



Certificateless Cryptography

Security of Certificateless Public Key Encryption

Breno de Medeiros

Department of Computer Science

Florida State University

PKI, IBC, and Certificateless Public-Key Cryptography

- ⑥ When using public key cryptography, an infrastructure must be provided to enable verifiable binding of public keys to identities.

PKI, IBC, and Certificateless Public-Key Cryptography

- ⑥ When using public key cryptography, an infrastructure must be provided to enable verifiable binding of public keys to identities.
- ⑥ The traditional approach, Public-Key Infrastructure (PKI) defines (locally and/or globally) trusted nodes, whose keys are (locally and/or globally) known a priori.

PKI, IBC, and Certificateless Public-Key Cryptography

- ⑥ When using public key cryptography, an infrastructure must be provided to enable verifiable binding of public keys to identities.
- ⑥ The traditional approach, Public-Key Infrastructure (PKI) defines (locally and/or globally) trusted nodes, whose keys are (locally and/or globally) known a priori.
 - △ All bindings must be certified (via digital signatures) from trusted nodes. The signed document attesting to the binding of a public key and a certificate is called a *certificate*.

PKI, IBC, and Certificateless Public-Key Cryptography

- ⑥ When using public key cryptography, an infrastructure must be provided to enable verifiable binding of public keys to identities.
- ⑥ The traditional approach, Public-Key Infrastructure (PKI) defines (locally and/or globally) trusted nodes, whose keys are (locally and/or globally) known a priori.
 - △ All bindings must be certified (via digital signatures) from trusted nodes. The signed document attesting to the binding of a public key and a certificate is called a *certificate*.
 - △ Chains of certificates that follow pre-specified trust inference rules may allow for certification of bindings that are not directly attested to by the (locally and/or globally) trusted nodes.

Identity-Based Cryptography

- ⑥ Identity-based cryptography eliminates the need for certificates. There is issue of binding identities to public keys because the identities *are* the public keys. The secret keys are extracted by (locally and/or globally) trusted nodes.

Identity-Based Cryptography

- ⑥ Identity-based cryptography eliminates the need for certificates. There is issue of binding identities to public keys because the identities *are* the public keys. The secret keys are extracted by (locally and/or globally) trusted nodes.
 - △ Essentially, what makes it work is that the public key algorithms (public key encryption, digital signature verification) in an IBE scheme implicitly encode the public key of the (locally and/or globally) trusted nodes.

Identity-Based Cryptography

- ⑥ Identity-based cryptography eliminates the need for certificates. There is issue of binding identities to public keys because the identities *are* the public keys. The secret keys are extracted by (locally and/or globally) trusted nodes.
 - △ Essentially, what makes it work is that the public key algorithms (public key encryption, digital signature verification) in an IBE scheme implicitly encode the public key of the (locally and/or globally) trusted nodes.
 - △ Using Hierarchical Identity-Based Cryptography (HIBC), one may re-create trust inference rules to extend the scheme and accommodate identities that are not registered at the (locally and/or globally) trusted nodes.

Identity-Based Cryptography

- ⑥ Identity-based cryptography eliminates the need for certificates. There is issue of binding identities to public keys because the identities *are* the public keys. The secret keys are extracted by (locally and/or globally) trusted nodes.
 - △ Essentially, what makes it work is that the public key algorithms (public key encryption, digital signature verification) in an IBE scheme implicitly encode the public key of the (locally and/or globally) trusted nodes.
 - △ Using Hierarchical Identity-Based Cryptography (HIBC), one may re-create trust inference rules to extend the scheme and accommodate identities that are not registered at the (locally and/or globally) trusted nodes.
 - △ A drawback of IBC/HIBC is that revocation of public keys is awkward, requiring revocation of names. Another is that the trusted nodes have knowledge of the private keys of users.

