Compatible Certificateless and Identity-Based Cryptosystems for Heterogeneous IoT

Rouzbeh Behnia[1], Attila Altay Yavuz[1], Muslum Ozgur Ozmen[2*], and Tsz Hon Yuen[3]

[1] University of South Florida, Tampa, Florida, USA
{behnia,attilaayavuz}@usf.edu
[2] Purdue University, West Lafayette, Indiana, USA
mozmen@purdue.edu
[3] The University of Hong Kong, Pokfulam, Hong Kong
thyuen@cs.hku.hk

Abstract. Certificates ensure the authenticity of users' public keys, however their overhead (e.g., certificate chains) might be too costly for some IoT systems like aerial drones. Certificate-free cryptosystems, like identity-based and certificateless systems, lift the burden of certificates and could be a suitable alternative for such IoTs. However, despite their merits, there is a research gap in achieving compatible identity-based and certificateless systems to allow users from different domains (identity-based or certificateless) to communicate seamlessly. Moreover, more efficient constructions can enable their adoption in resource-limited IoTs. In this work, we propose new identity-based and certificateless cryptosystems that provide such compatibility and efficiency. This feature is beneficial for heterogeneous IoT settings (e.g., commercial aerial drones), where different levels of trust/control is assumed on the trusted third party. Our schemes are more communication efficient than their public key based counterparts, as they do not need certificate processing. Our experimental analysis on both commodity and embedded IoT devices show that, only with the cost of having a larger system public key, our cryptosystems are more computation and communication efficient than their certificate-free counterparts. We prove the security of our schemes (in the random oracle model) and open-source our cryptographic framework for public testing/adoption.

Keywords: Identity-based cryptography · certificateless cryptography · IoT Systems · lightweight cryptography

1 Introduction

Mobile and heterogeneous IoT applications harbor large quantities of resource-limited and non-stationary IoT devices, each with different capabilities, configurations, and user domains. For instance, emerging commercial aerial drone

*Work done in part when Muslum Ozgur Ozmen was at University of South Florida.

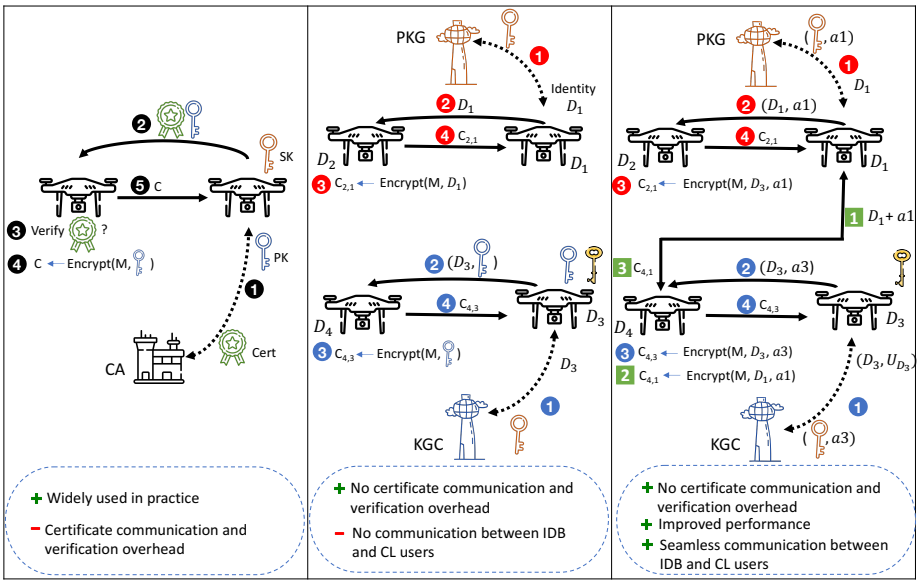
network protocols[4] need a near real-time communication and processing over a bandwidth-limited network. There are multiple hurdles of relying on traditional PKI for such systems: (i) The maintenance of PKI for such IoT networks demands a substantial infrastructure investment [25]. (ii) PKI requires transmission and verification of certificate chains at the sender's/verifier's side. This communication and computation overhead could create a major bottleneck for mobile IoT devices (e.g., aerial drones [21]) that potentially need to interact with a number of devices. In certain cases, these certificate chains might be larger than the actual measurements/commands being transmitted and therefore, might be the dominating cost for these applications. Figure 1-a depicts a high-level illustration of traditional PKI for mobile IoT applications.

Identity-based (IDB) and certificateless (CL) cryptosystems offer implicit certification [1,25,9], and therefore can mitigate the aforementioned hurdles. In IDB, the user's public key is derived from their identifying information, and the system relies on a fully-trusted third party (TTP), called the private key generator (PKG), to issue users' private keys. The top portion of Figure 1-b depicts IDB encryption, wherein the user authenticates itself to the PKG and receives a private key corresponding to its identity D_1. The sender can use D_1 as the public key to run encryption. IDB is potentially suitable for applications where the system setup is done and managed by a trusted centralized entity. In CL systems [1] the trust on the TTP is lowered by allowing the private key of the user to consist of two parts. One is computed by the user and the other is by the TTP (called the KGC). The bottom portion of Figure 1-b outlines CL encryption, where the user computes its key pair and then works as in IDB to receive the other part of the private key from the KGC. CL cryptosystems are suitable for architectures that might not assume a fully trusted third party where the trust level on the KGC is similar to traditional certification authorities.

IDB and CL cryptosystems have their own merits and drawbacks, and therefore might be used in different IoT applications. Hence, it is expected that there will be different user groups who rely on IDB and CL cryptosystems initiated in different domains/systems. For example, Amazon's Prime Air[5] would require drones, under the complete control of Amazon, to interact with other drones (e.g., personal) to ensure safe operation. By employing IDB cryptography on its drones, Amazon can have complete control over the operations of its delivery drones while avoiding the overheads of traditional PKI. However, it is a strong assumption that other drones, outside Amazon's network, will adopt a similar cryptographic setting to ensure safe and secure operations. For instance, personal users rarely trust any third party to have complete control and knowledge of their drones' activity. To the best of our knowledge, there is a significant research gap in enabling a seamless communication between users who are registered under different domains (e.g., IDB and CL). This is a potential obstacle to widely deploy efficient certificate-free solutions in heterogeneous environments. This limitation is mentioned in Figure 1-b. Moreover, it is important to further

[4] https://github.com/mavlink/mavlink
[5] https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

Fig. 1: Proposed IDB and CL Cryptosystems and Alternatives (High-Level)



improve the computational efficiency of IDB and CL techniques to offer a low end-to-end delay that is needed by delay-aware IoT applications.

Our Contribution. We propose a new series of public key encryption, digital signature, and key exchange schemes that permit users from different domains (IDB or CL) to communicate seamlessly. To our knowledge, this is the first set of certificate-free cryptosystems that achieve such compatibility and efficiency, and therefore a suitable alternative for resource-limited IoT systems such as commercial aerial drones. The idea behind our constructions is to create special key generation algorithms that harness the additive homomorphic property of the exponents and cover-free functions to enable the users to incorporate their private keys into the one provided by the TTP without falsifying it. As detailed in Section 4, this special design is applicable across our IDB and CL algorithms, and therefore it permits a seamless communication between our IDB and CL cryptosystems. This strategy also reduces the cost of online operations and enables our schemes to achieve a lower end-to-end delay compared to their counterparts. We elaborate on some desirable properties of our schemes as below.

- Compatible IDB and CL Schemes: Fig 1.c outlines the concept of *compatible* IDB and CL schemes where the users from different domains (and trust-levels) can use identical encryption, signature, and key exchange algorithms to communicate without any additional overhead.
- Computation & Communication Efficiency: Based on our analysis, new schemes offer performance advantages over their counterparts: (i) Similar to other IDB/CL cryptosystems, our schemes lift the hurdle of certificate transmission and verification, and therefore offer significant communication efficiency over some

of the most efficient PKI-based schemes. This advantage grows proportional to the size of the certificate chain. (ii) Our schemes outperform their certificate-free counterparts on the vast majority of the performance metrics. For instance, the end-to-end delay in our IDB/CL encryption schemes is $\approx 25\%$ lower than our most efficient counterpart in [29]. Our signature schemes achieve up to %52 faster end-to-end delay as compared to our counterparts. We also achieve a 65% lower end-to-end delay for our key exchange schemes.

• Open-Sourced Implementation: We implemented our schemes on a commodity hardware and an 8-bit AVR microprocessor, and compared their performance with a variety of their counterparts capturing some of the most efficient traditional PKI, IDB and CL schemes (see Section 6 for details). We open-source our implementations for broad testing, benchmarking, and adoption purposes.

