$ and $\mathsf{i}_\mathbb{F}$ a mapping from $\mathbb{F}$ to the set of integers.
  - **ECxq projection.** For $\mathcal{R} = \mathbb{Z}_q$ it is $\mathsf{p}(R) = \mathsf{i}_\mathbb{F}(R_x) \bmod q$.
  - **ECx2 projection.** For $\mathcal{R} = [\mathbb{Z}_q]_\#$ it is $\mathsf{p}(R) = \mathsf{i}_\mathbb{F}(R_x) \bmod 2^{\#q}$.
  - **ECaddq projection.** For $\mathcal{R} = \mathbb{Z}_q$ it is $\mathsf{p}(R) = \mathsf{i}_\mathbb{F}(R_x{+}R_y) \bmod q$.

  These functions are almost uniform and almost invertible. They are probably $\log q$-collision-resistant. They are not suitable as a random oracle.
- **KCDSA projection.** p is a hash function with output in $\mathcal{R}$, e.g. based on SHA-1.
  This function is uniform and collision-resistant, and suitable as a random oracle. It is not almost-invertible.
- **Permuted projection.** Any projection $\mathsf{p}' : \langle G\rangle \to \mathcal{R}$ can be composed with a random permutation $\mathsf{P} : \mathcal{R} \to \mathcal{R}$ to obtain $\mathsf{p} = \mathsf{P} \circ \mathsf{p}'$.
  The projection p inherits from properties of p', but is also suitable as a random oracle.

Projections with partial message recovery. Let $\mathsf{F} : \langle G\rangle \times \bar{\mathcal{M}} \to \mathcal{R}$ and $\mathsf{F}^{-1} : \langle G\rangle \times \mathcal{R} \to \bar{\mathcal{M}} \cup \{\mathit{fail}\}$ such that $\forall R \in \langle G\rangle$, $\mathsf{F}(R, \bar{m}) = r \Leftrightarrow \mathsf{F}^{-1}(R, r) = \bar{m}$. Then the function $\mathsf{p}(R) = \mathsf{F}(R, \bar{m})$ is a projection that allows partial message recovery. The verification $r \overset{?}{=} \mathsf{p}(R)$ is false if, and only if, $\mathsf{F}^{-1}(R, r)$ returns *fail*.

- **PVSSR projection.** It is the composition of an arbitrary encryption function E over $\mathcal{R}$ and with key in $\langle G\rangle$ and a redundancy function $\rho : \bar{\mathcal{M}} \to \mathcal{R}$. The definition is $\mathsf{F}(R, \bar{m}) = \mathsf{E}_R \circ \rho(\bar{m})$ and $\mathsf{F}^{-1}(R, r) = \rho^{-1} \circ \mathsf{E}_R^{-1}(r)$.
  The redundancy function ρ should have the following properties:
  - it is collision-free,
  - the inverse $\rho^{-1} : \mathcal{R} \to \bar{\mathcal{M}} \cup \{\mathit{fail}\}$ is easy to compute.
  - a random element $\tilde{m} \in \mathcal{R}$ is very unlikely the image of some $\bar{m} \in \bar{\mathcal{M}}$,

  If E is a secure cipher, then the projection is uniform, collision-resistant and suitable as a random oracle, but is not almost invertible.

- **Group projection.** It is a special case of the PVSSR projection, based on any other projection $\mathsf{p}' : \langle G \rangle \to \mathcal{G}$ such that $\mathcal{G}$ is a group with an action on $\mathcal{R}$. This defines a one-time-pad encryption scheme and the projection is $\mathsf{p}(R) = \mathsf{p}'(R) \cdot \rho(m)$.
  This projection has the same properties as p'.
- **NS projection.** This is the Group projection where $\mathcal{G} = \mathcal{R} = \mathbb{Z}_q$ with additive action. It is defined by the equation $\mathsf{p}(R) = \mathsf{p}'(R) + \rho(m) \bmod q$.
  This projection has the same properties as p'.
- **NR projection.** This is the Group projection where $\langle G \rangle \subset \mathbb{Z}_p^\times$ and $\mathcal{G} = \mathbb{Z}_p^\times$ has a multiplicative action on $\mathcal{R} = \mathbb{Z}_p$ and with a tweak of Schnorr projection $\mathsf{p}'(R) = R^{-1} \bmod p$. The NR projection is defined by the equation $\rho(m) = R \cdot \mathsf{p}(R) \bmod p$.

3.5 Categories

Definition and properties. The category is described by the sets $\mathcal{V} \subset \mathbb{Z}_q$, $\mathcal{S} \subset \mathbb{Z}_q$, $\mathcal{H}$ and $\mathcal{R}$, by the two functions ϕ and $\psi : \mathcal{H} \times \mathcal{R} \times \mathcal{S} \to \mathbb{Z}_q$ and a set $\mathcal{I} \subset \mathcal{H} \times \mathcal{R} \times \mathcal{V} \times \mathcal{K}$.
A category should meet some of the following properties.

- **Other functions.** Let $\mathcal{A}$ be the set of possible outputs for ϕ and $\mathcal{B}$ the set of possible outputs for ψ. Five additional functions can be defined.
  For all defined function in (ϕ, ψ, σ, λ_h, λ_s, λ_r, μ_h) there exists an efficient algorithm that computes the result.
  - $\sigma : \mathcal{I} \to \mathcal{S}$.
  - $\lambda_h : \mathcal{A} \times \mathcal{B} \times \mathcal{R} \to \mathcal{H}$
  - $\lambda_s : \mathcal{A} \times \mathcal{B} \times \mathcal{R} \to \mathcal{S}$
  - $\lambda_r : \mathcal{A} \times \mathcal{B} \times \mathcal{H} \to \mathcal{R}$
  - $\mu_h : \mathcal{S} \times \mathcal{R} \times \mathcal{V} \times \mathcal{K} \to \mathcal{H}$
- **Main properties.** These properties are mandatory for all DL-based schemes.

(𝔪1) For all $(h, r, v, k) \in \mathcal{I}$, the value $s = \sigma(h, r, v, k)$ is such that if $\alpha = \phi(h, r, s)$ and $\beta = \psi(h, r, s)$ then $k = \alpha + v \cdot \beta$.

