SIGNITC: Supersingular Isogeny Graph
Non-Interactive Timed Commitments

Knud Ahrens
University of Passau, Germany
knud.ahrens@uni-passau.de

Abstract
Non-Interactive Timed Commitment schemes (NITC) allow to open any commitment after a specified delay \( t_{fd} \). This is useful for sealed bid auctions and as primitive for more complex protocols. We present the first NITC without repeated squaring or theoretical black box algorithms like NIZK proofs or one-way functions. It has fast verification, almost arbitrary delay and satisfies IND-CCA hiding and perfect binding. Additionally, it needs no trusted setup. Our protocol is based on isogenies between supersingular elliptic curves making it presumably quantum secure, and all algorithms have been implemented as part of SQISign or other well-known isogeny-based cryptosystems.

Keywords: Non-interactive timed commitments, post-quantum, isogeny walks, Deuring correspondence.

1 Introduction

The concept of time-lock puzzles [21] has been around for more than twenty years, but timed commitments [5] are rather new and we will use the definition of Non-Interactive Timed Commitment schemes (NITC) by Katz, Loss, and Xu [19] from the year 2020. These protocols satisfy binding or non-malleability properties and efficient verification just like usual commitment schemes, but a commitment can be opened by anyone after some delay \( t_{fd} \). So hiding only lasts for this time \( t_{fd} \) and there are additional algorithms: one to verify that a commitment can be opened by others and another one to open the commitment forcefully in time at least \( t_{fd} \). A possible application is a sealed bid auction, where all bids can be revealed after time \( t_{fd} \) even if some of the bidders refuse to open their commitment. Other applications are listed in Katz et al. [19].

Our approach uses random walks in the isogeny graph of supersingular elliptic curves to construct a NITC, hence the name Supersingular Isogeny Graph Non-Interactive Timed Commitments or SIGNITC\(^1\) for short. The main idea is that computing isogenies of large or non-smooth degree is slow, but if we know the endomorphism ring of the starting curve, we can find a smooth shortcut. So we use a secret endomorphism ring for fast commitment and verification, but the forced decommitment has to compute the long isogeny and thus it needs time at least \( t_{fd} \).

\(^1\) pronounced like “signets”
The advantage of isogeny-based cryptography is that it is presumably quantum secure and relatively slow compared to other fields of post-quantum cryptography. Since we need a delay, this is a good thing. The field has undergone thorough scrutiny due to the candidates SIKE [18] and SQISign [9] in NIST competitions for post-quantum protocols and it is well studied by now. The protocol only uses (known) isogeny-based cryptography, so we do not need to know several fields and this facilitates correct and secure implementations. This also means that we have no theoretical black box algorithms like zero knowledge proofs, succinct non-interactive arguments of knowledge or one-way functions. In addition all needed calculations have already been implemented as sub-routines in other cryptosystems. To our knowledge this is the first quantum secure NITC scheme with explicit algorithms. The only drawbacks are that some algorithms are still quite involved and that we need to differ slightly from the original definition for hiding.

Related Work Thyagarajan et al. [23] present an approach based on class groups using non-interactive zero knowledge (NIZK) proofs. Katz et al. [19] and Chvojka and Jager [10] use protocols based on repeated squaring in a group of unknown order and NIZK proofs. Finally Ambrona et al. [2] avoid NIZK proofs but still use repeated squaring. None of these is quantum secure.

NITC schemes are related to verifiable delay functions (VDF) [6] in the sense that both have fast verification and a function that needs a long time to evaluate. The main difference is the handling of secrets. For VDFs finding the correct response for a given challenge has to be slow for everyone. For NITC schemes however someone has to compute the commitment and therefore already knows the output of the slow task, namely finding the message to a given commitment. So we can construct NITC schemes from VDFs, but the contrary is difficult or impossible, depending on the protocol.

VDFs have direct applications to blockchains and there already are several approaches. Many are based on repeated squaring for the delay. A new publication [4] suggests that this might not be sequential. So contrary to current belief, repeated squaring could be parallelizable, disqualifying it as a delay function. Additionally this is not quantum secure. There are even some isogeny-based candidates for VDFs, but they all still have some flaws. The pairing-based approach [11] is not quantum secure. Chavez-Saab et al. [8] use SNARGS and their verification time increases for larger delays. Finally there is one base on Kani’s criterion for abelian surfaces [13], but the authors state that it is not clear how to implement it. A different approach based on endomorphism rings [1] has the problem that the generation of a challenge also gives (a significant advantage in finding) the response. So it is closer to a NITC scheme and gave the initial idea for this article.

Structure of this Article The remainder of this paper is structured as follows. First we give a definition of NITC schemes and discuss their properties. Next we recall the necessary definitions and fix the notations of isogeny-based cryptography. Readers familiar with one of these topics can briefly skim through the respective sections as we aimed to use standard notations. The sole difference is a slight variation in Definition 3.5 of a IND-CCA security game. In Section 4 we present our protocol in full detail. Its security and its properties
are discussed in Section 5. Finally we give a short conclusion and outlook.

2 Non-Interactive Timed Commitments

In this section we recall NITC schemes and their properties. In their paper Katz et al. [19] gave the first formal definition of this concept.

Definition 2.1 (NITC [19]). A \((t_{\text{com}}, t_{\text{cv}}, t_{\text{dv}}, t_{\text{fd}})\)-non-interactive timed commitment scheme (NITC) is a tuple \(\mathcal{T}_C = (\text{PGen}, \text{Com}, \text{ComVrfy}, \text{DecVrfy}, \text{FDecom})\) of five algorithms with the following behaviour:

- The randomized parameter generation algorithm \(\text{PGen}\) takes as input the security parameter \(1^\kappa\) and outputs a common reference string \(\text{crs}\).
- The randomized commit algorithm \(\text{Com}\) takes as input a string \(\text{crs}\) and a message \(m\). It outputs a commitment \(C\) and proofs \(\pi_{\text{com}}, \pi_{\text{dec}}\) in time at most \(t_{\text{com}}\).
- The deterministic commitment verification algorithm \(\text{ComVrfy}\) takes as input a string \(\text{crs}\), a commitment \(C\) and a proof \(\pi_{\text{com}}\). It outputs accept (if \(C\) could be forcefully decommitted) or reject in time at most \(t_{\text{cv}}\).
- The deterministic decommitment verification algorithm \(\text{DecVrfy}\) takes as input a string \(\text{crs}\), a commitment \(C\), a message \(m\) and a proof \(\pi_{\text{dec}}\). It outputs accept or reject in time at most \(t_{\text{dv}}\).
- The deterministic forced decommitment algorithm \(\text{FDecom}\) takes as input a string \(\text{crs}\) and a commitment \(C\). It outputs a message \(m\) or invalid in time at least \(t_{\text{fd}}\).

We require that for all \(\kappa\), all \(\text{crs}\) output by \(\text{PGen}(1^\kappa)\), all \(m\) and all \(C, \pi_{\text{com}}, \pi_{\text{dec}}\) output by \(\text{Com}(\text{crs}, m)\), it holds that

\[
\text{ComVrfy}(\text{crs}, C, \pi_{\text{com}}) = \text{accept} = \text{DecVrfy}(\text{crs}, C, m, \pi_{\text{dec}})
\]

and \(\text{FDecom}(\text{crs}, C) = m\).

To be relevant for applications a NITC also needs to satisfy three further properties. First we give a proper definition of practicality and then recall definitions for hiding and binding in our notation.

Definition 2.2 (Practicality). A NITC scheme is practical, if verification is much faster than forcefully opening the commitment, so \(t_{\text{cv}}, t_{\text{dv}} \ll t_{\text{fd}}\).

We present two IND-CCA security games and define hiding in terms of the probability that an adversary \(A\) wins the games. In both cases the adversary has access to an oracle for \(\text{FDecom}\) and a query is considered to have only a small computational cost. The first game is the one used by Katz et al. [19].

