Kleptography from Standard Assumptions and Applications

Adam Young[1] and Moti Yung[2]

[1] Cryptovirology.com
[2] Google Inc. and Department of Computer Science, Columbia University

Abstract. Kleptography deals with employing and generating cryptographically secure covert channels as threats to unscrutinized (e.g., tamper-proof) cryptosystems and their applications. A prototypical example is a cryptosystem (or a protocol message employing a cryptosystem) where a cryptogram field (e.g., a public key, an encrypted message, a signature value) hosts an "inner cryptographic field" that is invisible (in the sense of indistinguishability) to all but the attacker, yet it is a meaningful ciphertext to the attacker (who is the designer/producer of the cryptosystem). The technical goal of Kleptography has been to identify "inner fields" as a way to embed cryptographic values in small bandwidth channel/sub-cryptogram inside a hosting system (RSA, DH based systems, etc.)

All asymmetric backdoors to date, that seamlessly embed an inner subliminal crypto field inside a hosting cryptographic value needed random oracle assumptions. This was used to make the inner value look "almost uniformly random" as part of its hosting random field. It was open whether the need for a random oracle is inherent, or, positively put: is there an algebraic cryptographic ciphertext that is embeddable inside another algebraic cryptographic field "as is"? In this work we achieve this goal for small bandwidth fields. To this end we present a new information hiding primitive that we call a "covert key exchange" that permits provably secure covert communications. Our results surpass previous work since: (1) the bandwidth that the subliminal channel needs is extremely small (bit length of a single compressed elliptic curve point), (2) the error probability of the exchange is negligible, and (3) our results are in the standard model. We use this protocol to implement the first kleptographic (i.e., asymmetric) backdoor in the standard model in RSA key generation and point at other applications. Key properties of the covert key exchange are that (1) both Alice's message to Bob and their shared secret appear to all efficient algorithms as uniformly random strings from $\{0,1\}^{k+1}$ and $\{0,1\}^{M}$, respectively (this is needed for the embedding), and (2) the fastest adversaries of the exchange run in time exponential in k, based on current knowledge (they have to solve DL over e-curves). We achieve this in the standard model based on the ECDDH assumption over a twisted pair of e-curves.

1 Introduction

Advances in information hiding reveal new threats by parties (e.g., hardware manufacturers, system designers) and uncover trust-related issues in systems. Information hiding in cryptographic algorithms/ protocols, fundamentally requires embedding one cryptographic primitive inside another. Often, kleptographic design is perhaps more demanding than simple cryptosystems (one has usually to prove security of two cryptosystems: one inside the other). In particular, it requires a random element in one cryptosystem to host, as a random substring, an element from another cryptographic primitive, and do so in a hidden (indistinguishable) fashion.

For example, an kleptographic backdoor in RSA key generation has been shown that works for a wide-range of RSA keys (e.g., 768 bits and above) with a complete proof given in [26]. However, the result relies on the random oracle model and ECDDH, thus it does not tell us about an algebraic embedding of a backdoor directly inside an RSA modulus, and is not a "direct relationship" between two algebraic cryptographic distributions. Another example is a highly space-efficient public key stegosystem [17]. But, it relies on the less conventional "Oracle Diffie-Hellman" assumption, which again does not imply feasibility based on a more general assumption such as DDH.

A new approach is needed to advance the state-of-the-art in areas like these, so that we may understand better the direct embedability relationships between algebraic primitives and natural cryptographic assumptions about them (so we get proofs that are not in an idealized world).

Here, we show a new method for the task of "hidden embedding". In our key exchange, the approach balances the entropy in Alice's message to Bob against that of their resulting shared secret, and it permits Alice's message to be subliminally embedded (in a provably indistinguishable sense) and at the same time their shared secret is ready for direct use (no entropy extraction needed). We call the approach "entropy balancing." We further say that such an exchange is *covert* since Alice's trapdoor value looks like a random bit string. This problem is strictly more demanding than simply requiring that one value in the exchange, e.g. the shared secret, appear as a uniformly random bit string.

To make things even more difficult, a covert key exchange is often conducted through a subliminal channel having narrow bandwidth (e.g., in kleptography). First, this implies that if the channel is inside a hosting distribution, one can first determine a random choice of its wishes and can nevertheless sample the hosting distribution (which is a prime property of subliminal channels). Second, the small size necessitates a space-efficient key exchange (e.g., using elliptic curve crypto with point compression). To put it another way, whereas in many cryptographic applications space-efficiency is merely a convenience or an "added benefit", in information hiding space-efficiency is often a mandatory design requirement (this is the case in the asymmetric backdoors that we present).

Our *complete* covert key exchange solution relies on the traditional ECDDH problem in the following sense. We use a twisted pair of elliptic curves that exhibit the following distinguishing characteristic. In one curve the Weierstrass coefficient b is a quadratic residue whereas in the other curve this coefficient is a quadratic non-residue. Both curves in the twist have prime order and half of the upper order bits of p are binary 1s. As we show, an adversary that breaks our scheme is able to solve ECDDH on at least one of these two types of curves.

We employ the notion of entropy balancing to provide the following contributions: (1) *space-efficient covert key exchange* as our building block. We prove the security of the exchange in the standard model using only the traditional ECDDH assumption (as opposed to a newer assumption such as oracle DH). Note that Alice's message size is optimized provided that ECDDH is exponentially hard on both curves in the twist. (2) Kleptographic attack on RSA key generation. This was the first and most researched kleptographic problem, and we give the first complete solution in the standard model. (3) Public key stegosystem secure against chosen plaintext attacks based on ECDDH. (4) We also give the first asymmetric backdoor in SSL in the standard model such that: (a) the backdoor is in effect in each session with overwhelming probability, and (b) there is no state-information for the backdoor stored across sessions (i.e., no key-exposure between sessions).

Organization: In Section 2 we present related work and background material on twisted elliptic curves. The definition of a covert key exchange is presented in Section 3 and a construction for it is given in Section 4. In Section 5 we present our applications. ECDDH is reviewed in Appendix A. The Twisted Decision Diffie-Hellman (TDDH) problem is reviewed in Appendix B. We prove that the key exchange is complete (i.e., terminates in agreement) in Appendix C. The security proofs for the key exchange are given in Appendix D.

2 Background

The background material spans results on key exchange protocols, and asymmetric backdoors in RSA key generation. We review the works in these areas that closely relate to the applications that we present.

Entropy extraction in key exchanges: Previous work has solved the problem of conducting a key exchange in which the shared secret is a uniformly random binary string. The *leftover hash lemma* [12] was used to derive symmetric keys properly from a Diffie-Hellman shared secret [7]. In other words, the symmetric key bits are drawn independently using the uniform distribution. An algebraic approach based on a twisted pair of curves was used to derive a shared secret that is a uniformly random binary string [2]. Related work is [9] that presents a secure hashed Diffie-Hellman transform over a non-DDH group G (a group in which DDH does not hold). Gennaro et al showed that for the hashed DH transform to be secure it is sufficient that G contain a sufficiently large Decision Diffie-Hellman subgroup. Note that unlike the above cases, entropy balancing has the stronger requirement that one of the key exchange messages that is sent in the clear must also be a random binary string. This notion applies to various information hiding applications and we concretely implement this notion in the form of a covert key exchange protocol.

Asymmetric Backdoors in SSL: An asymmetric backdoor in the Secure Sockets Layer (SSL)[1] was shown in [10]. The constructions are heuristic in nature and there are no formal security arguments made. Recent work on the problem also includes [11] that presents general attack ideas. Again, no formal security arguments are made in these works.

An asymmetric backdoor in SSL was presented in [27] that employs Kaliski's elliptic curve pseudorandom bit generator [14]. The construction is based on a key exchange and is proven secure under ECDDH in the standard model. However, key agreement fails with probability very close to 1/2 and this causes the backdoor in SSL to fail to take effect in each session with probability very close to 1/2. However, an extension to the backdoor attack is shown that retains state information across SSL connections (chaining), and the expectation is that the backdoor will remain in effect after the first few SSL sessions. The two problems with the approach are that: (1) completeness of the backdoor is not assured in the first few exchanges, and (2) there is key-exposure in the state information that is stored in the backdoor. We solve these problems.

Asymmetric backdoors in RSA key generation: The notion of an asymmetric (kleptographic) backdoor was introduced in [24,25] along with the first asymmetric backdoor in RSA key generation. The backdoor employs a well-known subliminal channel [21,22] in composites [16,5] to leak the private key. An asymmetric backdoor can only be used by the designer that plants it, even when the full specification of the backdoor is public (confidentiality property). It is applicable in black-box implementations in which the implementation is private. This is relevant, for example, in obfuscated software or tamper-proof hardware or open source code that no one scrutinizes!

