Penalty Deposits For Discrete Logarithms

Louis Goubin
CryptoExperts

Francisco Vial-Prado
Fortanix

Abstract

We propose a technique that allows a trusted party in some protocol to attach deposits to their own secrets, without impacting security, and convincing other parties using zero-knowledge proofs. The rationale is that, should this party compromise the protocol by blundering or revealing its secrets, they lose the attached deposit to attackers. As a result, stakeholders gain confidence on how this trusted party operates, and breaches become public as soon as deposits are claimed. We formalize this technique and related security notions, provide two practical constructions in generic discrete logarithm groups, and analyze its application to ECDLP-based protocols.

1 Introduction

In light of the multiple surveillance disclosures since Snowden [25], governments pushing legislation of new key surrender variants [2, 3], and deficient or outdated cryptography implementations being used far and wide [1], we believe that the assumption that trusted parties dutifully protect their secrets at all times is strong.

There are often transparency mechanisms to deincentivize or prevent deviations in established security protocols, such as recording actions in public ledgers. However, it seems common practice to instead move the trust elsewhere or split it among several actors. In this article, we propose a more drastic technique that allows the trustee to prove its worth by means of *attaching a deposit* to the secret they claim to protect, with the public guarantee that any actor that gains access to the protected secret is able to claim the deposit. This deposit is anything the trustee cannot afford to lose, such as another secret key, or a private key in a cryptocurrency network allowing to claim a large fund. The trustee is thus dissuaded from inappro-

priately handling the protected secret at the expense of their trustors, and conversely, trustors are more confident in how the trustee operates.

This does not remove the need for trust. The trustee could still use their secrets maliciously, without the need to reveal them. Even coercion does not stalemate the trustee, as they could perform secure multi-party computations that allow the trustee to hide the secret (without risking the deposit) and the coercing party to obtain what they want; in this way a certificate authority could be coerced by an agency to issue a bad certificate for a confidential target. Instead, this deposit scheme is effective against attackers, when the trustee is simply not up to security standards.

Additionally, the deposit acts as a warning system. In the case of cryptocurrency deposits, the public can monitor the deposit address and, as soon as a transaction is made, they have indication that corresponding protected secret may have been leaked. At the same time, the trustee can voluntarily move the deposit to a different address after security updates or key rotations, and duly inform trustors.

A trivial construction of the above is to have the trustee use the same secret for the protocol and the deposit [1] , but this has the drawback that any attacker claiming the deposit, say, by attacking the cryptocurrency network, is also able to break into the protocol. Considering this, it is desirable that the protocol secret stays within reasonable security levels even if the deposit is claimed by other means.

Crucially, the public also needs proof that a certain deposit is indeed attached to the protected secret, and this proof obviously cannot involve revealing either value. Since disclosing both elements and verifying they are attached is NP, there exists a zero-knowledge proof of the same statement [8]. We call these *Proofs of Projection* (ZK-PP) for brevity, and for reasons that will be apparent in Section 2.

A construction where the deposit is some one-way function of the protected secret (such as a preimage-resistant hash of the secret into a suitable key space) is only practical when the transformation is compatible with zero-knowledge proofs. This line of thought already allows to provide a generic construction in the random-oracle model.

In this article we formalize all of the above, and provide a practical construction that can be applied to protocols that depend on the difficulty of the discrete logarithm problem in generic groups.

[1] The same secret in two protocols should not be used, even if seemingly independent, as the underlying security reductions may no longer hold.

Related work. The idea of penalizing misbehaving parties in protocols is not new. For instance, in [7], parties operating a secure multi-party computation protocol are enforced to finish computation, by means of revealing private informations of aborting parties.

Our proposal is somewhat related to public traitor-tracing systems [4, 21]) in the sense that, if an individual or organization uses our construction with all its providers, by monitoring the deposit addresses they can publicly learn which provider was attacked and revoke the contract. Although, traitor-tracing schemes are focused in colluded parties that compile a pirate decoder, and traces them when this decoder is used, whereas here we trace the party when an attacker breaks its security, and we rely on different assumptions.

Finally, we believe zero-knowledge-friendly hash functions, such as [16], can be used to obtain more efficient constructions that approach our generic random oracle construction.

Contributions and Outline. In Section 2, we formalize the notion of attaching deposits to secrets by borrowing vocabulary from algebraic topology: the deposit and protocol key spaces Y, K are modeled as a topological covering space with a suitable projection $\pi : K \to Y$. The trustee first samples a secure deposit $y \in Y$, transfers a large public amount of cryptocurrency into it, and then samples the protected secret k in the fiber over y. We further require indistinguishability properties to ensure that security is not degraded when using such k in the protocol.

We provide a practical construction and security analysis in the context of discrete-logarithm groups in Section 3, and in Section 4, we apply this construction in ECDLP-based protocols. We research the implications of using well-known curves for the deposit scheme, and found that several are suited for our purposes. We also describe dedicated curves in Appendix A.

2 The Model

Here and henceforth, we identify *the deposit* to the secret that allows to claim the deposit. The protocol secret that the trustee is tasked to protect is referred to as simply *the secret*.

2.1 Uniform Covers

In order to capture the property that the deposit is lost whenever the secret is leaked, we model the key spaces Y, K as the base space and the total space

of covering map, where a projection $\pi : K \to Y$ is public and efficient. In this case, the deposit $y \in Y$ is an element of the base space, and the secret is a random element on the fiber over y (e.g., $\pi^{-1}(\{y\})$). This way, given a secret k, the deposit can be retrieved by computing the projection of k. Conversely, if π is preimage resistant, the knowledge of y does not imply any knowledge about a particular element $k \in K$ on the fiber over y. For cryptographic use, we request that the deposit is sampled uniformly and the secret is indistinguishable from random.

Definition 2.1 (uniform cover)**.** *Let Y be a (discrete) topological space and K be a cover space of Y with projection $\pi : K \to Y$. Let $\chi_{Y:K}$ be the distribution that samples $y \xleftarrow{R} Y$, $k \xleftarrow{R} \pi^{-1}(\{y\})$, and outputs (y, k). We say that $\chi_{Y:K}$ is a uniform cover if k is computationally indistinguishable from a uniform sample of K.*

In order to capture the necessary security notions, we reuse the existing security games from the secret and the deposit protocols.