Definition 2.3 (IND-CCA original [19]). For a NITC scheme \(\mathcal{T}_C\) and an algorithm \(A\), define the game IND-CCA\(^A_{\mathcal{T}_C}\) as follows:

1. Compute \(\text{crs} \leftarrow \text{PGen}(1^\kappa)\).
2. Run \(A(\text{crs})\) in a pre-processing phase with access to \(\text{FDecom}(\text{crs}, \cdot)\).
3. When $A$ outputs $(m_0, m_1)$, choose a uniform bit $b \leftarrow \{0, 1\}$ and then compute $(C_b, \pi_{\text{com}}, \pi_{\text{dec}}) \leftarrow \text{Com}^{\text{crs}}(m_b)$. Give $(C_b, \pi_{\text{com}})$ to $A$, who continues to have access to $\text{FDecom}^{\text{crs}}(\cdot)$ except that it may not query the oracle on the given commitment $C_b$.

4. When $A$ outputs a bit $b'$, it wins iff $b' = b$.

The commitment $C$ in our approach is a tuple $C = (E_s, K_T, u)$ and not a single value. Because of that we can only satisfy a slightly weaker variation of the IND-CCA security game. The new Definition 3.5 is given in Section 3.2 and is discussed in more detail in Section 5.1. Hiding is defined with respect to an IND-CCA game. This allows us to evaluate the security of our NITC in terms of both the original and our adapted definition. Broadly speaking hiding guarantees that it is impossible to infer information about the message from the commitment. In our case hiding should hold at least for the time $t_o$ it takes to open a commitment by force, so for all $t_o < t_{fd}$ in the following definition.

**Definition 2.4** (Hiding [19]). A NITC scheme $\text{TC}$ is $(t_p, t_o, \epsilon)$-CCA-secure if for all adversaries $A$ running in time at most $t_p$ in the pre-processing phase and time at most $t_o$ in the subsequent online phase,

\[ \Pr[A \text{ wins IND-CCA}_{\text{TC}}] \leq \frac{1}{2} + \epsilon. \]

Similar to hiding, binding is defined in terms of the probability that $A$ wins a BND-CCA security game. This time we do not need to adapt this for our approach.

**Definition 2.5** (BND-CCA [19]). For a NITC scheme $\text{TC}$ and an algorithm $A$, define the game $\text{BND-CCA}_{\text{TC}}$ as follows:

1. Compute $\text{crs} \leftarrow \text{PGen}(1^\kappa)$.
2. Run $A(\text{crs})$ with access to $\text{FDecom}(\text{crs}, \cdot)$.
3. $A$ outputs $(m, C, \pi_{\text{com}}, \pi_{\text{dec}}, m', \pi_{\text{dec}}')$ and wins iff $\text{ComVrfy}(\text{crs}, C, \pi_{\text{com}}) = \text{accept}$ and either:
   - $m \neq m'$, yet $\text{DecVrfy}(\text{crs}, C, m, \pi_{\text{dec}})$ and $\text{DecVrfy}(\text{crs}, C, m', \pi_{\text{dec}}')$ both output $\text{accept}$; or
   - $\text{DecVrfy}(\text{crs}, C, m, \pi_{\text{dec}}) = \text{accept}$ but $\text{FDecom}(\text{crs}, C) \neq m$.

Binding makes sure that a commitment cannot be opened to two different messages and that $\text{FDecom}$ gives the correct messages for valid commitments.

**Definition 2.6** (Binding [19]). A NITC scheme $\text{TC}$ is $(t, \epsilon)$-BND-CCA-secure if for all adversaries $A$ running in time $t$,

\[ \Pr[A \text{ wins BND-CCA}_{\text{TC}}] \leq \epsilon. \]

### 3 Isogeny-based Cryptography

In this section we provide the necessary basics for isogeny-based cryptography, quaternion algebras and the Deuring correspondence. We also discuss some computational problems in this area.
3.1 Elliptic Curves and the Quaternion Algebra

Elliptic curves have ties to different fields resulting in several equivalent definitions. We will mostly follow the notation of Silverman [22], but restrict ourselves to aspects relevant for this paper.

Definition 3.1 (Elliptic Curve). An elliptic curve is a pair \((E, \infty)\), where \(E\) is a curve of genus one and \(\infty \in E\). It is defined over a field \(K\), if it is defined over \(K\) as a curve and \(\infty \in E(K)\).

We can define an addition of points on the curve making \((E, +)\) an additive group where \(\infty\) is the neutral element. This permits scalar multiplication written as \([m]: E \to E\) and torsion subgroups \(E[m] := \{P \in E \mid [m]P = \infty\}\).

Definition 3.2 (Isogeny). Let \(E\) and \(E'\) be elliptic curves. Then a morphism \(\varphi: E \to E'\) such that \(\varphi(\infty) = \infty\) is called an isogeny. If a non-zero isogeny \(\varphi: E \to E'\) exists, then \(E\) and \(E'\) are called isogenous.

In fact, every isogeny between two curves is also a group homomorphism. The isogenies from a curve \(E\) into itself form the endomorphism ring \(\text{End} E\). Isogenies can be written as rational maps and their degree is defined by this map. Thus, the degree \(\deg(\varphi \circ \varphi') = \deg \varphi \deg \varphi'\) is multiplicative. In addition each isogeny \(\varphi: E \to E'\) has a unique dual isogeny \(\check{\varphi}: E' \to E\) such that the composition \(\check{\varphi} \circ \varphi = [\deg \varphi]\) is the multiplication by the degree. The isogenies of degree 1 are the isomorphisms, and each isomorphism class can be labelled by the so-called \(j\)-invariant. This allows to construct the \(\ell\)-isogeny graph that has those \(j\)-invariants as vertices and isogenies of degree \(\ell\) as edges.

Definition 3.3 (Supersingularity). Let \(K\) be a field of characteristic \(p > 0\) and \(E\) an elliptic curve defined over \(K\). The curve \(E\) is supersingular if the torsion group \(E[p]\) is trivial. Equivalently, this means that the endomorphism ring \(\text{End} E\) is an order in a quaternion algebra.

For the rest of this paper \(p > 3\) will be a large prime. This allows us to write every elliptic curve in short Weierstraß form as \(E: y^2 = x^3 + Ax + B\) with \(j(E) = 108(4A)^3/(4A^3 + 27B^2)\). For supersingular curves there is always a representation with \(A, B, j \in \mathbb{F}_p^2\). There are only \([p/12] + \varepsilon\) supersingular elliptic curves for fields with characteristic \(p\) where \(\varepsilon \in \{0, 1, 2\}\). Hence, the subset \(J_{SS} \subset \mathbb{F}_p^2\) of supersingular \(j\)-invariants has cardinality at least \([p/12]\).

We have already seen in Definition 3.3 that supersingular curves are related to quaternion algebras. We are interested in the quaternion algebra \(B_{p, \infty}\) ramified at \(p\) and infinity with \(\mathbb{Q}\)-basis \(\{1, i, j, k\}\) such that

\[
i^2 = -1, \quad j^2 = -p, \quad k = ij = -ji.
\]

The (reduced) norm of an element \(\alpha = a_1 + a_2i + a_3j + a_4k \in B_{p, \infty}\) is given by \(\alpha \bar{\alpha}\) for \(\bar{\alpha} = a_1 - a_2i - a_3j - a_4k\). An order in \(B_{p, \infty}\) is a lattice that is also a subring, and it is maximal if its discriminant equals \(p\). Now an elliptic curve \(E\) is supersingular if and only if \(\text{End} E\) is isomorphic to a maximal order \(\mathcal{O}\) in \(B_{p, \infty}\), i.e. \(\mathbb{Q} \otimes \text{End} E \cong B_{p, \infty}\).

Theorem 3.4 (Deuring Correspondence [14]). The isomorphism classes of supersingular elliptic curves correspond to the isomorphism classes of invertible left \(\mathcal{O}\)-ideals in the quaternion algebra.
This so-called Deuring correspondence also gives us that an \( \ell \)-isogeny \( \varphi \) starting at \( E \) corresponds to a left ideal \( I_\varphi \) of norm \( \ell \) in \( \mathcal{O} \cong \text{End} \, E \) and the image curve has an endomorphism ring isomorphic to the right order \( \mathcal{O}_R(I_\varphi) = \{ \alpha \in B_{p,\infty} \mid I_\varphi \alpha \subseteq I_\varphi \} \) of \( I_\varphi \), see [24, Ch. 42] for more details.