Certificateless Cryptography

- ⑥ An alternative approach is to use *Certificateless cryptography* (CL-PKC). In certificateless cryptography, the public key algorithms that require both the name and the public key as inputs.

Certificateless Cryptography

- ⑥ An alternative approach is to use *Certificateless cryptography* (CL-PKC). In certificateless cryptography, the public key algorithms that require both the name and the public key as inputs.
 - △ The (locally and/or globally) trusted node produce partial secret keys that are combined with user-generated private key material to form a public key.

Certificateless Cryptography

- ⑥ An alternative approach is to use *Certificateless cryptography* (CL-PKC). In certificateless cryptography, the public key algorithms that require both the name and the public key as inputs.
 - △ The (locally and/or globally) trusted node produce partial secret keys that are combined with user-generated private key material to form a public key.
 - △ Certification takes place implicitly. For instance, the recipient will be able to decrypt ciphertexts/sign messages only if he knows both components of the private key, which creates the binding between identity and public key.

Certificateless Cryptography

- ⑥ An alternative approach is to use *Certificateless cryptography* (CL-PKC). In certificateless cryptography, the public key algorithms that require both the name and the public key as inputs.
 - △ The (locally and/or globally) trusted node produce partial secret keys that are combined with user-generated private key material to form a public key.
 - △ Certification takes place implicitly. For instance, the recipient will be able to decrypt ciphertexts/sign messages only if he knows both components of the private key, which creates the binding between identity and public key.
 - △ As in IBC, the public key algorithms implicitly use the (locally and/or globally) trusted node's public key. Hierarchical constructions can be used to extend the scheme and accommodate identities that are not registered at the (locally and/or globally) trusted nodes.

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

setup: Takes an input the security parameter τ and outputs a master private-public key pairs (MPK, MSK) .

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

Setup: Takes as input the security parameter τ and outputs a master private-public key pairs (MPK, MSK) .

Partial-Private-Key-Extract: Takes as input parameters MSK and an identity ID_i and produces a partial private key PSK_i .

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

Setup: Takes as input the security parameter τ and outputs a master private-public key pairs (MPK, MSK) .

Partial-Private-Key-Extract: Takes as input parameters MSK and an identity ID_i and produces a partial private key PSK_i .

Set-User-Keys: Takes as input a PSK_i and computes a user's public-private key pair (UPK_i, USK_i) .

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

Setup: Takes as input the security parameter τ and outputs a master private-public key pairs (MPK, MSK) .

Partial-Private-Key-Extract: Takes as input parameters MSK and an identity ID_i and produces a partial private key PSK_i .

Set-User-Keys: Takes as input a PSK_i and computes a user's public-private key pair (UPK_i, USK_i) .

Encrypt: Takes as input a user's identity ID_i and public key UPK_i and a message m , and produces a ciphertext c .

Certificateless Public Key Encryption

- ⑥ The (probabilistic) algorithms defining a CL-PKE are:

Setup: Takes as input the security parameter τ and outputs a master private-public key pairs (MPK, MSK) .

Partial-Private-Key-Extract: Takes as input parameters MSK and an identity ID_i and produces a partial private key PSK_i .

Set-User-Keys: Takes as input a PSK_i and computes a user's public-private key pair (UPK_i, USK_i) .

Encrypt: Takes as input a user's identity ID_i and public key UPK_i and a message m , and produces a ciphertext c .

Decrypt: Takes as input a ciphertext c , and a user's private key USK_i and returns \perp or a message m .

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Extract-Partial(ID_i): \mathcal{S} returns to \mathcal{A} the results of running `Partial-Private-Key-Extract`(MSK, ID_i).

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Extract-Partial(ID_i): \mathcal{S} returns to \mathcal{A} the results of running $\text{Partial-Private-Key-Extract}(\mathcal{MSK}, ID_i)$.

Compromise(UPK_i): \mathcal{S} returns the value USK_i , if UPK_i is an honest user's public key.

