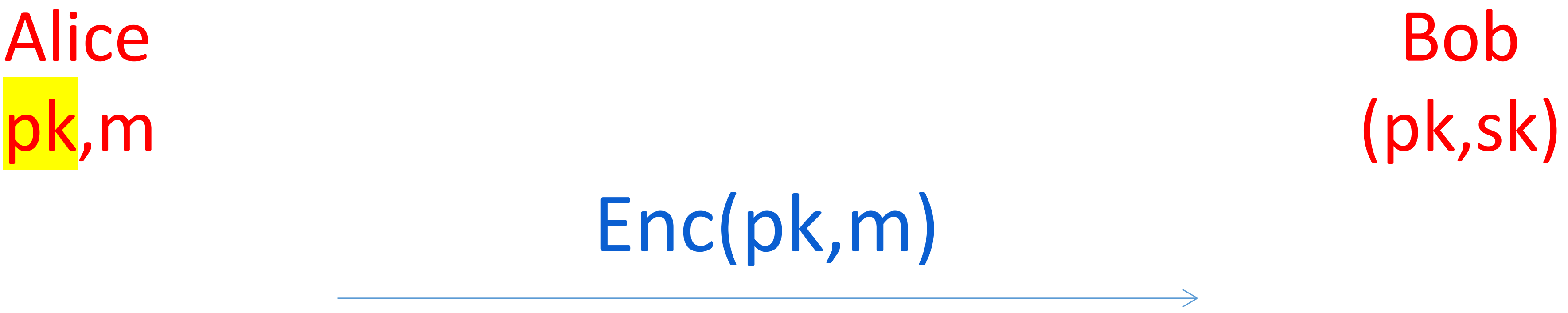
CSCI471/971
Modern Cryptography

Identity-based Cryptography

Motivations

Public-Key Encryption

Bob generates a key pair (pk,sk). pk is public key and published to all others, while sk is secret and only known by Bob.



Alice

pk,m

Bob

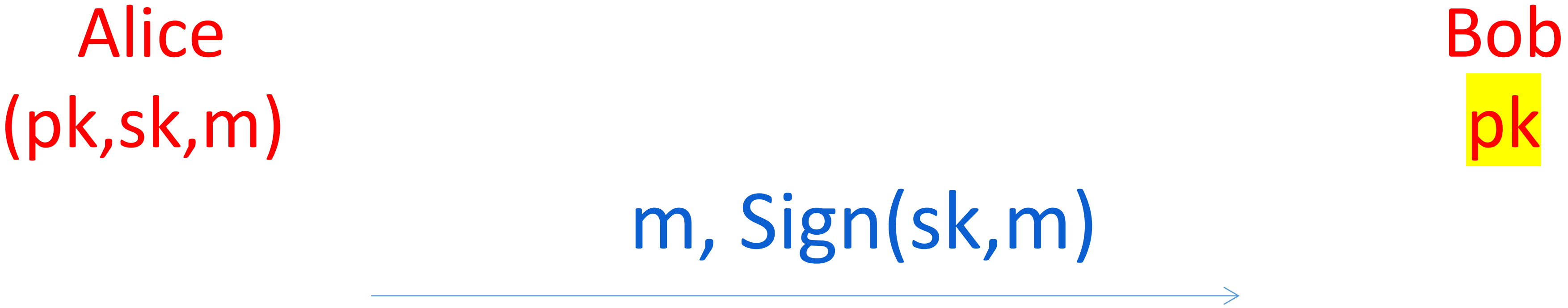
(pk,sk)

Enc(pk,m)

The data that Alice sent to Bob is unknown to the adversary.
Bob doens’t even need to know who is Alice (no need to share a secret key!)

Digital Signatures

Alice generates a key pair (pk,sk). pk is public key and published to all others, while sk is secret and only known by Alice.



Alice

(pk,sk,m)

Bob

pk

m, Sign(sk,m)

The data that Alice sent to Bob cannot be modified by the adversary.
Bob only needs to know that pk belongs to Alice (no need to share a secret key!)

In Public-key encryption and digital signatures, we assume that an entity pre-knows the public key of another entity.

A public key looks like:

```
30 82 01 0a 02 82 01 01 00 dd 9e e4 f6 88 b5 0d d7 a1 5f bc 25 6e cc 44 14 cf 34 e2 b5 73 09 1e e4
4b 12 52 38 95 36 1a c6 66 ed f0 c8 03 c9 b3 43 45 4e 0d 6a 92 4b 1b eb 94 60 5b 11 b9 15 79 b1
a5 f6 fc 5d bf a4 30 59 84 02 dd 3f 6d 21 6a 44 b7 18 1c 24 fc f5 02 2e 87 0c 20 3e c6 c5 b6 9f ad 16
1b 76 86 e9 73 9c 8d 31 60 3a a0 f0 2f da ad 8e f6 74 c9 81 d3 ea f7 5d ab 5d bb 05 63 b0 78 55 ed
72 13 a4 42 43 72 23 73 c0 de 33 9b 44 5c 89 a9 8a 90 d1 99 be bc f7 21 21 5f fb 22 8a 5c 50 b9 69
7c dc 87 92 ed 79 56 ed 32 55 41 9f af 41 f6 da d4 70 88 e9 a3 41 1d 66 9f a6 98 d2 7e 5b a9 52 38
1c 56 b4 cc 45 62 72 0c c7 f7 ef 2c 47 0a 3b 1a 7a ac e7 ae a9 a8 1e 98 43 b0 58 56 e9 41 44 72 e6
da 67 1c d7 b6 f4 e6 b4 90 5c b5 0a 98 b3 23 0c e7 35 6d 10 14 73 0e 94 5d 7c 4e 0a 18 f4 05 20 67
9f 02 03 01 00 01
```

This is because pk is computed from sk while sk is a random string.

Scenario: How to let Bob know pk

Alice

(pk,sk,m)

Bob

M →

M= My public key is “30 82 01 0a 02 82 01 01 00 dd 9e e4 f6 88 b5 0d d7 a1 5f bc 25 6e cc 44 14 cf 34 e2 b5 73 09 1e e4 4b 12 52 38 95 36 1a c6 66 ed f0 c8 03 c9 b3 43 45 4e 0d 6a 92 4b 1b eb 94 60 5b 11 b9 15 79 b1 a5 f6 fc 5d bf a4 30 59 84 02 dd 3f 6d 21 6a 44 b7 18 1c 24 fc f5 02 2e 87 0c 20 3e c6 c5 b6 9f ad 16 1b 76 86 e9 73 9c 8d 31 60 3a a0 f0 2f da ad 8e f6 74 c9 81 d3 ea f7 5d ab 5d bb 05 63 b0 78 55 ed 72 13 a4 42 43 72 23 73 c0 de 33 9b 44 5c 89 a9 8a 90 d1 99 be bc f7 21 21 5f fb 22 8a 5c 50 b9 69 7c dc 87 92 ed 79 56 ed 32 55 41 9f af 41 f6 da d4 70 88 e9 a3 41 1d 66 9f a6 98 d2 7e 5b a9 52 38 1c 56 b4 cc 45 62 72 0c c7 f7 ef 2c 47 0a 3b 1a 7a ac e7 ae a9 a8 1e 98 43 b0 58 56 e9 41 44 72 e6 da 67 1c d7 b6 f4 e6 b4 90 5c b5 0a 98 b3 23 0c e7 35 6d 10 14 73 0e 94 5d 7c 4e 0a 18 f4 05 20 67 9f 02 03 01 00 01”

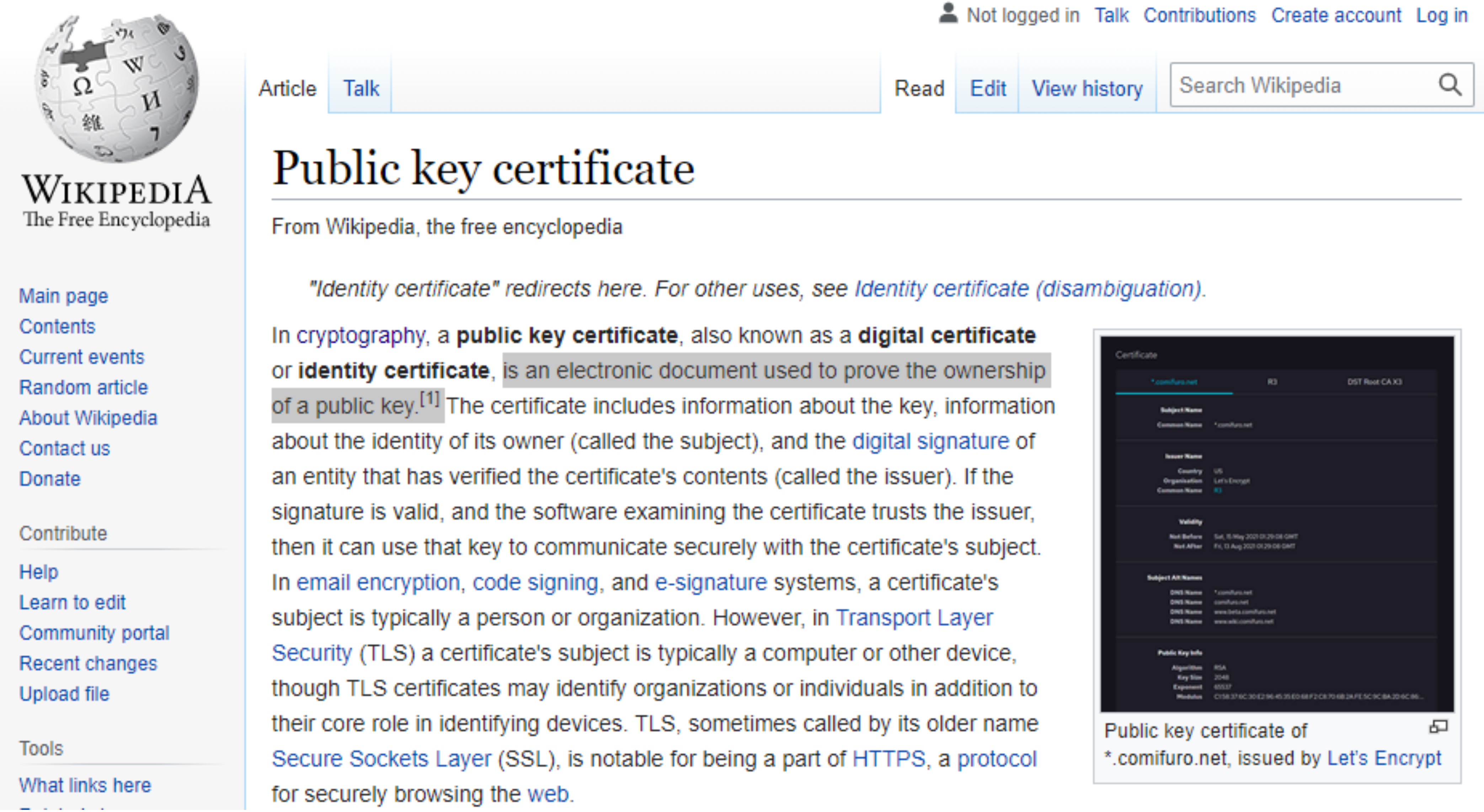
This is not secure!!!! (because the public key could be replaced by the adversary.)