3.2 Application to Cryptography

Many isogeny-based protocols rely on secret walks in isogeny graphs of supersingular elliptic curves. The fact that the endomorphism ring is non-commutative gives rise to presumably quantum secure protocols and the graphs have fast mixing properties, meaning that we reach an almost uniform distribution on the graph after a short random walk [16].

Taking \( n \) steps in the \( \ell \)-isogeny graph corresponds (up to isomorphism) to an isogeny \( \varphi : E \to E' \) of degree \( d = \ell^n \). For our purposes the degree of such isogenies will always be coprime to the characteristic \( p \) of the field and the isogeny \( \varphi \) is determined by a point \( K \) of order \( d \) on the staring curve \( E \). This point generates the kernel of \( \varphi \) and we write \( E' \cong E/\langle K \rangle \). In this case the \( d \)-torsion group \( E[d] \) has \( d^2 \) elements and can be generated by two integers \( a, b \) such that \( K = aP + bQ \). Note that although every supersingular elliptic curve has a representation in \( \mathbb{F}_{p^2} \), the kernel of an isogeny and hence its generators might be elements of extensions \( \mathbb{F}_{p^2} \). With this notation we can define the adapted security game mentioned in Section 2.

Definition 3.5 (IND-CCA adapted). For a NITC scheme \( TC \) and an algorithm \( A \), define the game IND-CCA\( _A^{TC} \) as follows:

1. Compute \( \text{crs} \leftarrow \text{PGen}(1^\kappa) \).
2. Run \( A(\text{crs}) \) in a pre-processing phase with access to \( \text{FDecom}(\text{crs}, \cdot) \).
3. When \( A \) outputs \( (m_0, m_1) \), choose a uniform bit \( b \leftarrow \{0, 1\} \) and then compute \( (C_b, \pi_{\text{com}}, \pi_{\text{dec}}) \leftarrow \text{Com}(\text{crs}, m_b) \). Give \( (C_b, \pi_{\text{com}}) \) to \( A \), who continues to have access to \( \text{FDecom}(\text{crs}, \cdot) \) except that it may not query the oracle on \( (E', K', \cdot) \) for \( E'/\langle K' \rangle \cong E_s/\langle K_T \rangle \) and \( C_b = (E_s, K_T, u_b) \).
4. When \( A \) outputs a bit \( b' \), it wins iff \( b' = b \).

Now we list some computational tasks that are relevant for isogeny-based cryptosystems. First we present tasks that can be solved efficiently and have a polynomial or even constant complexity.

**Task 1:** Compute isogenies of small or smooth degree.

**Task 2:** Given two elliptic curves \( E, E' \), an isogeny \( \varphi : E \to E' \) as well as the corresponding order \( \mathcal{O} \cong \text{End} \, E \) and ideal \( I_\varphi \), compute \( \mathcal{O}' \cong \text{End} \, E' \).

**Task 3:** Given two elliptic curves \( E, E' \), and the corresponding orders \( \mathcal{O} \cong \text{End} \, E, \mathcal{O}' \cong \text{End} \, E' \), compute a connecting ideal \( I \) corresponding to an isogeny \( \varphi_I : E \to E' \).

**Task 4:** Given a left ideal \( I \) of a maximal order \( \mathcal{O} \subset B_{p,\infty} \), find an equivalent ideal such that its norm is small or a prime power.
Task 5: Given $\mathcal{O} \cong \text{End } E$, translate between isogenies $\varphi: E \to E'$ and their corresponding left $\mathcal{O}$-ideals $I_{\varphi}$.

Task 6: Compute isogenies of large or non-smooth degree.

Task 1 can be solved using Vélu’s formulae [25]. For Task 2 we can compute $\mathcal{O}'$ as $\mathcal{O}_R(I_{\varphi})$ and the connecting ideal $I$ in Task 3 satisfies $\mathcal{O} = \mathcal{O}_L(I)$, where the left order $\mathcal{O}_L(I)$ is defined analogously to the right order $\mathcal{O}_R(I) = \mathcal{O}'$. Task 4 is solved by the KLPT algorithm [20] and Task 5 is a sub-routine of SQISign [12]. Depending on the degree we can use Vélu’s formulae or the $\sqrt{\text{velu}}$ algorithm [3] to solve Task 6.

To create a delay we need moderately hard problems, which are still polynomial in complexity but might take a considerable time to compute. In Section 5.2 we show that Task 6 can be made sufficiently slow. The following hard problems have an exponential complexity and are equivalent [26]. They are the basis for encryption or signature schemes like CSIDH [7] or SQISign [9]. In our case they ensure that there are no shortcuts for the forced decommitment.

Problem 3.6 (Isogeny Path Problem). Given two (isogenous) supersingular elliptic curves $E, E'$ and a prime $\ell$, find a path from $E$ to $E'$ in the $\ell$-isogeny graph.

Problem 3.7 (Endomorphism Ring Problem). Given a supersingular elliptic curve $E$, find four endomorphisms that generate $\text{End } E$ as a lattice.

Problem 3.8 (Maximal Order Problem). Given a supersingular elliptic curve $E$, find four quaternions in $B_{p,\infty}$ that generate a maximal order $\mathcal{O}$ such that $\mathcal{O} \cong \text{End } E$.

Remark 3.9. Knowledge of endomorphism rings can break the hard problems. If we know both endomorphism rings the first hard problem becomes polynomial using Tasks 3-5. If we know an isogeny from a curve with known endomorphism ring to our curve also the second hard problem becomes polynomial by Tasks 2 & 5. The third hard problem reduces to the second via Task 5.

Finding supersingular elliptic curves can be done in two ways. We can reduce an elliptic curve in characteristic 0 modulo a prime and check if the resulting curve is supersingular, or take a random isogeny starting at one of these curves. In both cases the endomorphism ring of the final curve can be computed either via reduction or by transport along the isogeny. But as discussed in Remark 3.9 this weakens the hard problems. Hence many cryptosystems require curves with unknown endomorphism ring. This in turn forces them to use a multi-party computation or a trusted authority in their setup to ensure that no single participant knows a complete path from a curve with known endomorphism ring to the one used.

4 The Protocol

Now we can combine the previous two sections and present our construction. First we give a high-level overview and discuss some challenges. Then we look at the algorithms and choices for the parameters.
4.1 Overview

At the heart of our protocol is an isogeny $\varphi_T$ of large degree $d_T$. Its domain is a public supersingular elliptic curve $E_s$ with secret $O_s \cong \text{End} E_s$ and its kernel is generated by a publicly known point $K_T$ on $E_s$. We use the $j$-invariant $j_T$ of the codomain $E_T$ of $\varphi_T$ to hide the message $m \in M$. Therefore an adversary needs to compute $E_T$ (or rather $j_T = j(E_T)$) in order to break hiding or to open the commitment by force. We can choose how long the commitment should be kept secret by setting the degree $d_T$ accordingly. This gives us hiding. Since $E_s$ and $K_T$ are part of the commitment, the codomain $E_T \cong E_s/(K_T)$ is fixed (up to isomorphism) and we have perfect binding.

For verification to be faster than forced opening, we need a more efficient way to compute $j_T$. This is where the first isogeny $\varphi_s$: $E_0 \to E_s$ comes into play. The starting curve $E_0$ has a known endomorphism ring, which allows us to compute the endomorphism ring of $E_s$ if we know $\varphi_s$. For the commitment and the verification we use $O_s \cong \text{End} E_s$ to find an equivalent isogeny $\varphi_T$: $E_s \to E_T$ of much smaller degree (Tasks 4 & 5 from Section 3.2). During the commitment we compute $O_s$ and give $\varphi_s$ to the verifier as part of the decommitment proof. An adversary only knows $E_s$, but not $\varphi_s$ and hence can neither compute $O_s \cong \text{End} E_s$ nor a shortcut $\varphi_T$. This gives us the preferred difference in speed for verification and forced opening. This is visualized in Figure 1.

$$
\begin{array}{cccc}
E_0 & \xrightarrow{\varphi_s} & E_s \cong E_0/(K_s) & \xrightarrow{\varphi_T} & E_T \cong E_s/(K_T)
\end{array}
$$

Figure 1: Walk in the isogeny graph with $\deg(\varphi_s) = d_s \ll d_T = \deg(\varphi_T)$ and $\deg(\varphi_T) \ll d_T$ for the equivalent isogeny $\varphi_T$.