2 Preliminaries

Notation. Given two primes p and q, we define a finite field $\mathbb{F}_p$ and a group $\mathbb{Z}_q$. We work on $E(\mathbb{F}_p)$ as an elliptic curve (EC) over $\mathbb{F}_p$, where $P \in E(\mathbb{F}_p)$ is the generator of the points on the curve. We denote a scalar and a point on a curve with small and capital letters, respectively. $x \xleftarrow{\$} S$ denotes a random uniform selection of x from a set S. We define the bit-length of a variable as $|x|$ (i.e., $|x| = \log_2 x$). EC scalar multiplication is denoted as xP, and all EC operations use an additive notation. Hash functions are $\mathtt{H}_1$: $E(\mathbb{F}_p) \times E(\mathbb{F}_p) \rightarrow \{0,1\}^\gamma$, $\mathtt{H}_2$: $\{0,1\}^n \times \{0,1\}^* \rightarrow \mathbb{Z}_q, \mathtt{H}_3$: $E(\mathbb{F}_p) \rightarrow \{0,1\}^n$ $\mathtt{H}_4$: $\{0,1\}^n \rightarrow \{0,1\}^n$ and $\mathtt{H}_5$: $\{0,1\}^n \times E(\mathbb{F}_p) \rightarrow \mathbb{Z}_q$, where all hash functions are random oracles [6]. FourQ [12] is a special EC that is defined by the complete twisted Edwards equation $\mathcal{E}/F_{p^2} : -x^2+y^2 = 1+dx^2y^2$. FourQ is known to be one of the fastest elliptic curves that admits 128-bit security level [12]. Moreover, with extended twisted Edwards coordinates, FourQ offers the fastest EC addition algorithms [12], that is extensively used in our optimizations. All of our schemes are realized on FourQ.

Definitions. We first give our intractability assumptions followed by the definitions of identity-based and certificateless encryption and signature schemes.

Definition 1. *Given points $P, Q \in E(\mathbb{F}_p)$, the Elliptic Curve Discrete Logarithm Problem (ECDLP) asks to find a, if it exists, such that $aP(\mathrm{mod}\, p) = Q$.*

Definition 2. *Given $P, aP, bP \in E(\mathbb{F}_p)$, the Computational Diffie-Hellman Problem (CDHP) asks to compute abP.*

Definition 3. *An identity-based encryption scheme is consisted of four algorithms* $\mathtt{IBE} = \{\mathtt{Setup}, \mathtt{Extract}, \mathtt{Enc}, \mathtt{Dec}\}$.
$(msk, params) \leftarrow \mathtt{IBE.Setup}(1^\kappa)$*: Given the security parameter κ, the PKG selects master secret key msk, computes master public key mpk and system parameters $params$ (an implicit input to all the following algorithms).*
$(sk_{ID}, Q_{ID}) \leftarrow \mathtt{IBE.Extract}(ID, msk)$: *Given an identity ID and msk, the PKG computes the commitment value Q_{ID} and the private key sk_{ID}.*

$c \leftarrow \texttt{IBE.Enc}(m, ID, Q_{ID})$: *Given a message* m *and* (ID, Q_{ID})*, the sender computes the ciphertext* c.
$m \leftarrow \texttt{IBE.Dec}(sk_{ID}, c)$: *Given the ciphertext* c *and the private key of the receiver* sk_{ID}*, the receiver returns either the corresponding plaintext* m *or* $\perp$ *(invalid).*

Definition 4. *An identity-based signature scheme is defined by four algorithms* $\texttt{IBS} = \{\texttt{Setup}, \texttt{Extract}, \texttt{Sig}, \texttt{Ver}\}$.
$(msk, mpk, params) \leftarrow \texttt{IBS.Setup}(1^{\kappa})$: *As in* `IBE.Setup` *in Definition 3.*
$(sk_{ID}, Q_{ID}) \leftarrow \texttt{IBS.Extract}(ID, msk)$: *As in* `IBE.Extract` *in Definition 3.*
$\sigma \leftarrow \texttt{IBS.Sign}(m, sk_{ID})$: *Given a message* m *and* sk_{ID}*, returns a signature* σ.
$d \leftarrow \texttt{IBS.Verify}(m, ID, Q_{ID}, \sigma)$: *Given* m*,* σ *and* (ID, Q_{ID}) *as input, if the signature is valid, it returns* $d = 1$*, else* $d = 0$.

Definition 5. *A certificateless encryption scheme is defined by six algorithms* $\texttt{CLE} = \{\texttt{KGCSetup}, \texttt{UserSetup}, \texttt{PartKeyGen}, \texttt{UserKeyGen}, \texttt{Enc}, \texttt{Dec}\}$.
$(msk, mpk, params) \leftarrow \texttt{CLE.KGCSetup}(1^{\kappa})$: *Give the security parameter* κ*, the KGC generates master secret key* msk*, master public key* mpk *and the system parameters* $params$ *(an implicit input to all the following algorithms).*
$(\alpha, U) \leftarrow \texttt{CLE.UserSetup}(\cdot)$: *The user* ID *computes her secret value* α *and its corresponding commitment* U.
$(w, Q_{ID}) \leftarrow \texttt{CLE.PartKeyGen}(ID, U, msk)$: *Given* ID*,* U*, and* msk *, the KGC computes partial private key* w *and its corresponding public commitment* Q_{ID}.
$x_{ID} \leftarrow \texttt{CLE.UserKeyGen}(w, \alpha)$: *Given* (w, α)*, the user* ID *computes* x_{ID}.
$c \leftarrow \texttt{CLE.Enc}(m, ID, Q_{ID})$: *Given* (m, ID, Q_{ID})*, sender computes ciphertext* c.
$m' \leftarrow \texttt{CLE.Dec}(x_{ID}, c)$: *Given the ciphertext* c *and the private key of the receiver* x_{ID}*, the receiver returns either corresponding plaintext* m *or* $\perp$ *(invalid).*

Definition 6. *A certificateless signature scheme is defined by six algorithms* $\texttt{CLS} = \{\texttt{KGCSetup}, \texttt{UserSetup}, \texttt{PartKeyGen}, \texttt{UserKeyGen}, \texttt{Sig}, \texttt{Ver}\}$. The definition of algorithms are as in Definition 5 except for `(CLS.Sig,CLS.Ver)`.
$\sigma \leftarrow \texttt{CLS.Sig}(m, x_{ID})$: Given a message m, and the signer's private key x_{ID}, it returns a signature σ.
$d \leftarrow \texttt{CLS.Ver}(m, ID, Q_{ID}, \sigma)$: Given m, σ and (ID, Q_{ID}) as input, if the signature is valid, it returns $d = 1$, else $d = 0$.

3 Security Model

The security model of identity-based schemes is slightly stronger than those for traditional PKI based schemes. More specifically, the adversary can query for the private key of any user ID, except for the target user ID^*. In this paper, we constructed our schemes by following the security model of Identity-based systems proposed in [9]. In certificateless systems, the private key of the users consists of two parts: (i) user secret key α, which is selected by the user, and (ii) partial private key w, which is supplied to the user by the KGC. Therefore, following [1], it is natural to consider two types of adversaries for such systems.

A Type-I adversary $\mathcal{A}_I$ does not have access to msk or the user's partial private key w but is able to replace any user's public key U with public key of its choice U'. However, in our security model, since we adopt the binding method [1], replacing the public key will result in falsifying the partial private key (and evidently the private key). Therefore, following [2], we allow $\mathcal{A}_I$ to query for the secret key of the user via $\alpha \leftarrow \mathcal{O}_{\texttt{SecKey}}(ID)$. Note that our model can also be extended to allow $\mathcal{A}_I$ to replace the public key of the user (see Section 5). A Type-II adversary $\mathcal{A}_{II}$ is assumed to be a malicious KGC. Having knowledge on msk, $\mathcal{A}_{II}$ can query the partial private key of the user via $w \leftarrow \mathcal{O}_{\texttt{PartKey}}(ID)$. Following [1], we allow the adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ to extract private key of users' private keys via the $x_{ID} \leftarrow \mathcal{O}_{\texttt{Corrupt}}(ID)$. We note that inspired by [3], many improvements on the security models of certificateless systems have been suggested (e.g., [2,3,17]). In this paper, we provide our proof in the original model proposed in [1,3], but note that many of those stronger security requirements can be enforced if needed.

Definition 7. *The indistinguishability of a* ***CLE*** *under chosen ciphertext attack (IND-CLE-CCA) experiment* $Expt_{\mathcal{A}}^{IND\text{-}CLE\text{-}CCA}$ *is defined as follows.*

- $\mathcal{C}$ runs $\texttt{CLE.KGCSetup}(1^\kappa)$ and returns mpk and $params$ to $\mathcal{A}$.
- $(ID^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\texttt{PartKey}}, \mathcal{O}_{\texttt{SecKey}}, \mathcal{O}_{\texttt{Corrupt}}, \mathcal{O}_{\texttt{Dec}}}(mpk, params)$
- $\mathcal{C}$ picks $b \xleftarrow{\$} \{0,1\}$, $c_b \leftarrow \texttt{CLE.Enc}(m_b, ID^*, params)$ and returns c_b to $\mathcal{A}$.
- $\mathcal{A}$ performs the second series of queries, with a restriction of querying ID^* or c_b to $\texttt{Corrupt}(\cdot)$ or $\texttt{CLE.Dec}(\cdot)$, respectively. Finally, $\mathcal{A}$ outputs a bit b'.