Adversary

Certificate (Idea)

- It is impossible to directly assume that an entity knows the public key of another entity.
- It is reasonable to assume that we ALL know the public key of a trusted third party.
- When we install such as chrome, the system beleives that we have trusted the third paties embeded in the chrome.

Certificate (Idea)



Public key certificate

From Wikipedia, the free encyclopedia

"Identity certificate" redirects here. For other uses, see Identity certificate (disambiguation).

In cryptography, a **public key certificate**, also known as a **digital certificate** or **identity certificate**, is an electronic document used to prove the ownership of a public key.[1] The certificate includes information about the key, information about the identity of its owner (called the subject), and the digital signature of an entity that has verified the certificate's contents (called the issuer). If the signature is valid, and the software examining the certificate trusts the issuer, then it can use that key to communicate securely with the certificate's subject. In email encryption, code signing, and e-signature systems, a certificate's subject is typically a person or organization. However, in Transport Layer Security (TLS) a certificate's subject is typically a computer or other device, though TLS certificates may identify organizations or individuals in addition to their core role in identifying devices. TLS, sometimes called by its older name Secure Sockets Layer (SSL), is notable for being a part of HTTPS, a protocol for securely browsing the web.

Certificate

*.comifuro.net | R3 | DST Root CA X3

Subject Name
Common Name *.comifuro.net

Issuer Name
Country US
Organisation Let's Encrypt
Common Name R3

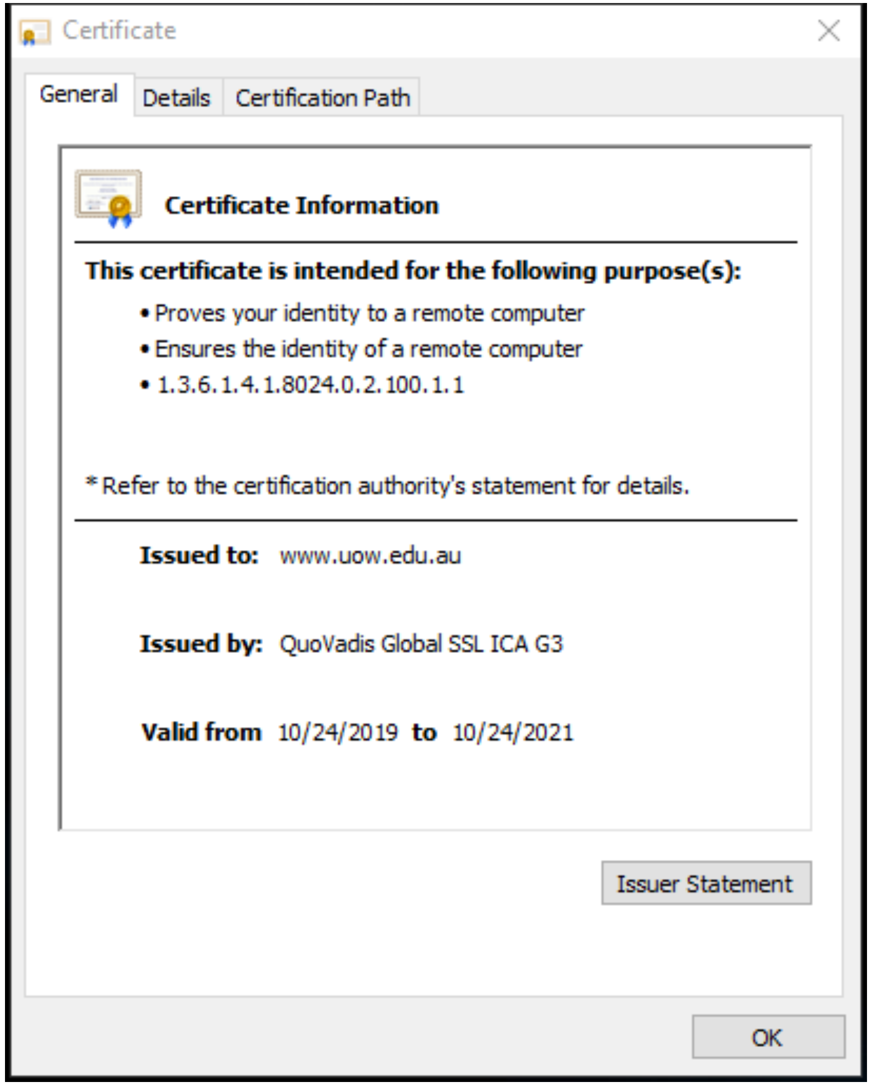
Validity
Not Before Sat, 15 May 2021 01:29:08 GMT
Not After Fri, 13 Aug 2021 01:29:08 GMT

Subject Alt Names
DNS Name *.comifuro.net
DNS Name comifuro.net
DNS Name www.beta.comifuro.net
DNS Name www.wiki.comifuro.net

Public Key Info
Algorithm RSA
Key Size 2048
Exponent 65537
Modulus C1:58:37:6C:30:E2:96:45:35:E0:68:F2:C8:70:6B:2A:FE:5C:9C:BA:2D:6C:86...

Public key certificate of *.comifuro.net, issued by Let's Encrypt

Certificate (Idea)



Certificate Information

This certificate is intended for the following purpose(s):

- Proves your identity to a remote computer
- Ensures the identity of a remote computer
- 1.3.6.1.4.1.8024.0.2.100.1.1

* Refer to the certification authority's statement for details.

Issued to: www.uow.edu.au

Issued by: QuoVadis Global SSL ICA G3

Valid from 10/24/2019 **to** 10/24/2021

- The public key of QuoVadis is pk*. (Certificate Authority)

(We trust this public key. Suppose that everyone trusts this)

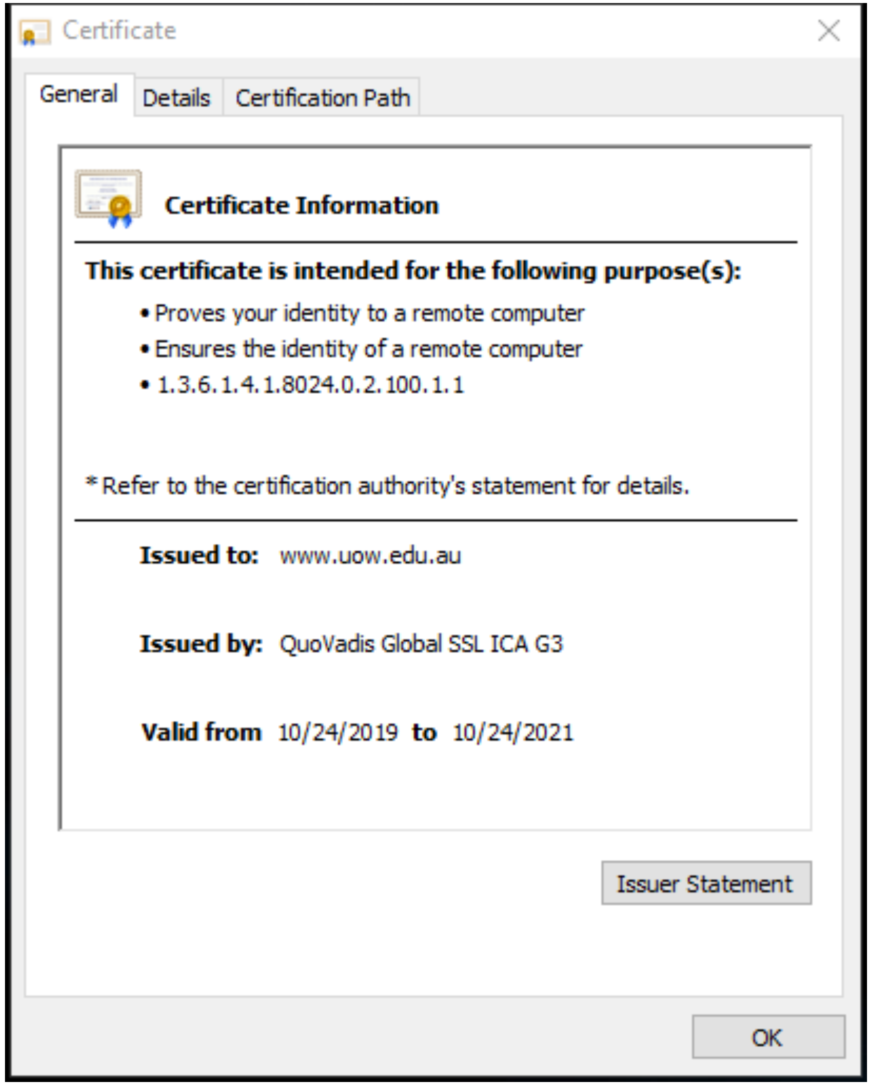
- When we browse “www.uow.edu.au”, the web server sends its public key pk to us. But we don’t know pk belongs to UOW or not.