(𝔪2) For all $v \in \mathcal{V}$ and $h \in \mathcal{H}$, $\Pr_{r \in \mathcal{R},\, k \in \mathcal{K}}[(h, r, v, k) \in \mathcal{I}] \geq \varepsilon_{\mathfrak{m}}$.

Property (𝔪1) implies that all signatures generated by the signing algorithm are valid. Property (𝔪2) implies that the expected number of random k needed for signature generation is less than $\frac{1}{\varepsilon_{\mathfrak{m}}}$.

- **Other properties.**
  - ($\mathfrak{o}1$) For all $(h,r,s) \in \mathcal{H} \times \mathcal{R} \times \mathcal{S}$ the equation $\lambda_h(\phi(h,r,s), \psi(h,r,s), r) = h$ holds.
  - ($\mathfrak{o}2$) For all $(h,r,s) \in \mathcal{H} \times \mathcal{R} \times \mathcal{S}$ the equation $\lambda_s(\phi(h,r,s), \psi(h,r,s), r) = s$ holds.
  - ($\mathfrak{o}3$) The function $s \mapsto \mu_h(s,r,v,k)$ is the inverse of $h \mapsto \sigma(h,r,v,k)$.
- **Additional properties for security with idealized p.**
  - ($\mathfrak{p}1$) For fixed (h,r,v) and uniform k such that $(h,r,v,k) \in \mathcal{I}$ the value $\sigma(h,r,v,k)$ is uniform in $\mathcal{S}$.
    NB: this property together with the hypothesis that the function $k \mapsto \mathsf{p}(G^k)$ is (almost-)uniform and one-way implies that the set of possible valid appendices (r,s) for a message m is uniformly distributed.
  - ($\mathfrak{p}2$) For fixed $h \in \mathcal{H}$ and $v \in \mathcal{V}$ and uniformly random $s \in \mathcal{S}$ and $r \in \mathcal{R}$, then $k = \phi(h,r,s) + v \cdot \psi(h,r,s)$ is uniformly random in $\mathcal{K}$.
    NB: failure may happen if $\mathcal{I} \neq \mathcal{H} \times \mathcal{R} \times \mathcal{V} \times \mathcal{K}$ and if $k \notin \mathcal{K}$. It is accepted that the probability of this failure is negligible.
  - ($\mathfrak{p}3$) Given random r and r', it is hard to find some (α,β) and messages m and m' such that $\lambda_h(\alpha,\beta,r) = \mathsf{H}(m,r)$ and $\lambda_h(\alpha,\beta,r') = \mathsf{H}(m',r')$.
- **Additional properties for security with idealized H.**
  - ($\mathfrak{h}1$) If $h = \lambda_h(\alpha,\beta,r)$ and $s = \lambda_s(\alpha,\beta,r)$ then $\alpha = \phi(h,r,s)$ and $\beta = \psi(h,r,s)$.
  - ($\mathfrak{h}2$) $\Pr_{\alpha\in\mathcal{A},\beta\in\mathcal{B}}[\lambda_h(\alpha,\beta,\mathsf{p}(G^\alpha V^\beta)) \in \mathcal{H} \text{ and } \lambda_s(\alpha,\beta,\mathsf{p}(G^\alpha V^\beta)) \in \mathcal{S}] \geq \varepsilon_{\mathfrak{h}}$.
- **Additional properties for security with idealized $\langle G \rangle$.**
  - ($\mathfrak{g}1$) For all (h,r,s) the equation $\lambda_r(\phi(h,r,s), \psi(h,r,s), h) = r$ holds.
  - ($\mathfrak{g}2$) For any (h,h',r,s), if $\lambda_r(\phi(h,r,s), \psi(h,r,s), h') = r$ then $h' = h$.

Simple categories. Those are the categories where $\mathcal{H} \subset \mathbb{Z}_q$ and $\mathcal{R} \subset \mathbb{Z}_q$ and where each of ϕ and ψ only does one operation in $\mathbb{Z}_q$. These are less general that Meta-ElGamal [13] or TEGTSS [6] schemes, but cover all actual published schemes.
Properties ($\mathfrak{m}1$), ($\mathfrak{m}2$), ($\mathfrak{o}1$), ($\mathfrak{o}2$), ($\mathfrak{o}3$), ($\mathfrak{p}1$), ($\mathfrak{p}2$), ($\mathfrak{h}1$) and ($\mathfrak{h}2$) hold for all the following examples.

- **ElGamal category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{R} = \mathcal{V} = \mathcal{K} = \mathcal{S} = \mathcal{B} = \mathbb{Z}_q^\times$ and $\mathcal{A} = \mathbb{Z}_q$. Because $\mathcal{I} = \{(h,r,v,k) | h + v \cdot r \in \mathbb{Z}_q^\times\}$ property ($\mathfrak{p}2$) can fail with negligible probability. ($\mathfrak{p}3$) is equivalent to div-collision-resistance of H. ($\mathfrak{g}1$) and ($\mathfrak{g}2$) hold with the restrictions $\mathcal{H} \subset \mathbb{Z}_q^\times$ and

$\mathcal{A} = \mathbb{Z}_q^\times$. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_\mathfrak{m} = \frac{\varphi(q)}{q}$ and $\varepsilon_\mathfrak{h} = \frac{|\mathcal{H}|}{q}$.

$$\begin{array}{ll}
\phi(h,r,s) = h/s & \lambda_h(\alpha,\beta,r) = \alpha\beta^{-1} \cdot r \\
\psi(h,r,s) = r/s & \lambda_s(\alpha,\beta,r) = \beta^{-1} \cdot r \\
\sigma(h,r,v,k) = (h + v \cdot r)/k & \lambda_r(\alpha,\beta,h) = \alpha^{-1}\beta \cdot h \\
\mu_h(s,r,v,k) = k \cdot s - v \cdot r &
\end{array}$$

- **Inverse ElGamal category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{R} = \mathcal{V} = \mathcal{K} = \mathcal{S} = \mathcal{B} = \mathbb{Z}_q^\times$ and $\mathcal{A} = \mathbb{Z}_q$. Because $\mathcal{I} = \{(h,r,v,k) | h + v \cdot r \in \mathbb{Z}_q^\times\}$ property $(\mathfrak{p}2)$ can fail with negligible probability. $(\mathfrak{p}3)$ is equivalent to div-collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ hold with the restrictions $\mathcal{H} \subset \mathbb{Z}_q^\times$ and $\mathcal{A} = \mathbb{Z}_q^\times$. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_\mathfrak{m} = \frac{\varphi(q)}{q}$ and $\varepsilon_\mathfrak{h} = \frac{|\mathcal{H}|}{q}$.