$\mathcal{A}$ wins the above experiment if $b = b'$ and the following conditions hold: (i) ID^* was never submitted to $\mathcal{O}_{\texttt{Corrupt}}$. (ii) If $\mathcal{A} = \mathcal{A}_I$, ID^* was never submitted to $\mathcal{O}_{\texttt{PartKey}}$. (iii) If $\mathcal{A} = \mathcal{A}_{II}$, ID^* was never submitted to $\mathcal{O}_{\texttt{SecKey}}$. The IND-CLE-CCA advantage of $\mathcal{A}$ is $\Pr[b = b'] \leq \frac{1}{2} + \epsilon$, for a negligible ϵ.

Definition 8. *The existential unforgeability under chosen message attack (EU-CLS-CMA) experiment* $Expt_{\mathcal{A}}^{EU\text{-}CLS\text{-}CMA}$ *for a certificateless signature* ***CLS*** *is defined as follows.*

- $\mathcal{C}$ runs $\texttt{CLS.KGCSetup}(1^\kappa)$ and returns mpk and $params$ to $\mathcal{A}$.
- $(ID^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\texttt{PartKey}}, \mathcal{O}_{\texttt{SecKey}}, \mathcal{O}_{\texttt{Corrupt}}, \mathcal{O}_{\texttt{Sign}}}(mpk, params)$

$\mathcal{A}$ wins the above experiment if $1 \leftarrow \texttt{CLS.Ver}(m^*, \sigma^*, ID)$, and the following conditions hold: (i) ID^* was never submitted to $\mathcal{O}_{\texttt{Corrupt}}$. (ii) If $\mathcal{A} = \mathcal{A}_I$, ID^* was never submitted to $\mathcal{O}_{\texttt{PartKey}}$. (iii) If $\mathcal{A} = \mathcal{A}_{II}$, ID^* was never submitted to $\mathcal{O}_{\texttt{SecKey}}$. The EU-CLS-CMA advantage of $\mathcal{A}$ is $\Pr[Expt_{\mathcal{A}}^{EU\text{-}CLS\text{-}CMA} = 1]$

4 Proposed Schemes

4.1 Proposed Identity-Based Cryptosystem

Most of pairing-free IDB schemes rely on the classical signatures (e.g., [24]) in their key generation to provide implicit certification. The use of such signatures

Algorithm 1 Identity-Based Encryption

$(msk, params) \leftarrow$ `IBE.Setup`(1^κ):

1: Select primes p and q and $(t, k) \in \mathbb{N}$ where $t >> k$.
2: **for** $i = 1, \ldots, t$ **do**
3: $\quad v_i \xleftarrow{\$} \mathbb{Z}_q,\ V_i \leftarrow v_i P \bmod p$
4: **return** $msk \leftarrow (v_1, \ldots, v_t)$, $mpk \leftarrow (V_1, \ldots, V_t)$ and $params \leftarrow (\mathtt{H}_1, \mathtt{H}_2, \mathtt{H}_3, \mathtt{H}_4, p, q, k, t, mpk)$

$(w, Q) \leftarrow$`IBE.Extract`(ID, U, msk):

1: $\beta \xleftarrow{\$} \mathbb{Z}_q,\ Q \leftarrow \beta P \bmod p$
2: $(j_1, \ldots, j_k) \leftarrow \mathtt{H}_1(ID, Q)$ where for all $i = 1, \ldots, t$, $1 < j_i < |t|$
3: $y \leftarrow \sum_{i=1}^{k} v_{j_i} \bmod q$
4: $x \leftarrow y + \beta \bmod q$
5: **return** (x, Q)

$c \leftarrow$ `IBE.Enc`(m, ID_a, Q_a): Bob encrypts message $m \in \{0,1\}^n$.

1: $\sigma \xleftarrow{\$} \{0,1\}^n,\ r \leftarrow \mathtt{H}_2(\sigma, m), R \leftarrow rP \bmod p$
2: $(j_1, \ldots, j_k) \leftarrow \mathtt{H}_1(ID_a, Q_a),\ Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$
3: $u \leftarrow \mathtt{H}_3(r(Y_a + Q_a) \bmod p) \oplus \sigma$, $v \leftarrow \mathtt{H}_4(\sigma) \oplus m$
4: **return** $c = (R, u, v)$

$m \leftarrow$ `IBE.Dec`(x_a, c): Alice decrypts the ciphertext c.

1: $\sigma' \leftarrow \mathtt{H}_3(x_a R \bmod p) \oplus u$
2: $m' \leftarrow v \oplus \mathtt{H}_4(\sigma')$,$r' \leftarrow \mathtt{H}_2(\sigma', m)$
3: **if** $r'P \pmod p = R$ **then return** m'
4: **else return** $\perp$

to construct IDB schemes usually require several expensive operations (e.g., scalar multiplication), and therefore may incur a non-negligible computation overhead. To reduce this cost, we exploit the message encoding technique and subset resilient functions (similar to [23]) along with the exponent product of powers property to generate keys. This permits an improved efficiency for both the PKG and user since it only requires a hash call and a few point additions.

Our IDB schemes use similar `IBE.Setup` and `IBE.Extract` functions whose key steps are outlined as follows. In the `IBE.Setup`, the PKG selects t values $v_i \leftarrow \mathbb{Z}_q$, and computes their commitments as $V_i \leftarrow v_i P \mod p$, for $i = 1, \ldots, t$, it then sets the master secret key $msk \leftarrow (v_1, \ldots, v_t)$ and the system-wide public key $mpk \leftarrow (V_1, \ldots, V_t)$. This is similar to the scheme in [23], where EC scalar multiplication is used as the one-way function. In `IBE.Extract`, the PKG picks a nonce $\beta \leftarrow \mathbb{Z}_q$ and computes its commitments $Q \leftarrow \beta P \mod p$. The PKG then derives indexes $(j_1, \ldots, j_k) \leftarrow \mathtt{H}_1(ID, Q)$, which select k-out-of-t elements from the master secret key v_{j_i} for $i = 1, \ldots, k$. Note that Q is implicitly authenticated by being included in input of $\mathtt{H}_1(\cdot)$, this is similar to the technique used in other pairing-free identity-based and certificateless systems [15,3]. In Steps 3-4, unlike the scheme in [23], where secret keys are exposed, we use the additive homomorphic property in the exponent to mask the one-time signature y (Step 3) via the nonce β (in line with [5,4]). The PKG will then sends (x, Q) to the user via a secure channel.

Identity-Based Encryption Scheme: In `IBE.Enc` (Algorithm 1, Step 2), the indexes obtained from $\mathtt{H}_1$ are used to retrieve the components V_{j_i} from the system-wide public key mpk. The input of $\mathtt{H}_3$ is the ephemeral key, which given the ciphertext $c = (R, u, v)$, can be recomputed by the receiver in the `IBE.Dec` algorithm. σ and r are computed in-line with the transformation proposed in [14].

Algorithm 2 Identity-Based Signature

$(msk, params) \leftarrow$ `IBS.Setup` (1^κ):
Description identical to `IBE.Setup` in Algorithm 1, except that only the description of $\mathtt{H}_1$ and $\mathtt{H}_5$ is included in *params*.

$(w, Q) \leftarrow$ `IBS.Extract`(ID, U, msk):
As in `IBE.Extract` in Algorithm 1.

$(s, e) \leftarrow$ `IBS.Sign`(m, x_a): Alice ID_a signs message m.

1: $r \xleftarrow{\$} \mathbb{Z}_q$, $R \leftarrow rP \bmod p$
2: $e \leftarrow \mathtt{H}_5(m, R)$
3: $s \leftarrow r - e \cdot x_a \bmod q$
4: **return** (s, e)

$\{0, 1\} \leftarrow$ `IBS.Verify`$(m, ID_a, Q_a, \langle s, e \rangle)$:
Bob verifies the signature (s, e).

1: $(j_1, \ldots, j_k) \leftarrow \mathtt{H}_1(ID_a, Q_a)$
2: $Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$
3: $R' \leftarrow sP + e(Y_a + Q_a) \bmod p$
4: **if** $e = \mathtt{H}_5(m, R')$ **then return** 1
5: **else return** 0

Identity-Based Signature Scheme: In `IBS.Verify`, the public key of the user Y_a is computed from $V_{j_i} \in mpk$ via the indexes retrieved from the output of $\mathtt{H}_1$. The key generation is as in Algorithm 1. The rest of the signing and verification steps are akin to Schnorr signatures [24].