An asymmetric backdoor attack is known as a secretly embedded trapdoor with universal protection (SETUP). The embedded trapdoor is the public key of the (malicious) designer. The attack revolves around a reference key generator that has no backdoor (it is a typical public specification of the key generator). An RSA key generation SETUP must satisfy the indistinguishability property. This holds when the ensemble corresponding to the key pair that is output by the backdoor key generator is polytime indistinguishable from the ensemble corresponding to the key pair that is output by the reference key generator. So, RSA key pairs with the backdoor are indistinguishable from key pairs without the backdoor. A secure SETUP satisfies both the confidentiality and indistinguishability properties.

Crépeau and Slakmon presented backdoor designs for RSA key generation in [4]. These designs emphasized speed and were symmetric backdoors as opposed to being asymmetric since the constructions assume that a secret key remains hidden even after reverse-engineering (i.e., when the backdoor is layed bare). The paper presents an approach that is intended to work even when Lenstra's composite generation method is used [16]. The authors made an important observation in RSA key generation backdoor design, namely, that by using Coppersmith's algorithm

[1] Freier, Karlton, and Kocher, Internet Draft "The SSL Protocol Version 3.0," Network Working Group, Nov. 18, 1996.

[3], the amount of information that needs to be leaked is small. In more detail, it is sufficient for the backdoor key generator to leak the upper half of the bits of the RSA prime p instead of (nearly) the whole prime. We employ this observation.

A recent asymmetric backdoor in RSA key generation was proposed that utilizes a twisted pair of binary curves [26]. The backdoor key generator was shown to be secure in the random oracle model under the ECDDH assumption. The backdoor we present, on the other hand, does not use the random oracle model. Note that from a basic research point of view, asymmetric backdoor designs are not merely a warning to users, but represent also properties of cryptographic mechanisms, thus our result establishes a purely algebraic relationship between the two cryptographic distributions.

We point out that straightforward Diffie-Hellman (DH) [6] using twisted curves does not solve our problem of implementing a *space-efficient* covert key exchange. The space-efficiency of ECDH is not the problem, since points can be compressed. However, the problem is making Alice's message to Bob and their shared secret appear as random binary strings. To solve this, the curves could be, e.g. binary or over $\mathbb{F}_p$ with p being a large prime with the upper half of the bits of p being 1. In both cases, a properly chosen twist has the property that a random encoded (compressed) point on it looks like a random binary string. In such a solution, Alice chooses a curve in the twist randomly[2] and uses it for a DH key exchange. The problem is that Alice's trapdoor value to Bob will be on one of the two curves and their shared secret will be on *the same* curve. This correlation causes the solution to fail to solve the problem.

Indeed this key exchange problem was addressed in [27]. As noted, this idea causes Bob to fail to learn the shared secret with probability very close to 1/2. The *partial* covert key exchange solution in [27] is therefore ill-suited for a backdoor in RSA key generation since the backdoor would only take effect with probability very close to 1/2 when a key pair is generated. This is a serious problem since RSA keys are typically generated infrequently (sometimes once every couple of years). It also appears undesirable as a basis for an elliptic curve public key stegosystem since it is not clear how to preserve space-efficiency given that the exchange is prone to fail. In this paper we solve these issues.

2.1 Notation and Conventions

Elliptic Curves: An elliptic curve $E_{a,b}(\mathbb{F}_p)$ is defined by the simplified Weierstrass equation $y^2 = x^3 + ax + b$ where $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Let $\#E_{a,b}(\mathbb{F}_p)$ denote the number of points on the curve $E_{a,b}(\mathbb{F}_p)$. Let $\mathcal{O}$ denote the point at infinity on $E_{a,b}(\mathbb{F}_p)$. We use uppercase to denote a point on an elliptic curve and lowercase to denote a scalar multiplier. So, xG denotes scalar multiplication. Recall that $0G = \mathcal{O}$, $1G = G$, $2G = G + G$, and so on.

In our review of Kaliski's work in [14] (and in particular his Lemma 6.6 that covers embedding using twisted curves) we use his notation $E_{a,b}(\mathbb{F}_p)$ and $E_{a',b'}(\mathbb{F}_p)$. However, we do not use this convention in our constructions. We

[2] In accordance with the number of points on each curve.

define $E_0(\mathbb{F}_p) = E_{a,b}(\mathbb{F}_p)$ and $E_1(\mathbb{F}_p) = E_{a',b'}(\mathbb{F}_p)$. This lets us select between the two curves using a single bit, which we do often. We also define $r_i = \#E_i(\mathbb{F}_p)$ to be the number of points on curve $E_i(\mathbb{F}_p)$ for $i = 0, 1$. Let $\mathcal{O}_i$ be the point at infinity on $E_i(\mathbb{F}_p)$ for $i = 0, 1$.

String Operations: If α is a bit string then $|\alpha|$ denotes the length in bits of α. However, if $\mathcal{S}$ is a set then $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. Let $\alpha||\beta$ denote the concatenation of strings α and β. Let $\texttt{LSB}(\alpha)$ be a function that returns the least significant bit of bit string α. Let $\oplus$ denote the infix bitwise exclusive-or operator that operates on two bit strings of equal length. $\alpha >> b$ denotes the following string operation. The string α is returned but with the rightmost b bits removed from α (this is right shifting).

Selection and Assignment: We use $s \in_R \mathcal{S}$ to denote the selection of an element s uniformly at random from set $\mathcal{S}$. However, unless otherwise stated, an element that is selected randomly is selected using the uniform distribution. The symbol $\leftarrow$ is used for assignment.

Integers vs. Strings: We are careful to treat integers as separate from bit strings. This is to avoid ambiguities that can result from the presence/absence of leading zeros. This is of particular importance in information hiding since the representation of information must be carefully controlled. We define algorithm $\texttt{StrToInt}(x_s)$ to take as input a bit string x_s and return the integer x corresponding to x_s in base-2. We define algorithm $\texttt{IntToStr}(x)$ to take as input a non-negative integer x and return the bit string x_s corresponding to x in base-2 (so the most significant bit is always 1, unless $x = 0$ in which case the output is $x_s = 0$). We define algorithm `Format` as follows.

```
Format($x, \ell$):
1. $x_s \leftarrow$ IntToStr($x$)
2. if ($|x_s| > \ell$) then output $0^\ell$ else output $0^{\ell-|x_s|}||x_s$
```

Flow Control: We use logical indentation to show the body of `if` statements, `for` loops, and so on. Also, an algorithm that terminates early uses the keyword `halt` in order to terminate.

2.2 Elliptic Curve Background

Twists using the general class of elliptic curves over $\mathbb{F}_p$ were studied by Kaliski [13,14,15]. Below we give Lemma 6.5 and Definition 6.1 from [14].

Lemma 1. *Let $\beta \neq 0$ be a quadratic nonresidue in the field $\mathbb{F}_p$ and let $E_{a,b}(\mathbb{F}_p)$ be an elliptic curve. Then for every value x, letting $y = \sqrt{x^3 + ax + b}$:*

1. *If y is a quadratic residue, then the points $(x, \pm y)$ are on the curve $E_{a,b}(\mathbb{F}_p)$.*
2. *If y is a quadratic nonresidue, then $(\beta x, \pm\sqrt{\beta^3} y)$ are on $E_{a\beta^2,b\beta^3}(\mathbb{F}_p)$.*
3. *If $y = 0$, then the point $(x, 0)$ is on the curve $E_{a,b}(\mathbb{F}_p)$ and the point $(\beta x, 0)$ is on the curve $E_{a\beta^2,b\beta^3}(\mathbb{F}_p)$.*

A corollary to this lemma is that the number of points *on* the two curves *is* $2p + 2$, two points for each value of x and two identity elements.

Definition 1. *Let $E_{a,b}(\mathbb{F}_p)$ be an elliptic curve of parameter k (i.e., p is k bits long) and let β be a quadratic nonresidue modulo p. A twisted pair $T_{a,b,\beta}(\mathbb{F}_p)$ of parameter k is the union of the elliptic curves $E_{a,b}(\mathbb{F}_p)$ and $E_{a\beta^2,b\beta^3}(\mathbb{F}_p)$.*

A twist may be a multiset, since $E_{a,b}(\mathbb{F}_p)$ and $E_{a\beta^2,b\beta^3}(\mathbb{F}_p)$ may intersect.