- The CA uses sk* to genera a digital signature S_UOW on

M=”pk belongs to UOW”

- With pk and S_UOW, we know that pk bleongs to UOW

Certificate (Idea)



Certificate Information

This certificate is intended for the following purpose(s):

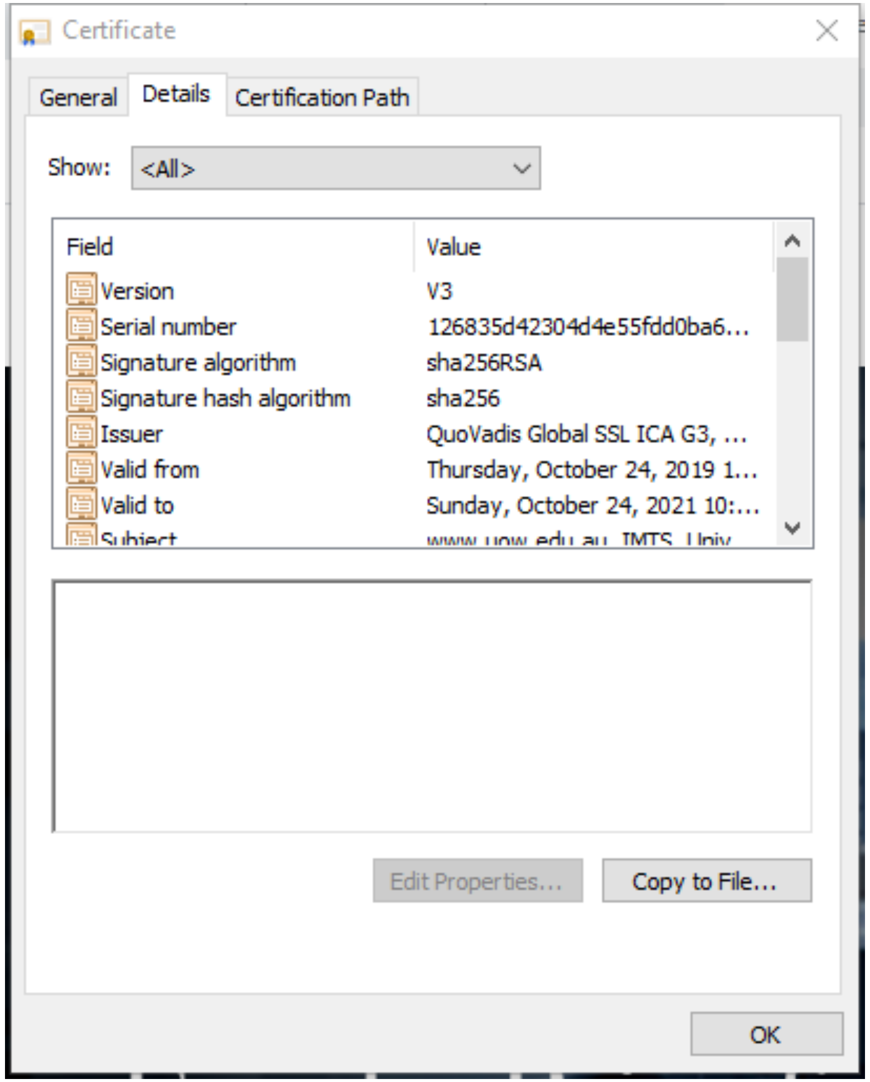
- Proves your identity to a remote computer
- Ensures the identity of a remote computer
- 1.3.6.1.4.1.8024.0.2.100.1.1

* Refer to the certification authority's statement for details.

Issued to: www.uow.edu.au

Issued by: QuoVadis Global SSL ICA G3

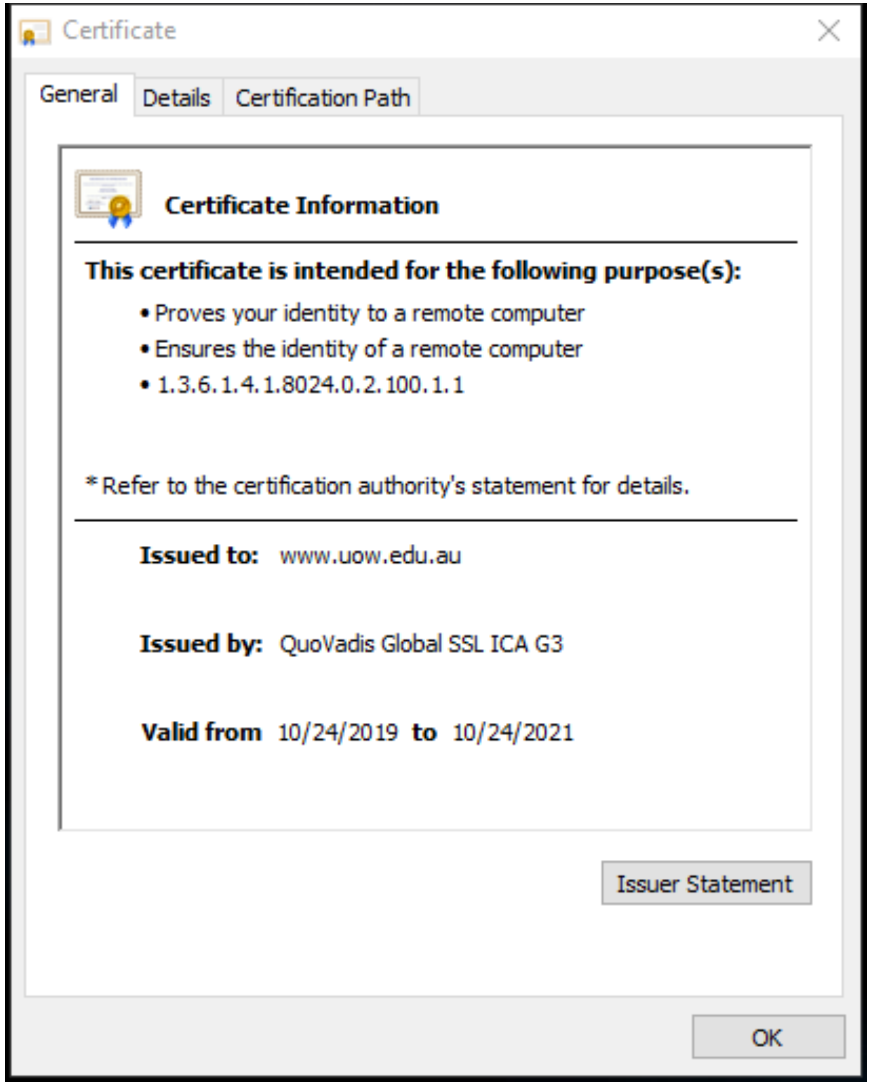
Valid from 10/24/2019 **to** 10/24/2021



Show: <All>

Field	Value
Version	V3
Serial number	126835d42304d4e55fdd0ba6...
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	QuoVadis Global SSL ICA G3, ...
Valid from	Thursday, October 24, 2019 1...
Valid to	Sunday, October 24, 2021 10:...
Subject	www.uow.edu.au, IMTS, Univ

Certificate (Disadvantages)



Certificate Information

This certificate is intended for the following purpose(s):

- Proves your identity to a remote computer
- Ensures the identity of a remote computer
- 1.3.6.1.4.1.8024.0.2.100.1.1

* Refer to the certification authority's statement for details.

Issued to: www.uow.edu.au

Issued by: QuoVadis Global SSL ICA G3

Valid from 10/24/2019 **to** 10/24/2021

- CA needs to generate a certificate for each user (pk,sk)
- CA needs to manage all certificates
- Encryptors need to receive certificate to verify pk

From **public-key cryptography** to identity-based crytography

- In PKC,

pk= random string computed from sk

sk = randomly chosen

We need to verify pk using certificate generated by CA.

- In IBC

pk= any identity information, UOW

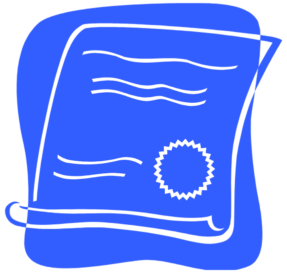
sk= computed by CA with pk and a (master) secret key msk

We don’t need to verify pk as long as we know the master public key mpk.