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Extract-Partial(ID_i): \mathcal{S} returns to \mathcal{A} the results of running $\text{Partial-Private-Key-Extract}(\mathcal{MSK}, ID_i)$.

Compromise(UPK_i): \mathcal{S} returns the value USK_i , if UPK_i is an honest user's public key.

Replace(UPK_i, UPK'_i): \mathcal{S} will replace the key of the i -th user.

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Extract-Partial(ID_i): \mathcal{S} returns to \mathcal{A} the results of running $\text{Partial-Private-Key-Extract}(\mathcal{MSK}, ID_i)$.

Compromise(UPK_i): \mathcal{S} returns the value USK_i , if UPK_i is an honest user's public key.

Replace(UPK_i, UPK'_i): \mathcal{S} will replace the key of the i -th user.

Decrypt(UPK_i, c): \mathcal{S} runs $\text{Decrypt}(USK_i, c)$ and returns results to \mathcal{A} .

Security model for CL-PKE

- ⑥ A simulator \mathcal{S} generates the master parameters and the public/private keys of honest parties, chooses random bit b .
- ⑥ An attacker \mathcal{A} can request that \mathcal{S} answer the following types of queries:

Extract-Partial(ID_i): \mathcal{S} returns to \mathcal{A} the results of running $\text{Partial-Private-Key-Extract}(\mathcal{MSK}, ID_i)$.

Compromise(UPK_i): \mathcal{S} returns the value USK_i , if UPK_i is an honest user's public key.

Replace(UPK_i, UPK'_i): \mathcal{S} will replace the key of the i -th user.

Decrypt(UPK_i, c): \mathcal{S} runs $\text{Decrypt}(USK_i, c)$ and returns results to \mathcal{A} .

Challenge(UPK_i, m_0, m_1): \mathcal{S} sends to \mathcal{A} the result of $\text{Encrypt}(ID_i, UPK_i, m_b)$.

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.
- ⑥ The simulation works in a find-then-guess paradigm. The adversary can ask queries, then chooses an identity ID_i , with current public key UPK_i and pair of messages m_0, m_1 , and asks to be challenged on that, receiving as answer ciphertext C .

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.
- ⑥ The simulation works in a find-then-guess paradigm. The adversary can ask queries, then chooses an identity ID_i , with current public key UPK_i and pair of messages m_0, m_1 , and asks to be challenged on that, receiving as answer ciphertext C .
- ⑥ The adversary is not allowed to:

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.
- ⑥ The simulation works in a find-then-guess paradigm. The adversary can ask queries, then chooses an identity ID_i , with current public key UPK_i and pair of messages m_0, m_1 , and asks to be challenged on that, receiving as answer ciphertext C .
- ⑥ The adversary is not allowed to:
 - △ Ask for the secret key corresponding to the challenge UPK_i .

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.
- ⑥ The simulation works in a find-then-guess paradigm. The adversary can ask queries, then chooses an identity ID_i , with current public key UPK_i and pair of messages m_0, m_1 , and asks to be challenged on that, receiving as answer ciphertext C .
- ⑥ The adversary is not allowed to:
 - △ Ask for the secret key corresponding to the challenge UPK_i .
 - △ Ask for decryption of C .

Security simulation rules

- ⑥ In a variant (type-II) simulation, \mathcal{S} gives the master secret key to \mathcal{A} . An attacker is labeled as type-I or type-II, respectively.
- ⑥ The simulation works in a find-then-guess paradigm. The adversary can ask queries, then chooses an identity ID_i , with current public key UPK_i and pair of messages m_0, m_1 , and asks to be challenged on that, receiving as answer ciphertext C .
- ⑥ The adversary is not allowed to:
 - △ Ask for the secret key corresponding to the challenge UPK_i .
 - △ Ask for decryption of C .
 - △ Ask for the partial secret key of the target identity UPK_i (in any phase) if the challenge was asked on a replaced key UPK'_i .