Below we review algorithms from [27] (which are built on [14]) that encode/decode points using fixed-length bit strings. Fact 1 is from [14]. The input P to `Encode` is a point originating on E_c where $c \in \{0, 1\}$. `Decode` outputs (P, c) where P resides on E_c where $c \in \{0, 1\}$. X_T, $X^{-1}_{T,even}$, and $X^{-1}_{T,odd}$ are defined in [14]. $\texttt{Encode}(T_{a,b,\beta}(\mathbb{F}_p), P, c) = \texttt{Format}(X_T[T_{a,b,\beta}(\mathbb{F}_p)](P, c), k + 1)$.

```
Decode($T_{a,b,\beta}(\mathbb{F}_p), P_s$):
1. set $ysgn \leftarrow \mathrm{LSB}(P_s)$ and set $\alpha \leftarrow \texttt{StrToInt}(P_s)$
2. if $(ysgn = 0)$ then output $(P, c) \leftarrow X^{-1}_{T,even}[T_{a,b,\beta}(\mathbb{F}_p)](\alpha)$ and halt
3. output $(P, c) \leftarrow X^{-1}_{T,odd}[T_{a,b,\beta}(\mathbb{F}_p)](\alpha)$
```

Fact 1. Let $T_{a,b,\beta}(\mathbb{F}_p)$ be a twisted pair. `Encode` is a polynomial time computable, probabilistic polynomial time invertible mapping between the set of points on the twisted pair $T_{a,b,\beta}(\mathbb{F}_p)$ and all $(k + 1)$-bit strings corresponding to the integers in the set $\{0, ..., 2p + 1\}$ padded with leading zeros as necessary. The inverse function of `Encode` is `Decode`.

3 The Covert Key Exchange

The covert key exchange protocol, denoted by Φ_1, is an implementation of the notion of entropy balancing for secret embeddings. The covert key exchange uses a twisted pair of curves over $\mathbb{F}_p$ where p is a k-bit prime. Informally, when we say that the covert key exchange is *space-efficient* what we mean is that the following properties hold: (1) Alice's message to Bob in the exchange is $k+1$ bits long and (2) the best known cryptanalytic algorithms against the exchange (confidentiality breaking/distinguishing) run in time exponential in k. Currently, the fastest known algorithm for solving ECDDH for curve $E_{a,b}(\mathbb{F}_p)$ runs in time $O(\sqrt{\#E_{a,b}(\mathbb{F}_p)})$ when $\#E_{a,b}(\mathbb{F}_p)$ is prime. It is from this that (2) holds based on current knowledge. This enables Alice to send a small key exchange message to Bob (in the hundreds of bits rather than in the thousands for an algorithm based on DL over a finite field or factoring), based on the state-of-the-art. Property (2) is mandatory to achieve the degree of space-efficiency that we desire (so it is part of the definition below that concretely defines the setting of the key exchange and its constraints).

Definition 2. *Let $\tau = (T_{a,b,\beta}(\mathbb{F}_p), G_0, G_1)$ be agreed upon system parameters where $|p| = k$, let $M \leq 10^4$ be a constant, let $\mathcal{T}_{\tau,\Phi_1}$ denote the probability ensemble corresponding to all possible $(Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$ resulting from Φ_1 and the probability distribution over them resulting from Φ_1, let $\mathcal{T}_{\tau,U}$*

denote $E_0(\mathbb{F}_p)^M \times E_1(\mathbb{F}_p)^M \times \{0,1\}^{k+1} \times \{0,1\}^M$ and the uniform distribution over it, and let Alice and Bob be probabilistic polytime algorithms. If in a 2-round protocol Φ_1 between Alice and Bob, Bob sends $(Y_{0,1},...,Y_{0,M},Y_{1,1},...,Y_{1,M})$ to Alice where $Y_{i,j} \in_R E_i(\mathbb{F}_p)$ for $i = 0,1$ and $j = 1,2,...,M$ (Bob knows the discrete logs $x_{i,j}$ of the $Y_{i,j}$), and Alice generates a $(k+1)$-bit message m_A and a M-bit shared secret m_k, and sends m_A to Bob, and

1. *(completeness) using m_A and private information (the $x_{i,j}$) Bob computes m_k with a probability that is overwhelming (in k), and*
2. *(security) the fastest algorithm that distinguishes ensemble $\mathcal{T}_{\tau,\Phi_1}$ from ensemble $\mathcal{T}_{\tau,U}$ with an advantage that is non-negligible (in k) runs in exponential time ($2^{\epsilon k}$ for a constraint $\epsilon > 0$),*

then Φ_1 is a k-secure ***space-efficient covert key exchange****.*

The upper bound on M is somewhat arbitrary (10^4 is simply large enough for practical use). Security implies covertness since it establishes that m_A appears as a uniformly random $(k+1)$-bit string. It also implies confidentiality since it establishes that m_k appears as a uniformly random M-bit string.

4 Key Exchange Construction

Alice and Bob agree on TDDH parameters τ. For our reductions to hold and our applications to be secure, they should generate τ using $\mathtt{IG}_1$ (see Appendix B for a review of TDDH) where $\mathtt{IG}_1$ adheres to the following list of constraints.

1. The prime p in τ is of the form $p = 2^k - \delta$ where δ satisfies $1 \leq \delta < \sqrt{2^k}$. The value δ may be randomly chosen until p of this form is generated.
2. The curves $E_0(\mathbb{F}_p)$ and $E_1(\mathbb{F}_p)$ have prime order. So, r_0 and r_1 are prime.
3. To prevent the attack on prime-field-anomalous curves [19,20,23], it is required that $r_0, r_1 \neq p$.
4. To prevent the Weil and Tate pairing attacks [18,8], it is required that r_0 does not divide $p^\nu - 1$ for $\nu = 1,2,3...$ up to, say, 20. The same holds for r_1.
5. More generally, $E_0(\mathbb{F}_p)$ and $E_1(\mathbb{F}_p)$ *must* provide suitable settings for the ECDDH problem.

From Subsection 2.2 it is the case that $r_0 + r_1 = 2p + 2$. We define $\mathcal{S}_{a,b,\beta,p}$ as follows.
$\mathcal{S}_{a,b,\beta,p} =$

$$\{s : P \in E_0(\mathbb{F}_p), s = \mathtt{Encode}(T_{a,b,\beta}(\mathbb{F}_p), P, 0)\}$$
$$\bigcup \{s : P \in E_1(\mathbb{F}_p), s = \mathtt{Encode}(T_{a,b,\beta}(\mathbb{F}_p), P, 1)\}$$

4.1 Intuition Behind the Covert Exchange

By glossing over some details and omitting others, it is possible to describe the covert key exchange algorithm at a high-level. We do so here.

Bob generates $2M$ public keys. Half are points on one curve in the twist and half are on the other. He gives these to Alice. It is her job to generate both her trapdoor value to Bob and their shared secret. Given her trapdoor value he will with overwhelming probability compute their shared secret using his $2M$ private keys.

Alice chooses one of the curves to use randomly. In this choice, the curves are selected in direct proportion to the number of points on them. So, the curve in the twist with the most points is slightly more likely than the other. Let the curve she selects be denoted by $E_u(\mathbb{F}_p)$ where $u \in \{0,1\}$. She generates a scalar multiplier k_1 randomly for this curve and computes her key exchange value to Bob using it. She then generates M shared secrets using k_1 and the M public keys of Bob on $E_u(\mathbb{F}_p)$. She Kaliski encodes them and concatenates the least significant bits of the M encodings together. The resulting string is their shared secret.

We defer to the proofs why the shared secret appears independently random and uniformly distributed. The method that Bob uses to compute the shared secret can be inferred from the above.

4.2 Key Exchange Protocol Φ_1

We define `SelCurve`$(T_{a,b,\beta}(\mathbb{F}_p))$ to be a randomized algorithm that outputs 0 with probability $\frac{r_0}{2p+2}$ and 1 with probability $\frac{r_1}{2p+2}$. Algorithm `DeriveBit` and algorithm `FillGap` are used in the key exchange.

```
DeriveBit($T_{a,b,\beta}(\mathbb{F}_p), P$):
Input: point $P$ on twist $T_{a,b,\beta}(\mathbb{F}_p)$
Output: $b \in \{0,1\}$
1. if ($P \in E_0(\mathbb{F}_p)$) then $c \leftarrow 0$ else $c \leftarrow 1$
2. if ($P = \mathcal{O}_c$) then output $b \in_R \{0,1\}$ and halt
3. output $b \leftarrow$ LSB($\psi_s$) where $\psi_s \leftarrow$ Encode($T_{a,b,\beta}(\mathbb{F}_p), P, c$)
```

```
FillGap($T_{a,b,\beta}(\mathbb{F}_p)$):
Input: twist $T_{a,b,\beta}(\mathbb{F}_p)$
Output: $(y, s_1, s_2) \in \{0,1\} \times \{0,1\}^{k+1} \times \{0,1\}^M$
1. choose $\mu \in_R \{0,1\}^{k+1}$
2. if ($\mu \notin \mathcal{S}_{a,b,\beta,p}$) then
3.      choose $\psi \in_R \{2p+2, 2p+3, ..., 2^{k+1}-1\}$
            and $s_2 \in_R \{0,1\}^M$
4.      compute $s_1 \leftarrow$ Format($\psi, k+1$),
            output $(1, s_1, s_2)$, and halt
5. output $(0, 0^{k+1}, 0^M)$
```

We refer to the following as protocol Φ_1. The value $M \leq 10^4$ is a constant.