From public-key cryptography to identity-based crytography

- IBC Public Key:

fuchun@uow.edu.au



X

- RSA Public Key:

Public exponent=0x10001

Modulus=135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

From **public-key cryptography** to identity-based crytography

- In PKC,

We cannot send a sensitive data to Bob without knowing his pk.

Suppoe Bob’s email address:

Bob@gmail.com

- In IBC

We can send a sensitive data to Bob as long as we know mpk.

1. We encrypt with Bob@gmail.com as the public key
2. Bob applies the secret key from CA.
3. Bob can decrypt CT then.

From **public-key cryptography** to **identity-based crytography**

- In PKC,

pk= random string computed from sk

sk = randomly chosen

pk is called public key

sk is called secret key

Using CA to generate certificates

- In IBC

pk= any identity information, UOW

sk= computed with pk and msk

pk= ID is called public key

sk= d_{ID} is called private key

Using PKG (private-key generator) to compute private keys

History of IBC

- The concept was formulated by Adi Shamir in 1984

Identity-Based Encryption (IBE)

Identity-based signatures (IBC)

- Concrete IBE schemes in 2001
  - **Boneh and Franklin [crypto 2001, SIAM J. of computing 2003]**
    - Originated from Sakai, Ohgishi, Kasahara
  - Cocks [IMA International Conference on Cryptography and Coding 2001]

Definition

Identity-Based Encryption (IBE)

- Setup(λ):Taking as input a security parameter λ, the P.P.T. algorithms returns (mpk,msk). mpk is the master public key and msk is the master secret key.

- KeyGen(ID, msk): Taking as input (ID, msk), the P.P.T. algorithms returns a private key d_{ID} for ID.

- Enc(M,mpk,ID): Taking as input a message M and (mpk,ID), the P.P.T. algorithm returns a ciphertext denoted by CT←Enc(M,mpk,ID)

- Dec(CT,mpk,d_{ID}): Taking as input (CT, mpk, d_{ID}), the P.P.T. algorithm returns M or ⊥(invalid ciphertext).

Identity-Based Signature (IBS)

- Setup(λ):Taking as input a security parameter λ, the P.P.T. algorithms returns (mpk,msk). mpk is the master public key and msk is the master secret key.

- KeyGen(ID, msk): Taking as input (ID, msk), the P.P.T. algorithms returns a private key d_{ID} for ID.

- Sign(M,mpk,d_{ID}): Taking as input a message M and (mpk,d_{ID}), the P.P.T. algorithm returns a signature denoted by S←Sign(M,mpk,d_{ID})

- Verify(M,S,mpk,ID): Taking as input (M,S,mpk,ID), the P.P.T. algorithm returns 1 or 0(invalid signature).

From **public-key cryptography** to identity-based crytography

- In PKE,

IND-CPA

IND-CCA

- In IBE

IND-ID-CPA

IND-ID-CCA

- Indistinguishable Security (IND): Given a ciphertext CT* and two messages M_0 and M_1 where CT* is created with one of random message, it is hard for the adversary to guess c from {0,1}.

- Chosen-Plaintext Attack (CPA): The adversary can choose any plaintext to know its ciphertext.

- Chosen-Ciphertext Attack (CCA):
  The adversary can choose any plaintext to know its ciphertext.
  The adversary can choose any ciphertxt to know its plaintext.

Security Model (IND-ID-CCA)

Setup: The challenger chooses a key pair (mpk,msk) and mpk is given to the adversary.

Phase 1:

The adversary can choose any identity ID for private key query.
The adversary can choose any (ID, CT) for decryption query.

Challenge: The adversary chooses any two different messages M_0 and M_1 and one challenge identity ID*. The challenger chooses a random c and computes the challenge ciphertext

CT*=Enc(M_c, mpk, ID*),

Phase 2: The same as Phase 1 except that no private key query on ID* and no decryption query on (ID*,CT*).

Guess: The adversary returns the guess c' and wins if c'=c.

We say that the encryption is secure if every P.P.T adversary can only win the game with **negligible** probability defined as

Pr[c'=c]-½

Security Model (IND-ID-CCA)

Phase 2: The same as Phase 1 except that no private key query on ID* and no decryption query on (ID*,CT*).

Question 1: Can the adversary make the decryption query on (ID’, CT*)?

Question 2: Can the adversary make the decryption query on (ID*, CT’)?

From **public-key cryptography** to identity-based crytography

- In PKS,

EUF-CMA

- In IBS

EUF-ID-CMA

- ***Existentially Unforgeable: (EUF)***
  Forge a (message, sig) pair for any message that has not yet been signed.

- ***Chosen message attack (CMA)***:
  The adversary can choose any message to know its signature before the attack.

Security Model (EUF-ID-CMA)

Setup: The challenger chooses a key pair (mpk,msk) and mpk is given to the adversary.

Query:

The adversary can choose any identity ID for private key query.
The adversary can choose any (ID, M) for signature query.

Forgery: The adversary outputs a forged signature (ID*, M*, S*) and wins the game if
No private key query on ID*
No signature query on (ID*, M*)

We say that the IBS is secure if every P.P.T adversary can only win the game with **negligible** probability defined as

Pr[Verify(mpk,ID*, M*, S*)=1]

Security Model (EUF-ID-CMA)

Forgery: The adversary outputs a forged signature (ID*, M*, S*) and wins the game if

No private key query on ID*
No signature query on (ID*, M*)

Question 1: Can the adversary make the signature query on (ID’, M*)?

Question 2: Can the adversary make the signature query on (ID*, M’)?

Construction

From Cyclic Group to Bilinear Pairing

Let (G,*g*,p) be a cyclic group.

- G is the set of all group elements.
- The set has p number of group elements.
- *g* is the generator of the group G.

$$G = \{g^0, g^1, g^2, g^3, \ldots\ldots, g^{p-1}\}$$

From Cyclic Group to Bilinear Pairing

Let (G,*g*,p) be a cyclic group.

- G is the set of all group elements.
- The set has p number of group elements.
- *g* is the generator of the group G.

$$G = \{g^0, g^1, g^2, g^3, \ldots\ldots, g^{p-1}\}$$

Given x and y, we can compute x+y mod p directly

Given x and y, we can compute x*y mod p directly

Given x, we can compute -x by computing p-x mod p.

Given x, we can run EEA algorithm to compute x^{-1} s.t. $x * x^{-1} = 1\, mod\ p$

From Cyclic Group to Bilinear Pairing

Let (G,*g*,p) be a cyclic group.

- G is the set of all group elements.
- The set has p number of group elements.
- *g* is the generator of the group G.

$$G = \{g^0, g^1, g^2, g^3, \dots\dots, g^{p-1}\}$$

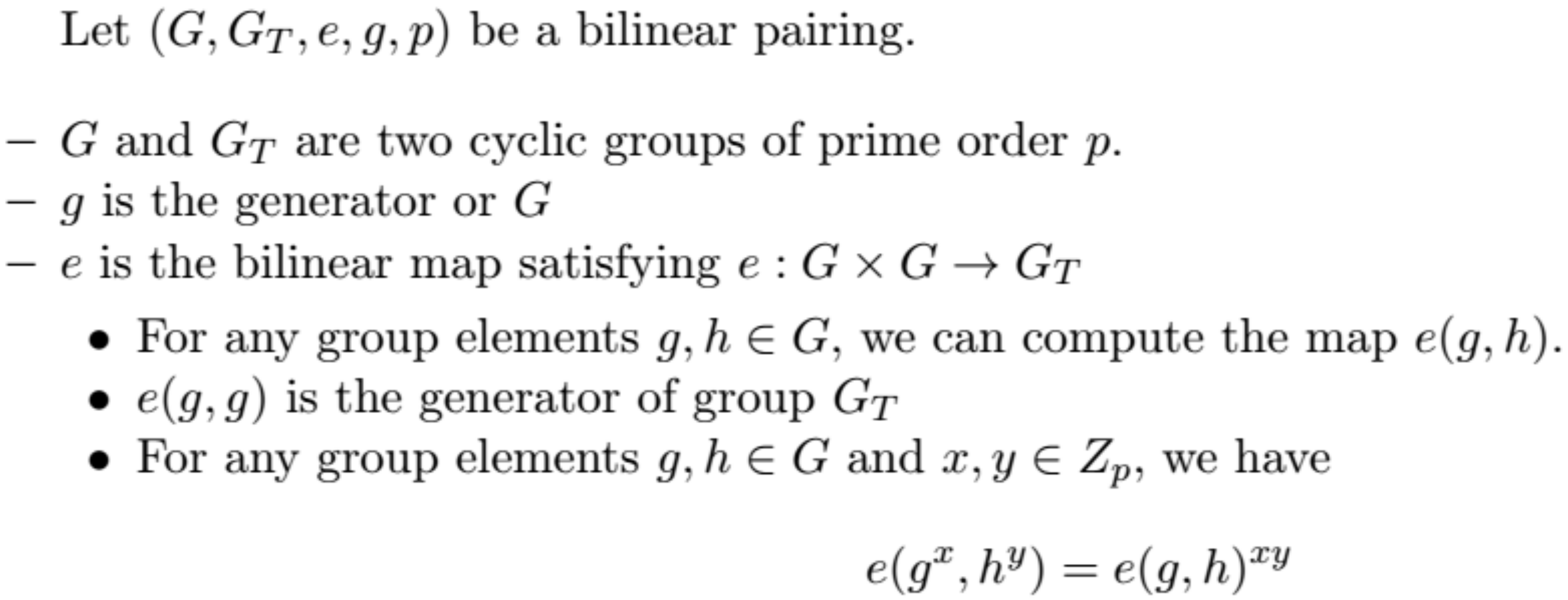
Given x and g, we can compute g^x in a fast way. (See Lecture 6 exponentiation)