Identity-Based Key Exchange Scheme: For the key exchange scheme, we run `IBE.Setup` and then let both parties, Alice and Bob, obtain (x_A, Q_A) and (x_B, Q_B) via the `IBE.Extract` algorithm, respectively. Alice then picks $z_A \xleftarrow{\$} \mathbb{Z}_q$, computes its commitment $M_A \leftarrow z_A P \bmod p$, and sends (M_A, Q_A) to Bob. Bob does the same and sends (M_B, Q_B) to Alice. Alice then computes $(j_1, \ldots, j_k) \leftarrow \mathtt{H}_1(ID_b, Q_b)$ and $Y_b \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$ and outputs the shared secret key as $K_a \leftarrow x_a(Y_b + Q_b) + z_a M_b \bmod p$. Bob works similarly, and outputs the shared key as $K_b \leftarrow x_b(Y_a + Q_a) + z_b M_a \bmod p$.

4.2 Proposed Certificateless Cryptosystem

For our CL schemes to achieve the same trust level (Level 3) [16] on the third party (KGC), as in traditional PKI, we use the binding method [1] in the `CLE.PartKeyGen` and `CLS.PartKeyGen` algorithms. Note that the same secure channel which is used for user authentication (e.g., SSL/TLS), can be used to send the user commitment U to the KGC. This permits an implicit certification of U, and therefore any changes of U, will falsify the private key.

The `CLE.KGCSetup` algorithm is as in `IBE.Setup` in Algorithm 1. The `CLE.PartKeyGen` algorithm is similar to the `IBE.Extract` in Algorithm 1, with the difference that the user commitment U is used to compute Q. In `CLE.UserKeyGen`, the correctness of the partial private key is checked first before the private key x is computed.

Certificateless Encryption Scheme: Note that the `CLE.Enc` and `CLE.Dec` algorithms are identical to `IBE.Enc` and `IBE.Dec` algorithms in Algorithm 1.

Certificateless Signature Scheme: The setup and key generation algorithms are as in Algorithm 3, and the `CLS.Sign` and `CLS.Verify` algorithms are as in `IBS.Sign` and `IBS.Verify` in Algorithm 2, respectively.

Certificateless Key Exchange Scheme: Given the compatibility of our IDB and CL schemes, after the initial algorithms (system setup and key generation)

Algorithm 3 Certificateless Encryption

$(msk, params) \leftarrow \texttt{CLE.KGCSetup}(1^\kappa)$:
As in `IBE.Setup` in Alg. 1.

$(\alpha, U) \leftarrow \texttt{CLE.UserSetup}(\cdot)$:
1: $\alpha \xleftarrow{\$} \mathbb{Z}_q, U \leftarrow \alpha P \bmod p$
2: **return** (α, U)

$(w, Q) \leftarrow \texttt{CLE.PartKeyGen}(ID, U, msk)$:
1: $\beta \xleftarrow{\$} \mathbb{Z}_q$, $W \leftarrow \beta P \bmod p$
2: $Q = U + W \bmod p$
3: $(j_1, \ldots, j_k) \leftarrow \texttt{H}_1(ID, Q)$ where for all $i = 1, \ldots, t$, $1 < j_i < |t|$
4: $y \leftarrow \sum_{i=1}^{k} v_{j_i} \bmod q$
5: $w \leftarrow y + \beta \bmod q$
6: **return** (w, Q)

$x \leftarrow \texttt{CLE.UserKeyGen}(w, \alpha)$:
1: $(j_1, \ldots, j_k) \leftarrow \texttt{H}_1(ID, Q)$, $Y \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$
2: $W' \leftarrow Q - U \bmod p$, $W'' := wP - Y \bmod p$
3: **if** $W' = W''$ **then return** $x \leftarrow w + \alpha \bmod q$ **else return** $\perp$

$c \leftarrow \texttt{CLE.Enc}(m, ID_a, Q_a)$: Bob encrypts message $m \in \{0,1\}^n$.
1: $\sigma \xleftarrow{\$} \{0,1\}^n$, $r \leftarrow \texttt{H}_2(\sigma, m), R \leftarrow rP \bmod p$
2: $(j_1, \ldots, j_k) \leftarrow \texttt{H}_1(ID_a, Q_a)$, $Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$
3: $u \leftarrow \texttt{H}_3(r(Y_a + Q_a) \bmod p) \oplus \sigma$, $v \leftarrow \texttt{H}_4(\sigma) \oplus m$
4: **return** $c = (R, u, v)$

$m \leftarrow \texttt{CLE.Dec}(x_a, c)$: Alice decrypts the ciphertext c.
1: $\sigma' \leftarrow \texttt{H}_3(x_a R \bmod p) \oplus u$
2: $m' \leftarrow v \oplus \texttt{H}_4(\sigma')$,$r' \leftarrow \texttt{H}_2(\sigma', m)$
3: **if** $r'P \pmod p = R$ **then return** m'
4: **else return** $\perp$

Algorithm 4 Certificateless Digital Signature

$(msk, params) \leftarrow \texttt{CLS.KGCSetup}(1^\kappa)$:
As in `CLE.KGCSetup` in Alg. 3, except that $\texttt{H}_1$ and $\texttt{H}_5$ are in *params*.

$(\alpha, U) \leftarrow \texttt{CLS.UserSetup}(params)$:
As in `CLE.UserSetup` in Alg. 3.

$(w, Q) \leftarrow \texttt{CLS.PartKeyGen}(ID, U, msk)$:
As in `CLE.PartKeyGen` in Alg. 3.

$x \leftarrow \texttt{CLS.UserKeyGen}(params, \alpha, w)$:
As in `CLE.UserKeyGen` in Alg. 3.

$(s, e) \leftarrow \texttt{CLS.Sign}(m, x_a)$: Alice ID_a signs message m.
1: $r \xleftarrow{\$} \mathbb{Z}_q$, $R \leftarrow rP \bmod p$
2: $e \leftarrow \texttt{H}_5(m, R)$
3: $s \leftarrow r - e \cdot x_a \bmod q$
4: **return** (s, e)

$\{0,1\} \leftarrow \texttt{CLS.Verify}(m, Q_a, \langle s, e \rangle)$:
Bob verifies the signature (s, e).
1: $(j_1, \ldots, j_k) \leftarrow \texttt{H}_1(ID_a, Q_a)$
2: $Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \bmod p$
3: $R' \leftarrow sP + e(Y_a + Q_a) \bmod p$
4: **if** $e = \texttt{H}_5(m, R')$ **then return** 1
5: **else return** 0

take place as in Algorithm 3, the CL key exchange will be identical to the one proposed in the identity-based key exchange scheme above.

4.3 Compatibility of Identity-Based and Certificateless Schemes

In our CL schemes, we utilize the additive homomorphic property of the exponents (i.e., w) when the KGC includes the addition of commitments ($W$and U) in the $\texttt{H}_1$. After receiving w, the user exploits the homomorphic property to

modify the key without falsifying it and obtain x. For instance, we observed that our counterparts (e.g., [9,1]) do not offer such a compatibility, since the partial private key is the KGC's commitment to the (hash of) user identity, without a homomorphic property. Moreover, the KGC does not output any auxiliary value to incorporate the user commitment with it.

As shown above, our IDB and CL schemes are compatible, thanks to the special design of their key generation algorithms (i.e., `Extract` in IDB, `UserSetup` and `PartKeyGen` in CL). Therefore, after the users computed/obtained their keys from the third party, the interface of the main cryptographic functions (e.g., encrypt, decrypt, sign, etc.) are identical in both systems, therefore, the users can communicate with uses in different domains seamlessly. For instance, ciphertext $c = (R, u, v)$ outputted by the `CLE.Enc` in Algorithm 1, can be decrypted by a user in the identity-based setting by the `IBE.Dec` algorithm in Algorithm 1. This also applies to the signature and the key exchange schemes proposed above.

5 Security Analysis

Theorem 1. *If an adversary $\mathcal{A}_I$ can break the IND-CLE-CCA security of the encryption scheme proposed in Algorithm 3 after q_{H_i} queries to random oracles H_i for $i \in \{1,2,3,4\}$, q_D queries to the decryption oracle and q_{sk} to the private key extraction oracle with probability ϵ. There exists another algorithm $\mathcal{C}$ that runs $\mathcal{A}_I$ as subroutine and breaks a random instance of the CDH problem (P, aP, bP) with probability ϵ' where: $\epsilon' > \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{\mathbf{e}(q_{sk}+1)} - \frac{q_{H_2}}{2^n} - \frac{q_D(q_{H_2}+1)}{2^n} - \frac{2q_D}{p}\right)$.*