Security Definition for CL-PKC

- ⑥ IND-CCA: A CL-PKE scheme is IND-CCA if the adversary has negligible advantage over simulations of either type, where its advantage is defined as usual, by:

$$\mathbf{Adv}_{\mathcal{CL-PKE}, \mathcal{A}_{\text{type}}}^{\text{IND-CCA-type}}(\tau) = \text{Prob} \left[\mathbf{Exp}_{\mathcal{CL-PKE}, \mathcal{A}_{\text{type}}}^{\text{IND-CCA-type-1}}(\tau) = 1 \right] \\ - \text{Prob} \left[\mathbf{Exp}_{\mathcal{CL-PKE}, \mathcal{A}_{\text{type}}}^{\text{IND-CCA-type-0}}(\tau) = 1 \right],$$

where type stands for either type I or type II simulations, and $\mathbf{Exp}_{\mathcal{CL-PKE}, \mathcal{A}_{\text{type}}}^{\text{IND-CCA-type-}b}(\tau)$ denotes a simulation where \mathcal{S} has input the security parameter τ and encrypts message m_b in the challenge.



CL-PKC

Construction of an IND-CCA CL-PKC Scheme

Breno de Medeiros

Department of Computer Science

Florida State University

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .
- ⑥ A pairing $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is:

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .
- ⑥ A pairing $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is:
 - △ Efficiently computable;

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .
- ⑥ A pairing $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is:
 - △ Efficiently computable;
 - △ Bilinear, i.e., $e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$, for any $a \in \mathbb{Z}_p$.

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .
- ⑥ A pairing $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is:
 - △ Efficiently computable;
 - △ Bilinear, i.e., $e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$, for any $a \in \mathbb{Z}_p$.
 - △ Non-degenerate, i.e., if g_1, g_2 generate $\mathbb{G}_1, \mathbb{G}_2$, respectively, then $e(g_1, g_2)$ generates \mathbb{G}_T .

Introduction to pairings

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p .
- ⑥ A pairing $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is:
 - △ Efficiently computable;
 - △ Bilinear, i.e., $e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$, for any $a \in \mathbb{Z}_p$.
 - △ Non-degenerate, i.e., if g_1, g_2 generate $\mathbb{G}_1, \mathbb{G}_2$, respectively, then $e(g_1, g_2)$ generates \mathbb{G}_T .
- ⑥ Pairings for cryptographic use are generated with \mathbb{G}_1 and \mathbb{G}_2 as prime order subgroups of special “elliptic curves of small embedding degree.” The details are irrelevant for this discussion.

Pairing-based cryptographic assumptions

- Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p , and $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a pairing. For simplicity, we will discuss only the case $\mathbb{G}_1 = \mathbb{G}_2$.

Pairing-based cryptographic assumptions

- Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p , and $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a pairing. For simplicity, we will discuss only the case $\mathbb{G}_1 = \mathbb{G}_2$.
- The Bilinear Diffie-Hellman problem is the following:

Pairing-based cryptographic assumptions

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p , and $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a pairing. For simplicity, we will discuss only the case $\mathbb{G}_1 = \mathbb{G}_2$.
- ⑥ The Bilinear Diffie-Hellman problem is the following:
 - △ Given (P_1, P_2, P_3, P_4) all points in \mathbb{G}_1 , compute $e(P_1, P_1)^{abc}$, where $P_2 = P_1^a$, $P_3 = P_1^b$, and $P_4 = P_1^c$.