$$\begin{array}{ll}
\phi(h,r,s) = h \cdot s & \lambda_h(\alpha,\beta,r) = \alpha\beta^{-1} \cdot r \\
\psi(h,r,s) = r \cdot s & \lambda_s(\alpha,\beta,r) = r^{-1} \cdot \beta \\
\sigma(h,r,v,k) = k/(h + v \cdot r) & \lambda_r(\alpha,\beta,h) = \alpha^{-1}\beta \cdot h \\
\mu_h(s,r,v,k) = k/s - v \cdot r &
\end{array}$$

- **GOST category.** Let $\mathcal{H} \subset \mathbb{Z}_q^\times$, $\mathcal{V} = \mathcal{K} = \mathcal{S} = \mathcal{A} = \mathbb{Z}_q$ and $\mathcal{R} = \mathcal{B} = \mathbb{Z}_q^\times$. $(\mathfrak{o}3)$ needs the restriction $\mathcal{K} = \mathbb{Z}_q^\times$. $(\mathfrak{p}3)$ is equivalent to div-collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ hold. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_\mathfrak{m} = 1$ and $\varepsilon_\mathfrak{h} = \frac{|\mathcal{H}|}{q}$.

$$\begin{array}{ll}
\phi(h,r,s) = s/h & \lambda_h(\alpha,\beta,r) = \beta^{-1} \cdot r \\
\psi(h,r,s) = r/h & \lambda_s(\alpha,\beta,r) = \alpha\beta^{-1} \cdot r \\
\sigma(h,r,v,k) = k \cdot h - v \cdot r & \lambda_r(\alpha,\beta,h) = \beta \cdot h \\
\mu_h(s,r,v,k) = (s + v \cdot r)/k &
\end{array}$$

- **GDSA category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{K} = \mathcal{S} = \mathcal{A} = \mathcal{B} = \mathbb{Z}_q$ and $\mathcal{R} = \mathcal{V} = \mathbb{Z}_q^\times$. $(\mathfrak{p}3)$ is equivalent to div-collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ hold with the restrictions $\mathcal{H} \subset \mathbb{Z}_q^\times$ and $\mathcal{A} = \mathbb{Z}_q^\times$. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_\mathfrak{m} = 1$ and $\varepsilon_\mathfrak{h} = \frac{|\mathcal{H}|}{q}$.

$$\begin{array}{ll}
\phi(h,r,s) = h/r & \lambda_h(\alpha,\beta,r) = \alpha \cdot r \\
\psi(h,r,s) = s/r & \lambda_s(\alpha,\beta,r) = \beta \cdot r \\
\sigma(h,r,v,k) = (k \cdot r - h)/v & \lambda_r(\alpha,\beta,h) = \alpha^{-1} \cdot h \\
\mu_h(s,r,v,k) = k \cdot r - v \cdot s &
\end{array}$$

- **KCDSAadd category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{R} = \mathcal{K} = \mathcal{S} = \mathcal{A} = \mathcal{B} = \mathbb{Z}_q$ and $\mathcal{V} = \mathbb{Z}_q^\times$. $(\mathfrak{p}3)$ is equivalent to add-collision-resistance of H. $(\mathfrak{g}1)$ and

$(\mathfrak{g}2)$ hold. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_{\mathfrak{m}} = 1$ and $\varepsilon_{\mathfrak{h}} = \frac{|\mathcal{H}|}{q}$.

$$\begin{array}{ll}
\phi(h,r,s) = h + r & \lambda_h(\alpha,\beta,r) = \alpha - r \\
\psi(h,r,s) = s & \lambda_s(\alpha,\beta,r) = \beta \\
\sigma(h,r,v,k) = (k - (h+r))/v & \lambda_r(\alpha,\beta,h) = \alpha - h \\
\mu_h(s,r,v,k) = (k - v \cdot s) - r &
\end{array}$$

- **KCDSAxor category.** Let $\mathcal{H} = \mathcal{R} = \mathcal{A} = [\mathbb{Z}_q]_{\#}$, $\mathcal{K} = \mathcal{S} = \mathcal{B} = \mathbb{Z}_q$ and $\mathcal{V} = \mathbb{Z}_q^{\times}$. $(\mathfrak{p}3)$ is equivalent to xor-collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ hold. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_{\mathfrak{m}} = 1$ and $\varepsilon_{\mathfrak{h}} = 1$.

$$\begin{array}{ll}
\phi(h,r,s) = h \oplus r & \lambda_h(\alpha,\beta,r) = \alpha \oplus r \\
\psi(h,r,s) = s & \lambda_s(\alpha,\beta,r) = \beta \\
\sigma(h,r,v,k) = (k - (h \oplus r))/v & \lambda_r(\alpha,\beta,h) = \alpha \oplus h \\
\mu_h(s,r,v,k) = (k - v \cdot s) \oplus r &
\end{array}$$

- **Schnorr category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{V} = \mathcal{K} = \mathcal{S} = \mathcal{A} = \mathbb{Z}_q$ and $\mathcal{B} = \mathcal{H}$. The variable t is not used and is taken from an arbitrary set $\mathcal{R}$. $(\mathfrak{p}3)$ is implied by the collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ do not hold because λ_r cannot be defined. $(\mathfrak{o}3)$ needs the restriction $\mathcal{V} = \mathbb{Z}_q^{\times}$. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_{\mathfrak{m}} = 1$ and $\varepsilon_{\mathfrak{h}} = 1$.

$$\begin{array}{ll}
\phi(h,r,s) = s & \lambda_h(\alpha,\beta,r) = \beta \\
\psi(h,r,s) = h & \lambda_s(\alpha,\beta,r) = \alpha \\
\sigma(h,r,v,k) = k - v \cdot h & \\
\mu_h(s,r,v,k) = (k - s)/v &
\end{array}$$