Step 1: Bob chooses $x_{i,j} \in_R \{0,1,2,...,r_i - 1\}$ and computes $Y_{i,j} \leftarrow x_{i,j}G_i$ for $i = 0,1$ and $j = 1,2,...,M$. Bob sends $(Y_{0,1},...,Y_{0,M},Y_{1,1},...,Y_{1,M})$ to Alice.

Step 2: Alice sends the key exchange message m_A to Bob where $(m_A, m_k) \leftarrow \texttt{ExchAlg}_1(\tau, Y_{0,1},...,Y_{0,M},Y_{1,1},...,Y_{1,M})$.

```
ExchAlg$_1(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M})$:
Input: TDDH parameters $\tau$, points $Y_{i,j}$ on
    $E_i(\mathbb{F}_p)$ for $i = 0, 1$, $j = 1, 2, ..., M$
Output: $(m_A, m_k) \in \{0,1\}^{k+1} \times \{0,1\}^M$
1. $(y, s_1, s_2) \leftarrow$ FillGap$(T_{a,b,\beta}(\mathbb{F}_p))$
2. if $(y = 1)$ then output $(s_1, s_2)$ and halt
3. $u \leftarrow$ SelCurve$(T_{a,b,\beta}(\mathbb{F}_p))$
4. choose $k_1 \in_R \mathbb{Z}_{r_u}$ and compute $U \leftarrow k_1 G_u$
5. $m_A \leftarrow$ Encode$(T_{a,b,\beta}(\mathbb{F}_p), U, u)$
6. for $j = 1$ to $M$ do:
7.     compute $P_j \leftarrow k_1 Y_{u,j}$ and
           $b_j \leftarrow$ DeriveBit$(T_{a,b,\beta}(\mathbb{F}_p), P_j)$
8. set $m_k \leftarrow b_M || b_{M-1} || ... || b_1$ and output $(m_A, m_k)$
```

Step 3: Bob receives the message m_A from Alice. Bob computes Recover$(T_{a,b,\beta}(\mathbb{F}_p), m_A, x_{0,1}, ..., x_{0,M}, x_{1,1}, ..., x_{1,M})$, thereby obtaining (t, m_k).

```
Recover$(T_{a,b,\beta}(\mathbb{F}_p), m_A, x_{0,1}, ..., x_{0,M}, x_{1,1}, ..., x_{1,M})$:
Input: twist $T_{a,b,\beta}(\mathbb{F}_p)$, $m_A \in \{0,1\}^{k+1}$,
    $x_{i,j} \in \mathbb{Z}_{r_i}$ for $i = 0, 1$, $j = 1, 2, ..., M$
Output: $(t, m_k) \in \{0,1\} \times \{0,1\}^M$
1. if $(m_A \notin \mathcal{S}_{a,b,\beta,p})$ then output $(0, 0^M)$ and halt
2. $(U, u) \leftarrow$ Decode$(T_{a,b,\beta}(\mathbb{F}_p), m_A)$
3. for $j = 1$ to $M$ do:
4.    compute $P_j \leftarrow x_{u,j} U$ and
         $b_j \leftarrow$ DeriveBit$(T_{a,b,\beta}(\mathbb{F}_p), P_j)$
5. set $m_k \leftarrow b_M || b_{M-1} || ... || b_1$ and output $(1, m_k)$
```

We define *failure* of protocol Φ_1 to be a condition in which any of the following occur:

1. $m_A \in \mathcal{S}_{a,b,\beta,p}$ and the decoding of m_A is on $E_u(\mathbb{F}_p)$ and $\exists\ j \in \{1, 2, ..., M\}$ such that $Y_{u,j} = \mathcal{O}_u$.
2. $m_A \in \{0,1\}^{k+1} \setminus \mathcal{S}_{a,b,\beta,p}$.
3. $m_A \in \mathcal{S}_{a,b,\beta,p}$ and the decoding of m_A is $\mathcal{O}_0$ or $\mathcal{O}_1$ (i.e., ExchAlg$_1$ chooses $k_1 = 0$).

The rationale behind this definition is as follows. These are the conditions that cause one or more bits of m_k to be derived directly from a fair coin flip. So, Bob cannot be certain that he receives m_k correctly in these cases.

An alternate definition is of course possible. For example, we could define success as the raw probability that Bob decides on an m_k that is the same as the m_k that Alice selected (Bob may have to guess 1 or more bits). However, we have decided to use a definition of success that measures Bob's certainty regarding his computation of m_k.

The following Lemma is proven in Appendix C where the definition of a negligible function is reviewed.

Lemma 2. *(completeness) Failure occurs in protocol Φ_1 with a probability that is negligible in k.*

In our failed attempts at designing the exchange we found that the following seemed to occur: when we succeeded in removing the bias in m_A, the construction would introduce bias into m_k in the process, and vice-versa. The two are inextricably linked. $\texttt{ExchAlg}_1$ succeeds in balancing the entropy in m_A and m_k in the sense that they are both random binary strings. This is the origin of our terminology.

We prove Theorem 1 in Appendix D. Theorem 1 establishes Property 2 of Definition 2.

Theorem 1. *(security) The ECDDH problem over $(E_0(\mathbb{F}_p), G_0)$ or ECDDH over $(E_1(\mathbb{F}_p), G_1)$ polytime reduces to the problem of distinguishing $\mathcal{T}_{\tau,\Phi_1}$ from $\mathcal{T}_{\tau,U}$ with an advantage that is non-negligible (in k).*

Lemma 2 establishes property 1 of Definition 2. So, Lemma 2 and Theorem 1 imply Theorem 2.

Theorem 2. *If ECDDH requires exponential time (in k) on $(E_0(\mathbb{F}_p), G_0)$ and $(E_1(\mathbb{F}_p), G_1)$ then Φ_1 is a k-secure space-efficient covert key exchange.*

The above theorem is proven in the appendix. The proof establishes that the exchanged value from Alice is random and that the key established is random and hard to compute to any polynomial time adversary (under the proper decisional DH assumption). The basic intuition of the proof idea is that the exchanged value is random and then the decisional assumption implies that the resulting shared secret is indistinguishable from a random value in the target group, and therefore the shared key extracted itself should not help distinguishing between them and thus is indistinguishable from a pseudorandomly chosen value and thus from a randomly chosen value.

5 Applications

Kleptographic RSA key generation in the standard model: The following describes how to build an asymmetric backdoor in RSA key generation in which the RSA public exponent e can be fixed for all users. The attacker Bob places his public key $(Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M})$ in Alice's key generation device. Key generation performs Alice's side of the covert key exchange. The shared secret m_k is embedded in the upper order bits of the RSA prime p being generated. The corresponding key exchange value m_A is encoded in the upper order bits of the RSA modulus being generated. Under this constraints an otherwise random public composite is chosen. The attacker obtains Alice's public key from a public channel (e.g., CA, a certificate, etc.). He extracts m_A and uses his private key to compute m_k. Using this and Coppersmith's factoring algorithm [3], he factors the public modulus.

The security adheres to the notion of a SETUP (secretly embedded trapdoor with universal protection). It can be shown that indistinguishability holds provided ECDDH is hard on both curves in the twist. Confidentiality can be shown under this and the assumption that integer factorization is hard.

Public Key Stegosystem: The stegosystem construction is a straightforward implementation of ElGamal based on the covert key exchange. The primary difference between this version of the ElGamal and traditional implementations (ECC and otherwise) is that we exclusive-or the plaintext with the shared secret. Informally, the indistinguishability property dictates that a ciphertext must appear as a fixed-length bit string, where the bits appear as fair coin tosses. We can show that this holds provided ECDDH is hard on both curves in the twist. Furthermore, we can show that if ECDDH is hard on both curves in the twist then the stegosystem is semantically secure against chosen plaintext attacks.

Kleptographic backdoor in SSL: It is well-known that the 28 byte `hello` nonce in SSL is a subliminal channel that is visible in the clear to passive eavesdroppers on the network. The client sends such a nonce to the server and vice-versa. The pre-master secret is 48 bytes and it contains a 46 byte nonce. The pre-master secret nonce is chosen by the client and is sent securely to the server. Knowledge of the two hello nonces and the pre-master secret nonce implies the ability to eavesdrop on the SSL session.

We build an asymmetric backdoor into the client in the SSL protocol. Let $k = 223$ and $M = 368$ for the covert key exchange. The attacker Bob places his elliptic curve public key $(Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M})$ into Alice's SSL client.

When Alice initiates an SSL session with a server, the asymmetric backdoor takes over the generation of the client hello nonce and the pre-master secret nonce. The backdoor runs $\texttt{ExchAlg}_1(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M})$ to obtain (m_A, m_k). It sets the hello nonce to be m_A and the pre-master secret nonce to be m_k.