Proof. Our proof technique is similar to the one in [3]. $\mathcal{C}$ *simulates* the real environment for $\mathcal{A}_I$. It knows the t secret values $v_i's$ in the scheme, and tries to embed a random instance of the CDH problem (P, aP, bP). $\mathcal{C}$ sets aP as a part of the target user's (ID^*) public key (i.e., $Q_{ID^*} \leftarrow aP$) and bP as a part of the challenge ciphertext (i.e., $R^* \leftarrow bP$). $\mathcal{C}$ uses four lists, namely $\texttt{List}_{\texttt{H}_1}$, $\texttt{List}_{\texttt{H}_2}$, $\texttt{List}_{\texttt{H}_3}$, and $\texttt{List}_{\texttt{H}_4}$, to keep track of the random oracle responses and following the IND-CLE-CCA experiment $Expt_{\mathcal{A}}^{IND\text{-}CLE\text{-}CCA}$ (Definition 7), $\mathcal{C}$ responds to $\mathcal{A}_I$ queries as follows.
Queries to $\texttt{H}_1(ID_i, Q_i)$: If the entry $(\langle ID_i, Q_i\rangle, h_{1,i})$ exists in $\texttt{List}_{\texttt{H}_1}$, $\mathcal{C}$ returns $h_{1,i}$, otherwise, it chooses $h_{1,i} \xleftarrow{\$} \gamma$, and inserts $(\langle ID_i, Q_i\rangle, h_{1,i})$ in $\texttt{List}_{\texttt{H}_1}$.
Queries to $\texttt{H}_2(\sigma_i, m_i)$: If the entry $(\langle \sigma_i, m_i\rangle, h_{2,i})$ exists in $\texttt{List}_{\texttt{H}_2}$, $\mathcal{C}$ returns $h_{2,i}$, otherwise, it chooses $h_{2,i} \xleftarrow{\$} \mathbb{Z}_q$, and inserts $(\langle \sigma_i, m_i\rangle, h_{2,i})$ in $\texttt{List}_{\texttt{H}_2}$.
Queries to $\texttt{H}_3(K_i)$: If the entry $(K_i, h_{3,i})$ exists in $\texttt{List}_{\texttt{H}_3}$, $\mathcal{C}$ returns $h_{3,i}$, otherwise, it chooses $h_{3,i} \xleftarrow{\$} \{0,1\}^n$, and inserts $(K_i, h_{3,i})$ in $\texttt{List}_{\texttt{H}_3}$.
Queries to $\texttt{H}_4(\sigma_i)$: If the entry $(\sigma_i, h_{4,i})$ exists in $\texttt{List}_{\texttt{H}_4}$, $\mathcal{C}$ returns $h_{4,i}$, otherwise, it chooses $h_{4,i} \xleftarrow{\$} \{0,1\}^n$, and inserts $(\sigma_i, h_{4,i})$ in $\texttt{List}_{\texttt{H}_4}$.
Public key request: Upon receiving a public key request on ID_i, $\mathcal{C}$ works as follows. If $(\langle ID_i, U_i, Q_i\rangle, \zeta_i)$ exists in $\texttt{List}_{PK}$, then it returns (ID_i, U_i, Q_i). Else, it flips a fair coin where $\Pr[\zeta = 0] = \delta$, and works as follows (δ will be determined later in the proof). If $\zeta = 0$, it runs the partial key extraction oracle below first, update $\texttt{List}_{PK}$ and then output (ID_i, U_i, Q_i). If $\zeta = 1$, pick $t \xleftarrow{\$} \mathbb{Z}_q$, set $Q_i \leftarrow$

$aP \mod p$, adds $(ID_i, U_i, \langle \perp, Q_i \rangle)$ to $\texttt{List}_{\texttt{PartialSK}}$ and adds $(\langle ID_i, U_i, Q_i \rangle, \zeta_i)$ to $\texttt{List}_{PK}$, before outputting (ID_i, U_i, Q_i).
Partial key extraction: Upon receiving a partial key extraction query on (ID_i, U_i), $\mathcal{C}$ works as follow:

- If $(ID_i, U_i, \langle w_i, Q_i \rangle) \in \texttt{List}_{\texttt{PartialSK}}$, return (w_i, Q_i).
- Else,
  - $w_i \xleftarrow{\$} \mathbb{Z}_q$, $Z_i \leftarrow w_i P \mod p$, $(j_1, \ldots, j_k) \xleftarrow{\$} [1, \ldots, t]$, $Q_i \leftarrow Z_i - \sum_{i=1}^{k} V_{j_i} + U_i \mod p$.
  - If $(ID_i, Q_i, \ldots) \in \texttt{List}_{\texttt{H}_1}$, aborts. Else, adds $(\langle ID_i, Q_i \rangle, h_{1,i})$ to $\texttt{List}_{\texttt{H}_1}$, where $h_{1,i} \leftarrow (j_1, \ldots, j_k)$ and output the partial private key as (w_i, U_i, Q_i) after adding it to $\texttt{List}_{\texttt{PartialSK}}$.

Secret key request: Upon receiving a secret key request on ID_i, $\mathcal{C}$ checks if there exists a pair $(ID_i, u_i, U_i) \in \texttt{List}_{\texttt{SecretKey}}$, it returns u_i. Otherwise, selects $u_i \xleftarrow{\$} \mathbb{Z}_q$, computes $U_i \leftarrow u_i P \mod p$ and inserts (ID_i, u_i, U_i) in $\texttt{List}_{\texttt{SecretKey}}$.
Private key request: To answer a private key request on (ID_i, U_i), $\mathcal{C}$ runs the public key request oracle above to get $(\langle ID_i, U_i, Q_i \rangle, \zeta_i) \in \texttt{List}_{PK}$ and finds (ID_i, u_i, U_i) in $\texttt{List}_{\texttt{SecretKey}}$. If $\zeta = 0$, finds $((ID_i, U_i, \langle w_i, Q_i \rangle) \in \texttt{List}_{\texttt{PartialSK}}$ and returns $w_i + u_i$ as the response. Otherwise, it aborts.
Decryption query: Upon receiving a decryption query on $(ID_i, Q_i, c_i = \langle R_i, u_i, v_i \rangle)$, $\mathcal{C}$ works as follows.

- Searches $\texttt{List}_{PK}$ for an entry $(\langle ID_i, U_i, Q_i \rangle, \zeta_i)$. If $\zeta = 0$, works as follows.
  - Searches $\texttt{List}_{\texttt{PartialSK}}$ for a tuple $(ID_i, U_i, \langle w_i, Q_i \rangle)$ and searches for $(ID, \langle w, Q \rangle)$ in $\texttt{List}_{\texttt{PartialSK}}$, set $\sigma' \leftarrow \texttt{H}_3((w + \alpha)R \mod p) \oplus u$, $m' = v \oplus \texttt{H}_4(\sigma')$, $r' := \texttt{H}_2(\sigma', m)$.
  - Checks if $R = r'P \mod p$ holds, outputs m'
- Else, if $\zeta = 1$, works as follows.
  - Runs the oracle for $\texttt{H}_1$ to get $h_{1,i}$ (to compute the public key Y_i) and checks lists $\texttt{List}_{\texttt{H}_2}$, $\texttt{List}_{\texttt{H}_3}$ and $\texttt{List}_{\texttt{H}_4}$ for tuples $(\langle \sigma_i, m_i \rangle, h_{2,i})$, $(K_i, h_{3,i})$, and $(\sigma_i, h_{4,i})$, such that $R_i = h_{2,i} P \bmod p$, $u = h_{3,i} \oplus \sigma_i$ and $v = h_{4,i} \oplus m_i$ exists. Checks if $K_i = r_i(Y_i + Q_i)$ holds, outputs m_i, else, aborts.

After the first round of queries, $\mathcal{A}_I$ outputs ID^* and two messages m_0 and m_1 on which it wishes to be challenged on. We assume that ID^* has been already queried to $\texttt{H}_1$ and was not submitted to the *private key request oracle*. $\mathcal{C}$ checks $(\langle ID^*, U^*, Q^* \rangle, \zeta) \in \texttt{List}_{PK}$ if $\zeta = 0$, it aborts. Otherwise, it computes the challenge ciphertext as follows. $\beta^* \xleftarrow{\$} \{0,1\}$, $\sigma^* \xleftarrow{\$} \{0,1\}^*$, $u^* \leftarrow \{0,1\}^n$, $b \xleftarrow{\$} \{0,1\}$. $R^* \leftarrow aP$ (this implicitly implies that $a = \texttt{H}_2(\sigma^*, m_b)$), $\texttt{H}_3(K_{ID^*}) \leftarrow u^* \oplus \sigma^*$ and $v^* \leftarrow \texttt{H}_4(\sigma^*) \oplus m_b$. Return (R^*, u^*, v^*).

$\mathcal{A}_I$ initiates the second round of queries similar as above, with the restrictions defined in Definition 5. When $\mathcal{A}_I$ outputs its decision bit b', $\mathcal{C}$ returns a set $\Lambda = \{K_i - R_i^{y_i}$,where K_is are the input queries to $\texttt{H}_3\}$.

Notice that if $\mathcal{C}$ does not abort, and $\mathcal{A}_I$ outputs its decision bit b', then the public key must have the $Q_{ID^*} = aP$, and given how the challenge ciphertext is

formed (e.g., $R^* = bP$), $K_{ID^*} = y_{ID^*}abP$ should hold, where y_{ID^*} is known to $\mathcal{C}$. Hence, the answer to a random instance of the the CDH problem (P, aP, bP), can be derived from examining the $\mathcal{A}_I$'s choice of public key and $\mathtt{H}_3$ queries.