- **Swapped-Schnorr category.** Let $\mathcal{H} \subset \mathbb{Z}_q$, $\mathcal{K} = \mathcal{S} = \mathcal{B} = \mathbb{Z}_q$, $\mathcal{V} = \mathbb{Z}_q^{\times}$ and $\mathcal{A} = \mathcal{H}$. The variable t is not used and is taken from an arbitrary set $\mathcal{R}$. $(\mathfrak{p}3)$ is implied by the collision-resistance of H. $(\mathfrak{g}1)$ and $(\mathfrak{g}2)$ do not hold because λ_r cannot be defined. $(\mathfrak{m}2)$ and $(\mathfrak{h}2)$ hold with $\varepsilon_{\mathfrak{m}} = 1$ and $\varepsilon_{\mathfrak{h}} = 1$.

$$\begin{array}{ll}
\phi(h,r,s) = h & \lambda_h(\alpha,\beta,r) = \alpha \\
\psi(h,r,s) = s & \lambda_s(\alpha,\beta,r) = \beta \\
\sigma(h,r,v,k) = (h - k)/v & \\
\mu_h(s,r,v,k) = v \cdot s + k &
\end{array}$$

3.6 Examples of published signature schemes

- ElGamal scheme [11] is defined on the multiplicative group $\mathbb{Z}_p^{\times}$, with a slight variant of ElGamal category (where $-r$ replaces r), ElGamal projection and type I hash.

- DSA scheme [25] is defined on a prime order subgroup of the multiplicative group $\mathbb{Z}_p^\times$, with ElGamal category, DSA projection and type I hash.
- ECDSA scheme [17] is defined on a prime order elliptic curve subgroup with ElGamal category, ECxq projection and type I hash.
- GOST 34.10 scheme [22] is defined on a prime order multiplicative subgroup of $\mathbb{Z}_p$, with a slight variant of GOST category (where $-r$ replaces r), DSA projection and type I.
- KCDSA scheme [18] is defined on a prime order multiplicative subgroup of $\mathbb{Z}_p$ or on a prime order elliptic curve subgroup, with KCDSAxor category, KCDSA projection and type I hash, where some certification data is hashed together with the message.
- ECGDSA scheme [2] is defined on a prime order elliptic curve subgroup, with GDSA category, ECxq projection and type I hash.
- DSA-II scheme [6] is defined on a prime order multiplicative subgroup of $\mathbb{Z}_p$, with ElGamal category, KCDSA projection and type II hash.
- ECDSA-II scheme [19] is defined on a prime order elliptic curve subgroup, with ElGamal category, ECxq projection and type II hash.
- ECDSA-III scheme [19] is defined on a prime order elliptic curve subgroup, with ElGamal category, ECaddq projection and type II hash.
- Schnorr scheme [34] is defined on a prime order multiplicative subgroup of $\mathbb{Z}_p$, with a slight variant of Schnorr category (where $-h$ replaces h), ElGamal projection and type II hash.
- Nyberg-Rueppel scheme [29, 30] is a scheme with total message recovery: no variable m. It is defined on a prime order multiplicative subgroup of $\mathbb{Z}_p$, with Schnorr category, NR projection and type II hash defined by $\mathsf{H}(r) = r \bmod q$.
- PVSSR scheme (Pintsov-Vanstone Signature Scheme with message Recovery [31]) is defined on a prime order elliptic curve subgroup with a slight variant of Schnorr category (where $-h$ replaces h), PVSSR projection and type II hash.
- Naccache-Stern scheme [24] is defined on a prime order elliptic curve subgroup with ElGamal category, NS projection based on ECxq projection and type I hash.
- Abe-Okamoto scheme [1] is a scheme with total message recovery: no variable m. It is defined on a prime order elliptic curve subgroup with Schnorr category, the xor variant of NS projection based on ECx2 projection and type II hash $\mathsf{H}(r)$.

4 Security lemmas

4.1 Random oracle model and almost invertible functions

The random oracle model builds an oracle for a one-way function f, that answers to queries for $\mathsf{f}(x)$ with some uniformly distributed value y. Suitable functions have uniform output, are collision-resistant, etc.
If the function f is almost invertible, then the random oracle model should also allow queries for $\mathsf{f}^{-1}(x)$.

Results that are proven for the random oracle model applied to collision-resistant one-way functions with uniform output are also valid for the random oracle model applied to collision-resistant almost invertible functions with uniform output.

The simulator builds an input/output table for f in answer to the oracle queries. To show that one-wayness is not required for the random oracle model, and that almost-invertibility cannot change a security result, it is sufficient to show that the answers given to f^{-1} queries have a negligible influence to the answers given to f queries.
A f query is influenced by a previous f^{-1} query if either the input of the f query was the output of the f^{-1} query or the output of the f query was the input of f^{-1} query. The second event is unlikely since f is collision-resistant. The first event does not learn to the attacker anything new.

4.2 The forking lemmas

This is a family of lemmas, found e.g. in [32], and is a tool for proofs of security in the random oracle model. The forking lemmas holds when the scheme has the following property: each forgery can be linked to a unique "critical" query to the random oracle. The critical query is an input x such that knowing $x \overset{\mathsf{f}}{\mapsto} y$ is necessary to check if the forgery is valid.

The forking lemmas show that if a simulator can obtain a forgery with respect to a given choice for the random oracle, then it is possible to use the same simulator to obtain another forgery with same critical query but a different choice for the random oracle.

The forking lemmas also holds in the random oracle model for a almost invertible function, if the critical query hypothesis holds.

Forking lemma with non-uniform security proof. This lemma is taken from [32]. Let q_S be the number of signature queries, q_H the number of oracle queries, n_H the number of possible outputs for the random oracle and ε the probability that the forger outputs a valid forgery.

There exists constants c_0 and c_1 such that if $\varepsilon \geq c_0 \cdot q_H/n_H$ then after less than of $c_1 \cdot q_H/\varepsilon$ replays of the simulation with different choices for the random oracle, one can obtain (with some probability ε') another forgery with the same critical query having another uniform random answer.

In [32, *Lemma 8*] we have $c_0 = 7$, $c_1 = 2(7 + \frac{1}{q_H})$ and $\varepsilon' = 3/25$.

Forking lemma with expected running time. This lemma is taken from [32].

There exists constants c_0 and c_1 such that if $\varepsilon \geq c_0 \cdot q_H/n_H$ then after an expected number of $c_1 \cdot q_H/\varepsilon$ replays of the simulation with different choices for the random oracle, one can obtain (with probability 1) another forgery with the same critical query having another uniform random answer.