To efficiently verify the validity of a commitment, we need to map the $j$-invariant $j_T$ into the group of messages $M$. This map has to satisfy the following property. Otherwise the commitment might leak information about $j_T$.

**Definition 4.1 (Inverse Resistant Functions).** A function $f: X \to Y$ is $\lambda$-inverse resistant, if for uniform $x \in X$ the probability $\Pr[A(f(x)) = x]$ is at most $2^{-\lambda}$ for all algorithms $A$.

This definition is weaker than one-way functions, since finding an element in the preimage is allowed as long as the probability to find the correct one is sufficiently small. It also differs from hash functions, which are mostly considered to be collision resistant. A simple projection with a sufficiently large preimage set satisfies this definition but is neither a one-way function nor a proper hash function.

4.2 Algorithms

As seen in Definition 2.1 we have five algorithms $\text{PGen}$, $\text{Com}$, $\text{ComVrfy}$, $\text{DecVrfy}$ and $\text{FDecom}$. In this subsection we give pseudocode for each algorithm and discuss their (relative) speed and some sub-routines.
**Parameter Generation** The parameter generation \( \text{PGen} \) defines the security of the whole protocol and fixes the delay \( T \). It sets all general parameters like the characteristic \( p \) of the finite fields, the degrees \( d_s \) and \( d_T \), as well as the message group \( (M, \oplus) \) and the inverse resistant function \( F: J_{SS} \rightarrow M \). It also provides the starting curve \( E_0 \). \( O_0 \cong \text{End} E_0 \) and generators of the \( d_s \) and \( d_T \) torsion groups to improve the speed of the commitment. Its output is the common reference string \( \text{crs} \). A detailed description can be found in Algorithm 1. The speed is dominated by finding generators of \( E_0[dt] \), since we have to check the order and linear independence of two random points in \( E_0(\mathbb{F}_{p^{2s}}) \).

**Algorithm 1 Parameter generation algorithm \( \text{PGen} \)**

<table>
<thead>
<tr>
<th>Require: Security parameter ( 1^\kappa )</th>
<th>Ensure: ( \text{crs} = ((p, E_0, O_0, \eta_0, \theta_0, M, F, d_s, P_0, Q_0), (d_T, P_s', Q_s', e)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Choose prime ( p ) of right size</td>
<td>8: ( \text{crs}_0 = (p, E_0, O_0, \eta_0, \theta_0, M, F, d_s, P_0, Q_0) \quad \triangleright \text{Depends only on ( \kappa )}</td>
</tr>
<tr>
<td>2: Choose supersingular elliptic curve ( E_0 ) with known ( O_0 \cong \text{End} E_0 )</td>
<td>9: Choose ( c, d_T \in \mathbb{N} ) such that ( d_T ) is coprime to ( d_s ) and ( E_0[dt] \subseteq E_0(\mathbb{F}_{p^{2s}}) )</td>
</tr>
<tr>
<td>3: Choose ( \eta_0, \theta_0 \in O_0 ) corresponding to orthogonal endomorphisms in ( \text{End} E_0 )</td>
<td>10: Find ( P_s', Q_s' \in E_0(\mathbb{F}_{p^{2s}}) ) such that ( (P_s', Q_s') = E_0[dt] )</td>
</tr>
<tr>
<td>4: Choose a group ( (M, \oplus) ) with efficient membership testing as message space</td>
<td>11: ( \text{crs}_T = (d_T, P_s', Q_s', e) \quad \triangleright \text{Fixes delay} T )</td>
</tr>
<tr>
<td>5: Choose an efficient, inverse resistant function ( F: J_{SS} \rightarrow M )</td>
<td>12: \textbf{return} ( (\text{crs}_0, \text{crs}_T) )</td>
</tr>
</tbody>
</table>

**Commitment** The commitment algorithm \( \text{Com} \) takes as input a message \( m \in M \) and outputs a tuple \( \text{(C, } \pi_{\text{com}}, \pi_{\text{dec}} \text{)} \). First it chooses a random isogeny \( \varphi_s: E_0 \rightarrow E_s \), of degree \( d_s \) and computes the secret order \( O_s \cong \text{End} E_s \). Then it chooses a second random isogeny \( \varphi_T: E_s \rightarrow E_T \) of large degree \( d_T \). It uses \( O_s \) to find an equivalent isogeny \( \tilde{\varphi}_T: E_s \rightarrow E_T \) of smooth and much smaller degree \( d_T \). If \( d_T \) is still too big it tries to find an other equivalent isogeny or chooses different random isogenies \( \varphi_s \) and \( \varphi_T \) before proceeding. This allows it to efficiently compute the \( j \)-invariant \( j_T = j(E_T) \) and \( u = m \circ F(j_T) \in M \). The commitment itself \( \text{C} = (E_s, K_T, u) \) is again a tuple of a supersingular elliptic curve \( E_s \), a point \( K_T \) on \( E_s \) that generates the kernel of \( \varphi_T \) and \( u \in M \). While the commitment proof \( \pi_{\text{com}} \) is empty, the decommitment proof \( \pi_{\text{dec}} \) allows to reconstruct the secret isogeny \( \varphi_s \). The individual steps are given in Algorithm 2. In SQISign [9] the authors state that converting between isogenies and ideals is the bottleneck of their computation. Therefore we assume that the slowest part of this algorithm is computing \( \tilde{\varphi}_T \), since it also contains operations in \( \mathbb{F}_{p^{2s}} \). Note however, that SQISign is still fast and our commitment algorithm will be faster than computing the long isogeny \( \varphi_T \) (over \( \mathbb{F}_{p^{2s}} \)) directly. A more detailed discussion can be found in Section 5.3.

**Commitment Verification** Algorithm 3 shows the commitment verification \( \text{ComVrfy} \). It is fast since it only needs to check if the three parts of the commitment are of the correct form. Namely, \( E_s \) is an elliptic curve, \( K_T \) is a point
Algorithm 2 Commitment algorithm Com

Require: Common reference string crs, message \( m \in M \)
Ensure: \((C, \pi_{\text{com}}, \pi_{\text{dec}}) = ((E_s, K_T, u), (\), s)\)
1: Choose random \( s \in [0, d_s) \) and compute \( K_s = P_0 + sQ_0 \in E_0[d_s] \)
2: Compute \( E_s \cong E_0/(K_s) \) via Vélu’s formulae
3: Compute ideal \( I_s \) corresponding to isogeny \( \varphi_s : E_0 \to E_s \) with kernel \( \langle K_s \rangle \)
   \( \triangleright \) Here we use the orthogonal elements \( \eta_0, \theta_0 \in O_0 \)
4: Compute \( O_s \cong \text{End } E_s \) as right order of ideal \( I_s \)
5: Choose \( \eta_s, \theta_s \in O_s \) corresponding to orthogonal endomorphisms in \( \text{End } E_s \)
6: Choose random \( t \in [0, d_T) \) and compute \( K_T = \varphi_s(P_s') + t\varphi_s(Q_s') \in E_s[d_T] \)
7: Compute ideal \( I_T \) corresponding to \( \varphi_T : E_s \to E_T \) with kernel \( \langle K_T \rangle \)
   \( \triangleright \) Here we use the orthogonal elements \( \eta_s, \theta_s \in O_s \)
8: Compute equivalent ideal \( I_T \) with smooth norm \( d_T \ll d_T \)
9: Compute corresponding isogeny \( \tilde{\varphi}_T \) of degree \( d_T \)
10: Compute \( E_T \cong E_s/\langle K_T \rangle \) as codomain of \( \tilde{\varphi}_T \)
11: Compute \( j_T = j(E_T) \) and \( u = m \odot F(j_T) \in M \)
12: \( C = (E_s, K_T, u) \) \( \triangleright \) Commitment
13: \( \pi_{\text{com}} = () \) \( \triangleright \) Commitment proof (empty)
14: \( \pi_{\text{dec}} = s \) \( \triangleright \) Decommitment proof
15: return \((C, \pi_{\text{com}}, \pi_{\text{dec}})\)