Here, we provide an indistinguishability argument for the above simulation. First we look at the simulation of the decryption algorithm. If $\zeta = 0$, we can see that the simulation is perfect. For $\zeta = 1$, an error might occur in the event that c_i is valid, but (σ_i, m_i), K_i, and σ_i were never queried to $\mathtt{H}_2, \mathtt{H}_3$, and $\mathtt{H}_4$, respectively. For the first two hash functions, the probability that the c_i is valid, given a query to $\mathtt{H}_3$ was never made, considers the query to $\mathtt{H}_2$ as well (not considering the checking phase in the simulation). Therefore, the probability that this could occur is $\frac{q_{\mathtt{H}_2}}{2^n} + \frac{1}{p}$. When considering $\mathtt{H}_4$, this probability is $\frac{1}{2^n} + \frac{1}{p}$. Given the number of decryption queries q_D, we have the probability of decryption error $\frac{q_D(q_{\mathtt{H}_2}+1)}{2^n} + \frac{2q_D}{p}$.

$\mathcal{C}$ will also fail in simulation during the partial key extraction queries if the entry $(ID_i, Q_i, \dots)$ already exists in $\mathtt{List}_{\mathtt{H}_1}$. This will happen with probability $\frac{q_{\mathtt{H}_1}}{2^\gamma}$.

The probability that $\mathcal{C}$ does not abort in the simulation is $\delta^{q_{sk}}(1-\delta)$ which is maximized at $\delta = 1 - \frac{1}{q_{sk}+1}$. Therefore, the probability that $\mathcal{C}$ does not abort is $\frac{1}{\mathbf{e}(q_{sk}+1)}$, where $\mathbf{e}$ is the base of natural logarithm. Given the argument above, we know that if (σ^*, m_b), (K^*) were never queried to $\mathtt{H}_2$ and $\mathtt{H}_3$ oracles, then $\mathcal{A}_I$ cannot gain any distinguishing advantage more than $\frac{1}{2}$. Given all the above arguments, the probability that K_{ID^*} has been queried to $\mathtt{H}_3$ is $\geq \frac{2\epsilon}{\mathbf{e}(q_{sk}+1)} - \frac{q_{\mathtt{H}_2}}{2^n} - \frac{q_D(q_{\mathtt{H}_2}+1)}{2^n} - \frac{2q_D}{p}$.

Therefore, if the above probability occurs, $\mathcal{C}$ can solve the CDH problem by finding and computing $K_{ID^*} = y_{ID^*}abP$ from the list Λ. Given the size of the list Λ (i.e., $q_{\mathtt{H}_3}$), the probability for $\mathcal{C}$ to be successful in solving CDH is: $\epsilon' > \frac{1}{q_{\mathtt{H}_3}}\big(\frac{2\epsilon}{\mathbf{e}(q_{sk}+1)} - \frac{q_{\mathtt{H}_2}}{2^n} - \frac{q_D(q_{\mathtt{H}_2}+1)}{2^n} - \frac{2q_D}{p}\big)$

Theorem 2. *If an adversary $\mathcal{A}_{II}$ can break the IND-CLE-CCA security of the encryption scheme proposed in Algorithm 3 after $q_{\mathtt{H}_i}$ queries to random oracles $\mathtt{H}_i$ for $i \in \{1,2,3,4\}$, q_D queries to the decryption oracle and q_{sk} to the secret key extraction oracle with probability ϵ. There exists another algorithm $\mathcal{C}$ that runs $\mathcal{A}_{II}$ as subroutine and breaks a random instance of the CDH problem (P, aP, bP) with probability ϵ' where: $\epsilon' > \frac{1}{q_{\mathtt{H}_3}}\big(\frac{2\epsilon}{\mathbf{e}(q_{sk}+1)} - \frac{q_{\mathtt{H}_2}}{2^n} - \frac{q_D(q_{\mathtt{H}_2}+1)}{2^n} - \frac{2q_D}{p}\big)$.*

Proof (Sketch). Having access to random oracles, and by keeping lists similar to above, the challenger $\mathcal{C}$ can simulate an indistinguishable environment for $\mathcal{A}_{II}$ and respond to its queries similar to the above proof. Note that following Definition 7, $\mathcal{A}_{II}$ can query for the secret key of all the users, except for the target user ID^*.

$\mathcal{C}$ knows the t private values $v_i's$ in the scheme, and tries to embed a random instance of the CDH problem (P, aP, bP). By flipping a fair coin, as in the *public key request* query above, $\mathcal{C}$ defines the probability to embed aP in the target U_{ID^*} value. $\mathcal{C}$ sets bP as a part of the challenge ciphertext (i.e., $R^* \leftarrow bP$).

After the $\mathcal{A}_{II}$ outputs a forgery, $\mathcal{C}$ can extract the solution to the CDH problem since it has knowledge over the t secret values and β^*.

Lemma 1. *A public key replacement attack by $\mathcal{A}_I$ is not practical since it will falsify the private key.*

Proof. Note that if $\mathcal{A}_I$ replaces U with a new value U' (which it might know the corresponding secret key), then the existing Q will be falsified since $Q = U + W$ and this will also falsify the current partial private key component y since it is computed based on the indexes that are obtained by computing $\mathtt{H}_1(ID, Q)$. Also note that, if $\mathcal{A}_I$ can obtain the original (α, U), given Q is public, it can compute W, however, W is merely the commitment of β and it does not disclose any information about β. The public key replacement attack in our security proof is possible if $\mathcal{A}_I$ requests a new partial private key for each new U' .

Lemma 2. *If an adversary $\mathcal{A}_I$ can break the EU-CMA security of the signature scheme proposed in Algorithm 4 , then one can build another algorithm $\mathcal{C}$ that runs $\mathcal{A}_I$ as subroutine and breaks a random instance of the ECDLP (P, aP).*

Proof. Due to the space constraint, here we give the high level idea of our proof. We let $\mathcal{A}_I$ be as in Definition 8, then we can build another algorithm $\mathcal{C}$ that uses $\mathcal{A}_I$ as a subroutine, and upon $\mathcal{A}_I$'s successful forgery, solves a random instance of the ECDLP (P, aP). $\mathcal{C}$ knows the t secret values v_i's and, similar to the proof of Theorem 1, it sets aP as a part of the target user's public key (i.e., $Q_{ID^*} \leftarrow aP$). Most of the simulation steps are like the ones in the proof of Theorem 1. At the end of the simulation phase, $\mathcal{A}_I$ outputs a forgery signature (s_1^*, e_1^*), the proof then uses the forking lemma [22] to run the adversary again to obtain a second forgery (s_2^*, e_2^*), using the same random tape. Our proof will follow the same approach as in [15] which is very similar to the proof in [24]. Given two forgeries and the knowledge of $\mathcal{C}$ on the $v_i's$ and α_{ID}^*, $\mathcal{C}$ can compute a and solve the ECDLP. Note that similar to Schnorr [24] the security of the scheme will be non-tight due to the forking lemma.

Parameters Selection for (t, k). Parameters (t, k) should be selected such that the probability $\frac{q_{\mathtt{H}_1} \cdot k!}{2^\gamma}$ is negligible. Considering that $\gamma = k \log_2 t$ (since k indexes that are $\log_2 t$-bit long are selected with the hash output), this gives us $\frac{q_{\mathtt{H}_1} \cdot k!}{2^{k|t|}}$. We further elaborate on some choices of (t, k) along with their performance implications in Section 6.

6 Performance Analysis and Comparison

We first present the analytical and then experimental performance analysis and comparison of our schemes with their counterparts. We focus on the *online* operations (e.g., encryption, signing, key exchange) for which both our IDB and CL schemes have the same algorithms, rather than one-time (offline) processes like setup and key generation. Since the online operations are identical in IDB and CL systems in our case, we refer to them as "Our Schemes" in the following

Table 1: Analytical Comparison of Public Key Encryption Schemes

Scheme	sk	**Enc**	**Comm.**	**Dec**	PK	mpk	**System Type**	$\kappa^{\dagger}$
ECIES [26]	$\|q\|$	$2\,m_{EC} + dm_{EC}$	$2\|p\| + b + d + CF$	m_{EC}	$\|p\|$	-	TD	128
BF [9]	$\|p\|$	$bp + m_{EC} + ex$	$\|p\| + b + \|M\|$	$bp + m_{EC}$	$\|p\|$	$\|p\|$	IB	80
AP [1]	$\|p\|$	$3\,bp + m_{EC} + ex$	$\|p\| + b + \|M\|$	$bp + m_{EC}$	$2\|p\|$	$\|p\|$	CL	80
BSS [3]	$2\,\|q\|$	$4\,ex + m$	$\|p\| + b + \|M\|$	$3\,ex$	$2\|p\|$	$\|p\|$	CL	128
WSB [29]	$\|p\| + \|q\|$	$3\,m_{EC} + 2\,a_{EC}$	$2\|q\| + \|c\|$	$2\,m_{EC}$	$2\|p\|$	$\|p\|$	CL	128
Our Schemes	$\|p\|$	$2\,m_{EC} + k\,a_{EC}$	$\|p\| + b + \|M\|$	$2\,m_{EC}$	$\|p\|$	$t\,\|p\|$	IB/CL	128

† Denotes the security bit. **Enc**, **Dec**, and **Comm.** represent encryption, decryption, and communication load (bi-directional), respectively. m_{EC}, a_{EC}, and dm_{EC} denote the costs of EC scalar multiplication, EC addition, and double scalar multiplication over modulus p, respectively. m, ex and bp denote multiplication, exponentiation and pairing operation, respectively. k is the BPV parameter that shows how many precomputed pairs are selected in the online phase. b, d and CF denote block/key size for symmetric key encryption, message digest (i.e., MAC) size and size of the certificate, respectively. M denotes message space size. TD, IDB, and CL represent traditional public key cryptography, identity-based cryptography, and certificateless cryptography, respectively.