2.2 Security Games

Let $\mathcal{G}_Y, \mathcal{G}_K$ denote two challenger-adversary games where in the definition of each game, the challenger starts by sampling one secret value in $y \xleftarrow{R} Y, k \xleftarrow{R} K$ respectively (to fix ideas, think of $\mathcal{G}_Y, \mathcal{G}_K$ as the security games of guessing discrete logarithms in different groups). Note by $\mathcal{G}_Y(y)$ the game $\mathcal{G}_Y$ such that instead of sampling uniformly from Y in step 1, the challenger uses the given $y \in Y$. Similarly, define $\mathcal{G}_K(k)$. We consider a challenger and an adversary playing these games in parallel, with the constraint that the challenger samples y, k according to a cover $\chi_{Y:K}$ as above instead of uniformly from Y, K:

Steal or Breach Game (`s||b`): $\mathcal{S}_{\mathcal{G}_Y,\mathcal{G}_K}(\chi_{Y:K})$

1. Challenger samples $(y, k) \leftarrow \chi_{Y:K}$.
2. Adversary and Challenger play $\mathcal{G}_Y(y)$ and $\mathcal{G}_K(k)$ in parallel. Adversary wins if they win either game.

The intuition is that $\mathcal{G}_Y$ is the game of stealing the deposit and $\mathcal{G}_K$ is the game of breaking the protocol. Security against this adversary is paramount, since winning this game implies that the security of either game is reduced when sampling according to the uniform cover. A second, stronger adversary, already knows how to win $\mathcal{G}_Y(y)$. In other words, they already claimed the deposit, and now try to break the protocol.

Steal then Breach Game (`s;b;`): $\mathcal{S}_{\mathcal{G}_K|\mathcal{G}_Y}(\chi_{Y:K})$

1. Challenger samples $(y, k) \leftarrow \chi_{Y:K}$.
2. Challenger sends y to Adversary.
3. Adversary and Challenger play $\mathcal{G}_K(k)$.

It is desirable to defend against this attacker – for instance, an attack against a cryptocurrency network should not impact the security of the protocol.

2.3 Proofs of Projection

For uniform covers to be useful in practice, trustors need some guarantee that the secret indeed projects into a known deposit public key. For this, the party knowing the deposit and the secret constructs a zero-knowledge proof asserting that they know both of these values, and the deposit is a projection of the secret (rather than being unrelated). We refer to these as *Proofs of Projection* (ZK-PP). While it is known that such proofs always exist [8], there might be no efficient constructions. Adapting the well-known zero-knowledge arguments of [13], we provide an efficient ZK-PP for our discrete-log construction, in Section 3.1.1. Recently, similar proofs of knowledge for lattices have been proposed[20], and we believe this enables a lattice-based construction. This is reserved for future work.

2.4 Uniform Cover from Random Oracles

Let us give a construction that covers any secret key space into itself using a random oracle. This is of theoretical interest, as it is optimal in the sense that fibers are singletons (e.g. K is a permutation of Y).

In the random oracle model, we can instantiate universal access to an oracle $h : \{0,1\}^* \to Y$ that produces unpredictable, uniform elements of Y. For $y \in Y$, we define the fiber over y as $h^{-1}(\{y\}) \subset \{0,1\}^*$. This already gives a uniform cover $\chi_{Y:Y}$ as described in Definition 2.1. In other words, let $\chi_{Y:Y}$ be the distribution that samples $k \leftarrow Y$ uniformly, computes $y := h(k)$ and outputs (y, k); it is easy to see that this distribution is a uniform cover of Y onto itself. Also, note that leakage of k implies leakage of y, since we can query the oracle again with k. Conversely, as y is uniformly chosen by the oracle, it is unfeasible to recover k given y.

While theoretically optimal, this construction is obviously not practical. Additionally, a ZK-PP exists only generically. In the following, we focus in giving a practical uniform cover that allow to write zero-knowledge proofs of sampling.

3 Uniform Covers of Discrete Logarithms

In this section, we provide two uniform covers of discrete logarithms with other discrete logarithms. In the first scenario, we consider a cyclic group with a large subgroup of prime order, and in the second, we consider two *resonant* cyclic groups of primer order, in the sense that one order is almost a multiple of the other.

These constructions are of cryptographic interest since they can be applied in any protocols whose security reduces to the discrete logarithm problem in some group.

3.1 Uniform Covers in Composite Groups

Here, we describe a uniform cover of discrete logarithms in a cyclic group of prime order with discrete logarithms in a cyclic group containing it, e.g., a uniform cover of $Y = \mathbb{Z}_p$ with $K = \mathbb{Z}_n$ where p is a prime factor of n.

Throughout this section, let p be a prime[2], q a positive integer with $\gcd(p, q) = 1$, G a cyclic group of order $n = pq$ generated by g, and $H \subset G$ be the subgroup of order p and generator h.

Definition 3.1. *Let p be a prime factor of n, and $\lambda_{p:n}$ be the distribution that samples integers $(y, r) \xleftarrow{R} \mathbb{Z}_p \times \mathbb{Z}_n$, computes $k := y + pr \ (mod\, n)$ and outputs $(y, k) \in \mathbb{Z}_p \times \mathbb{Z}_n$.*

Note that $\lambda_{p:n}$ is equivalent to the distribution that samples $k \xleftarrow{R} \mathbb{Z}_n$ and outputs $(k \pmod p, k)$, which makes more sense for implementation. We have decided to present it as above, since it could be desirable in some contexts to add the constraint that y was committed to before k was sampled (this would probably need some interactive step using randomness from a verifier).

It is easy to see that $\lambda_{p:n}$ is a uniform cover, since k is indistinguishable from uniform in $\mathbb{Z}_n$. The projection of an element $k \in \mathbb{Z}_n$ is simply $y = k \pmod p$; this holds because p divides n. Conversely, it is computationally difficult to compute k from the sole knowledge of y, since the fiber has q elements.