Algorithm 3 Commitment verification algorithm ComVrfy

Require: Common reference string crs, commitment \( C \) and proof \( \pi_{\text{com}} \)
1: Check if \( E_s \) is an elliptic curve over \( \mathbb{F}_{p^2} \), \( K_T \in E_s \) and \( u \in M \)
   Optional: check if \( K_T \in \mathbb{F}_{p^2}^2 \) \( \triangleright \) Check upper bound for degree of \( \varphi_T \)
2: return (accept/reject)

Decommitment Verification The decommitment verification DecVrfy (Algorithm 4) is similar to the commitment algorithm. It first reconstructs \( \varphi_s \) from \( \pi_{\text{dec}} \) and verifies \( \varphi_s : E_0 \to E_s \). Then it computes \( O_s \cong \text{End } E_s \) and uses it to find a faster isogeny \( \tilde{\varphi}_T : E_s \to E_T \) equivalent to \( \varphi_T : E_s \to E_T \) where the kernel of \( \varphi_T \) is generated by \( K_T \). With this short isogeny it computes \( j_T = j(E_T) \) and checks if \( u \oplus F(j_T) = m \). As stated above, we assume the slowest part of this algorithm to be the computation of \( \tilde{\varphi}_T \). Again, this is still faster than forced decommitment, as \( \deg \tilde{\varphi}_T \) is smooth and smaller than \( \deg \varphi_T \) (cf. Section 5.3).
Decommitment verification algorithm $\text{ComVrfy}$

**Require:** Common reference string $\text{crs}$, commitment $C$, message $m$, decommitment proof $\pi_{\text{dec}}$

1. Compute $K_s = P_0 + sQ_0 \in E_0[d_s]$ and check $E_s \cong E_0/(K_s)$
   ▶ We use the orthogonal elements $\eta_0, \theta_0 \in \mathcal{O}_0$
2. Compute ideal $I_s$ corresponding to isogeny $\varphi_s: E_0 \to E_s$ with kernel $\langle K_s \rangle$
   ▶ Here we use the orthogonal elements $\eta_s, \theta_s \in \mathcal{O}_s$
3. Compute $O_s \cong \text{End}_{E_s}$ as right order of ideal $I_s$
4. Compute orthogonal endomorphisms in $\text{End}_{E_s}$
5. Compute ideal $I_T$ corresponding to $\varphi_T: E_s \to E_T$ with kernel $\langle K_T \rangle$
   ▶ Here we use the orthogonal elements $\eta_s, \theta_s \in \mathcal{O}_s$
6. Compute equivalent ideal $\tilde{I}_T$ with smooth norm $d_T \ll d_T$
7. Compute corresponding isogeny $\varphi_T$ of degree $d_T$
8. Compute $E_T \cong E_s/(K_T)$ as codomain of $\varphi_T$
9. Compute $j_T = j(E_T)$ and check $u \oplus F(j_T) = m$
10. Return (accept/reject)

**Forced Decommitment** In terms of the number of tasks the forced decommitment algorithm is rather simple. It just computes $E_T$ as codomain of the isogeny $\varphi_T$ given by the point $K_T$ that generates its kernel. From there it recovers the message $m = u \oplus F(j(E_T))$. Computing an isogeny $\varphi_T$ of large degree $d_T$ is slow (cf. Theorem 5.12), especially when the calculations have to be done in a field extension $\mathbb{F}_{p^{2e}}$. This allows us to make Algorithm 5 (almost) arbitrarily slow.

**Algorithm 5 Forced decommitment algorithm $\text{FDecom}$**

**Require:** Common reference string $\text{crs}$, commitment $C$

**Ensure:** Message $m$

1. Compute $E_T \cong E_s/(K_T)$ via Vélu’s formulae or $\sqrt{\text{velu}}$ algorithm
2. Compute $j_T = j(E_T)$ and $m = u \oplus F(j_T)$
3. Return $m$

**4.3 Parameter Sizes and other Choices**

The algorithms above do not specify all properties of the parameters. Therefore we now discuss the necessary and some optional choices. For example, the hiding property sets requirements on the size of some parameters and we also propose some choices for implementing this protocol.

The delay $t_{\text{id}}$ should be large, but it has to be polynomial in $\kappa$ (or $\log p$). On one hand the main idea of NITC schemes is that we can forcefully open a commitment (in polynomial time) with $\text{FDecom}$, if someone refuses to open it themselves. On the other hand generic algorithms to solve Problems 3.6 - 3.8 could be faster than $\text{FDecom}$ and therefore violate hiding, if $t_{\text{id}}$ were superpolynomial. In particular we need $t_{\text{id}} < \min\{d_s^{1/4}, p^{1/4}\}$ and $t_{\text{id}} \gg t_{\text{cv}}, t_{\text{dv}}$.

**Prime $p$, starting curve $E_0$ and isogenies $\varphi_s$ and $\varphi_T$** In order to satisfy the hiding property, $p$ and $d_s$ have to have a certain size. It has to be infeasible to pre-compute $\mathcal{O}_s$ for all possible $E_s$ or to find an isogeny from $E_0$ to $E_s$ in time
less than \( t_{\text{id}} \) in the online phase. Therefore we choose \( p \approx 2^{2\kappa} \) and \( \sqrt{p} \lesssim d_s \lesssim p \) or equivalently \( 2^\kappa \lesssim d_s \lesssim 2^{2\kappa} \). The degree \( d_T \) is chosen such that computing an isogeny of degree \( d_T \) takes at least time \( t_{\text{id}} \), but not much more, and we require \( d_s \leq d_T \). Since \( P_s = \varphi_s(P'_s) \) and \( Q_s = \varphi_s(Q'_s) \) have to generate \( E_s[d_T] \), we need the degree \( d_s \) of \( \varphi_s \) to be coprime to \( d_T \). In Section 5 we give a more detailed justification of these numbers.

The starting curve could be any supersingular elliptic curve \( E_0 \) with a known efficient representation of \( \mathcal{O}_0 \). For our protocol we choose \( E_0 \) to be the curve \( E_0: y^2 = x^3 + x \) with \( (p + 1)^2 \) points over \( \mathbb{F}_p^2 \) and \( \mathcal{O}_0 = \langle 1, i, 1 + i, \frac{1 \pm i}{2} \rangle_\mathbb{Z} \) for \( p \equiv 3 \mod 4 \). Usually, we would want \( d_s \) and \( d_T \) to be smooth numbers both dividing \( p + 1 \) in order to have fast evaluation of the corresponding isogenies. So \( d_s \) should be smooth and divide \( p + 1 \) (or \( p^2 - 1 \)). However, evaluating \( \varphi_T \) does not need (in fact should not) be efficient, since it is only evaluated by \( \text{FDecom} \). Therefore \( d_T \) can contain large prime factors and should be large.

For a supersingular curve with \((p+1)^2\) points over \( \mathbb{F}_p^2 \) we have \((p^e - (-1)^e)^2\) points over \( \mathbb{F}_p^{2e} \) and the largest fully \( \mathbb{F}_p^{2e} \)-rational torsion group is the \((p^e - (-1)^e)\)-torsion. This means that for large \( d_T \) we need to go to extensions of \( \mathbb{F}_p \) to find a basis for the \( d_T - \text{torsion} \) group of \( E_0 \) or \( E_s \). Higher extensions and larger \( p \) slow down the computations, therefore we want to minimize the degree of the extension and the size of \( p \) to increase efficiency. Since the size of \( p \) affects almost all computations, whereas the size of \( e \) only influences computations related to the \( K_T \) or \( \varphi_T \), it can be beneficial to choose a smaller \( p \) and a larger \( e \) when dealing with large \( d_T \).