In [32, *Theorem 10*] we have $c_0 = 7$ and $c_1 = 84480$.

The improved forking lemma with non-uniform security proof. This lemma is taken from [6] and is used to idealize $\ell+1$-collision-resistant functions.

There exists constants c_0 and c_1 such that if $\varepsilon \geq c_0 \cdot q_H/n_H$ then after an expected number of $c_1 \cdot q_H/\varepsilon$ replays of the simulation with different choices for the random oracle, one can obtain (with some probability ε') ℓ other forgeries with the same critical query but having other uniform random answers.

In [6, *Lemma 10*] we have $c_0 = 4/q_H$, $c_1 = 24\ell \log(2\ell) + \frac{1}{q_H}$ and $\varepsilon' = 1/96$.

4.3 Proofs of the forking lemmas

In the next version of this document, this section will review the results from [32, 6] and explain the values for c_0, c_1 and ε'.

5 Security proofs

Lemma (unique representation). *If the discrete logarithm of $V \in \langle G \rangle$ is hard to compute and if two representations $R = G^{\alpha}V^{\beta}$ and $R = G^{\alpha'}V^{\beta'}$ have been computed then $\alpha = \alpha'$ and $\beta = \beta'$.*

This is proven by $(\alpha - \alpha') = (\beta' - \beta) \cdot \log V$.

5.1 Security proof with idealized p

This proof is based on one of the results from [6]. In this proof, H may be a Type I or Type II hash function, and p may be almost invertible.

A DL-based signature scheme is existentially unforgeable (and non-malleable) under adaptive chosen message attacks if the discrete logarithm is hard, if H is collision-resistant, if p is a random oracle and if the category has properties $(\mathfrak{o}1)$, $(\mathfrak{o}2)$, $(\mathfrak{p}2)$, $(\mathfrak{p}1)$, and $(\mathfrak{p}3)$. The security reduction is loose.

To answer a signature query for m, the simulator generates a random r and a random s, computes $h = \mathsf{H}(m, r)$ and $R = G^{\phi(h,r,s)}V^{\psi(h,r,s)}$. With property $(\mathfrak{p}2)$, the value R is uniformly distributed and with property $(\mathfrak{p}1)$ the value s has same distribution as for the signing algorithm. The simulator sets the oracle table $\mathsf{p}(R) := r$. The signed message is (m, r, s). p-oracle queries that were not defined by a signature query are answered with a random value. If p is almost invertible, then p^{-1}-oracle queries are answered with some $R = G^{\alpha'}V^{\beta'}$ for random α' and β'. The probability that the oracle table cannot be set is the probability of a collision in R, which is low if $(q_H + q_S)^2 \leq q$.

When the forger outputs its forgery (m, r, s), the critical query is the value $R = G^{\alpha}V^{\beta}$ where $\alpha = \phi(h, r, s)$, $\beta = \psi(h, r, s)$ and $h = \mathsf{H}(m, r)$. Let us suppose that the critical query was part of a signature query for some m' that answered $(m', r', s') \neq (m, r, s)$. We define $h' = \mathsf{H}(m', r')$, $\alpha' = \phi(h', r', s')$ and $\beta = \psi(h', r', s')$. Validity of the signature means that $R = G^{\alpha'}V^{\beta'}$ and the unique representation of R implies $\alpha' = \alpha$ and $\beta' = \beta$. We also have $r' = r = \mathsf{p}(R)$ and property $(\mathfrak{o}1)$ implies $h = \lambda_h(\alpha, \beta, r) = h'$ and property $(\mathfrak{o}2)$ implies $s = \lambda_s(\alpha, \beta, r) = s'$. Therefore $(m', r', s') \neq (m, r, s)$ implies $m' \neq m$ with $\mathsf{H}(m', r') = \mathsf{H}(m, r')$. We found a collision in H.

Let us suppose that the critical query was a p-oracle query for R. The forking lemma allows to have another forgery $(m', r', s') \neq (m, r, s)$ with same critical R but a different oracle for p. The unique representation of R implies $\alpha' = \alpha$ and $\beta' = \beta$. Therefore the simulator got α, β, m and m' such that $\lambda_h(\alpha, \beta, r) = \mathsf{H}(m, r)$ and $\lambda_h(\alpha, \beta, r') = \mathsf{H}(m', r')$ for random r and r', which is intractable if ($\mathfrak{p}3$) holds.

Let us suppose that the critical query was an p^{-1}-oracle query that returned $R = G^{\alpha'} V^{\beta'}$. The unique representation of R implies $\alpha' = \alpha$ and $\beta' = \beta$, which is very unlikely because α' and β' were kept secret.

5.2 Security proof with idealized H

This proof is based on one of the results from [6]. In this proof, H is a type II hash function.

A DL-based signature scheme is existentially unforgeable under adaptive chosen message attacks if the discrete logarithm is hard, if H *is a random oracle with large output set, if* p *is almost uniform and* $\ell + 1$*-collision-resistant and if the category has properties* ($\mathfrak{o}1$)*,* ($\mathfrak{o}2$)*,* ($\mathfrak{h}1$)*, and* ($\mathfrak{h}2$)*. Collision-resistance of* p *also implies non malleability. The security reduction is loose.*

To answer a signature query, the simulator generates random $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ and computes $R = G^\alpha V^\beta$, $r = \mathsf{p}(R)$, $h = \lambda_h(\alpha, \beta, r)$ and $s = \lambda_s(\alpha, \beta, r)$, until $h \in \mathcal{H}$ and $s \in \mathcal{S}$. This is equivalent to using the signature generation algorithm with $k = \alpha + v \cdot \beta$ therefore this simulation has the same output distribution. Property ($\mathfrak{h}2$) says that the expected number of random α, β needed is less than $\frac{1}{\varepsilon_{\mathfrak{h}}}$. The simulator sets the oracle table $\mathsf{H}(m, r) := h$. The signed message is (m, r, s).

Oracle queries that were not defined by a signature query are answered with a random value. The value of R is uniformly distributed for random α and β. If p is $\frac{1}{n}$-almost uniform then the probability that the oracle table cannot be set is bounded by the probability of a collision in r, which is low if $(q_H + q_S)^2 \leq n$.

When the forger outputs its forgery (m, r, s), the critical query is the H-oracle query of (m, r).