Given g and h, we can compute g·h (basic group operation)

Given g and x, we can compute $g^{-x} = g^{p-x}. \mathit{In\ particuar,\ we\ can\ compute}\ g^{-1}$

Given g and x, we can compute $g^{\frac{1}{x}}$ s.t. $x * \frac{1}{X} = 1 \, mod \, p$

- From Cyclic Group to Bilinear Pairing

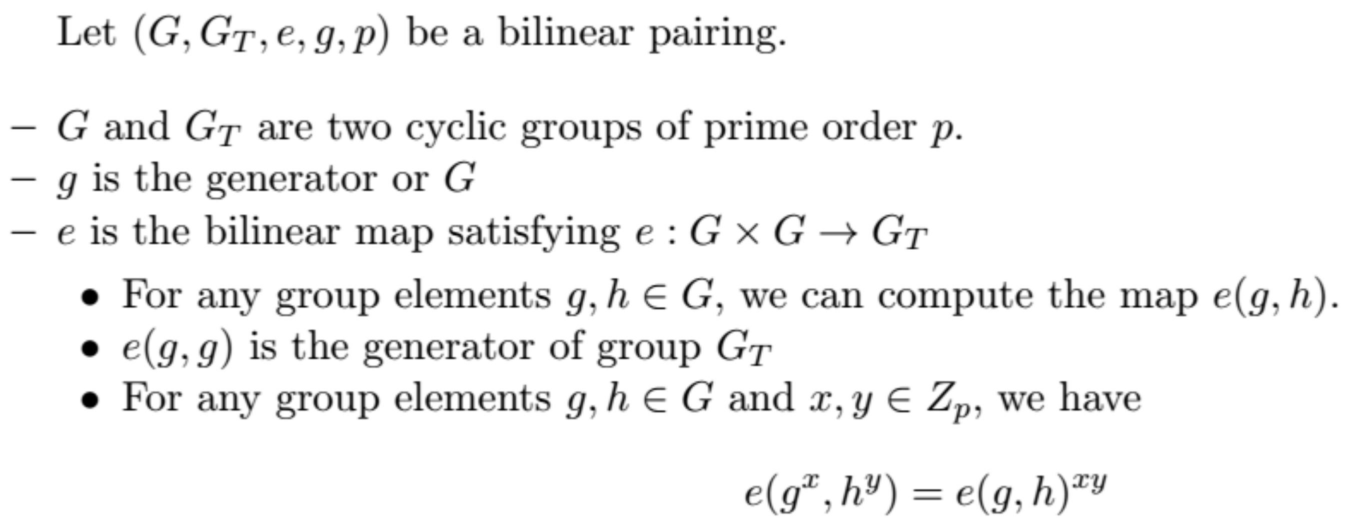


Let (G, G_T, e, g, p) be a bilinear pairing.

- G and G_T are two cyclic groups of prime order p.
- g is the generator or G
- e is the bilinear map satisfying $e : G \times G \rightarrow G_T$
  - For any group elements $g, h \in G$, we can compute the map $e(g, h)$.
  - $e(g, g)$ is the generator of group G_T
  - For any group elements $g, h \in G$ and $x, y \in Z_p$, we have

$$e(g^x, h^y) = e(g, h)^{xy}$$

- From Cyclic Group to Bilinear Pairing



Let (G, G_T, e, g, p) be a bilinear pairing.

- G and G_T are two cyclic groups of prime order p.
- g is the generator or G
- e is the bilinear map satisfying $e : G \times G \rightarrow G_T$
  - For any group elements $g, h \in G$, we can compute the map $e(g, h)$.
  - $e(g, g)$ is the generator of group G_T
  - For any group elements $g, h \in G$ and $x, y \in Z_p$, we have

$$e(g^x, h^y) = e(g, h)^{xy}$$

Let $f(x)=g^x$. We have $f(x)\cdot f(y)= f(x+y)$

Let $F(x)=e(g,g)^x$. We have $F(x)\cdot F(y)= F(x+y)$

We have the bilinear map satisfies:

$$e(f(x), f(y))= F(x*y)$$

From Cyclic Group to Bilinear Pairing

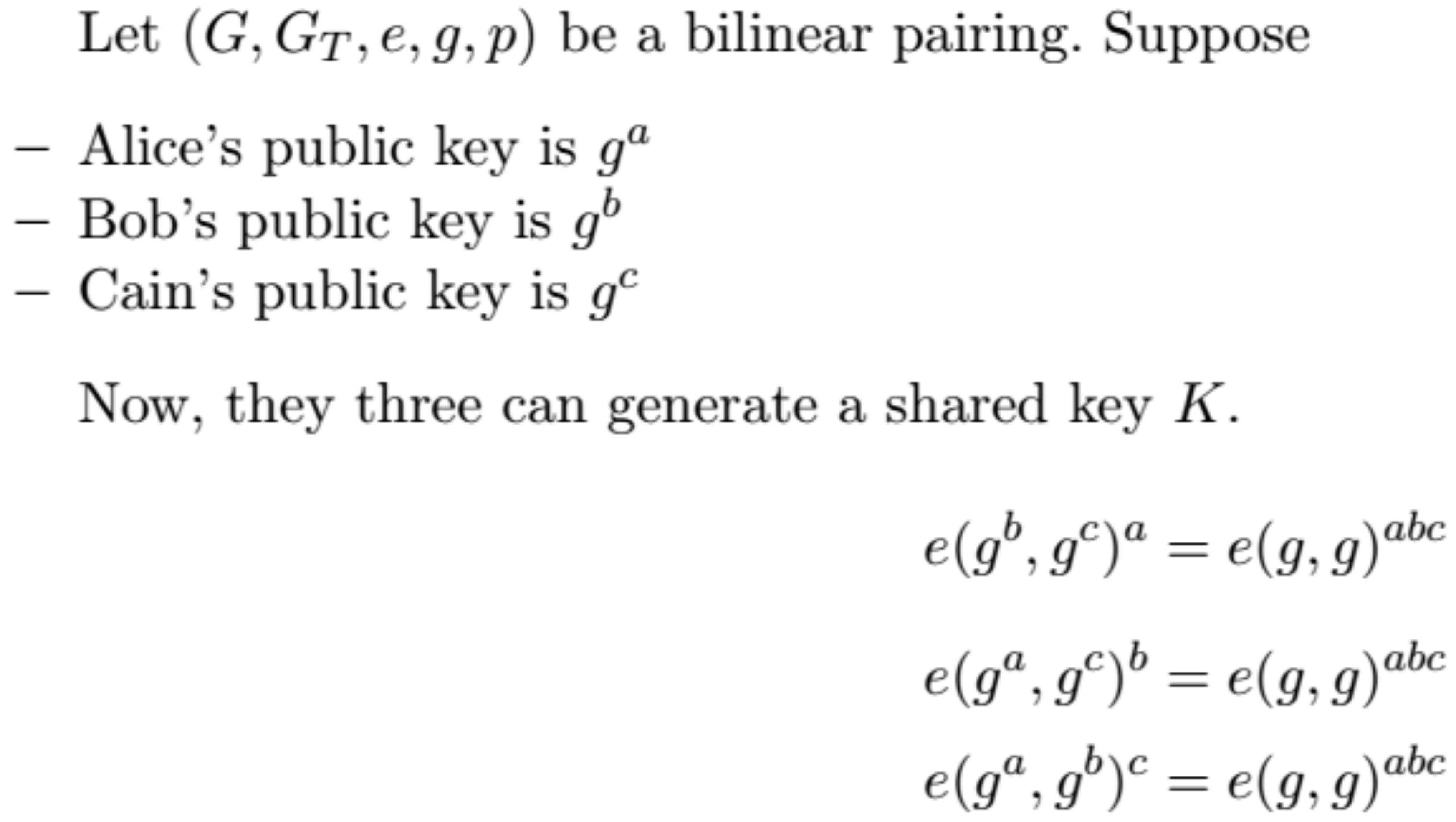
$$e(g^x, h^y) = e(g, h)^{xy}$$

$$e(g_1 g_2, h) = e(g_1, h) \cdot e(g_2, h)$$

Suppose $g_1 = g^{x_1}$ and $g_2 = g^{x_2}$.

$$\begin{aligned} e(g_1 g_2, h) &= e(g^{x_1 + x_2}, h) \\ &= e(g, h)^{x_1 + x_2} \\ &= e(g, h)^{x_1} \cdot e(g, h)^{x_2} \\ &= e(g_1, h\ \ ) \cdot e(g_2, h) \end{aligned}$$