For an implementation we can choose a prime \( p \) such that \( p + 1 \) contains a smooth factor \( d_s = 2^\kappa \). This ensures that the first isogeny \( \varphi_s: E_0 \to E_s \) can be evaluated efficiently. We can even force \( d_T \) to be a power of 2 for efficient evaluation of \( \varphi_T \). After choosing a prime, we find an extension degree \( e \) such that \( p^e - (-1)^e \) contains a sufficiently large factor \( d_T \) that is coprime to \( d_s \). The primes used in SQISign and SIKE allow to choose \( d_s \) (and \( d_T \)) this way. So there are already known primes with the right properties for different security levels. The primes for SQISign even allow \( d_s \approx p \) in \( \mathbb{F}_p^4 \) and therefore \( t_{\text{id}} \) almost as large as \( p^{1/4} \) instead of \( p^{1/5} \). This could be a good trade-off for large delays.

**Message space \( M \) and function \( F \)** We choose \( M \) to be a finite group \( M = \mathbb{Z}/N\mathbb{Z} \) for an integer \( N \in \mathbb{N} \). This gives us very efficient testing and group operations. The size of \( N \) depends on the needed length of a message \( m \) and the prime \( p \). If \( N \) is larger than \([p/12] + 2\), then \( F: J_{SS} \to M \) can not be surjective and therefore \( u = m \oplus F(j_T) \) might leak information about the message \( m \).

As mentioned before, computing \( j_T \) from \( F(j_T) \) has to be infeasible or at least slow. In order to satisfy hiding we choose the function \( F \) to be \( \lambda \)-inverse resistant with \( \lambda = \kappa \approx \log \sqrt{p} \). In addition it has to be fast since \( \text{Com} \) and \( \text{DecVrfy} \) have to compute \( F(j_T) \). An easy way to accomplish this is to take a function that is not injective. The larger the kernel of \( F \), i.e. smaller \( N \), the more information is lost. A simple projection \( \mathbb{F}_p \supset J_{SS} \to \mathbb{F}_p \) onto one of the components or even their sum will leak information, since there is a subset of \( j \)-invariants that already are in \( \mathbb{F}_p \). If we use a simple map like \((a, b) \mapsto b \mod N \) or \((a, b) \mapsto a + b \mod N \), we thus need to use \( N \ll p \). For an implementation
we can identify $J_{SS} \subset \mathbb{F}_p^2$ with a subset of $\mathbb{F}_p[i] \cong \mathbb{F}_p^2$ and choose

$$F : J_{SS} \to M = \mathbb{Z}/N\mathbb{Z}, \quad a + bi \mapsto a + b \mod N$$

with $N = \lfloor \sqrt{p}/12 \rfloor$. Then we can expect every element in $M$ to be the image of about $\sqrt{p} \approx 2^{16}$ elements in $J_{SS}$. There is no direct way of finding the supersingular $j$-invariants. Hence, one would have to compute the preimage in $\mathbb{F}_p^2$ (about $12p^{3/2}$ elements) and check if they are $j$-invariants of supersingular elliptic curves. This is sufficiently inverse resistant in practice.

**Remark 4.2.** If $d_s = N \approx \sqrt{p}$ (or $d_s$ slightly smaller) we can add $v = s \odot F(j_T)$ to the commitment $C$ to make the scheme publicly verifiable. In this case $s \in [0, d_s]$ can be uniquely recovered from $v$ if we know $F(j_T)$, so $\text{FDecom}$ could also provide the decommitment proof $\pi_{dec} = s$ and everyone could use $\text{DecVrfy}$ to verify the output of $\text{FDecom}$ instead of computing it themselves. Since $s$ can be considered as a random number in $M$ (in this case), the additional $v$ in the commitment will neither leak information about $F(j_T)$ nor about $s$ unless we already know $F(j_T)$.

## 5 Security

We show that our protocol satisfies the Definition 2.1 of a NITC scheme by Katz et al. [19] and prove the three properties practicality, hiding and binding. In order to prove practicality, we need assumptions for the relative speed of some algorithms. Remember that our timings are the number of operations rather than real world times.

**Remark 5.1.** Operations in $\mathbb{F}_{p^{2\kappa}}$ are slower than operations in $\mathbb{F}_{p^2}$. In particular, the majority of operations of $\text{FDecom}$ are in extension fields, but for $\text{Com}$, $\text{ComVrfy}$ and $\text{DecVrfy}$ most operations can be done in $\mathbb{F}_{p^2}$. So our timings are rather conservative.

Our algorithms have the correct input and output arguments and for all $\kappa$ and $m \in M$ every set of honestly generated $(\kappa, m, \text{crs}, C, \pi_{com}, \pi_{dec})$ satisfies verification $\text{ComVrfy}(\text{crs}, C, \pi_{com}) = \text{accept} = \text{DecVrfy}(\text{crs}, C, m, \pi_{dec})$ and forced decommitment $\text{FDecom}(\text{crs}, C) = m$. This makes it a NITC scheme.

### 5.1 Hiding and Binding

For hiding we use the same (non-malleability) Definition 2.4 as Katz et al. [19]. First we show why we need an adapted security game. In Definition 2.3 the adversary $A$ sends two messages $m_0, m_1$ and receives the commitment $C_b = (E_b, K_T, u_b)$ corresponding to message $m_b$ for a uniform $b \in \{0, 1\}$. It is allowed to query an oracle for $\text{FDecom}(\cdot)$ except for $\text{FDecom}(\text{crs}, C_b)$.

**Lemma 5.2.** An adversary $A$ can break hiding with the original security game from Definition 2.3.

**Proof.** Since $m_{1-b} \oplus m_b \oplus u_b = u_{1-b}$, querying $\text{FDecom}(\text{crs}, (E_b, K_T, u_{1-b}))$ with $u_+ = (m_0 \oplus m_1) \oplus u_b$ and $u_- = \ominus (m_0 \ominus m_1) \ominus u_b$ gives $m_{1-b}$ and a random message $m'$. For $|M| = 2$ we have $u_+ = u_-$ and get $m_{1-b}$. For $|M| > 2$ however,
we can assume $m_0 \neq m' \neq m_1$. This allows $A$ to output the correct $b' = b$ with high probability.

Even worse, if we replace $K_T$ by any other point $K'$ such that $\langle K' \rangle = \langle K_T \rangle$, e.g. $K' = [\ell] K_T$ for $\ell$ coprime to $d_T$, or apply an isomorphism such that $E'/\langle K' \rangle \cong E_T \cong E_s/\langle K_T \rangle$ then $\text{FDecom}(\text{crs}, (E', K', u_b))$ will return $m_0$.

Thus, it is reasonable to disallow queries of the form $\text{FDecom}(\text{crs}, (E', K', \cdot))$ for $E'/\langle K' \rangle \cong E_s/\langle K_T \rangle$, i.e. using the adapted security game in Definition 3.5. This is still in the spirit of the original definition, since it prohibits the “de-
cription” of the commitment in question. In our case the security arises from the secret isogeny $\varphi_\varepsilon: E_0 \to E_s$ and the long isogeny $\varphi_T: E_s \to E_T$ with kernel $\langle K_T \rangle$, and the “key” is $F(j_T)$ for $j_T = j(E_T)$. Such queries would enable $A$ to find $F(j_T)$ and would hence basically allow to query $\text{FDecom}(\text{crs}, (E_s, K_T, u_b))$ by proxy, which is forbidden in the original definition.

**Assumption 5.3.** We assume that the probability to find the correct output in the online phase (step 3) of the security game from Definition 3.5 in time $t_o < t_{td}$ is less than $2^{-\kappa}$ if $F$ is a $\kappa$-inverse resistant function.

Let us justify this assumption by looking at the security game from Definition 3.5. Assume that $F$ is a $\kappa$-inverse resistant function as specified in the protocol. In the online phase $A$ sends two messages $m_0, m_1$ and receives the output $(E_s, K_T, u_b)$ of $\text{Con}(\text{crs}, m)$ for a uniform $b \in \{0, 1\}$. The adversary $A$ knows that $F(j_T)$ is equal to $F_0 = \oplus u_b \oplus m_0$ or $F_1 = \oplus u_b \oplus m_1$, but for each $i \in \{0, 1\}$ there are at least $2^{\kappa}$ $j$-invariants such that $F(j) = F_i$ and none of them is more likely than the other. To verify one of them, $A$ would have to compute $E_s/\langle K_T \rangle$.