Table 2: Analytical Comparison of Digital Signature Schemes

Scheme	sk	**Sign**	**Comm.**	**Verify**	PK	mpk	**System Type**	κ
Schnorr [24]	$\|q\|$	m_{EC}	$2\|q\| + CF$	$2dm_{EC}$	$\|p\|$	-	TD	128
GG [15]	$\|q\|$	m_{EC}	$2\|p\| + \|q\|$	$2\,dm_{EC}$	$\|p\|$	$\|p\|$	IDB	128
AP [1]	$\|p\|$	$2\,m_{EC} + a_{EC} + bp$	$\|q\| + \|p\|$	$4\,bp + ex$	$2\|p\|$	$\|p\|$	CL	80
KIB [18]	$\|q\|$	m_{EC}	$\|q\| + \|p\|$	$3\,m_{EC}$	$3\,\|p\|$	$2\,\|p\|$	CL	128
Our Schemes	$\|q\|$	m_{EC}	$2\|q\|$	$dm_{EC} + k\,a_{EC}$	$\|p\|$	$t\,\|p\|$	IDB/CL	128

tables/discussions. We consider the cost of certificate verification for schemes in traditional PKI. We only consider the cost of verifying and communicating the cost of one certificate, which is highly conservative since in practice (i.e., X.509) there are at minimum two certificates in a certificate chain. This number could be as high as ten certificates in some scenarios.

Analytical Performance Analysis and Comparison We present a detailed analytical performance comparison of our schemes with their counterparts for public key encryption/decryption, digital signature and key exchange in Table 1, Table 2 and Table 3, respectively.

Our schemes have significantly lower communication overhead than their PKI-based counterpart in all cryptosystems as they do not require the transmission of certificates. As discussed above and also elaborated in Section 6, this translates into substantial bandwidth gain as well as computational efficiency since the certification verification overhead is also lifted. Moreover, in almost all instances, our schemes also offer a lower end-to-end computational overhead compared to their PKI-based counterparts. Our schemes also offer a lower end-to-end computational delay than that of all of their IDB and CL counterparts in all cryptosystems, with generally equal private and public key sizes. However, the master public key size of our scheme is larger than all of their counterparts.

Experimental Performance Analysis and Comparison: We now further elaborate on the details of our performance analysis and comparison with experimental results. We conduct experiments on both commodity hardware and low-end embedded devices that are typically found in IoT systems to objec-

Table 3: Analytical Comparison of Key Exchange Schemes

Scheme	sk	**User Comp.**	**Comm.**	PK	mpk	**System Type**
Ephemeral ECDH [13]	$\|q\|$	$2m_{EC}$	$2\|p\| + CF$	$\|p\|$	-	TD
ECHMQV [19]	$\|q\|$	$3\,m_{EC}$	$2\|p\| + CF$	$\|p\|$	-	TD
TFNS [28]	$\|p\|$	$bp + 5\,m_{EC}$	$\|p\|$	$\|p\|$	$\|p\|$	IDB
AP [1]	$\|p\|$	$4\,bp + ex$	$3\|p\|$	$2\|p\|$	$\|p\|$	CL
YT [30]	$2\|q\| + \|p\| + \|s\|^{\dagger}$	$3\,dm_{EC} + 5\,m_{EC}$	$3\|p\| + \|s\|$	$2\|p\| + \|s\|$	$2\,\|p\|$	CL
Our Scheme	$\|q\|$	$3\,m_{EC} + (k+1)\,a_{EC}$	$2\|p\|$	$\|p\|$	$t\,\|p\|$	IDB/CL

User Comp. denotes user computation.
†$|s|$ denotes the output of a signature scheme that the authors in [30] use in their scheme.

Table 4: Public Key Encryption Schemes on Commodity Hardware

Scheme	sk	**Enc**	**Comm.**	**Dec**	PK	mpk	**E2E Delay**
ECIES [26]	32	55	$690^{\dagger} + \|M\|$	21	32	-	76
BF [9]	32	≈ 2000	$48 + \|M\|$	≈ 2000	32	32	≈ 4000
AP [1]	32	≈ 6000	$48 + \|M\|$	≈ 2000	64	32	≈ 8000
BSS [3]	64	73	$64 + \|M\|$	53	64	32	126
WSB [29]	64	53	$64 + \|M\|$	41	64	32	94
Our Schemes	32	39	$48 + \|M\|$	33	32	32K	72

All sizes are in Bytes, and all computations are in microseconds.
† We assume the certificate size is 578 Bytes, the size is given in RFC 5280 [11].

tively assess the performance of our schemes as well as their counterparts. Our open-sourced implementation is available via the following link.

https://github.com/Rbehnia/CertFreeSystems

• *Experiments on Commodity Hardware:* We used an i7 Skylake laptop equipped with a 2.6 GHz CPU and 12 GB RAM in our experiments. We implemented our schemes on the FourQ curve [12] which offers fast elliptic curve operations for $\kappa =$ 128-bit security. We instantiated our random oracles with blake2 hash function[6], which offers high efficiency and security. For our parameters, we selected $k = 18$ and $t = 1024$. We conservatively estimated the costs of our counterparts based on the microbenchmarks on our evaluation setup of (i) FourQ curve for schemes that do not require pairing and (ii) PBC library[7] on a curve with $\kappa = 80$-bit security (we used the most efficient alternative for them) for schemes that require pairing.

As depicted in Table 4, the encryption and decryption algorithms of our schemes are more efficient than their counterparts in the identity-based and certificateless settings. More specifically, the end-to-end delay of our schemes is $\approx 25\%$ lower than that in [29], which is specifically suitable for aerial drones. One could also notice how the communication overhead is lower in certificateless and identity-based schemes since there is no need for certificate transmission.

As shown in Table 5, our schemes enjoy from the fastest verification algorithms among all its counterparts. This is again due to the novel way the user keys are derived and results in 30% and 52% faster end-to-end delay as compared

[6] http://131002.net/blake/blake.pdf
[7] https://crypto.stanford.edu/pbc/

Table 5: Digital Signature Schemes on Commodity Hardware

Scheme	*sk*	**Sign**	**Comm.**	**Verify**	*PK*	*mpk*	**E2E Delay**
Schnorr [24]	32	12	642	44	32	-	56
GG [15]	32	12	96	44	32	32	56
AP [1]	32	$\approx$ 2000	64	$\approx$ 8000	64	32	$\approx$ 10000
KIB [18]	32	20	64	61	96	64	81
Our Schemes	32	12	64	27	32	32K	39

All sizes are in Bytes, and all computations are in microseconds.

Table 6: Key Exchange Schemes on Commodity Hardware

Scheme	*sk*	**User Comp.**	**Comm.**	*PK*	*mpk*	**E2E Delay**
Ephemeral ECDH [13]	32	55	642	32	-	110
ECHMQV [19]	32	74	642	32	-	148
TFNS [28]	32	$\approx$ 2000	32	32	32	$\approx$ 4000
AP [1]	32	$\approx$ 8000	96	64	32	$\approx$ 16000
YT† [30]	160	157	160	128	64	314
Our Scheme	32	57	64	32	32K	114

All sizes are in Bytes, and all computations are in microseconds.
† The signature size is considered as 64 bytes.

to its most efficient identity-based [15] and certificateless [18] counterparts, respectively. *One may notice that although schemes in [15,24,18], along with our schemes, all require a scalar multiplication in their signature generation (see Table 2), their experimental costs differ. The reason for this discrepancy is the fact that the cost of scalar multiplication over the generator P is faster than the scalar multiplication over any curve points, and these differences are considered in the experimental evaluations.*

As shown in Table 6, our schemes' performance is similar to that in their counterparts in traditional PKI setting [13,19]. However, they outperform the most efficient counterpart in certificateless setting [30] by having 65% lower end-to-end delay and 60% smaller load for communication.