Bob passively eavesdrops on the SSL session. He learns both hello nonces since he sees them in the clear. He sets m_A to be Alice's hello nonce. Bob then runs `Recover` and obtains m_k. He is then able to decipher the SSL session.

We note that this attack is a perfect example of the power of our covert key exchange primitive (of Section 4). This is because: (1) the subliminal channel is extremely narrow, a mere 224 bits, and therefore space-efficiency is an obvious requirement, and (2) the kleptographic application is incredibly simple. Observe that the host distribution over (hello nonce,pre-master secret nonce) is exactly the uniform distribution over $\{0, 1\}^{224} \times \{0, 1\}^{368}$. It is not likely to get simpler than this.

For security note that the indistinguishability and confidentiality properties of the asymmetric backdoor in SSL follow from Theorem 1. It is therefore a provable asymmetric backdoor on top of a secure subliminal channel.

References

1. Boneh, D.: The Decision Diffie-Hellman Problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
2. Chevassut, O., Fouque, P., Gaudry, P., Pointcheval, D.: The Twist-AUgmented Technique for Key Exchange. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 410–426. Springer, Heidelberg (2006)

3. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
4. Crépeau, C., Slakmon, A.: Simple backdoors for rsa key generation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 403–416. Springer, Heidelberg (2003)
5. Desmedt, Y.: Abuses in cryptography and how to fight them. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 375–389. Springer, Heidelberg (1990)
6. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
7. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
8. Frey, G., Rück, H.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. of Computation 62(206), 865–874 (1994)
9. Gennaro, R., Krawczyk, H., Rabin, T.: Secure hashed Diffie-Hellman over non-DDH groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 361–381. Springer, Heidelberg (2004)
10. Goh, E.-J., Boneh, D., Pinkas, B., Golle, P.: The design and implementation of protocol-based hidden key recovery. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 165–179. Springer, Heidelberg (2003)
11. Golebiewski, Z., Kutylowski, M., Zagorski, F.: Stealing secrets with SSL/TLS and SSH—kleptographic attacks. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 191–202. Springer, Heidelberg (2006)
12. Impagliazzo, R., Levin, L., Luby, M.: Pseudo-random generation from one-way functions. In: Symp. on the Theory of Comput.—STOC 1989, pp. 12–24 (1989)
13. Kaliski, B.S.: A pseudo-random bit generator based on elliptic logarithms. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 84–103. Springer, Heidelberg (1987)
14. Kaliski, B.S.: Elliptic curves and cryptography: A pseudorandom bit generator and other tools. PhD Thesis. MIT (February 1988)
15. Kaliski, B.S.: One-way permutations on elliptic curves. Journal of Cryptology 3(3), 187–199 (1991)
16. Lenstra, A.K.: Generating RSA moduli with a predetermined portion. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 1–10. Springer, Heidelberg (1998)
17. Möller, B.: A public-key encryption scheme with pseudo-random ciphertexts. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 335–351. Springer, Heidelberg (2004)
18. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. on Info. Theory 39(5), 1639–1646 (1993)
19. Satoh, T., Araki, K.: Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Commentarii Mathematici Universitatis Sancti Pauli 47, 81–92 (1998)
20. Semaev, I.: Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. Math. of Computation 67(221), 353–356 (1998)
21. Simmons, G.J.: The prisoners' problem and the subliminal channel. In: McCurley, K.S., Ziegler, C.D. (eds.) Advances in Cryptology—Crypto 1983. LNCS, vol. 1440, pp. 51–67. Springer, Heidelberg (1999)
22. Simmons, G.J.: Subliminal channels: past and present. European Transactions on Telecommunications 5(4), 459–473 (1994)

23. Smart, N.: The discrete logarithm problem on elliptic curves of trace one. Journal of Cryptology 12(3), 193–196 (1999)
24. Young, A., Yung, M.: The dark side of black-box cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)
25. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997)
26. Young, A., Yung, M.: A space efficient backdoor in RSA and its applications. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 128–143. Springer, Heidelberg (2006)
27. Young, A., Yung, M.: Space-efficient kleptography without random oracles. In: Furon, T., Cayre, F., Doërr, G., Bas, P. (eds.) IH 2007. LNCS, vol. 4567, pp. 112–129. Springer, Heidelberg (2008)

A Review of ECDDH

A group family denoted by $\mathcal{G}$ is a set of finite cyclic groups $\mathcal{G} = \{E_{a,b}(\mathbb{F}_p)\}$ where each group has prime order. Let IG_0 be an instance generator for $\mathcal{G}$ that on input k (in unary) generates $(E_{a,b}(\mathbb{F}_p), G)$ where $E_{a,b}(\mathbb{F}_p)$ is from $\mathcal{G}$, G is a generator of $E_{a,b}(\mathbb{F}_p)$, and $r = \#E_{a,b}(\mathbb{F}_p)$. The ECDDH assumption is that no polytime algorithm A_0 exists for $\mathcal{G}$. We define the *superpolynomial* ECDDH assumption to be that no superpolynomial time algorithm A_0 exists for $\mathcal{G}$.

Definition 3. *An ECDDH algorithm A_0 for $\mathcal{G}$ satisfies, for some fixed $\alpha > 0$ and sufficiently large k:*

$$|\Pr[A_0(E_{a,b}(\mathbb{F}_p), G, aG, bG, abG) = 1] - \Pr[A_0(E_{a,b}(\mathbb{F}_p), G, aG, bG, cG) = 1]| > \frac{1}{k^\alpha}$$

The probability is over the random choice of $(E_{a,b}(\mathbb{F}_p), G)$ according to the distribution induced by $\mathrm{IG}_0(k)$, the random choice of integers a, b, c satisfying $0 \le a, b, c \le r - 1$, and the bits used by A_0.

We now review the ECDDH randomization method from [1] adapted for the case of elliptic curves. Let the $(E_{a,b}(\mathbb{F}_p), G, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$ be an ECDDH problem instance. Algorithm $\mathtt{f}$ chooses scalars u_1, u_2, v randomly satisfying the inequality $0 \le u_1, u_2, v \le r - 1$. The function $\mathtt{f}(E_{a,b}(\mathbb{F}_p), G, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$ outputs $(v\mathcal{X} + u_1G, \mathcal{Y} + u_2G, v\mathcal{Z} + u_1\mathcal{Y} + vu_2\mathcal{X} + u_1u_2G)$.

B Review of Twisted DDH

We now review the twisted DDH problem that is covered in [27]. Let TW_k be the set of all twists of parameter k in which both groups (curves) in each twist have prime order. Let IG_1 be an instance generator for TW_k that on input the value k (in unary) generates $\tau = (T_{a,b,\beta}(\mathbb{F}_p), G_0, G_1)$ where G_0 is a generator of $E_0(\mathbb{F}_p)$ and G_1 is a generator of $E_1(\mathbb{F}_p)$ (these curves are defined in Subsection 2.1). Our results in this paper require that $\mathtt{IG}_1$ generate TDDH parameters in accordance with Section 4.

Definition 4. *A TDDH algorithm* $\mathtt{A}_1$ *for* TW_k *satisfies, for some fixed* $\alpha > 0$ *and sufficiently large* k*:*

$$|\Pr[\mathtt{A}_1(\tau,(a_0G_0,b_0G_0,a_0b_0G_0),(a_1G_1,b_1G_1,a_1b_1G_1)) = 1] - \Pr[\mathtt{A}_1(\tau,(a_0G_0,b_0G_0,c_0G_0),(a_1G_1,b_1G_1,c_1G_1)) = 1]| > \frac{1}{k^\alpha}$$

The probability is over the random choice of τ *according to the distribution induced by* $\mathrm{IG}_1(k)$*, the random choice of* $a_0,b_0,c_0 \in \{0,1,...,r_0-1\}$*, the random choice of* $a_1,b_1,c_1 \in \{0,1,2,...,r_1-1\}$*, and the bits used by* $\mathtt{A}_1$*.*

The twisted DDH assumption (TDDH) is that no such polytime $\mathtt{A}_1$ exists for TW_k. The *superpolynomial* TDDH assumption is that no such superpolynomial time $\mathtt{A}_1$ exists for TW_k.

Theorem 3 is straightforward to show. Being able to compute ECDDH for just one of the two curves in the twist breaks TDDH.