3.1.1 ZK-PP of Discrete Logarithms in composite groups

Let g, h, G, H and $(y, k) \leftarrow \lambda_{p:n}$ be as described above. The goal is to establish a zero-knowledge proof of the facts that the prover knows the

[2]The primality of p can be easily relaxed, we assume it only for the sake of simplicity.

discrete logarithms of $A = h^y$ and $B = g^k$, and that the projection $\pi : k \mapsto y = k \pmod p$ holds. Note in particular that the projection is linear in $\mathbb{Z}_p$, therefore, we adapt the well-known results of [13]. The prover selects a hash-function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^l$ (modeled as a random oracle) and proceeds as follows.

1. $(y', k') \leftarrow \lambda_{p:n}$, $t_1 = h^{y'}, t_2 = g^{k'}$.
2. $c = \mathcal{H}(g, h, A, B, t_1, t_2)$,
3. $r_1 = y' - cy \pmod p$ and $r_2 = k' - ck \pmod n$.

The proof is (c, r_1, r_2). It can be verified by first reconstructing the commitments $t_1' = h^{r_1} A^c$ and $t_2' = g^{r_2} B^c$ and checking the equations

$$c \stackrel{?}{=} \mathcal{H}(g, h, A, B, t_1', t_2') \text{ and } r_1 - r_2 \stackrel{?}{=} 0 \pmod p.$$

3.1.2 Security analysis

If the factorization of the order of a group is known, the difficulty of computing a discrete logarithm in the group is controlled by the size of its largest subgroup. As before, we assume that G has order $n = pq$, $\gcd(p, q) = 1$, p is prime, and H is a subgroup of order p.

Recall also the `s||b` game of Section 2.2. An adversary playing this game in the context of our construction above, faces two discrete logarithm problems $A = h^y$ in H and $B = g^k$ in G but with the additional information that the solutions verify $k \equiv y \pmod p$.

Note that this additional information may allow the adversary to distinguish (A, B) from random $(A', B') \leftarrow H \times G$. For instance, if the generator of H is chosen as $h := g^q \in H$, then $A^{-1} \cdot B^q = 1$. Following this line of thought, the "cross-subgroup" problem of deciding whether $s \stackrel{?}{=} t$ given $(h^s, g^t) \in H \times G$ is a reduction of the above (the special case where $k = y + 0p \pmod n$), and is easy if the discrete logarithm of h with respect to g is known, since for some $z \in \mathbb{Z}_p$ such that $h = g^z$, the solver simply checks if $h^s \stackrel{?}{=} (g^t)^z$ in H. Perhaps this problem is difficult if such z is not known, but it is not realistic to impose such a condition: First, we do not want security to depend on one specific instance of the discrete logarithm where generators may be special, and second, industrial consumers of discrete-logarithm cryptography use parameters issued by organizations, often controlled by governments, instead of sampling their own.

The security of linear relations between discrete logarithms has already been considered before. In the Chaum-Pedersen signature scheme [14], a secret key is an integer x, the public key is $P = g^x$ in a generic group of prime

order, and each new signature is an element of the form $\sigma = m^x$ (where m is some encoding of the data being signed), along with a zero-knowledge proof that indeed $\log_g P = \log_m \sigma$. This also generalizes to discrete logarithms constrained to general linear relations g^u, m^v with $v = au + b \pmod q$, as it suffices to submit $m' = m^a$ and multiply the public key by g^{-b} to simulate the challenger.

Let us discuss security in terms of the `s||b` and `s;b;` games of Section 2.2. For any group G, consider the game DL_G of computing a discrete logarithm in G (e.g., Challenger chooses a generator g and a random exponent x, sends g^x to Adversary, and Adversary attempts to guess x), and the game $\mathrm{DL}_G(g)$ of computing a discrete logarithm in G with base g.

Proposition 1. *Let G be a cyclic group of order $n = pq$ for a prime p and $\gcd(p, q) = 1$. Let $H \subset G$ a subgroup of order p. If an adversary wins $\mathcal{S}_{DL_H, DL_G}(\lambda_{p:n})$ with probability ρ, they can win DL_G or DL_H with probability ρ.*

Proof. Challenger chooses a secret value $y \xleftarrow{R} \mathbb{Z}_n$ and sends $g^y \in G$ to Adversary. Adversary first selects a random values r, s and computes the pair $(g^y \cdot g^{pr}, (g^y)^{qs}) = (g^{y+pr}, (g^{qs})^y) = (g^k, h^y)$, by letting $k := y + pr \pmod n$ and $h := g^{qs}$. The tuple (g^k, h^y) simulates a uniformly random instance of $\mathcal{S}_{\mathrm{DL}_H, \mathrm{DL}_G}(\lambda_{p:n})$. By the definition of `s||b`, they win one of DL_H, DL_G with probability ρ. □

Proposition 2. *Let G be a cyclic group of order $n = pq$ for a prime p with generator g and $\gcd(p, q) = 1$. Let $H \subset G$ be the subgroup of order p and H' the subgroup of order q, generated by g^p. An adversary that wins $\mathcal{S}_{DL_G | DL_H}(\lambda_{p:n})$ with probability ρ, can win $DL_{H'}(g^p)$ with probability ρ.*

Proof. The proof is similar to the proof of Proposition 1. First, Challenger samples $r \xleftarrow{R} \mathbb{Z}_q$ and sends $(g^p)^r \in H'$ to Adversary. Adversary simulates another Challenger, by sampling $y \xleftarrow{R} \mathbb{Z}_p$ and computing $g^y (g^p)^r = g^{k'}$. The pair $(y, g^{k'})$ simulates a uniformly random instance of $\mathcal{S}_{\mathrm{DL}_G | \mathrm{DL}_H}(\lambda_{p:n})$. □

3.2 Uniform Covers in Resonant Groups

Here, we describe a uniform cover between discrete logarithms of two cyclic groups H and G of prime orders p and q, with the additional property defined as follows:

Definition 3.2 (resonant primes)**.** *Let $p < q$ be two prime numbers. We say that p and q are resonant if there is a small positive integer $e \ll p$ such that $q = \lfloor q/p \rfloor p + e$, this is, $q \bmod p$ is small.*

Definition 3.3 (resonant groups)**.** *We say that two cyclic groups of prime order are resonant if their orders are resonant primes.*

Throughout this section, let H, G be resonant groups or prime orders p, q, and let $e = q \bmod p$. We will define a uniform cover of discrete logarithms in H with *a strict subset of* discrete logarithms in G, more precisely, a uniform cover of $Y = \mathbb{Z}_p$ with $K = [0, \lfloor q/p \rfloor p) = [0, q - e) \subsetneq \mathbb{Z}_q$:

Definition 3.4. *Let p, q be two distinct primes such $q = \lfloor q/p \rfloor p + e$. Let $\theta_{p:q}$ be the distribution that samples an integer $k \xleftarrow{R} [0, q - e)$, computes $y := k \bmod p$ and outputs (y, k).*

Note that k is indistinguishable from a uniform integer in $[0, q - e)$ and, since $p \mid q - e$, we also have that y is a uniform integer in $[0, p)$. It follows that $\theta_{p:q}$ is a uniform cover and $k \bmod p$ equals $y \in \mathbb{Z}_p$.

This cover is more practical than the previous one, as known groups can be used (for instance, two distinct prime resonant elliptic curves). On the other hand, there is a penalty on the secret key in the larger group G, introducing an additional security assumption: the hardness of the discrete logarithm problem with known bits.

The security analysis follows from Section 3.1.2, with the following considerations: As mentioned above, the discrete logarithm in G has known most-significant bits, as e is known. This reduces the security level against `s||b` in at least $\log_2(e)$ bits. Second, consider the `s;b;` attacker who already knows y. They face a discrete logarithm instance on G of the form $A = g^{y+pr}$, e.g., $A' = (g')^r$; this is a discrete logarithm instance where the attacker knows that the solution lies in $[0, \lfloor q/p \rfloor)$, therefore, the above security assumption needs to be reused. In particular, if $\lfloor q/p \rfloor$ is small, the attacker can brute-force this instance.

Addressing these two points, it is desirable to have large $\lfloor q/p \rfloor$ and small $q \bmod p$, for example, by selecting H, G of prime orders p, q such that $q = r \cdot p^2 + e$ for a constant r and small e. In this case, assuming that DLOG with known bits is hard, both attackers ultimately deal with discrete-log instances of around $\log(p)$ bits.

3.2.1 ZK-PP of Discrete Logarithms in Resonant Groups

The proof of projection is similar to the previous one (Section 3.1.1). Let g, h, G, H and $(y, k) \leftarrow \theta_{p:q}$ be as described above. The goal is to establish a zero-knowledge proof of the facts that the prover knows the discrete logarithms of $A = h^y$ and $B = g^k$, and that the projection $\pi : k \mapsto y = k \pmod p$ holds. The prover selects a hash-function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^l$ (modeled as a random oracle) and proceeds as follows.

1. $(y', k') \leftarrow \theta_{p:q}$, $t_1 = h^{y'}, t_2 = g^{k'}$.
2. $c = \mathcal{H}(g, h, A, B, t_1, t_2)$,
3. $r_1 = y' - cy \pmod p$ and $r_2 = k' - ck \in \mathbb{Z}$.

The proof is (c, r_1, r_2). It can be verified by first reconstructing the commitments $t_1' = h^{r_1} A^c$ and $t_2' = g^{r_2} B^c$ and checking the equations

$$c \stackrel{?}{=} \mathcal{H}(g, h, A, B, t_1', t_2') \text{ and } r_1 - r_2 \stackrel{?}{=} 0 \pmod p.$$

4 Covering Elliptic Curve Secrets

It is believed that random elliptic curves over finite fields behave like generic groups [23], therefore, we can specialize the constructions of Section 3 to elliptic curve groups. For the composite groups cover (Section 3.1) we use extended curves, e.g., an elliptic curve $E/\mathbb{F}_q$ for the deposit and its second-degree extension $E/\mathbb{F}_{q^2}$ for the secret. As we explain below, this already has the desired group structure. As for the resonant groups cover (Section 3.2) we use any two (resonant) curves.

We also explore which known curves are appropriate in terms of security against `s||b` and `s;b;` adversaries.

4.1 Covers in extended curves

As we explain here, curves defined over $\mathbb{F}_{q^2}$ for a prime power q present the desired group structure, as they always have composite order. In this case, the cover $\lambda_{n:nm}$ from Definition 3.1 can be used to sample the deposit $y \in \mathbb{Z}_n$ and the secret $k \in \mathbb{Z}_{nm}$, and along with the ZK-PP of Section 3.1.1, these secrets may be used in any protocol whose security reduces to ECDLP. We refer to the largest prime factors of n, m as the order and the *co-order*, respectively. The order measures the difficulty of claiming the deposit, while the co-order measures the difficulty of breaking the protocol *once the attacker has already claimed the deposit*, e.g., a `s;b;` adversary.

Fix a prime p, a power $q = p^d$ with $d \geq 1$, and a cyclic elliptic curve $E(\mathbb{F}_q)$ of order n and generator G. Consider the curve $E(\mathbb{F}_{q^2})$. Note that $E(\mathbb{F}_q)$ is a subgroup of $E(\mathbb{F}_{q^2})$ generated by G, therefore, n must divide $|E(\mathbb{F}_{q^2})|$. Moreover, by Weil conjectures we know that $|E(\mathbb{F}_{q^k})| = q^k + 1 - a_k$, where

$$\begin{cases} a_0 &= 2, \\ a_{k+2} &= a_1 \cdot a_{k+1} - q a_k, \end{cases} \tag{1}$$

and $a_1 = q + 1 - |E(\mathbb{F}_p)|$ is the trace of Frobenius (see for instance [24], exercise 5.13). This yields

$$|E(\mathbb{F}_{q^2})| = n \cdot (q + 1 + a_1) = n \cdot (2q + 2 - n).,$$

In the optimal scenario, we are looking for curves defined over $\mathbb{F}_{q^2}$ such that $E(\mathbb{F}_q)$ has prime order n, and prime co-order $m = 2q + 2 - n$. We call these curves *balanced*. For unbalanced curves, we define *second cofactor* as m divided by its largest prime factor.