Pairing-based cryptographic assumptions

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p , and $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a pairing. For simplicity, we will discuss only the case $\mathbb{G}_1 = \mathbb{G}_2$.
- ⑥ The Bilinear Diffie-Hellman problem is the following:
 - △ Given (P_1, P_2, P_3, P_4) all points in \mathbb{G}_1 , compute $e(P_1, P_1)^{abc}$, where $P_2 = P_1^a$, $P_3 = P_1^b$, and $P_4 = P_1^c$.
- ⑥ The Generalized Bilinear Diffie-Hellman problem is the following:

Pairing-based cryptographic assumptions

- ⑥ Let \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T be groups of same prime order p , and $e(,) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a pairing. For simplicity, we will discuss only the case $\mathbb{G}_1 = \mathbb{G}_2$.
- ⑥ The Bilinear Diffie-Hellman problem is the following:
 - △ Given (P_1, P_2, P_3, P_4) all points in \mathbb{G}_1 , compute $e(P_1, P_1)^{abc}$, where $P_2 = P_1^a$, $P_3 = P_1^b$, and $P_4 = P_1^c$.
- ⑥ The Generalized Bilinear Diffie-Hellman problem is the following:
 - △ Given (P_1, P_2, P_3, P_4) all points in \mathbb{G}_1 , find Q in \mathbb{G}_1 and compute $e(P_1, Q)^{abc}$, with a , b , and c as above.

Basic CL-PKE scheme, only IND-OW secure

- ⑥ We now consider the construction of a basic CL-PKE scheme, which is secure only in simulations where the attacker does not make any ciphertext-decryption queries, and the attacker's objective is to decrypt a target ciphertext (not distinguish between two ciphertexts).

Basic CL-PKE scheme, only IND-OW secure

- ⑥ We now consider the construction of a basic CL-PKE scheme, which is secure only in simulations where the attacker does not make any ciphertext-decryption queries, and the attacker's objective is to decrypt a target ciphertext (not distinguish between two ciphertexts).

Setup(τ): The trusted party PKG generates group \mathbb{G}_1 , \mathbb{G}_T , of order p , and pairing $e(\cdot, \cdot)$ to match the security parameter τ . It chooses generator g_1 of \mathbb{G}_1 , a random s in \mathbb{Z}_p , computes $y_1 = g_1^s$, and cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$, and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^n$, where n is the length of ciphertexts. It computes

Basic CL-PKE scheme, only IND-OW secure

- ⑥ We now consider the construction of a basic CL-PKE scheme, which is secure only in simulations where the attacker does not make any ciphertext-decryption queries, and the attacker's objective is to decrypt a target ciphertext (not distinguish between two ciphertexts).

Setup(τ): The trusted party PKG generates group $\mathbb{G}_1, \mathbb{G}_T$, of order p , and pairing $e(\cdot, \cdot)$ to match the security parameter τ . It chooses generator g_1 of \mathbb{G}_1 , a random s in \mathbb{Z}_p , computes $y_1 = g_1^s$, and cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$, and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^n$, where n is the length of ciphertexts. It computes

- △ $\mathcal{MPK} = (\mathbb{G}_1, \mathbb{G}_T, g_1, y_1, H_1(\cdot), H_2(\cdot),)$ and

Basic CL-PKE scheme, only IND-OW secure

- ⑥ We now consider the construction of a basic CL-PKE scheme, which is secure only in simulations where the attacker does not make any ciphertext-decryption queries, and the attacker's objective is to decrypt a target ciphertext (not distinguish between two ciphertexts).

Setup(τ): The trusted party PKG generates group \mathbb{G}_1 , \mathbb{G}_T , of order p , and pairing $e(\cdot, \cdot)$ to match the security parameter τ . It chooses generator g_1 of \mathbb{G}_1 , a random s in \mathbb{Z}_p , computes $y_1 = g_1^s$, and cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$, and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^n$, where n is the length of ciphertexts. It computes

- △ $MPK = (\mathbb{G}_1, \mathbb{G}_T, g_1, y_1, H_1(\cdot), H_2(\cdot),)$ and
- △ $MSK = (MPK; s)$.