Theorem 3. *The TDDH problem polytime reduces to the ECDDH problem over* $(E_0(\mathbb{F}_p),G_0)$ *or the ECDDH problem over* $(E_1(\mathbb{F}_p),G_1)$*.*

Theorem 4. *The ECDDH problem over* $(E_0(\mathbb{F}_p),G_0)$ *or the ECDDH problem over* $(E_1(\mathbb{F}_p),G_1)$ *polytime reduces to TDDH.*

Proof. Suppose there exists a distinguisher $\mathtt{D}$ that solves TDDH. Both $E_0(\mathbb{F}_p)$ and $E_1(\mathbb{F}_p)$ are as defined in Subsection 2.1. Let the values t_0 and t_1 be ECDDH problem instances where $t_i = (E_i(\mathbb{F}_p),G_i,X_i,Y_i,Z_i)$ for $i=0,1$.

```
$\mathtt{M}_0(E_0(\mathbb{F}_p),G_0,X_0,Y_0,Z_0)$:
1. $u_0 \leftarrow \mathtt{f}(E_0(\mathbb{F}_p),G_0,X_0,Y_0,Z_0)$
2. generate a random 3-tuple $u_1$ over $(E_1(\mathbb{F}_p),G_1)$
3. output $(\tau,u_0,u_1)$
```

```
$\mathtt{M}_1(E_1(\mathbb{F}_p),G_1,X_1,Y_1,Z_1)$:
1. $u_1 \leftarrow \mathtt{f}(E_1(\mathbb{F}_p),G_1,X_1,Y_1,Z_1)$
2. generate a random DH triple $u_0$ over $(E_0(\mathbb{F}_p),G_0)$
3. output $(\tau,u_0,u_1)$
```

Clearly $\mathtt{M}_0$ and $\mathtt{M}_1$ run in time polynomial in k. Let $S_{i,DH}$ be the set of all DH triples over $(E_i(\mathbb{F}_p),G_i)$ for $i=0,1$. Let $S_{i,T}$ be the set of all 3-tuples over $(E_i(\mathbb{F}_p),G_i)$ for $i=0,1$.

Without loss of generality we may suppose that the TDDH distinguisher $\mathtt{D}$ outputs 1 with advantage δ_1 in k when both 3-tuples are DH triples and 0 with advantage δ_0 in k when both 3-tuples are random 3-tuples, where δ_1 and δ_0 are non-negligible. Observe that a slightly less powerful distinguisher can be used to construct $\mathtt{D}$, e.g., one in which δ_1 is non-negligible but δ_0 is negligible.

Consider the case that $v_0 \in_R S_{0,DH}$ and $v_1 \in_R S_{1,T}$. There are 3 cases:

Case 1: Consider the case that $\mathtt{D}(\tau,v_0,v_1)$ outputs 0 with probability $1/2 \pm \gamma(k)$ where the function γ is negligible. Let $d \leftarrow \mathtt{D}(\mathtt{M}_0(E_0(\mathbb{F}_p),G_0,X_0,Y_0,Z_0))$. $\mathtt{M}_0$ generates u_1 to be a random 3-tuple over $(E_1(\mathbb{F}_p),G_1)$. Suppose that (X_0,Y_0,Z_0)

is a DH triple. Then by the correctness of `f`, u_0 is a random DH triple. So, in this case $d = 0$ with probability $1/2 \pm \gamma(k)$ (negligible advantage). Suppose that (X_0, Y_0, Z_0) is not a DH triple. Then by the correctness of `f`, u_0 is a random 3-tuple. So, $d = 0$ with probability $1/2 + \delta_0(k)$ (non-negligible advantage). There is a polynomial time observable difference in behavior here. Therefore, $\mathtt{D}(\mathtt{M}_0(E_0(\mathbb{F}_p), G_0, X_0, Y_0, Z_0))$ solves the ECDDH problem over $(E_0(\mathbb{F}_p), G_0)$.

Case 2: Suppose that $\mathtt{D}(\tau, v_0, v_1)$ outputs 0 with probability $1/2 - \delta_2(k)$ and 1 with probability $1/2 + \delta_2(k)$ where the value δ_2 is non-negligible. Let $d \leftarrow \mathtt{D}(\mathtt{M}_0(E_0(\mathbb{F}_p), G_0, X_0, Y_0, Z_0))$. Machine $\mathtt{M}_0$ generates u_1 to be a random 3-tuple over $(E_1(\mathbb{F}_p), G_1)$. Suppose that (X_0, Y_0, Z_0) is a DH triple. Then by the correctness of `f`, u_0 is a random DH triple. So, in this case $d = 1$ with probability $1/2 + \delta_2(k)$. Suppose that (X_0, Y_0, Z_0) is not a DH triple. Then by the correctness of `f`, u_0 is a random 3-tuple. So, $d = 0$ with probability $1/2 + \delta_0(k)$. Therefore, $\mathtt{D}(\mathtt{M}_0(E_0(\mathbb{F}_p), G_0, X_0, Y_0, Z_0))$ solves ECDDH over $(E_0(\mathbb{F}_p), G_0)$.

Case 3: Suppose that $\mathtt{D}(\tau, v_0, v_1)$ outputs 0 with probability $1/2 + \delta_3(k)$ and 1 with probability $1/2 - \delta_3(k)$ where the value δ_3 is non-negligible. Let $d \leftarrow \mathtt{D}(\mathtt{M}_1(E_1(\mathbb{F}_p), G_1, X_1, Y_1, Z_1))$. Algorithm $\mathtt{M}_1$ generates u_0 to be a random DH triple over $(E_0(\mathbb{F}_p), G_0)$. Suppose that (X_1, Y_1, Z_1) is a DH triple. Then by the correctness of `f`, u_1 is a random DH triple. So, in this case $d = 1$ with probability $1/2 + \delta_1(k)$. Suppose that (X_1, Y_1, Z_1) is not a DH triple. Then by the correctness of `f`, u_1 is a random 3-tuple. So, $d = 0$ with probability $1/2 + \delta_3(k)$. Therefore, $\mathtt{D}(\mathtt{M}_1(E_1(\mathbb{F}_p), G_1, X_1, Y_1, Z_1))$ solves ECDDH over $(E_1(\mathbb{F}_p), G_1)$.

It follows that Theorem 5 holds (equivalence).

Theorem 5. *TDDH is polytime equivalent to ECDDH over $(E_0(\mathbb{F}_p), G_0)$ or ECDDH over $(E_1(\mathbb{F}_p), G_1)$*

C Completeness Proof

Definition 5. *ν is a negligible function if for every constant $c \geq 0$ there exists an integer k_c such that $\nu(k) < \frac{1}{k^c}$ for all $k \geq k_c$.*

The following is the proof of Lemma 2, namely, that failure occurs in Φ_1 with a probability that is negligible in k.

Proof. Let $p_1(k)$ denote the success probability of protocol Φ_1 having security parameter k. Let k_c be 64 (see Definition 5). $p_1(k) =$

$$(\tfrac{r_0-1}{r_0})^M * 1 * \tfrac{2p+2}{2^{k+1}} \tfrac{r_0}{2p+2} \tfrac{r_0-1}{r_0} + 1 * (\tfrac{r_1-1}{r_1})^M \tfrac{2p+2}{2^{k+1}} \tfrac{r_1}{2p+2} \tfrac{r_1-1}{r_1}$$

$$p_1(k) = (\tfrac{r_0-1}{r_0})^M \tfrac{r_0-1}{2^{k+1}} + (\tfrac{r_1-1}{r_1})^M \tfrac{r_1-1}{2^{k+1}}$$

Hasse showed that $|\#E_{a,b}(\mathbb{F}_p) - (p+1)| \leq 2\sqrt{p}$ for an elliptic curve $E_{a,b}(\mathbb{F}_p)$. So, $r_u - 1 \geq p - 2\sqrt{p}$ for $u = 0, 1$.

$$p_1(k) \geq ((\frac{r_0-1}{r_0})^M \frac{p-2\sqrt{p}}{2^{k+1}} + (\frac{r_1-1}{r_1})^M \frac{p-2\sqrt{p}}{2^{k+1}}) = \frac{p-2\sqrt{p}}{2^{k+1}}((\frac{r_0-1}{r_0})^M + (\frac{r_1-1}{r_1})^M)$$

From the Binomial Theorem it follows that,

$$(-\frac{1}{r_u}+1)^M = 1 - \frac{M}{r_u} + \sum_{\ell=2}^{M} \binom{M}{\ell}(-\frac{1}{r_u})^\ell$$

Observe that if M is even, then the last term in the summation above is positive. So, we can get rid of it and use an inequality. So, let $L = M$ if M is odd and set $L = M - 1$ if M is even. Then,

$$(-\frac{1}{r_u}+1)^M \geq 1 - \frac{M}{r_u} + \sum_{\ell=2}^{L} \binom{L}{\ell}(-\frac{1}{r_u})^\ell$$

$$(-\frac{1}{r_u}+1)^M \geq 1 - \frac{M}{r_u} + \sum_{\ell=1}^{\frac{L-1}{2}} (\frac{1}{r_u})^{2\ell}\binom{L}{2\ell}(1 - \frac{1}{r_u}\frac{L-2\ell}{2\ell+1})$$

Since $M \leq 10^4$ and $k \geq k_c = 64$ it follows from Hasse's Theorem that the term $1 - \frac{1}{r_u}\frac{L-2\ell}{2\ell+1} > 0$ for $u = 0, 1$ and $\ell = 1, 2, ..., (L-1)/2$. So,

$$(-\frac{1}{r_u}+1)^M \geq 1 - \frac{M}{r_u}$$

Recall that $p = 2^k - \delta$ and $1 \leq \delta < \sqrt{2^k}$. So, $p - 2\sqrt{p} > 2^k - 3 * 2^{k/2}$. Since $r_0, r_1 > p - 2\sqrt{p}$,

$$p_1(k) \geq \frac{p-2\sqrt{p}}{2^{k+1}}(2 - \frac{2M}{p-2\sqrt{p}}) > \frac{2^k-3*2^{k/2}}{2^{k+1}}(2 - \frac{2M}{p-2\sqrt{p}}) = (1 - \frac{3}{2^{k/2}})(1 - \frac{M}{p-2\sqrt{p}})$$

It follows from Definition 5 that the failure probability is negligible in k.