For example, one curve with prime order and prime co-order is defined over $\mathbb{F}_7$ by the equation $E : y^2 = x^3 + x + 6$, since $|E(\mathbb{F}_7)| = 11$ and $|E(\mathbb{F}_{49})| = 5 \cdot 11$, as desired. We found many known curves to be balanced, such as P-384 (and secp384), 3 of the *Nothing-Up-My-Sleeves* curves [11], the Oakley Random 384-bit [17], and all of the BADA55 [10] curves. We also found random balanced elliptic curves, by searching among curves with prescribed prime order and looking at their quadratic extensions. All of these curves are suitable for use in our deposit-protocol scheme for a discrete-log security level of around $\log(n)/2$ bits. Some unbalanced curves with small cofactors and extended cofactors also suited for our purposes, e.g. Curve25519 [9] and *The Million Dollar Curve* [5]. In the remainder of this Section, we discuss the implications of using this method.

As before, fix an odd prime power q and a known cyclic elliptic curve group $E(\mathbb{F}_q)$ of prime order n, with generator T. Consider the elliptic curve group $E(\mathbb{F}_{q^2})$ of order $nm = n(2q + 2 - n)$ and generator S, and let r be the largest prime factor of m.

Recall that the deposit is a discrete logarithm y in $E(\mathbb{F}_q)$, and the protocol secret is a discrete logarithm k in $E(\mathbb{F}_{q^2})$ of the form $k = y + nl \pmod{nm}$. Both are held by the same party, who publishes T^y and S^k, and a ZK-PP (as described in 3.1.1). As proved in Section 3.1.2, an attacker that succeeds in stealing either k or y with some probability, is also able to break discrete log instances of size n or m. Therefore, against `s||b` attackers, any curve of prime order is suitable[3]

[3]It is easy to relax the prime order condition. If E is a curve with cardinality $n'h$ where

For unbalanced curves with $r < n$, a `s;b;` attacker has an advantage for breaking the protocol, as they can exploit the lack of balance to solve smaller ECDLP instances. For instance, using P-256, there is a 15-bit sacrifice, meaning that the attacker confronts 241-bit ECDLP instances. Using Bitcoin's secp256k1, the co-order has 220 bits, e.g. a 36-bit penalty. In some extreme cases, such as Brainpool256r1, the order and co-order have lengths 256 and 89 bits respectively, making the protocol secret completely vulnerable to `s;b;` attackers.

On the other side of the spectrum, there are curves such that $r > n$. This is desirable, since the `s;b;` faces a larger ECDLP instance when trying

n' is the largest prime factor of the cardinality (far and wide called *the order*) and h is the cofactor, then $\lambda_{n':nm}$ is also a uniform cover.

Curve	(order, co-order)	(cofactor, 2nd cofactor)
BLS12-446	(299, 444)	(148-bit integer, 4)
Curve22103	(218, 220)	(8, 4)
Ed448	(446, 447)	(4, 4)
P-384	(384, 385)	(1, 1)
Curve25519/Ed25519	(253, 253)	(8, 4)
MDC201601	(254, 254)	(4, 4)
BADA55-R-256	(256, 256)	(1, 1)
Tweedledee	(255, 247)	(1, 147)
P-256	(256, 241)	(1, 34905)
gost256	(256, 241)	(1, 17955)
Fp512BN	(512, 471)	(1, 42-bit integer)
BLS12-381	(255, 168)	(126-bit integer, 214-bit integer)
brainpoolP256r1	(256, 89)	(1, 168-bit integer)
ssc-320	(320, 147)	(1, 173-bit integer)
B-283	(282, 87)	(2, 197-bit integer)
bn350	(350, 104)	(1, 246-bit integer)

Table 1: A selection of the 208 curves we analyzed. The order measures the difficulty of the of the deposit ECDLP instance. As usual, $|E(\mathbb{F}_q)| =$ order $\times$ cofactor, and let $m = |E(\mathbb{F}_{q^2})|/|E(\mathbb{F}_q)| =$ co-order $\times$ 2^{nd}cofactor, where the co-order is the largest prime factor of m. It measures the size of the ECDLP instance faced by a `s;b;` adversary. Curves with cofactors (1, 1) are *balanced*; they have prime order and prime co-order. The full set is in Figure 1.

to break the protocol, than the one they faced when claiming the deposit. Note that $m = 2q + 2 - n$ is roughly of the size of q, while the order n can be smaller. See for instance BLS12-446 [6] (top score in Table 1). Another possibility is to sacrifice efficiency and use specialized curves in higher degree extensions (see Section 4.1 below).

4.1.1 Discrete Logarithms in Extended Curves

The extended structure for curves defined over $\mathbb{F}_{q^d}$ allows for non-generic attacks, such as GHS. In a nutshell, these are mounted by writing polynomial equations over $\mathbb{F}_{q^d}$ as d polynomial equations over $\mathbb{F}_q$. These equations define an algebraic variety with the transported group law, where the problem could be easier due to index calculus attacks. Although this has led to practical attacks in some cases (e.g. in combination with decomposition attacks over $\mathbb{F}_{q^6}$ [19]), Pollard methods are more efficient than index calculus for curves over $\mathbb{F}_{q^2}$ and $\mathbb{F}_{q^3}$, hence there is no security reduction due to the GHS attack (see e.g. [15]).