Let us suppose that the critical query was part of a signature query. It returned a valid (m, r, s') with the same oracle. Therefore $h' = h$. If p is collision-resistant, then $R = R'$ and its unique representation implies $\alpha' = \alpha$ and $\beta' = \beta$ and property ($\mathfrak{o}2$) implies $s' = s$.

Let us suppose that the critical query was an oracle query for (m, r). The improved forking lemma allows to have ℓ other forgeries (m, r, s_i)

with same critical (m, r) but different oracle for H. Since all $\mathsf{p}(R_i) = r$, the $\ell + 1$ collision-resistance of p implies that there exist a pair where $R_i = R_j$. Unique representation implies $\alpha_i = \alpha_j$ and $\beta_i = \beta_j$. Property ($\mathfrak{o}1$) implies a unique possible value $h_i = h_j$, which is unlikely to be the one given by the two different oracles for H, because the output set is large.

5.3 Security proof with idealized $\langle G \rangle$

The generic group model was introduced by Shoup [36] and extended by Brown [7] to prove the security of ECDSA.

A DL-based signature scheme is existentially unforgeable under adaptive chosen message attacks in the generic group model if H *is uniform and collision-resistant, if* p *is almost uniform and almost invertible and if the category has properties* ($\mathfrak{g}1$) *and* ($\mathfrak{g}2$). *The security reduction is tight.*

We don't include in this document the proof given in [7] but we show below how it can be adapted to other schemes than ECDSA, using our general framework.
The proof was written for a type I hash function, but it also works for a type II hash. It was written for ElGamal category, but it works for all category with properties ($\mathfrak{g}1$) and ($\mathfrak{g}2$).

- In [Table 1] step 3 of **Hint** is replaced with
  $s_{m+1} = z_1 \cdot \sigma(h_{m+1}, \mathsf{p}(A_{m+1}), z_2 z_1^{-1}, z_{m+1})$.
- In [Table 2] steps 1 and 2 of **Hint** are replaced with
  $C_{(m+1)1} = \phi(h_{m+1}, \mathsf{p}(A_{m+1}), s_{m+1})$ and
  $C_{(m+1)2} = \psi(h_{m+1}, \mathsf{p}(A_{m+1}), s_{m+1})$.
- In [Table 4] step 2.b should use $\mathsf{p}^{-1}(\lambda_r(C_{i1}, C_{i2}, e))$.
- In [Table 7] step 1.b.iii should use $\mathsf{p}^{-1}(\lambda_r(C_{i1}, C_{i2}, \hat{e}_i))$.

Property ($\mathfrak{g}2$) is used when the proof shows that
$r = ... = \lambda_r(C_{m1}, C_{m2}, \hat{e}_i) = \lambda_r(\phi(\mathsf{H}(m), r, s_m), \psi(\mathsf{H}(m), r, s_m), \hat{e}_i)$
and then deduces that $\hat{e}_i = \mathsf{H}(m)$.

6 The PECDSA proposal

6.1 Comments on the security proofs

Comparison with the results from [6]. Two above results follow closely the proofs from [6], but their interaction with the components of the scheme are more clearly detailed. Property ($\mathfrak{p}3$) was not clearly defined in term of interaction between the category and H.

Moreover, we showed that the proof with an idealized p also works if p is almost invertible.

Comparison with the results from [7]. Our result is more general but overlooked some details that are found in [7]. For example we don't consider zero-finder-resistance, because our toolbox restricts ElGamal category to $\mathcal{H} \subset \mathbb{Z}_q^\times$ to meet properties ($\mathfrak{g}1$) and ($\mathfrak{g}2$).

6.2 Comments on the toolbox

Projections. None of the projections previously proposed in the literature have all the required properties for our three proofs. This is the reason why we described how to build a permuted projection. The permuted EC projections probably have all the required properties.
If partial message recovery is useful, Group projections are the best candidates.

Categories. ElGamal category is the choice with worst properties, even if it is the most widely used in practice.
Schnorr category is the simplest choice, but it has some drawbacks: it does not allow to build schemes proven with an idealized $\langle G \rangle$ and its use may be patented. Swapped-Schnorr only has the first drawback.

NB: for Schnorr category, the order q can have intractable factorization, because no inverse is computed. For the KCDSA categories and Swapped-Schnorr category, computing inverses in $\mathbb{Z}_q^\times$ is only needed for key generation. If the signer's only private information is v^{-1}, neither the verifier nor the signer need to know the factorization of q. Intractable factorization might be useful for an identity based scheme [21].

6.3 Description of PECDSA

Rationale for our design. PECDSA means Provable Elliptic Curve Digital Signature Algorithm.
Taking the best for each component, we propose a Type II signature scheme, with $\mathcal{H} = [\mathbb{Z}_q]_\#$ so we can rely on standard hash functions, with permuted EC projection and with KCDSAxor category. This scheme is proven secure against existential forgery in a chosen message attack if any one of the components $\langle G \rangle$, H or p is idealized.
To be secure in the multi key setting and against parameter manipulation, the parameters for the curve and the public key are included in the components H and P, as it is done for KCDSA [18]. See also [26, 12].

- **Domain parameters.** The security parameter is an integer κ, e.g. 160. Let $\langle G \rangle$ be an elliptic curve group of known prime order $q \in [2^\kappa, 2^{\kappa+1}]$ and known generator G, let H be a hash function with κ bits of output and P a symmetric cipher on blocks of κ bits and with key of κ bits.
- **Key generation algorithm.** The key generation algorithm chooses a random $v \in \mathbb{Z}_q^\times$ and sets $\mathsf{pk} = V = G^v$ and $\mathsf{sk} = v^{-1}$.
  The certification value z is the κ bits long hash of the certification data, which contains at least the description of $\langle G \rangle$ and the values G and V.
- **Verification algorithm.** The verification of $(m, r, s) \in \mathcal{M} \times [\mathbb{Z}_q]_\# \times \mathbb{Z}_q$ begins with $h = \mathsf{H}(z, m, r)$, $R = G^{h \oplus r} V^s$, and checks if $r \overset{?}{=} \mathsf{P}_z(R_x \bmod 2^\kappa)$.
- **Signing algorithm.** To sign the message m one takes a random $k \in \mathbb{Z}_q$, computes $R = G^k$, $r = \mathsf{P}_z(R_x \bmod 2^\kappa)$, $h = \mathsf{H}(z, m, r)$ and $s = v^{-1} \cdot (k - (h \oplus r))$. The signed message is (m, r, s).