D Security Proof

Algorithm $\mathtt{f}_1$ chooses scalars u_1, u_2, v randomly satisfying the inequality $0 \leq u_1, u_2, v \leq r-1$. However, unlike $\mathtt{f}$, algorithm $\mathtt{f}_1$ gives the additional output u_2. $\mathtt{f}_1(E_{a,b}(\mathbb{F}_p), G, \mathcal{X}, \mathcal{Y}, \mathcal{Z}) = (v\mathcal{X} + u_1 G, \mathcal{Y} + u_2 G, v\mathcal{Z} + u_1\mathcal{Y} + vu_2\mathcal{X} + u_1u_2G, u_2)$ Algorithm $\mathtt{f}_2$ chooses scalars u_1, v randomly satisfying the inequality $0 \leq u_1, v \leq r-1$. Algorithm $\mathtt{f}_2(E_{a,b}(\mathbb{F}_p), G, \mathcal{X}, \mathcal{Y}, \mathcal{Z}, u_2)$ returns the following tuple $(v\mathcal{X} + u_1G, \mathcal{Y} + u_2G, v\mathcal{Z} + u_1\mathcal{Y} + vu_2\mathcal{X} + u_1u_2G)$.

Let $(X, Y, Z) = \mathtt{f}_2(E_{a,b}(\mathbb{F}_p), G, \mathcal{X}, \mathcal{Y}, \mathcal{Z}, u_2)$. We partition the set of possible inputs to $\mathtt{f}_2$ into two sets, DH triples and non-DH triples.

Suppose that $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ is a DH triple. Let $\mathcal{X} = xG$, $\mathcal{Y} = yG$ and $\mathcal{Z} = xyG$. So, $X = (vx + u_1)G$, $Y = (y + u_2)G$, and,

$$Z = (vxy + u_1y + vu_2x + u_1u_2)G = (vx + u_1)(y + u_2)G$$

It follows that (X, Y, Z) is $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ transformed as follows. The scalar y is replaced by $y + u_2$ and the scalar x is replaced by the random scalar $vx + u_1$. So, (X, Y, Z) is a DH triple. We say that such a DH triple is a *one-scalar randomized* DH triple of $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$.

Now suppose that $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ is not a DH triple. Then $X, Z \in_R E_{a,b}(\mathbb{F}_p)$. This claim needs justification. Let $\mathcal{Z} = zG$. Observe that $z = xy + c$ for some scalar

$c > 0$. So, $Z = (vx + u_1)(y + u_2)G + v(cG)$. Define $X' = (vx + u_1)G$, $Y' = (y + u_2)G$, and $Z' = (vx + u_1)(y + u_2)G$. Then $(X, Y, Z) = (X', Y', Z' + v(cG))$ and cG is a generator since $c > 0$.

We now consider an "exchange" Φ_2 in which Bob really has no hope of recovering m_k (we will show why later on). Let Φ_2 be the same as Φ_1 except that $\texttt{ExchAlg}_1$ is replaced with $\texttt{ExchAlg}_2$.

```
ExchAlg_2($\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}$):
Input: TDDH parameters $\tau$, points $Y_{i,j}$ on
    $E_i(\mathbb{F}_p)$ for $i = 0, 1$, $j = 1, 2, ..., M$
Output: $(m_A, m_k) \in \{0,1\}^{k+1} \times \{0,1\}^M$
1. $(y, s_1, s_2) \leftarrow$ FillGap($T_{a,b,\beta}(\mathbb{F}_p)$)
2. if $(y = 1)$ then output $(s_1, s_2)$ and halt
3. $u \leftarrow$ SelCurve($T_{a,b,\beta}(\mathbb{F}_p)$)
4. choose $k_1 \in_R \mathbb{Z}_{r_u}$ and compute $U \leftarrow k_1 G_u$
5. $m_A \leftarrow$ Encode($T_{a,b,\beta}(\mathbb{F}_p), U, u$)
6. for $j = 1$ to $M$ do:
7.     choose $P_j \in_R E_u(\mathbb{F}_p)$ and compute
           $b_j \leftarrow$ DeriveBit($T_{a,b,\beta}(\mathbb{F}_p), P_j$)
8. set $m_k \leftarrow b_M || b_{M-1} || ... || b_1$ and output $(m_A, m_k)$
```

Lemma 3. *In algorithm* $\texttt{ExchAlg}_2$, $m_A \in_R \{0,1\}^{k+1}$.

Proof. Consider the operation of $\texttt{ExchAlg}_2$. Let s be any string in $\{0,1\}^{k+1} \setminus \mathcal{S}_{a,b,\beta,p}$. Then it follows from the definitions of algorithm $\texttt{FillGap}$ and algorithm $\texttt{Format}$ that $\Pr[m_A = s] = \frac{2\delta-2}{2p+2}\frac{1}{2\delta-2}$. Let P be any point on $E_u(\mathbb{F}_p)$. Then it follows from the definitions of $\texttt{FillGap}$, $\texttt{SelCurve}$, and $\texttt{Encode}$ that the probability that m_A is the Kaliski encoding of P is $\frac{2p+2}{2^{k+1}}\frac{r_u}{2p+2}\frac{1}{r_u}$. It follows that each string contained in $\{0,1\}^{k+1}$ is selected by $\texttt{ExchAlg}_2$ and output as m_A with probability $\frac{1}{2^{k+1}}$.

Fact 1: The following is from the Group Law for $E_{a,b}(\mathbb{F}_p)$ when the prime $p \neq 2, 3$. *Negatives*: If $P = (x, y) \in E_{a,b}(\mathbb{F}_p)$ then $(x, y) + (x, -y) = \mathcal{O}$. The point $(x, -y)$ is denoted by $-P$ and is referred to as the *negative* of P. So, $-P$ is in fact a point on $E_{a,b}(\mathbb{F}_p)$. It is also the case that $-\mathcal{O} = \mathcal{O}$.

Lemma 4. *If* $\#E_{a,b}(\mathbb{F}_p)$ *is odd then there are no points with an ordinate of zero on* $E_{a,b}(\mathbb{F}_p)$.

Proof. Let $f(x) = x^3 + ax + b$ with coefficients in $\mathbb{F}_p$. It is well-known that $f(x) = 0$ has 0 solutions, 1 solution x_1, or 3 solutions x_2, x_3, x_4 in $\mathbb{F}_p$. We consider these 3 cases in turn.

Case 1: There are 0 solutions. Let $\mathcal{S} = E_{a,b}(\mathbb{F}_p) \setminus \{\mathcal{O}\}$. Then $|\mathcal{S}| = \#E_{a,b}(\mathbb{F}_p) - 1$. From Fact 1, all points in $\mathcal{S}$ have the following property: $(x, y) \in \mathcal{S} \Leftrightarrow (x, -y) \in \mathcal{S}$ with $(x, y) \neq (x, -y)$. We now make a partitioning argument. Consider the following two sets.

$$\mathcal{S}_0 = \{(x, y) : (x, y) \in \mathcal{S}, y < p/2\} \qquad \mathcal{S}_1 = \{(x, y) : (x, y) \in \mathcal{S}, y > p/2\}$$

Clearly $\mathcal{S} = \mathcal{S}_0 \bigcup \mathcal{S}_1$, $\mathcal{S}_0 \bigcap \mathcal{S}_1 = \emptyset$, and $|\mathcal{S}_0| = |\mathcal{S}_1|$. So, $|\mathcal{S}| = 2|\mathcal{S}_0|$. It follows that $|\mathcal{S}|$ is even, therefore $\#E_{a,b}(\mathbb{F}_p)$ is odd. This supports our claim.