But since this is equivalent to computing $\text{FDecom}(\text{crs}, (E_s, K_T, u_b))$, it can not be done in time less than $t_{td}$. Similar, querying $\text{FDecom}(\text{crs}, (E_s, [\ell] K_T, u_b))$ for $\ell | d_T$ gives $m_\ell = u_b \oplus F(j_\ell)$ and hence $F(j_\ell) = \oplus u_b \oplus m_\ell$ for $j$-invariants $j_\ell$ of intermediate curves of the long isogeny $\varphi_T$. But since $F$ is a $\kappa$-inverse resistant function, there are at least $2^{\kappa}$ undistinguishable candidates for each $j_\ell$. For a (small) prime $\ell$ a match between the $(\ell + 1)$ neighbours of each candidate for $j_\ell$ in the $\ell$-isogeny graph and the candidates for $j_T$ from each $F_0$ and $F_1$ has to be found. The probability to find such a match is less than $t_o 2^{-2\kappa} < 2^{-\kappa}$ using $t_o < t_{td} < p^{1/4} < 2^{\kappa}$ and the fact that not all of time $t_o$ can be spent on this task. Replacing $E_s$ and $K_T$ by a curve $E'$ and point $K'$ such that $E'/\langle K' \rangle$ is un-
related to $E_s/\langle K_T \rangle$ or intermediate curves the query $\text{FDecom}(\text{crs}, (E', K', u_b))$ will give completely unrelated results.

**Theorem 5.4.** For a $\kappa$-inverse resistant function $F$ and under Assumption 5.3, SIGNITC is $(t_p, t_o, \varepsilon)$-CCA-secure (hiding by Definition 2.4) with security game from Definition 3.5 for $t_p \ll 2^{\kappa}$ polynomial in $\kappa$, $t_o < t_{td}$ and $\varepsilon = 2^{-\kappa}$.

**Proof.** The pre-computation phase can only provide a negligible advantage for an adversary $A$. The computation of $\text{Con}(\text{crs}, m)$ includes choosing random $K_s \in E_0[d_s]$ and $K_T \in E_s[d_T]$ of maximal order. Since $2^\kappa \approx \sqrt{p} \lesssim d_s \leq d_T$, it is infeasible to pre-compute (and store) a significant subset of all possibilities in time $t_p \ll 2^{\kappa}$ polynomial in $\kappa$. For the online phase Assumption 5.3 gives us that the advantage over guessing is less than $2^{-\kappa}$.

The proof for binding works with the original Definition 2.6 and security game from Definition 2.5. With our protocol we even achieve perfect binding.
Lemma 5.8. Let for the lower bound we get the following lemma.

\[ \text{multiplications to take a similar time as } \text{mult}(c), \]

where \( c \) is a (small) constant larger than this and for highly composite \( d \) the crossover point for optimized algorithms is at \( q \approx 100 \) and we denote the time it takes to compute an isogeny of prime degree \( q \) with \( \text{eval}_{\text{prime}}(q) \). Computing isogenies efficiently is a well studied topic and we will assume that these timings are close to optimal.

Lemma 5.6. There is a (small) constant \( c_p \) such that evaluating an isogeny of prime degree \( q \) takes time \( \text{eval}_{\text{prime}}(q) \leq c_p q \).

Now let us look at an isogeny \( \varphi \) with a kernel that is generated by a point \( K_0' \) of order \( q^l \). We can decompose \( \varphi = \varphi_l \circ \cdots \circ \varphi_1 \) into isogenies \( \varphi_i \) of degree \( q \). In each step we compute the points \( K_i = [q^{l-i}]K_{i-1}' \) generating the kernel of \( \varphi_i \) and \( K_i' = \varphi_i(K_{i-1}) \) generating the kernel of \( \varphi_i' = \varphi_l \circ \cdots \circ \varphi_{i+1} \). So every step takes time \( \text{eval}_{\text{prime}}(q) \) plus the time it takes to compute the point multiplication. Generalizing this to isogenies of arbitrary composite degree gives us bounds for the time \( \text{eval}(d) \) it takes to compute an isogeny of degree \( d \).

Assumption 5.7. We assume that evaluating an isogeny \( \varphi: E \to E' \) of large degree \( d \) is slower than computing the multiple \([m]P\) of a point \( P \in E \) of order \( d \) for \( 0 \leq m < d \).

We justify this assumption by the following observations. For prime degree \( q \) we get \( \text{eval}_{\text{prime}}(q) \geq \min\{q, \sqrt{q}(\log q)^2\} \) and the time of a point multiplication is \( \log m \leq \text{mult}(m) \leq 2 \log m \) operations on \( E \) or \( \Theta(\log m) \) field operations, so \( \text{eval}_{\text{prime}}(q) \geq \text{mult}(q) \) for large primes \( q \). For composite degrees \( d = \prod q_i \) with \( q_1 \geq q_2 \geq \cdots \) there is a point multiplication in every intermediate step and their sum is faster than \( \text{mult}(d) \) by at most \( 2 \log q_1 + \log q_2 \) operations on \( E \), so by \( O(\log q_1) \) field operations. The sum \( \sum \text{eval}(q_i) \) can be expected to be larger than this and for highly composite \( d \) we can even estimate the sum of the multiplications to take a similar time as \( \text{mult}(d) \). If we ignore the multiplications for the lower bound we get the following lemma.

Lemma 5.8. Let \( d = \prod_{i=1}^{r} q_i^{e_i} \) be the prime factorization of the degree \( d \). There is a (small) constant \( c_c \geq 1 \) such that the time \( \text{eval}(d) \) it takes to evaluate an isogeny of degree \( d \) is bounded by

\[ \sum_{i=1}^{r} e_i \cdot \text{eval}_{\text{prime}}(q_i) \leq \text{eval}(d) \leq c_c \sum_{i=1}^{r} e_i \cdot \text{eval}_{\text{prime}}(q_i) \].
This allows us to choose $d_T$ such that $\text{eval}(d_T) \geq t_B$. Combining these results we get an upper bound for the computation time of isogenies of smooth degree.

**Lemma 5.9.** Evaluating an isogeny of $B$-smooth degree $d$ with prime factorization $d = \prod_{i=1}^{r} q_i^{e_i}$ takes time $\text{eval}(d) \in O\left(\frac{B}{\log B} \log d\right)$.

*Proof.* We use Lemmas 5.8 and 5.6 to write

$$\text{eval}(d) \leq c_e \sum_{i=1}^{r} e_i \text{eval}_{\text{prime}}(q_i) \leq c_e c_p \sum_{i=1}^{r} e_i q_i.$$  

Since $q_i < B$ for all $1 \leq i \leq r$, we get $q_i \leq \log q_i \frac{B}{\log B}$ and

$$\text{eval}(d) \leq c_e c_p \sum_{i=1}^{r} e_i \log q_i \frac{B}{\log B} = c_e c_p B \log d. \quad \square$$

According to Eisenträger et al. [15] the fastest (currently known) algorithms for solving the (equivalent) general Isogeny Path Problem, general Endomorphism Ring Problem or general Maximal Order Problem (cf. Section 3.2) over $\mathbb{F}_p^2$ take time $O(p^{1/2})$ for classical computations and $O(p^{1/4})$ with a quantum computer. Since $E_0$ and $E_s$ are known to be connected by a $d_s$-isogeny there is also a meet-in-the-middle or claw-finding attack in classical time $O(d_s^{1/2})$ and $O(d_s^{1/4})$ when applying Grover’s Algorithm [17].

**Assumption 5.10** (General Isogeny Problem Assumption). We assume that the fastest algorithms to solve the general Isogeny Path Problem, the general Endomorphism Ring Problem or the general Maximal Order Problem over $\mathbb{F}_p^2$ need at least $p^{1/2}$ or $p^{1/4}$ operations for classical or quantum algorithms, respectively.