Diffie-Hellman Key Exchange



Let (G, G_T, e, g, p) be a bilinear pairing. Suppose

- Alice's public key is g^a
- Bob's public key is g^b
- Cain's public key is g^c

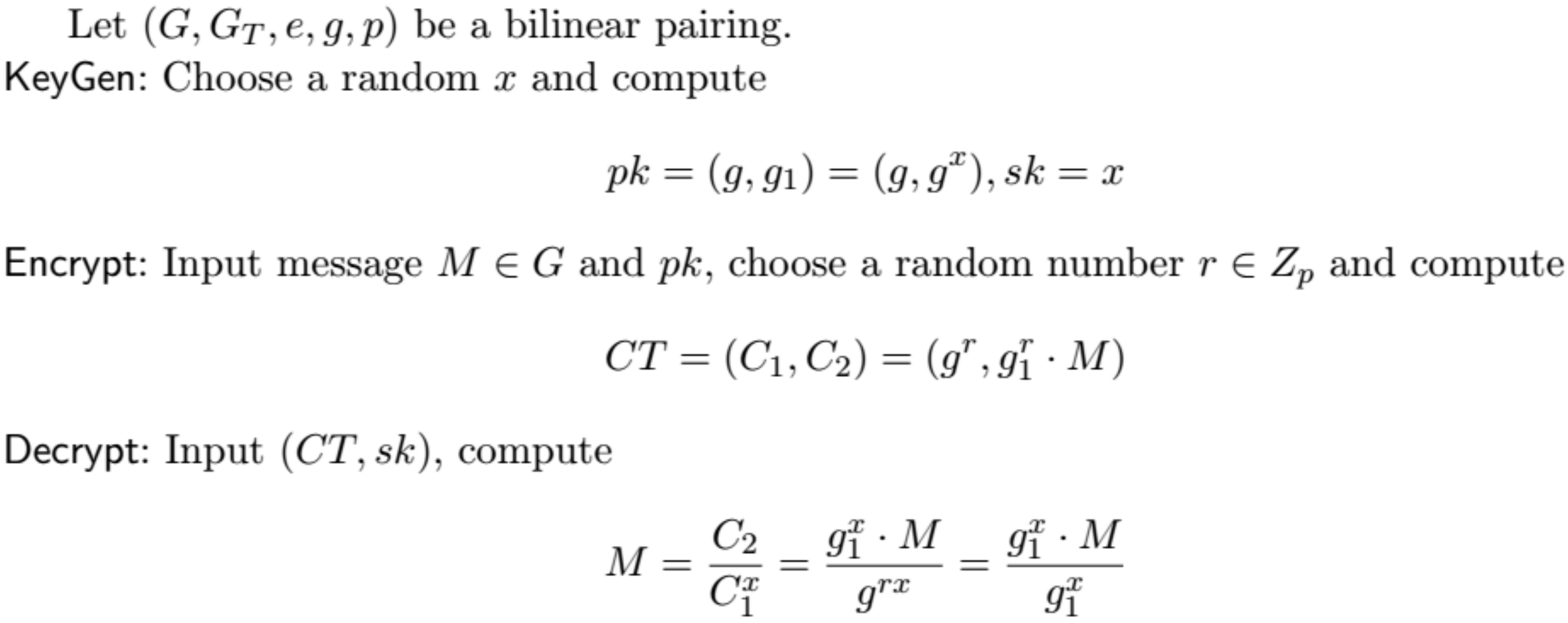
Now, they three can generate a shared key K.

$$e(g^b, g^c)^a = e(g,g)^{abc}$$

$$e(g^a, g^c)^b = e(g,g)^{abc}$$

$$e(g^a, g^b)^c = e(g,g)^{abc}$$

ElGamal Encryption (PKE)



Let (G, G_T, e, g, p) be a bilinear pairing.

KeyGen: Choose a random x and compute

$$pk = (g, g_1) = (g, g^x), sk = x$$

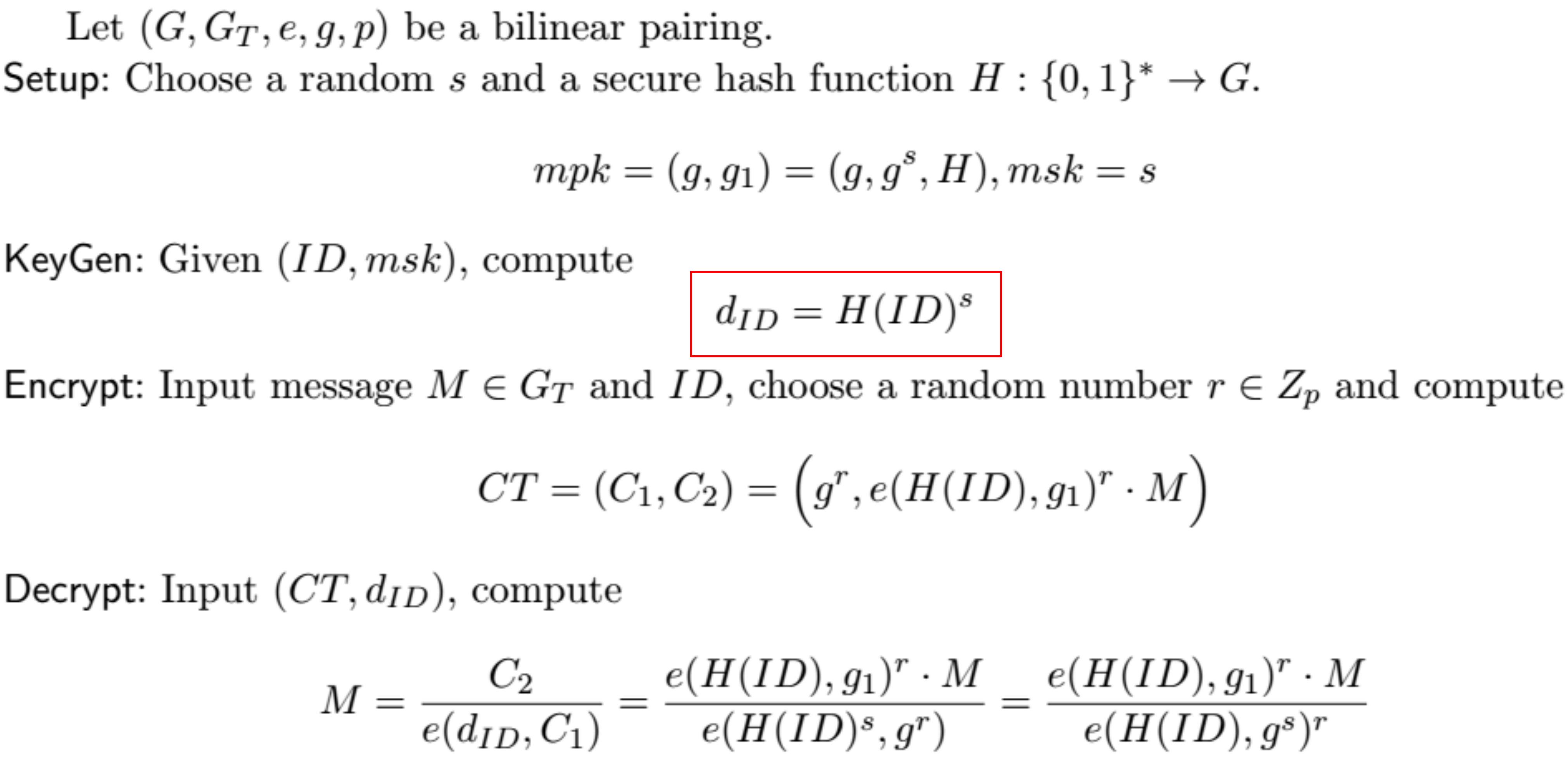
Encrypt: Input message $M \in G$ and pk, choose a random number $r \in Z_p$ and compute

$$CT = (C_1, C_2) = (g^r, g_1^r \cdot M)$$

Decrypt: Input (CT, sk), compute

$$M = \frac{C_2}{C_1^x} = \frac{g_1^x \cdot M}{g^{rx}} = \frac{g_1^x \cdot M}{g_1^x}$$

Boneh-Franklin Encryption (IBE 2001)



Let (G, G_T, e, g, p) be a bilinear pairing.

Setup: Choose a random s and a secure hash function $H : \{0,1\}^* \to G$.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk), compute