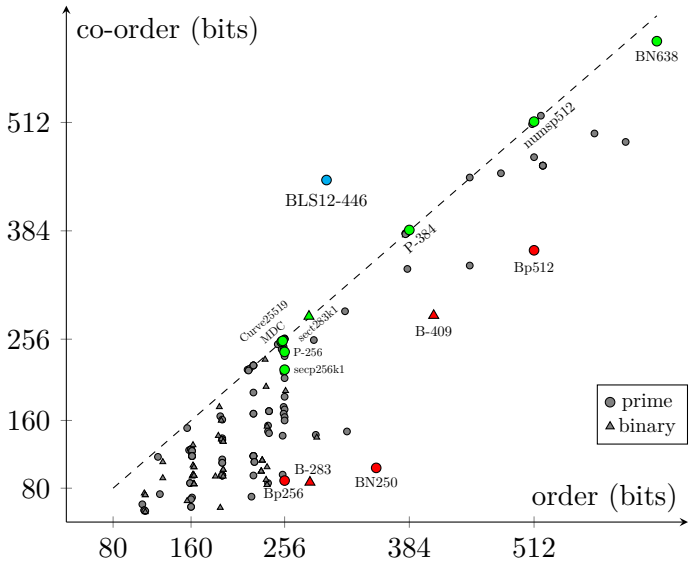

Figure 1: Most curves we analyzed present some security degradation against the `s;b;` attacker, with notable exceptions near the balanced (dashed) line. An interesting case is BLS12-446: as a consequence of having the prescribed embedding degree of 12, the order is 147 bits shorter than the prime, but the co-order is not affected.

4.1.2 Third-degree Extension Curves

Let $E/\mathbb{F}_q$ be an elliptic curve, and let $n = |E(\mathbb{F}_q)|$ as before. We explore the venue of hardening security against a `s;b;` attacker, by considering $E/\mathbb{F}_{q^3}$. As we show here, it has order nl where $l \approx n^2$, hence, security against the `s;b;` adversary is controlled by the largest prime factor of l. Letting $t = q + 1 - n$ denote the trace of Frobenius, the second-degree recurrence relation 1 above solves to

$$a_k = \left(\frac{t + \sqrt{t^2 - 4q}}{2}\right)^k + \left(\frac{t - \sqrt{t^2 - 4q}}{2}\right)^k,$$

which gives us $|E(\mathbb{F}_{p^3})| = n(3(q^2 + q + 1) - 3n(q + 1) + n^2)$. Therefore, as before we can define $l := 3(q^2+q+1)-3n(q+1)+n^2$, and the *third cofactor* as l divided by its largest prime factor. If a curve with small cofactor and small third cofactor is chosen, the security against the `s;b;` attacker, who presumably knows how to compute discrete logarithms in groups of prime order n, faces an ECDLP instance of size l. We do not know if there are curves with prime order and prime l, but we found a curve defined over $\mathbb{F}_p$ with a 256-bit prime p, with prime l and a 10-bit cofactor, in Appendix A.1.

4.2 Covers in resonant elliptic curve groups

There is no reason for a pair of established curves (say, Curve25519 and P-521) to be resonant. However, it is feasible to find pairs of resonant curves, or to find a curve resonant to any given curve, using generating methods of prescribed order. In Appendix A.2 we display a pair of resonant curves with 192-bit and 384-bit orders p, q such that $q = p^2 + 292$, a pair of resonant curves with 256-bit and 512-bit orders p, q such that $q = p^2+840$ and a curve with 512-bit order that is resonant to with NIST's P-256 with $e = 1278$ (this is, an 11-bit penalty on the larger discrete logarithm).

5 Conclusion

In this article, we introduced a technique that incentivizes trustees of cryptographic protocols to dutifully protect their secrets, as if they reveal these secrets by coercion, bad intentions, or deficient security, they may lose a large deposit. Conversely, if they behave as expected, they keep the deposit and trustors gain confidence.

Even if our construction is practical, we believe that there exist more efficient coverings, in the spirit of the random oracle model construction as described in Section 2.4. The main obstacle is to find a projection that is preimage- and collision- resistant, at the same time compatible with proofs of knowledge. With the logarithm-based construction, there is an efficiency cost given by arithmetic in the extended curve, but taking into account, for example, the efficiency of the Four$\mathbb{Q}$ curve (also operating over a degree-2 extension field), it is possible that there is a trade-off favoring the protocol efficiency, for some curves.

As mentioned in Section 2.3, there have been recent advances in lattice-based proofs of knowledge. Indeed, linear relation and range proofs using lattice techniques are already available [20]. For instance, it would be interesting to have specialized proofs of knowledge that allow to cover a discrete-log secret with short vectors of a lattice. This would allow to attach cryptocurrency deposit to secrets in post-quantum resistant security protocols.

Finally, as individuals usually share same data with multiple providers (name, email address), they often cannot know which provider leaked their data. It would be interesting to have a construction where deposits are linked to data instead of keys.

6 Acknowledgments

Author F.VP. would like to thank Jethro Beekman for helpful discussion.

References

[1] Cryptographic competitions - disasters. `https://competitions.cr.yp.to/disasters.html`, consulted in April 2022.

[2] Eff tells e.u.: Don't break encryption. `www.eff.org/deeplinks/2022/03/eff-tells-eu-commission-dont-break-encryption`, consulted in April 2022.

[3] Republican senators introduce bill that tech advocates have warned would weaken privacy. `www.cnbc.com`, consulted in April 2022.

[4] S. Agrawal, S. Bhattacherjee, D. H. Phan, D. Stehlé, and S. Yamada. Efficient public trace and revoke from standard assumptions: Extended

abstract. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[5] T. Baignères, C. Delerablée, M. Finiasz, L. Goubin, T. Lepoint, and M. Rivain. Trap me if you can: The million dollar curve. `https://cryptoexperts.github.io/million-dollar-curve/`.

[6] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Proceedings of the 3rd International Conference on Security in Communication Networks*, SCN'02, page 257–267, Berlin, Heidelberg, 2002. Springer-Verlag.

[7] C. Baum, B. David, and R. Dowsley. Insured mpc: Efficient secure computation with financial penalties. In J. Bonneau and N. Heninger, editors, *Financial Cryptography and Data Security*, pages 404–420, Cham, 2020. Springer International Publishing.

[8] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 37–56, New York, NY, 1988. Springer New York.