Variant with partial message recovery. If message recovery is useful, then a group projection similar to NS projection (with XOR) is used. The redundancy function ρ concatenates $\kappa/2$ zeroes at the end of a $\kappa/2$ message $\bar{m}$.

- **Verification algorithm.** The verification of $(\hat{m}, r, s) \in \hat{\mathcal{M}} \times [\mathbb{Z}_q]_\# \times \mathbb{Z}_q$ begins with $h = \mathsf{H}(z, \hat{m}, r)$, $R = G^{h \oplus r} V^s$ computes $(\bar{m}, t) = r \oplus \mathsf{P}_z(R_x \bmod 2^\kappa)$ and checks if $t \overset{?}{=} 0...0$.
- **Signing algorithm.** To sign the message $(\hat{m}, \bar{m})$ one takes a random $k \in \mathbb{Z}_q$, computes $R = G^k$, $r = (\bar{m}, 0...0) \oplus \mathsf{P}_z(R_x \bmod 2^\kappa)$, $h = \mathsf{H}(z, \hat{m}, r)$ and $s = v^{-1} \cdot (k - (h \oplus r))$. The signed message is $(\hat{m}, r, s)$.

The generation of *k*. If the initial seed k used for signature generation is partilly known to the attacker, then it may be possible to break the scheme [27, 28].

For this reason the use of an external generator of random numbers may add weaknesses to some implementations of the scheme. A suggested technique similar to [12] is proposed: a random value Δ is added to the secret key, a hash function H_q with output in $\mathbb{Z}_q$ is used. The signature generation for a message m computes $h' = \mathsf{H}(z, m)$, then $k = \mathsf{H}_q(\Delta, h')$, R and r as before and $h = \mathsf{H}(h', r)$. The verification computes $h = \mathsf{H}(\mathsf{H}(z, m), r)$. It is straightforward to check that the security proofs still hold for this deterministic variant.

6.4 Why is it useful to propose this new scheme.

So many DL-based schemes have been proposed that a new one may seem useless. However, we feel that this new scheme has important advantages.

- **Security.** The two actual de facto standards (DSA and ECDSA) have no convincing security proof. ECDSA can be proven in the generic group model [7] but an elliptic curve group has automorphisms and the generic group model does not apply directly [37].
  Most of the published weaknesses of DSA or ECDSA [5, 33, 37, 38] are solved by KCDSA, but this scheme can only be proven with idealized p.
  We feel that current advances in the field of provable security and the long term security needed by some applications of digital signature are a strong argument for a design where all components have the best security properties.

  Standardized hash functions don't have their output in some $\mathbb{Z}_q$ therefore $\mathcal{H} = [\mathbb{Z}_q]_\#$ should be used. The efficiency of the security proof with idealized H is optimal if $\varepsilon_{\mathfrak{h}} = 1$, which is the case for Schnorr, Swapped-Schnorr or KCDSAxor categories.
- **Performance.** In all these schemes, the running time is dominated by the cost of exponentiations in $\langle G \rangle$ and eventually by the hashing of a large message. Therefore the easyness of implementing the scheme is a better evaluation.
  Because there exist no simple technique to generate a valid k for the ElGamal category, any implementation of DSA or ECDSA either may fail to sign (with extremely low probability) or has to loop to choose another k (which is required by the standards but complicates the implementation).
  Some categories need to avoid non invertible values for h or k or r. The test is fast to implement but makes the code grow and can be the source of errors. ElGamal category needs to compute inverses in $\mathbb{Z}_q$ both for signature and verification, GOST and GDSA categories only need this for verification, KCDSA and Schnorr categories don't.
  Resistance against implementation-dependant attacks is also linked to the simplicity of the implementation.

References

1. M. Abe and T. Okamoto. A signature scheme with message recovery as secure as discrete logarithms. In K.-Y. Lam, E. Okamoto, and C. Xing, editors, *Proceedings*

of Asiacrypt'99, number 1716 in Lecture Notes in Computer Science, pages 378–389. Springer-Verlag, 1999.

2. Algorithms group of the EESSI-SG. Elliptic Curve German Digital Signature Algorithm. Described in [16], 1999.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of Conference on Computer and Communications Security – CCS'93*, pages 62–73. ACM Press, Nov. 1993. Full paper available at `http://www-cse.ucsd.edu/users/mihir/papers/ro.html`.
4. M. Bellare and P. Rogaway. The exact security of digital signature – how to sign with RSA and Rabin. In U. Maurer, editor, *Proceedings of Eurocrypt'96*, number 1070 in Lecture Notes in Computer Science, pages 399–416. Springer-Verlag, 1996. Revised version: `http://ww-cse.ucsd.edu/users/mihir/papers/crypto-papers.html`.
5. D. Bleichenbacher. Generating ElGamal signatures without knowing the secret key. In U. Maurer, editor, *Proceedings of Eurocrypt'96*, number 1070 in Lecture Notes in Computer Science, pages 10–18. Springer-Verlag, 1996.
6. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In H. Imai and Y. Zheng, editors, *Proceedings of Public Key Cryptography – PKC'00*, number 1751 in Lecture Notes in Computer Science, pages 276–292. Springer-Verlag, 2000. Available at `http://www.di.ens.fr/~pointche/pub.php?reference=BrPoVaYu00`.
7. D. R. L. Brown. Generic groups, collision resistance, and ECDSA. Available at `http://eprint.iacr.org/2002/026/`, 2002.
8. D. R. L. Brown and D. B. Johnson. Formal security proofs for a signature scheme with partial message recovery. In D. Naccache, editor, *Proceedings of CT-RSA'01*, number 2020 in Lecture Notes in Computer Science, pages 126–142. Springer-Verlag, 2001. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-39.pdf`.
9. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of Symposium on Theory of Computing – STOC'98*, pages 209–218. ACM Press, 1998. Available at `http://theory.lcs.mit.edu/~oded/rom.html`.
10. A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Y. Zheng, editor, *Proceedings of Asiacrypt'02*, Lecture Notes in Computer Science. Springer-Verlag, 2002. Available at `http://eprint.iacr.org/2002/086/`.
11. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
12. L. Granboulan. How to repair ESIGN. In *Proceedings of SCN'02*, Lecture Notes in Computer Science. Springer-Verlag, 2002. Available at `http://eprint.iacr.org/2002/074/`.
13. P. Horster, M. Michels, and H. Petersen. Meta-ElGamal signature schemes. In *Proceedings of Conference on Computer and Communications Security – CCS'94*. ACM Press, 1994. Full paper available at `http://www.geocities.com/CapeCanaveral/Lab/8967/TR-94-5.ps.gz`.
14. P. Horster, M. Michels, and H. Petersen. Meta message recovery and meta blind signature schemes based on the discrete logarithm problem and their applications. In J. Pieprzyk and R. Safavi-Naini, editors, *Proceedings of Asiacrypt'94*, number 917 in Lecture Notes in Computer Science, pages 224–237. Springer-Verlag, 1994. Full paper available at `http://www.geocities.com/CapeCanaveral/Lab/8967/TR-94-9.ps.gz`.