$$\boxed{d_{ID} = H(ID)^s}$$

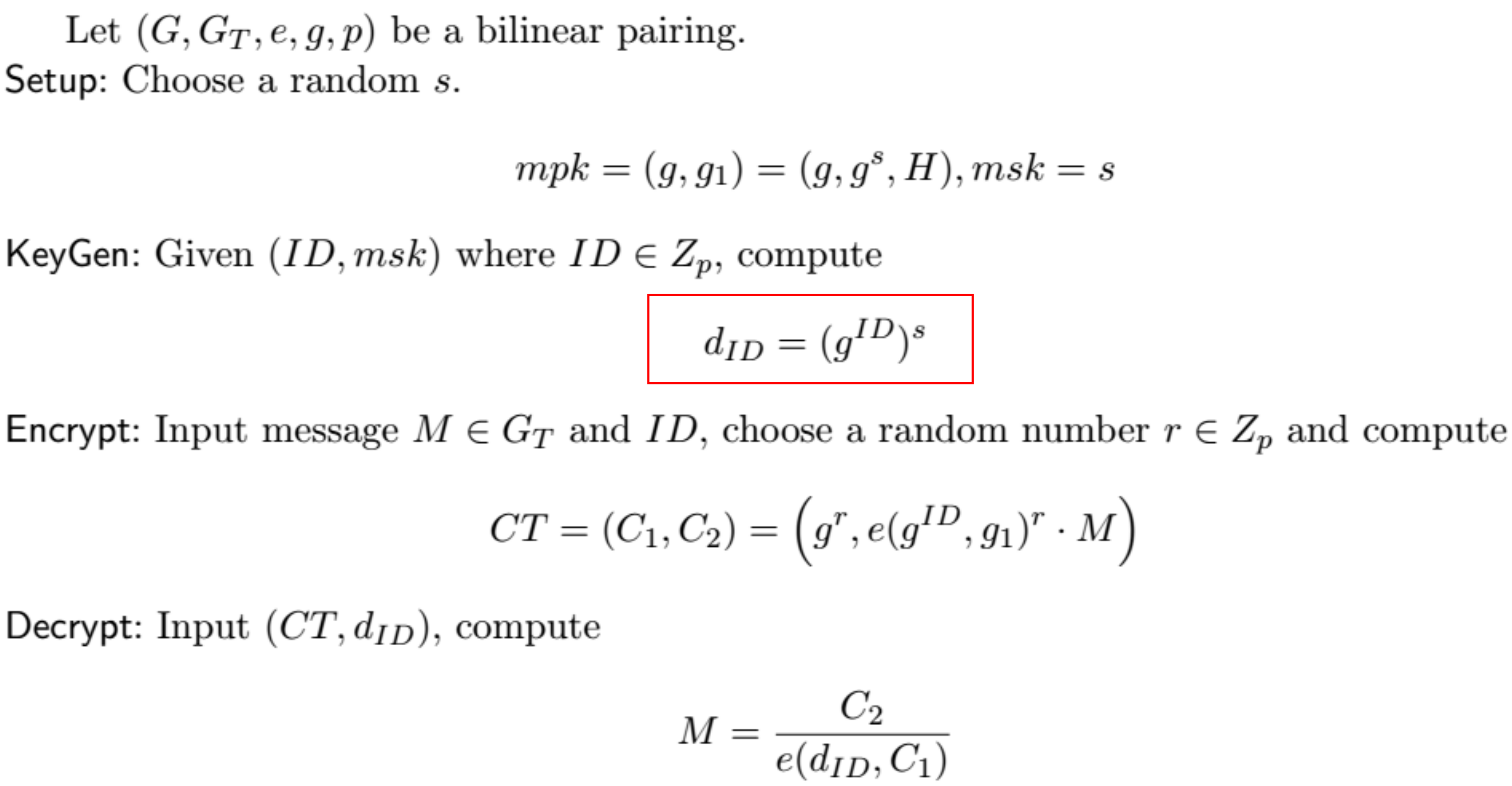
Encrypt: Input message $M \in G_T$ and ID, choose a random number $r \in Z_p$ and compute

$$CT = (C_1, C_2) = \left(g^r, e(H(ID), g_1)^r \cdot M\right)$$

Decrypt: Input (CT, d_{ID}), compute

$$M = \frac{C_2}{e(d_{ID}, C_1)} = \frac{e(H(ID), g_1)^r \cdot M}{e(H(ID)^s, g^r)} = \frac{e(H(ID), g_1)^r \cdot M}{e(H(ID), g^s)^r}$$

Boneh-Franklin Encryption (IBE 2001 Insecure)



Let (G, G_T, e, g, p) be a bilinear pairing.

Setup: Choose a random s.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk) where $ID \in Z_p$, compute

$$\boxed{d_{ID} = (g^{ID})^s}$$

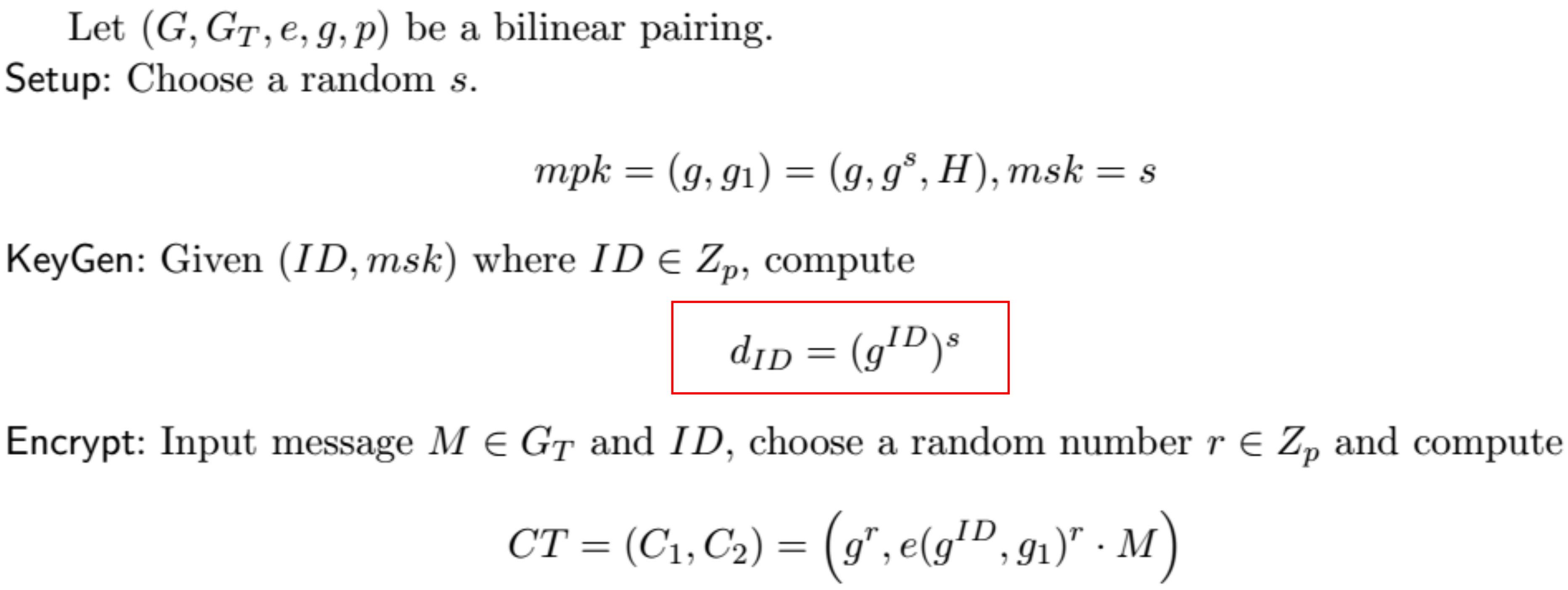
Encrypt: Input message $M \in G_T$ and ID, choose a random number $r \in Z_p$ and compute

$$CT = (C_1, C_2) = \Big(g^r, e(g^{ID}, g_1)^r \cdot M\Big)$$

Decrypt: Input (CT, d_{ID}), compute

$$M = \frac{C_2}{e(d_{ID}, C_1)}$$

Boneh-Franklin Encryption (IBE 2001 Insecure)



Let (G, G_T, e, g, p) be a bilinear pairing.

Setup: Choose a random s.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk) where $ID \in Z_p$, compute

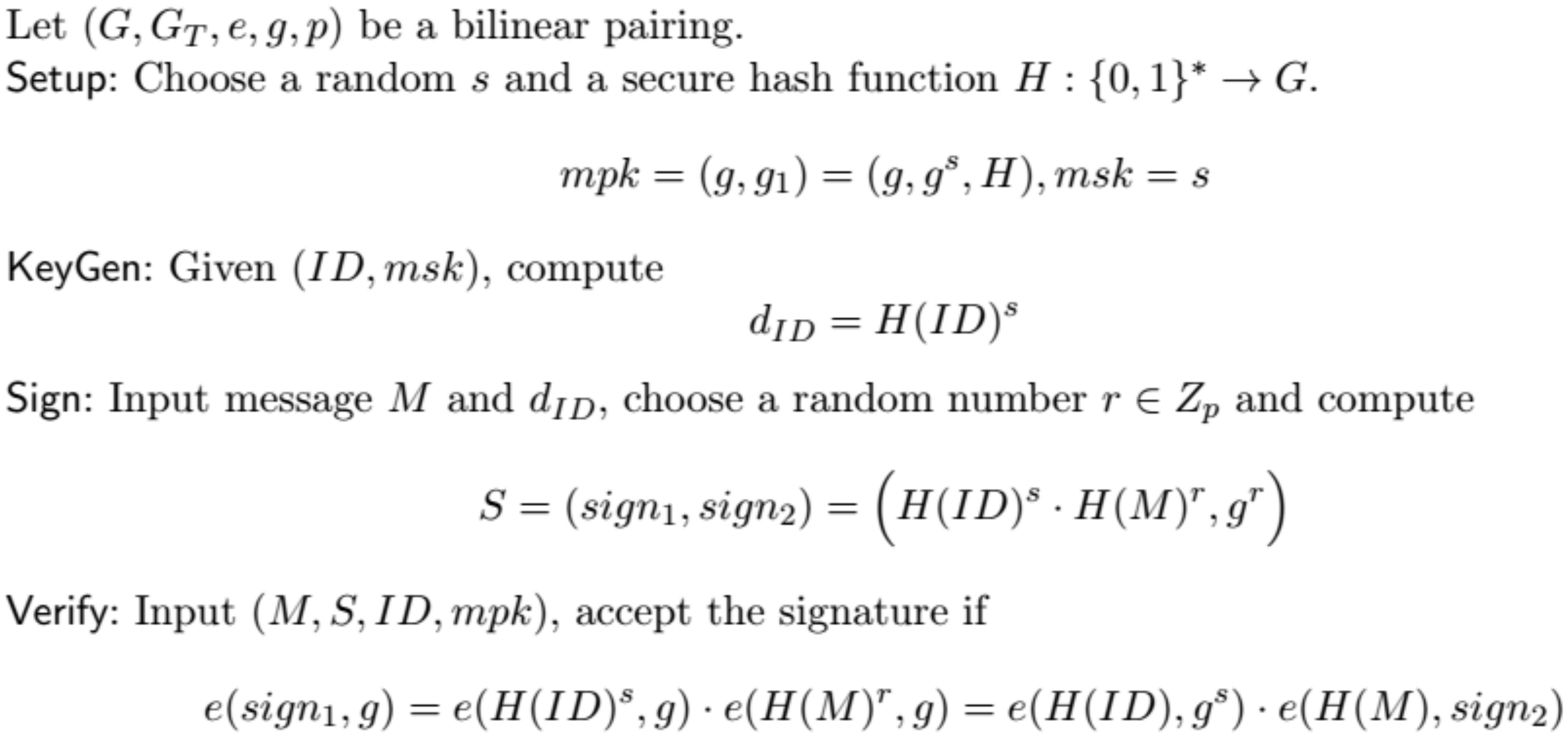
$$\boxed{d_{ID} = (g^{ID})^s}$$

Encrypt: Input message $M \in G_T$ and ID, choose a random number $r \in Z_p$ and compute