[9] D. J. Bernstein. Curve25519: New diffie-hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography - PKC 2006*, pages 207–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[10] D. J. Bernstein, T. Chou, C. Chuengsatiansup, A. Hülsing, E. Lambooij, T. Lange, R. Niederhagen, and C. van Vredendaal. How to manipulate curve standards: A white paper for the black hat https://bada55.cr.yp.to. In L. Chen and S. Matsuo, editors, *Security Standardisation Research*, pages 109–139, Cham, 2015. Springer International Publishing.

[11] J. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: An efficiency and security analysis. *Journal of Cryptographic Engineering*, 2015.

[12] R. Bröker and P. Stevenhagen. Elliptic curves with a given number of points. In D. Buell, editor, *Algorithmic Number Theory*, pages 117–131, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[13] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

[14] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 89–105, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[15] S. Galbraith and P. Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78, 01 2016.

[16] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. Cryptology ePrint Archive, Paper 2019/458, 2019. `https://eprint.iacr.org/2019/458`.

[17] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, RFC Editor, November 1998.

[18] J. Jančár and K. A. Korinsky. ecgen: Tool for generating elliptic curve domain parameters. `https://github.com/J08nY/ecgen`, 2022.

[19] A. Joux and V. Vitse. Cover and decomposition index calculus on elliptic curves made practical. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 9–26, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] V. Lyubashevsky. Lattice-based zero-knowledge and applications. Available at `crypto.sjtu.edu.cn/cis2019/slides/Vadim.pdf`, 2019.

[21] R. Nishimaki, D. Wichs, and M. Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 388–419, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[22] E. Savaş, T. A. Schmidt, and Ç. K. Koç. Generating elliptic curves of prime order. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 142–158, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[23] C.-P. Schnorr and M. Jakobsson. Security of discrete log cryptosystems in the random oracle and the generic model, 2005.

[24] J. H. Silverman. The arithmetic of elliptic curves. In *Graduate texts in mathematics*, 1986.

[25] E. Snowden. *Permanent Record*. Pan Books, 2020.

A Dedicated Elliptic Curves

It is well-known that it is efficiently possible to produce elliptic curves with prescribed order [22, 12] or embedding degree [6], by means of the Complex Multiplication method. In this section we use these methods to generate dedicated curves.

A.1 Dedicated Curves with prime order and co-order

Using Sage and the elliptic curve generator [18], we quickly found dedicated Weierstrass curves that are optimal for our constructions. Our strategy is to loop through curves of prime order, randomly generated by the efficient algorithm given in [12], until $m = 2p + 2 - n$ is prime. See Tables 2 and 3

We also explored hardened security against the `s;b;` attacker, by means of using curves with prime co-order in third degree extensions (see Section 4.1 for details) and Tables 4, 5.

	Bal-256
p	938d0fdad3293ea9bb3b94499dd782de c14af06877ab9723ec892d528e75892d
a	7b95bc541df473bedd7fe7c2b189a0e4 abfcc6977f96f5fded926f2edeed7052
b	40e8bb5c396a8dad529b6a77bd273edf 3d25a261d139f165b93b0449a3934fa4
n	938d0fdad3293ea9bb3b94499dd782de 6b98bbac80e14245c09aa2b1a15b01c1
m	938d0fdad3293ea9bb3b94499dd782df 16fd25246e75ec021877b7f37b90109b
G_x	563f2ba773e23e785c25db8decedd6ca 03bdfc1d4cb3ca746250e9c164a56949
G_y	7fd0be0dcbb2fea3b7b5a641def420da 1275eb557fa19e8e8194550855cc5ac2

Table 2: 256-bit Curve with prime order and prime co-order.

	Bal-384
p	d5b6623346184219ac0661fcf53b26e9db5a47b451b23c0e a313f93bf6a44bf74104c284d778ebfb1215ad4a8db25c3b
a	0a75ba92a8ea031b8b92af49adf2b5ec3226a0f781af35e1 1f2b240b70d3b3f7ab2131f46f31f6e96e784e8af6da1bc0
b	9ada13d6a669ab8593334cad4df33432d6c622bb9fca6ca6 10a861200a33266790e1ff3d5388c3a869374e81c001c1c9
n	d5b6623346184219ac0661fcf53b26e9db5a47b451b23c0f 19df5ffabb6cdeb2ea3213f253ce359081963b18664689fb
m	d5b6623346184219ac0661fcf53b26e9db5a47b451b23c0e 2c48927d31dbb93b97d771175b23a265a2951f7cb51e2e7d
G_x	9929955d6c0f383231ebb0eb2de9c43a4bdb7be091ce95e4 6e6055e286932e9558d517225f7b271f4c83b9ed8118dbe7
G_y	c80d69ab24a8719aac5168f81d61e832290771bfa08b06f7 1a9632076ef93c9bdf2e34471ad7458868976162a6a9aad9

Table 3: 384-bit Curve with prime order and prime co-order.

	Cof3-128
p	e21ab98d62a881e88dc7bdf90b4abcc5
a	c1ff3bbbb26d9b7a307c770875bdf27c
b	24a999a01a276dd1d57d07c088681cd6
n	710d5cc6b15440f3ae5e7ab03ffd32d7
G_x	5b74e44b8fa405811558efec904fffee
G_y	8e03410c0a93e368d40472ea157c164b
l	c7b33267d96736476a637ebe4ac3634b 29efbb5bbe1093b0a31e7530217581a5
cof	2

Table 4: 128-bit Curve with third cofactor of 1, and small cofactor.

A.2 Resonant Curves

Finally, we found pairs of resonant curves by sampling a curve of prime order p, letting q be the first prime after p^2, and finding a curve with prescribed order q, using Sage. See Tables 6, 7 and 8.