Basic CL-PKE scheme (2)



Basic CL-PKE scheme (2)

Partial-Private-Key-Extract(\mathcal{MSK}, ID_i): Returns
 $\mathcal{PSK}_i = sQ_i \in \mathbb{G}_1^*$, where $Q_i = H_1(ID_i)$.

Basic CL-PKE scheme (2)

Partial-Private-Key-Extract(\mathcal{MSK}, ID_i): Returns

$$\mathcal{PSK}_i = sQ_i \in \mathbb{G}_1^*, \text{ where } Q_i = H_1(ID_i).$$

Set-User-Keys(\mathcal{PSK}_i): Generate x_i at random in \mathbb{Z}_p^* , and sets

$$\mathcal{UPK}_i = (X_i, Y_i) = (g_1^{x_i}, y_1^{x_i}), \text{ and } \mathcal{USK}_i = \mathcal{PSK}_i^{x_i}.$$

Basic CL-PKE scheme (2)

Partial-Private-Key-Extract(\mathcal{MSK}, ID_i): Returns

$$\mathcal{PSK}_i = sQ_i \in \mathbb{G}_1^*, \text{ where } Q_i = H_1(ID_i).$$

Set-User-Keys(\mathcal{PSK}_i): Generate x_i at random in \mathbb{Z}_p^* , and sets

$$\mathcal{UPK}_i = (X_i, Y_i) = (g_1^{x_i}, y_1^{x_i}), \text{ and } \mathcal{USK}_i = \mathcal{PSK}_i^{x_i}.$$

Encrypt($ID_i, \mathcal{UPK}_i; m$): Check that $(X_i, Y_i) = \mathcal{UPK}_i$ satisfies $e(X_i, y_1) = e(Y_i, g_1)$; if not, abort and output \perp . Else, compute $Q_i = H_1(ID_i)$, choose r at random in \mathbb{Z}_p^* , and return:

$$C = (g_1^r, m \oplus H_2(e(Q_i, Y_i)^r))$$

Basic CL-PKE scheme (2)

Partial-Private-Key-Extract(MSK, ID_i): Returns

$$PSK_i = sQ_i \in \mathbb{G}_1^*, \text{ where } Q_i = H_1(ID_i).$$

Set-User-Keys(PSK_i): Generate x_i at random in \mathbb{Z}_p^* , and sets

$$UPK_i = (X_i, Y_i) = (g_1^{x_i}, y_1^{x_i}), \text{ and } USK_i = PSK_i^{x_i}.$$

Encrypt($ID_i, UPK_i; m$): Check that $(X_i, Y_i) = UPK_i$ satisfies $e(X_i, y_1) = e(Y_i, g_1)$; if not, abort and output \perp . Else, compute $Q_i = H_1(ID_i)$, choose r at random in \mathbb{Z}_p^* , and return:

$$C = (g_1^r, m \oplus H_2(e(Q_i, Y_i)^r))$$

Decrypt(USK_i, C): Parse C as (U, V) , with $U \in \mathbb{G}_1$ and $V \in \{0, 1\}^n$.
Return $V \oplus H_2(e(USK_i, U))$.

Full, IND-CCA CL-PKE scheme

- ⑥ The IND-CCA scheme is a modification to prevent malleability of ciphertexts. The setup is the same as the basic scheme, except that the trusted party chooses two more hash functions, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_p^*$, and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$. The generation of private and public keys is as in the basic scheme. The changes are:

Full, IND-CCA CL-PKE scheme

- ⑥ The IND-CCA scheme is a modification to prevent malleability of ciphertexts. The setup is the same as the basic scheme, except that the trusted party chooses two more hash functions, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_p^*$, and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$. The generation of private and public keys is as in the basic scheme. The changes are:

Encrypt($ID_i, UPK_i; m$): Check that $(X_i, Y_i) = UPK_i$ satisfies $e(X_i, y_1) = e(Y_i, g_1)$; if not, abort and output \perp . Else:

Full, IND-CCA CL-PKE scheme

- ⑥ The IND-CCA scheme is a modification to prevent malleability of ciphertexts. The setup is the same as the basic scheme, except that the trusted party chooses two more hash functions, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_p^*$, and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$. The generation of private and public keys is as in the basic scheme. The changes are:

Encrypt($ID_i, UPK_i; m$): Check that $(X_i, Y_i) = UPK_i$ satisfies $e(X_i, y_1) = e(Y_i, g_1)$; if not, abort and output \perp . Else:

- △ Choose random $\sigma \in \{0, 1\}^n$, and set $r = H_3(\sigma, m)$.

Full, IND-CCA CL-PKE scheme

- ⑥ The IND-CCA scheme is a modification to prevent malleability of ciphertexts. The setup is the same as the basic scheme, except that the trusted party chooses two more hash functions, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_p^*$, and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$. The generation of private and public keys is as in the basic scheme. The changes are:

Encrypt($ID_i, UPK_i; m$): Check that $(X_i, Y_i) = UPK_i$ satisfies

$e(X_i, y_1) = e(Y_i, g_1)$; if not, abort and output \perp . Else:

- △ Choose random $\sigma \in \{0, 1\}^n$, and set $r = H_3(\sigma, m)$.
- △ Compute $Q_i = H_1(ID_i)$ and

$$C = (g_1^r, \sigma \oplus H_2(e(Q_i, Y_i)^r), m \oplus H_4(\sigma))$$

Full, IND-CCA CL-PKE scheme (2)



Full, IND-CCA CL-PKE scheme (2)

Decrypt(USK_i, C): Parse C as (U, V, W) , where $U \in \mathbb{G}_1$, and V, W in $\{0, 1\}^n$.

Full, IND-CCA CL-PKE scheme (2)

Decrypt(USK_i, C): Parse C as (U, V, W) , where $U \in \mathbb{G}_1$, and V, W in $\{0, 1\}^n$.

⑥ Compute $\sigma' = V \oplus H_2(e(USK_i, U))$ and $M' = W \oplus H_4(\sigma')$.

Full, IND-CCA CL-PKE scheme (2)

Decrypt(USK_i, C): Parse C as (U, V, W) , where $U \in \mathbb{G}_1$, and V, W in $\{0, 1\}^n$.

- ⑥ Compute $\sigma' = V \oplus H_2(e(USK_i, U))$ and $M' = W \oplus H_4(\sigma')$.
- ⑥ Compute $r' = H_3(\sigma', M')$ and check if $U = g_1^{r'}$.

Full, IND-CCA CL-PKE scheme (2)

Decrypt(USK_i, C): Parse C as (U, V, W) , where $U \in \mathbb{G}_1$, and V, W in $\{0, 1\}^n$.

- ⑥ Compute $\sigma' = V \oplus H_2(e(USK_i, U))$ and $M' = W \oplus H_4(\sigma')$.
- ⑥ Compute $r' = H_3(\sigma', M')$ and check if $U = g_1^{r'}$.
- ⑥ If the check fails, output \perp . Else output M' .

Full, IND-CCA CL-PKE scheme (2)

Decrypt(USK_i, C): Parse C as (U, V, W) , where $U \in \mathbb{G}_1$, and V, W in $\{0, 1\}^n$.

- ⑥ Compute $\sigma' = V \oplus H_2(e(USK_i, U))$ and $M' = W \oplus H_4(\sigma')$.
- ⑥ Compute $r' = H_3(\sigma', M')$ and check if $U = g_1^{r'}$.
- ⑥ If the check fails, output \perp . Else output M' .

Theorem [S. Al-Riyami, K. Patterson (2003)] The above scheme is IND-CCA secure under the assumption that the GBDH problem is hard and if $H_1(\cdot)$, $H_2(\cdot)$, $H_3(\cdot)$, and $H_4(\cdot)$ are modeled as random oracles.