• *Experiments on Low-End Device:* We used an 8-bit AVR ATmega 2560 microprocessor to evaluate the costs of our schemes on an IoT device. AVR ATmega 2560 is a low-power microprocessor with 256 KB flash, 8 KB SRAM, 4 KB EEPROM, and operates at 16 MHz frequency. We used the 8-bit AVR library of the FourQ curve presented in [20]. For our counterparts, we again conservatively estimated their costs based on microbenchmarks in (i) FourQ curve 8-bit AVR implementation [20], and (ii) NanoECC [27], that implements a curve that supports pairings on 8-bit AVR microprocessors and offers $\kappa = 80$-bit security.

As depicted in Table 7, our schemes outperform all of their identity-based and certificateless counterparts and have a more efficient encryption algorithm than [26]. Our decryption algorithms, while being more efficient than all of their identity-based and certificateless counterparts, are slightly less efficient than the one in [26]. Similar to the trend in the analytical performance, our signature schemes outperform their counterparts. As Table 8 shows, our schemes' signing algorithm are amongst the most efficient ones, while the verification algorithm outperforms all the counterparts with similar communication overhead.

Limitations: The main limitation of our schemes is the size of the master public key. Note that if there are different TTP in different domains and users

Table 7: Public Key Encryption Schemes on 8-bit AVR processor

Scheme	sk	**Enc**	**Comm.**	**Dec**	PK	mpk
ECIES [26]	32	17 950 967	$610 + \|c\|$	6 875 861	32	-
BF [9]	32	60 802 555	$48 + \|c\|$	56 278 012	32	32
AP [1]	32	166 213 087	$48 + \|c\|$	58 912 609	64	32
BSS [3]	64	22 791 835	$64 + \|M\|$	16 590 321	64	32
WSB [29]	64	17 091 636	$64 + \|c\|$	13 631 755	64	32
Our Schemes	32	11 789 632	$48 + \|c\|$	9 883 161	32	32K

All sizes are in Bytes, and all computations are in CPU Cycles.

Table 8: Digital Signature Schemes on 8-bit AVR processor

Scheme	sk	**Sign**	**Comm.**	**Verify**	PK	mpk
Schnorr [24]	32	4 263 298	642	17 902 958	32	-
GG [15]	32	4 263 298	96	17 902 958	32	32
AP [1]	32	62 487 032	64	221 226 015	64	32
KIB [18]	32	7 025 861	64	20 617 583	96	64
Our Schemes	32	4 263 298	64	10 955 369	32	32K

All sizes are in Bytes, and all computations are in CPU Cycles.

often communicate with the users in those domains, it would make sense to store different mpk. Otherwise, the users only need to store mpk for their own systems. We can reduce the size of the mpk in exchange for a small performance loss. For instance, with $k = 32$, we can reduce the size of the mpk by four times.

7 Related Work

There is a comprehensive literature covering different aspects of IDB and CL systems. Remark that most of the closely related works have been discussed in Section 3 and 5 in terms of security models and performance metrics. Overall, the main difference in our work is to focus on the achievement of inter-compatibility between IDB and CL with a high efficiency, with respect to existing alternatives.

The idea of IDB cryptography was proposed by Shamir [25]. However, the first practical instance of such schemes was proposed later by Boneh and Franklin [9] using bilinear pairing. To get the full adaptive-identity, chosen-ciphertext security guarantees without sacrificing performance, Boyen [10] described an augmented versions of the scheme in [8] in the random-oracle model. However, the augmented version also requires multiple pairing computations in the decryption algorithm. Following [7], several pairing-free signature scheme were proposed. Galindo and Garcia [15] proposed a lightweight IDB signature scheme based on [24] with reduction to the discrete logarithm problem.

CL cryptography [1] was proposed to address the private key escrow problem in IDB systems. In the same paper, the authors proposed an IND-CCA encryption scheme along with a signature and key exchange schemes. Following their work, Baek et al. [3] proposed the first IND-CCA secure certificateless encryption scheme without pairing. The scheme is constructed using Schnorr-like signatures in partial private key generation algorithm. Recently Won et al. [29] proposed another efficient IND-CCA encryption scheme that is specifically used for key encapsulation mechanisms. There has been a number of works that focus on the

Table 9: Key Exchange Schemes on 8-bit AVR processor

Scheme	*sk*	**User Comp.**	**Comm.**	*PK*	*mpk*
Ephemeral ECDH [13]	32	18 039 710	642	32	-
ECHMQV [19]	32	24 601 857	642	32	-
TFNS [28]	32	82 356 781	32	32	32
AP [1]	32	221 226 015	96	64	32
YT [30]	160	54 212 824	160	128	64
Our Scheme	32	18 015 493	64	32	32K

All sizes are in Bytes, and all computations are in CPU Cycles.

security models of certificateless systems. In most of the proposed models (e.g., [1]) a Type-II adversary is assumed to generate the keys honestly, and initiate the attacks only after the setup phase.

Acknowledgments. This work is supported by the Department of Energy Award DE-OE0000780 and NSF Award #1652389.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.S. (ed.) Advances in Cryptology - ASIACRYPT 2003. pp. 452–473. Springer Berlin Heidelberg (2003)
2. Au, M.H., Mu, Y., Chen, J., Wong, D.S., Liu, J.K., Yang, G.: Malicious kgc attacks in certificateless cryptography. In: 2nd ACM Symposium on Information, Computer and Communications Security. pp. 302–311. ASIACCS (2007)
3. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) Information Security. pp. 134–148. Springer Berlin Heidelberg (2005)
4. Behnia, R., Ozmen, M.O., Yavuz, A.A.: ARIS: Authentication for real-time IoT systems. In: IEEE International Conference on Communications (ICC). pp. 1855–1867. ICC, ACM, New York, NY, USA (2019)
5. Behnia, R., Ozmen, M.O., Yavuz, A.A., Rosulek, M.: Tachyon: Fast signatures from compact knapsack. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1855–1867. CCS '18, ACM, New York, NY, USA (2018)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93). pp. 62–73. ACM, NY, USA (1993)
7. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification andăsignature schemes. Journal of Cryptology 22(1), 1–61 (Jan 2009)
8. Boneh, D., Boyen, X.: Efficient Selective-ID secure identity-based encryption without random oracles. In: Theory and Applications of Cryptographic Techniques (EUROCRYPT '04). pp. 223–238 (2004)
9. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Advances in Cryptology - CRYPTO 2001. pp. 213–229 (2001)
10. Boyen, X.: A tapestry of identity-based encryption: practical frameworks compared. IJACT 1(1), 3–21 (2008)
11. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor (May 2008)

12. Costello, C., Longa, P.: Four$Q$: Four-dimensional decompositions on a Q-curve over the mersenne prime. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 214–235. Springer Berlin Heidelberg (2015)
13. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22, 644–654 (November 1976)
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology - CRYPTO '99. pp. 537–554 (1999)
15. Galindo, D., Garcia, F.D.: A schnorr-like lightweight identity-based signature scheme. In: Progress in Cryptology - AFRICACRYPT 2009. pp. 135–148 (2009)
16. Girault, M.: Self-certified public keys. In: Proceedings of the 14th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '91). pp. 490–497 (1991)
17. Huang, X., Mu, Y., Susilo, W., Wong, D.S., Wu, W.: Certificateless signature revisited. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) Information Security and Privacy. pp. 308–322. Springer Berlin Heidelberg (2007)
18. Karati, A., Islam, S.H., Biswas, G.: A pairing-free and provably secure certificateless signature scheme. Information Sciences 450, 378 – 391 (2018)
19. Krawczyk, H.: HMQV: A high-performance secure diffie-hellman protocol. In: Advances in Cryptology - CRYPTO 2005. pp. 546–566 (2005)
20. Liu, Z., Longa, P., Pereira, G.C.C.F., Reparaz, O., Seo, H.: Four$Q$ on embedded devices with strong countermeasures against side-channel attacks. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 665–686. Springer International Publishing, Cham (2017)
21. Ozmen, M.O., Behnia, R., Yavuz, A.A.: Iod-crypt: A lightweight cryptographic framework for internet of drones. CoRR abs/1904.06829 (2019), http://arxiv.org/abs/1904.06829
22. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96. pp. 387–398. Springer Berlin Heidelberg (1996)
23. Reyzin, L., Reyzin, N.: Better than BiBa: Short one-time signatures with fast signing and verifying. In: Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02). pp. 144–153. Springer-Verlag (2002)
24. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
25. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Advances in Cryptology. pp. 47–53. Springer Berlin Heidelberg (1985)
26. Shoup, V.: A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112 (2001), https://eprint.iacr.org/2001/112
27. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In: Verdone, R. (ed.) Wireless Sensor Networks. pp. 305–320. Springer Berlin Heidelberg (2008)
28. Tomida, J., Fujioka, A., Nagai, A., Suzuki, K.: Strongly secure identity-based key exchange with single pairing operation. In: Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II. pp. 484–503 (2019)
29. Won, J., Seo, S., Bertino, E.: Certificateless cryptographic protocols for efficient drone-based smart city applications. IEEE Access 5, 3721–3749 (2017)
30. Yang, G., Tan, C.H.: Strongly secure certificateless key exchange without pairing. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. pp. 71–79. ASIACCS, ACM, New York, NY, USA (2011)