	Cof3–256
p	496cbacfcad9b6cfe64153fade2c20a5
	4d7c231a6d6f46240d78b1a7c029ca4b
a	36337ee3f7e460042d6f3dfef0a7a343
	caab70d642d0ee8cab5f50b4fc2ccf18
b	110b7dd9789e260ec3135829d00d9020
	f0fbb41c0dbb6f610a21b89b6943e5d4
n	0757ac47faaf5f14ca39bb9916379cdd
	3f63d196b6845929c8bebf8c0bc1154b
G_x	25ce8de35d8356d44b6965d479ac123b
	6e691c33057bbcd1ff5163fcfd985ff6
G_y	30c389ffe056d6cb7a660a73c870c74c
	c779dd9604aa9181dfee4c752bded74d
l	150f30b8a955ed62db339578c06eb1e8
	cf390e333ba06b8e114bdf864a435b45
	a08666b6e04ee315fec5eed74f548d63
	bf3f09b51407aa1fbbe55240afc6ba1b
cof	10

Table 5: 256-bit Curve with third cofactor of 1, and small cofactor.

	Res-A-192
p	8de7e43bbcbb4840e4001e299b8ee0210a02e380ec2b5ab3
a	1fbc10a3f9830eba5e55c35b42cf911953a25d0f186e17bf
b	54b6c1f99063e0fa8e57a32092fb85f69e0f9df9a60ac582
n	8de7e43bbcbb4840e4001e28482e8c0292c15bf31316af57
G_x	16ce7645808ad7532e73d1d015dd14b04e9c9793ba229d94
G_y	145382efe897f5e3c6f5afc6a2f9a0c603064c836f49206a
cof	1
	Res-A-384
p	4ea943777d2f5db2026e189a0c0d0689c61abaf022130628
	2dfa798f815917192f7d6f10955c861f7a7ca1cd84a87c75
a	2b5d423b408af905f1d6717a94002482a876b8b06e616e37
	d9817cca3040afe16133a056f3e2d9aa958503e4c3111711
b	42dcda91aa9bdf33f3e08d91492a5edfc8f6a62955942dfd
	c6e2d0bf72546876613032785ed8197c8c4ce3a3c03b4927
n	4ea943777d2f5db2026e189a0c0d0689c61abaf022130627
	1876b14077190a0221ff399216b543eefc531933850c10b5
G_x	37db3a8959b6c0f92a45469e4babea4d88bc7c26701321e7
	aad96be1ad86dcb4235ebb76a7c5e600388f1624d940bbc4
G_y	2fd10884e3d844ee490ce3be1bc505ce10bb85b5c0742a6f
	1f683a5af1a8cfaf5a69157b39ccd42498f60e47c687e388
cof	1

Table 6: Resonant curves with prime orders n_1, n_2 such that $n_2 = n_1^2 + 292$

	Res-B-256
p	cc68d8383be7d8320c9ef3ca6a426b6e 9fbb46bfeb139e254326acdd63914b53
a	4c60aa49f73e3462ff713b37963327b9 263362eba7b2c0866d7e5df73ee55445
b	2319880de290411e26a7905ddcf91dad c14b6254b8ce745c47e1e4cac44a8b74
n	cc68d8383be7d8320c9ef3ca6a426b6f 1e7fbd5ad8ce095852a0262f0df9aca7
G_x	76b4c653c91018f5d519f8a6421a0575 e0f38754e603eae42ca259e33ad3ae93
G_y	7793695dccaaaefc64f78b428981cbaa f84a646294a880829d9dc376640f33e9
cof	1

	Res-B-512
p	a337438a03c91e60ed0f50816178f75f63a15af90c 955c6af594336cd9e8d34e67d04f1eb32ea01d055b f88b34a04bd91d196cb31b0478c1a2716952502ce6a1
a	89c1fe6fbff146c0bf3a80f12cf8c3fc7cd88bfb7c 90f5ab5f882bef0949eb96d70aa18a34fddef57d0b 7b0b8ad97a3a7f614fa90b70a6bfbb9bdbf76c53383b
b	97ea7f8f285fe234f347e90af7ef554da86cc8cfcf e43c978a970e1faa5d1ff42922c445cd82d01bca35 9417bd0a8d6cf29edadb22e77ffe56fa314d4b4a2947
n	a337438a03c91e60ed0f50816178f75f63a15af90c 955c6af594336cd9e8d34f395e26d03c57553e5bd6 c14bb7d0972ff4f25892bc05dc4f0e72485c474ed839
G_x	285cd147e6e8aff4038f126a7c9416494644b9b087 cbd54873f8ff3830db3668abe35940a51f21791e5d 151528d7754ae79cb43075a66e88a4eeadf216fdb65
G_y	6c5ff600c98cb13925b2241256ae94d3a2fc037888 4de29b64a5b7fd4687f580f72f55aaae1520142558 078a27e1a9e6a1f71d4e6793252260fc76d5b1403e58
cof	1

Table 7: Resonant curves with prime orders n_1, n_2 such that $n_2 = n_1^2 + 840$

	Res-P256-512
p	fffffffe00000002fffffffe0000000079cdf55bd4
	6147ae13124dd75f81f2246b9711a0daa4a13613d0
	2332284aea3c657eba98d5f5de7fba043b090c59ba49
a	708b1a266c238e361455a6a7e5fdefdd4f7f96ee41
	31d82f125ce6e2747438c0287137236df03f4af7d6
	e12aa430e4089b11316823ce5f3afe7d4ef05fb30ed0
b	bcc974c62c09155527b64c78eacc7b365d4fb342a1
	a4bf6e2c897e782a5b60ee43af2829b9cdfa9e7df5
	80929bedfa18a094f73110eb1bcf40ebeb6b40913d43
n	fffffffe00000002fffffffe0000000079cdf55bd4
	6147ae13124dd75f81f2260043661f1d819d019a02
	fcd85d724aa132ad5e5de469c27bab0dbaa15a16889f
G_x	2be1e3b4a3d12112fee2e659f761ed5b8a10dc74d8
	672c139e82ca553ee6b130a921462a312e0c142590
	78b1dc97db6d827518023319aa08d76fbfd451c660af
G_y	74328767592b58e787d2083be0eee3e201d0acad5f
	b17130379040c7f6d3aac9161a001888af312035fa
	953aec1487c408ca1bf22bcfd38075d41fd668cf038
cof	1

Table 8: Let n_{P256} be the order of NIST curve P-256. Res-P256-512 is a curve resonant to P-256 with 512-bit order n such that $n = n_{P256}^2 + 1278$.