Case 2: There is one solution x_1 and therefore $(x_1, 0)$ is the only point on $E_{a,b}(\mathbb{F}_p)$ with an ordinate of zero. Let $\mathcal{S} = E_{a,b}(\mathbb{F}_p) \setminus \{\mathcal{O}, (x_1, 0)\}$. Then $|\mathcal{S}| = \#E_{a,b}(\mathbb{F}_p) - 2$. From Fact 1, all points in $\mathcal{S}$ have the following property: $(x, y) \in \mathcal{S} \Leftrightarrow (x, -y) \in \mathcal{S}$ with $(x, y) \neq (x, -y)$. Using the partitioning argument it follows that $|\mathcal{S}|$ is even and therefore $\#E_{a,b}(\mathbb{F}_p)$ is even.

Case 3: The integers x_2, x_3, x_4 are solutions to $f(x) = 0$ and therefore the points $(x_2, 0), (x_3, 0), (x_4, 0)$ are the only points with an ordinate of zero on $E_{a,b}(\mathbb{F}_p)$. Let $\mathcal{S} = E_{a,b}(\mathbb{F}_p) \setminus \{\mathcal{O}, (x_2, 0), (x_3, 0), (x_4, 0)\}$. Then $|\mathcal{S}| = \#E_{a,b}(\mathbb{F}_p) - 4$. By a similar argument, $\#E_{a,b}(\mathbb{F}_p)$ is even.

Lemma 5. *For $u = 0, 1$, the ordered execution of, $P_j \in_R E_u(\mathbb{F}_p)$, followed by $b_j \leftarrow$ `DeriveBit`$(T_{a,b,\beta}(\mathbb{F}_p), P_j)$ causes b_j to be a fair coin flip.*

Proof. Let u be any element in $\{0, 1\}$. Since r_u is odd it follows from Lemma 4 that there are no points with an ordinate of zero on $E_u(\mathbb{F}_p)$. So, aside from the point at infinity, if $(x, y) \in E_u(\mathbb{F}_p)$ then $(x, -y) \in E_u(\mathbb{F}_p)$ where $y \neq 0$. It follows from the definition of Kaliski's X_T function that $E_u(\mathbb{F}_p) \setminus \{\mathcal{O}_u\}$ contains exactly $\frac{r_u-1}{2}$ points with Kaliski encodings that have an LSB of 0. Similarly, $E_u(\mathbb{F}_p) \setminus \{\mathcal{O}_u\}$ contains exactly $\frac{r_u-1}{2}$ points with Kaliski encodings that have an LSB of 1. `DeriveBit` returns a fair coin flip on input $\mathcal{O}_u$. So, $\Pr[b_j = 1] = \frac{\frac{r_u-1}{2}}{r_u} + \frac{1}{r_u}\frac{1}{2} = \frac{1}{2}$.

Lemma 6. *In algorithm `ExchAlg`$_2$, $m_k \in_R \{0, 1\}^M$.*

Proof. Consider algorithm `FillGap`. Clearly, $\Pr[\mu \notin \mathcal{S}_{a,b,\beta,p}] = \frac{2\delta-2}{2^{k+1}}$. If $\mu \notin \mathcal{S}_{a,b,\beta,p}$ then it follows from the definition of `FillGap` that $s_2 = m_k$ is chosen randomly from $\{0, 1\}^M$. Also, $\Pr[\mu \in \mathcal{S}_{a,b,\beta,p}] = \frac{2p+2}{2^{k+1}}$. If $\mu \in \mathcal{S}_{a,b,\beta,p}$ then from Lemma 5, `ExchAlg`$_2$ chooses $m_k \in_R \{0, 1\}^M$. Either $\mu \in \mathcal{S}_{a,b,\beta,p}$ or not and in both cases it follows that $m_k \in_R \{0, 1\}^M$.

Algorithm `InstTrans` transforms an input TDDH problem instance into a tuple that looks like it is from Φ_1 or from Φ_2. `InstTrans` is used in several proofs in this paper.

```
InstTrans($\tau, t_0, t_1$):
Input: TDDH problem instance $(\tau, t_0, t_1)$ where
    $t_i = (\mathcal{X}^{(i)}, \mathcal{Y}^{(i)}, \mathcal{Z}^{(i)})$ are points on $E_i(\mathbb{F}_p)$ for $i = 0, 1$
Output: $(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$
1. $(y, s_1, s_2) \leftarrow$ FillGap($T_{a,b,\beta}(\mathbb{F}_p)$)
2. if $(y = 1)$ then
3.     choose $Y_{i,j} \in_R E_i(\mathbb{F}_p)$ for $i = 0, 1$, $j = 1, 2, ..., M$
4.     $(m_A, m_k) \leftarrow (s_1, s_2)$
5.     halt with $(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$
6. $u \leftarrow$ SelCurve($T_{a,b,\beta}(\mathbb{F}_p)$)
```

```
7. $(X, Y, Z, u_2) \leftarrow \texttt{f}_1(E_u(\mathbb{F}_p), G_u, \mathcal{X}^{(u)}, \mathcal{Y}^{(u)}, \mathcal{Z}^{(u)})$
8. $m_A \leftarrow \texttt{Encode}(T_{a,b,\beta}(\mathbb{F}_p), Y, u)$
9. for $j = 1$ to $M$ do:
10.    $(X_j, Y_j, Z_j) \leftarrow \texttt{f}_2(E_u(\mathbb{F}_p), G_u, \mathcal{X}^{(u)}, \mathcal{Y}^{(u)}, \mathcal{Z}^{(u)}, u_2)$
11.    set $Y_{u,j} \leftarrow X_j$ and choose $Y_{1-u,j} \in_R E_{1-u}(\mathbb{F}_p)$
12.    set $P_j \leftarrow Z_j$ and $b_j \leftarrow \texttt{DeriveBit}(T_{a,b,\beta}(\mathbb{F}_p), P_j)$
13. $m_k \leftarrow b_M || b_{M-1} || ... || b_1$
14. output $(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$
```

Lemma 7. *TDDH polytime reduces to the problem of distinguishing $\mathcal{T}_{\tau,\Phi_1}$ from $\mathcal{T}_{\tau,\Phi_2}$ with an advantage that is non-negligible (in k).*

Proof. Suppose there exists an algorithm D that distinguishes $\mathcal{T}_{\tau,\Phi_1}$ from $\mathcal{T}_{\tau,\Phi_2}$ with an advantage that is non-negligible in k. Consider the polytime (in k) algorithm InstTrans that takes as input a problem instance (τ, t_0, t_1) for TDDH.

Suppose that both t_0 and t_1 are DH triples. From the correctness of $\texttt{f}_1$ and $\texttt{f}_2$, the (X_j, Y_j, Z_j) for $j = 1, 2, ..., M$ in algorithm InstTrans are all one-scalar randomized DH triples of the input Diffie-Hellman triple $(\mathcal{X}^{(u)}, \mathcal{Y}^{(u)}, \mathcal{Z}^{(u)})$ where $u \in \{0, 1\}$ is selected using SelCurve in algorithm InstTrans. So, the output of algorithm InstTrans given by the tuple $(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$ is drawn from the same set and probability distribution as in Φ_1.

Suppose t_0 and t_1 are not DH triples. From the correctness of $\texttt{f}_1$ and $\texttt{f}_2$, $(X_j, Z_j) \in_R E_u(\mathbb{F}_p) \times E_u(\mathbb{F}_p)$ for $j = 1, 2, ..., M$. So, the output of algorithm InstTrans given by the tuple $(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M}, m_A, m_k)$ is drawn from the same set and probability distribution as in Φ_2.

Therefore, D(InstTrans(τ, t_0, t_1)) solves TDDH.

Let protocol Φ_3 be the same as protocol Φ_2 except that algorithm $\texttt{ExchAlg}_2$ is replaced with algorithm $\texttt{ExchAlg}_3$.

```
$\texttt{ExchAlg}_3(\tau, Y_{0,1}, ..., Y_{0,M}, Y_{1,1}, ..., Y_{1,M})$:
Input: TDDH parameters $\tau$, points $Y_{i,j}$ on
    $E_i(\mathbb{F}_p)$ for $i = 0, 1$, $j = 1, 2, ..., M$
Output: $(m_A, m_k) \in \{0,1\}^{k+1} \times \{0,1\}^M$
1. choose $m_A \in_R \{0,1\}^{k+1}$ and $m_k \in_R \{0,1\}^M$
2. output $(m_A, m_k)$
```

Lemma 8. *$\mathcal{T}_{\tau,\Phi_2}$ is perfectly indistinguishable from $\mathcal{T}_{\tau,\Phi_3}$.*

Lemma 9. *TDDH polytime reduces to the problem of distinguishing $\mathcal{T}_{\tau,\Phi_1}$ from $\mathcal{T}_{\tau,U}$ with an advantage that is non-negligible (in k).*

Lemma 8 follows from Lemmas 3, 6. But, $\mathcal{T}_{\tau,U} = \mathcal{T}_{\tau,\Phi_3}$. So, lemmas 7 and 8 give Lemma 9. Theorem 5 and Lemma 9 imply Theorem 1.