**Assumption 5.11** (Special Isogeny Problem Assumption). We assume that the fastest algorithms to find an isogeny between two $d$-isogenous curves over $\mathbb{F}_p^2$ with $d < p$ take at least $d^{1/2}$ or $d^{1/4}$ operations for classical or quantum algorithms, respectively.

With these assumptions we can prove that computing the codomain of an isogeny can be made almost arbitrarily slow.

**Theorem 5.12.** Let $E$ be a supersingular elliptic curve over $\mathbb{F}_p^2$ with unknown $\mathcal{O} \cong \text{End } E$, but $d'$-isogenous to a curve $E_0$ with known endomorphism ring and $d' < p$. Let further $K$ be a point on $E$ of order $d_s$, such that computing the corresponding isogeny takes at least time $t$, according to Lemma 5.8. Then for $t < \min\{d^{1/4},p^{1/4}\}$ and under Assumptions 5.10 and 5.11, computing $E_K \cong E/\langle K \rangle$ takes at least time $t$.

*Proof.* The isogeny $\varphi : E \to E_K$ with kernel $\langle K \rangle$ has degree $d$. Efficiently calculating an equivalent isogeny $\hat{\varphi} : E \to E_K$ requires knowledge of $\mathcal{O} \cong \text{End } E$. Finding the endomorphism ring $\text{End } E$ or the order $\mathcal{O} \cong \text{End } E$ without an isogeny $\varphi : E_0 \to E$ or finding an isogeny $\hat{\varphi}$ without $\mathcal{O} \cong \text{End } E$ are hard problems. By Assumption 5.10 solving these problems takes time at least $p^{1/4} > t$ and by Assumption 5.11 finding an isogeny $\varphi'$ needs at least time $d'^{1/4} > t$. Therefore computing $E_T \cong E_s/(K_T)$ takes at least time $t$. \quad \square
The algorithm \textsc{FDecom} only has \texttt{crs} and \( C = (E_s, K_T, u) \) as input. In order to compute \( m = u \oplus F(j_T) \) it has to calculate the \( j \)-invariant \( j_T \) of the secret curve \( E_T \equiv E_s / \langle K_T \rangle \). Theorem 5.12 gives us the following corollary:

**Corollary 5.13.** For \( t_{fd} < \min\{d_s^{1/4}, p^{1/4}\} \) and under the Assumptions 5.10 and 5.11, the forced decommitment \textsc{FDecom} takes at least time \( t_{fd} \).

Note that the restriction \( t_{fd} < \min\{d_s^{1/4}, p^{1/4}\} \) is based on the quantum timings in Assumptions 5.10 and 5.11. For classical algorithms \( t_{fd} < \min\{d_s^{1/2}, p^{1/2}\} \) would be sufficient, but since our protocol should be quantum secure we chose the more general bound including quantum algorithms.

### 5.3 Practicality

We show that \textsc{ComVrfy} and \textsc{DecVrfy} can be computed efficiently and that we achieve a practical NITC scheme. We chose \( p = 2^{2^{128}} \), \( \sqrt{p} \lesssim d_s \lesssim p \) and \( t_{fd} < \min\{d_s^{1/4}, p^{1/4}\} \) to get \( \kappa \) bits of classical and \( \kappa/2 \) bits of quantum security for the pre-computation phase in hiding. In this subsection “efficiently” means a running time at most polynomial in \( \log p \).

**Lemma 5.14.** The maximal number of operations \( t_{cv} \) for algorithm \textsc{ComVrfy} is a small constant.

**Proof.** The algorithm has to complete three tasks. First it has to check if \( E_s \) is an elliptic curve. To do that, it suffices to check that the discriminant is non-zero. For curves in short Weierstraß form \( E : y^2 = x^3 + Ax + B \) this is just \( 4A^3 \neq -27B^2 \). To check if \( K_T \) is a point on \( E_s \) it can simply compute if \( K_T \) satisfies the curve equation. Finally, membership testing for \( u \in M \) is efficient by definition of \( M \). For \( M = \mathbb{Z}/N\mathbb{Z} \) this means checking if \( u \) is an integer (and if \( 0 \leq u < N \)). So all this can be done in very few operations and their number is independent of the size of \( d_s \), \( p \) and \( \kappa \).

**Lemma 5.15.** The decommitment verification algorithm \textsc{DecVrfy} takes time \( t_{dv} \in \text{poly}(\log p) \).

**Proof.** The number of operations on \( E_0 \) for computing \( K_s = P_0 + sQ_0 \) is linear in \( \log d_s \) and by Lemma 5.9 we can find \( E_s \equiv E_0 / \langle K_s \rangle \) via Vélu’s formulae in time \( O(\frac{B}{\log B} \log d_s) \) if \( d_s \) is \( B \)-smooth. SQLSign [12] shows that computing \( \mathcal{O}_s \equiv \text{End} E_s \) if we know \( \mathcal{O}_0 \equiv \text{End} E_0 \) and an isogeny \( \varphi_s : E_0 \rightarrow E_s \) of smooth degree can be done efficiently. By heuristics from [9], the degree of an equivalent isogeny can be expected to be roughly \( \sqrt{p} \) or smaller. We can see that finding an equivalent isogeny of smooth degree \( \tilde{d}_T \) can be done efficiently via the algorithms from [9]. Evaluating this isogeny to find \( E_T \equiv E_s / \langle K_T \rangle \) is in \( O(\frac{B}{\log B} \log \sqrt{p}) \) if \( \tilde{d}_T \) is \( B \)-smooth. Finally we have to compute \( j_T = j(E_T) \) and \( u = m \odot F(j_T) \). Since we chose \( F \) and the group operation in \( M \) to be efficiently computable, \( d_s \lesssim p \) and \( d_s, \tilde{d}_T \) smooth, we get that the algorithm takes time \( t_{dv} \in \text{poly}(\log p) \).}

Note that even for low security levels like \( \kappa = 128 \) we get that \( \log p \ll p^{1/8} \lesssim d_s^{1/4} \). Since \( t_{fd} \) can be almost as large as \( \min\{d_s^{1/4}, p^{1/4}\} \) and \( d_s \lesssim p \), the previous Lemmas 5.14 – 5.15 show that we can choose \( t_{fd} \) such that \( t_{cv}, t_{dv} \ll t_{fd} \). This gives us the following theorem:
**Theorem 5.16.** SIGNITC is practical under Assumptions 5.10 and 5.11.

Now we take a look at the running time of the commitment $\text{Com}$ and show that we can expect it to be faster than forced decommitment.

**Lemma 5.17.** The commitment $\text{Com}$ takes time $t_{\text{com}} \in O(\log d_T)$.

**Proof.** The only difference to $\text{DecVrfy}$ is, that we need to choose random $s \in [0,d_s)$, $t \in [0,d_T)$ and compute $K_T = P_s + tQ_s$. Finding $P_s = \varphi_s(P'_s)$ and $Q_s = \varphi_s(Q'_s)$ is in $O(\log d_s)$ by Lemma 5.9 since $d_s$ is smooth. Computing $[t]Q_s$ however takes time $\text{mult}(t) < \text{mult}(d_T)$ which is in $O(\log d_T)$ and $\log d_T \gg \log d_s$. So the running time is dominated by this step. 

Assumption 5.7 shows that we can assume $\text{eval}(d_T) > \text{mult}(d_T)$ and we can even expect the difference to be at least linear in $\log d_T$. So even with the other computations (linear in $\log d_s$) we can assume that $t_{\text{com}} < t_{\text{fa}}$ in practice.

**Conclusion**

We showed that SIGNITC is a practical NITC that satisfies hiding and binding. It is the first NITC without repeated squaring or black box algorithms, it needs no trusted setup and all sub-routines have already been implemented for other cryptosystems. Since it uses only isogeny-based cryptography, it is presumably quantum secure. Since repeated squaring might not be a good candidate for creating a delay anymore, this could also be an interesting starting point for isogeny-based delay in other settings. The next step is to implement this protocol to get some benchmarks for (relative) real world timings.

**References**