15. P. Horster, M. Michels, and H. Petersen. Meta signature schemes giving message recovery based on the discrete logarithm problem. In *Proceedings of 2nd int. workshop on IT-Security*, 1994. Full paper available at `http://www.geocities.com/CapeCanaveral/Lab/8967/TR-94-4.ps.gz`.
16. International Organization for Standardization. ISO/IEC FCD 15946-2: Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures, 1999. Final Committee Draft.
17. D. B. Johnson and S. Blake-Wilson. ECDSA. Primitive submitted to NESSIE by Certicom, Sept. 2000.
18. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. In K. Ohta and D. Pei, editors, *Proceedings of Asiacrypt'98*, number 1514 in Lecture Notes in Computer Science, pages 175–186. Springer-Verlag, 1998. Also available at `http://grouper.ieee.org/groups/1363/P1363a/PSSigs.html` as an IEEE P1363a submission.
19. J. Malone-Lee and N. P. Smart. Modifications of ECDSA. In *Proceedings of Selected Areas in Cryptography – SAC'02*, Lecture Notes in Computer Science. Springer-Verlag, 2002.
20. U. M. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D. W. Davies, editor, *Proceedings of Eurocrypt'91*, number 547 in Lecture Notes in Computer Science, pages 498–507. Springer-Verlag, 1991.
21. U. M. Maurer and Y. Yacobi. A non-interactive public-key distribution system. *Designs, Codes, and Cryptography*, 9(3):305–316, 1996. Available from `http://www.crypto.ethz.ch/~maurer/publications.html`, final version of [20].
22. M. Michels, D. Naccache, and H. Petersen. GOST 34.10. A brief overview of Russia's DSA. *Computers and Security*, 8(15):725–732, 1996. Available at `http://www.geocities.com/CapeCanaveral/Lab/8967/TR-95-14.zip`.
23. A. Miyaji. A message recovery signature scheme equivalent to DSA over elliptic curves. In K. Kim and T. Matsumoto, editors, *Proceedings of Asiacrypt'96*, number 1163 in Lecture Notes in Computer Science, pages 1–14. Springer-Verlag, 1996.
24. D. Naccache and J. Stern. Signing on a postcard. In Y. Frankel, editor, *Proceedings of Financial Cryptography – FC'00*, number 1962 in Lecture Notes in Computer Science, pages 121–135. Springer-Verlag, 2000. Available at `http://grouper.ieee.org/groups/1363/Research/contributions/Postcard.ps`.
25. National Institute of Standards and Technology. FIPS-186: Digital Signature Standard (DSS), Nov. 1994. Available at `http://csrc.nist.gov/publications/fips/`.
26. NESSIE consortium. NESSIE Security report. Deliverable report D20, NESSIE, 2002. Available from `http://www.cryptonessie.org/`.
27. P. Q. Nguyen and I. Shparlinsky. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15:151–176, 2002. Available at `ftp://ftp.ens.fr/pub/dmi/users/pnguyen/PubDSA.ps.gz`.
28. P. Q. Nguyen and I. Shparlinsky. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Design, Codes and Cryptography*, 2002. Available at `ftp://ftp.ens.fr/pub/dmi/users/pnguyen/PubECDSA.ps.gz`.
29. K. Nyberg and R. A. Rueppel. A new signature scheme based on the dsa giving message recovery. In *Proceedings of Conference on Computer and Communications Security – CCS'93*, pages 58–61. ACM Press, Nov. 1993.
30. K. Nyberg and R. A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In A. De Santis, editor, *Proceedings of Eurocrypt'94*, number 950 in Lecture Notes in Computer Science, pages 182–193. Springer-Verlag, 1994.

31. L. A. Pintsov and S. A. Vanstone. Postal revenue collection in the digital age. In Y. Frankel, editor, *Proceedings of Financial Cryptography – FC'00*, number 1962 in Lecture Notes in Computer Science, pages 105–120. Springer-Verlag, 2000. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2000/corr2000-43.ps`.
32. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. Available at `http://www.di.ens.fr/~pointche/pub.php?reference=PoSt00`.
33. T. Rosa. On key-collisions in (EC)DSA schemes. Crypto'02 rump session, 2002. Available at `http://eprint.iacr.org/2002/129/`.
34. C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Proceedings of Crypto'89*, number 435 in Lecture Notes in Computer Science, pages 239–252. Springer-Verlag, 1989.
35. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
36. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of Eurocrypt'97*, number 1233 in Lecture Notes in Computer Science, pages 256–266. Springer-Verlag, 1997. Revised version available at `http://shoup.net/papers/dlbounds1.pdf`.
37. J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart. Flaws in applying proof methodologies to signature schemes. In M. Yung, editor, *Proceedings of Crypto'02*, number 2442 in Lecture Notes in Computer Science, pages 93–110. Springer-Verlag, 2002. Available at `http://www.di.ens.fr/~pointche/pub.php?reference=MaPoSmSt02`.
38. S. Vaudenay. Hidden collisions on DSS. In N. Koblitz, editor, *Proceedings of Crypto'96*, number 1109 in Lecture Notes in Computer Science, pages 83–88. Springer-Verlag, 1996.
39. D. H. Youm and P. J. Lee. Security proof for KCDSA under the random oracle model. In *Proceedings of CISC'99*, pages 173–180, 1999.