$$CT = (C_1, C_2) = \left(g^r, e(g^{ID}, g_1)^r \cdot M\right)$$

$$\boxed{e(C_1, g_1)^{ID} = e(g^r, g_1)^{ID} = e(g, g_1)^{ID \cdot r}}$$

Identity-Based Signature



Let (G, G_T, e, g, p) be a bilinear pairing.

Setup: Choose a random s and a secure hash function $H : \{0,1\}^* \to G$.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk), compute

$$d_{ID} = H(ID)^s$$

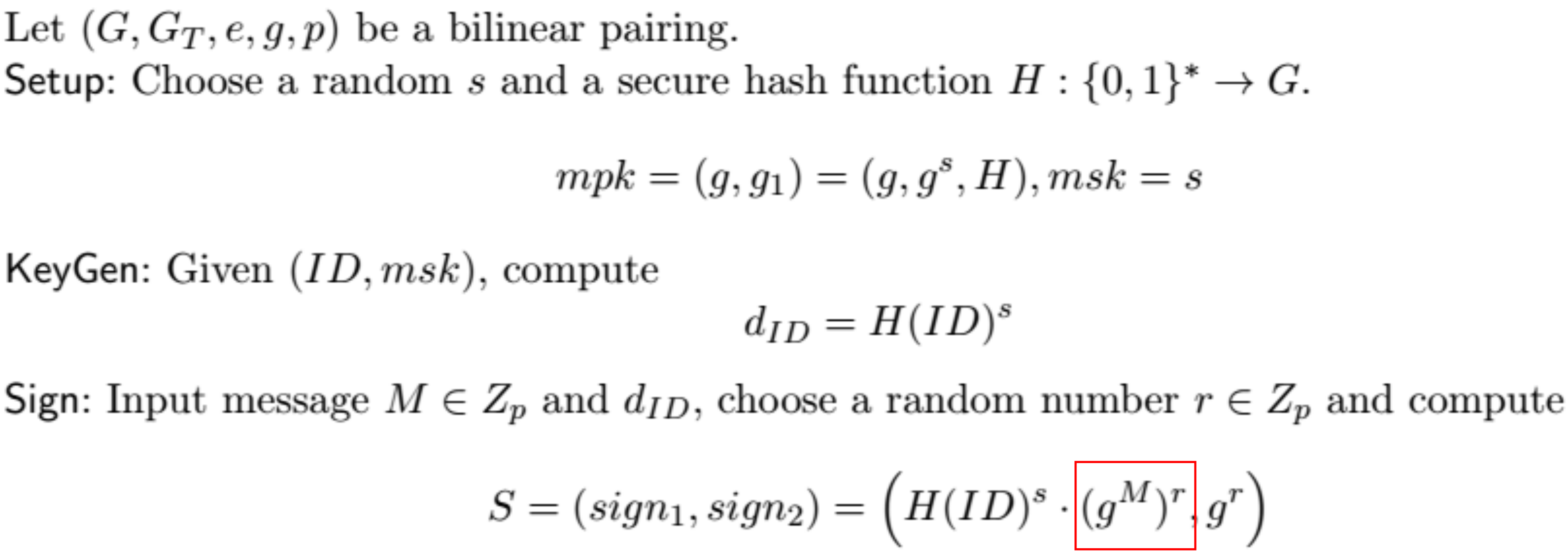
Sign: Input message M and d_{ID}, choose a random number $r \in Z_p$ and compute

$$S = (sign_1, sign_2) = \Big(H(ID)^s \cdot H(M)^r, g^r\Big)$$

Verify: Input (M, S, ID, mpk), accept the signature if

$$e(sign_1, g) = e(H(ID)^s, g) \cdot e(H(M)^r, g) = e(H(ID), g^s) \cdot e(H(M), sign_2)$$

Identity-Based Signature (insecure)



Let (G, G_T, e, g, p) be a bilinear pairing.

Setup: Choose a random s and a secure hash function $H : \{0,1\}^* \rightarrow G$.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

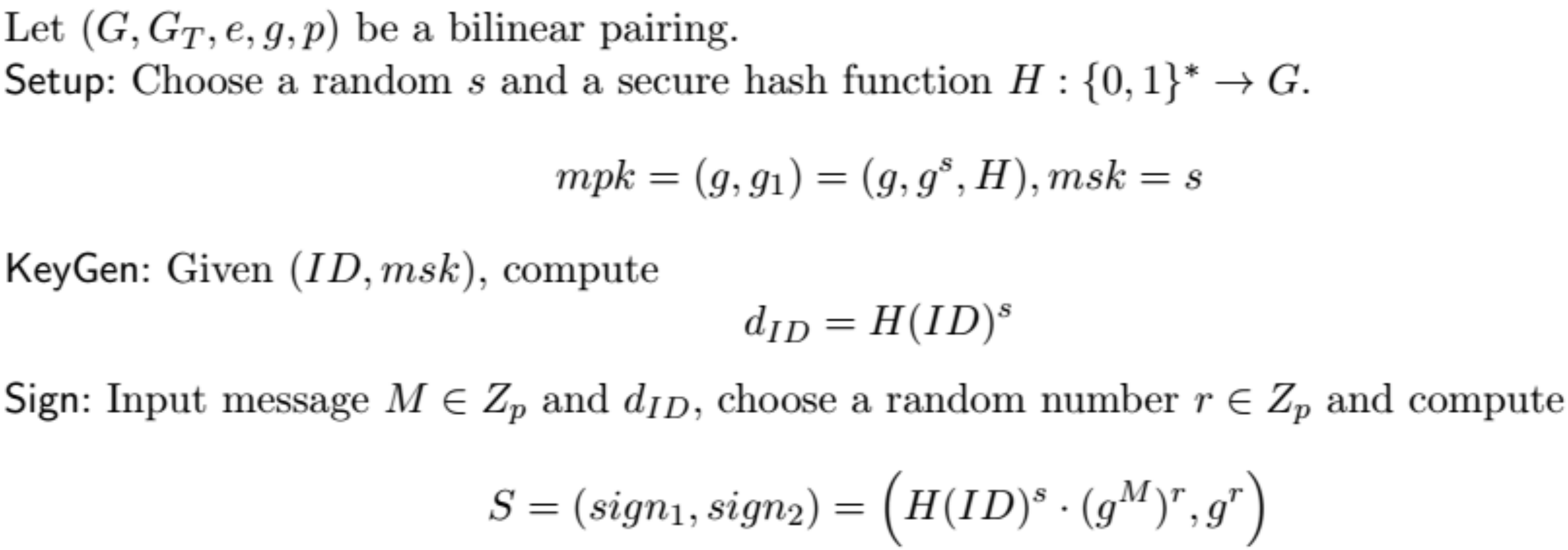
KeyGen: Given (ID, msk), compute

$$d_{ID} = H(ID)^s$$

Sign: Input message $M \in Z_p$ and d_{ID}, choose a random number $r \in Z_p$ and compute

$$S = (sign_1, sign_2) = \left(H(ID)^s \cdot \boxed{(g^M)^r}, g^r\right)$$

Identity-Based Signature (insecure)



Let (G, G_T, e, g, p) be a bilinear pairing.
Setup: Choose a random s and a secure hash function $H : \{0,1\}^* \rightarrow G$.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk), compute

$$d_{ID} = H(ID)^s$$

Sign: Input message $M \in Z_p$ and d_{ID}, choose a random number $r \in Z_p$ and compute

$$S = (sign_1, sign_2) = \left(H(ID)^s \cdot (g^M)^r, g^r\right)$$

Given a signature, we can compute its private key.

Identity-Based Signature (insecure)

Let (G, G_T, e, g, p) be a bilinear pairing.
Setup: Choose a random s and a secure hash function $H : \{0,1\}^* \rightarrow G$.

$$mpk = (g, g_1) = (g, g^s, H), msk = s$$

KeyGen: Given (ID, msk), compute

$$d_{ID} = H(ID)^s$$

Sign: Input message $M \in Z_p$ and d_{ID}, choose a random number $r \in Z_p$ and compute

$$S = (sign_1, sign_2) = \left(H(ID)^s \cdot (g^M)^r, g^r\right)$$

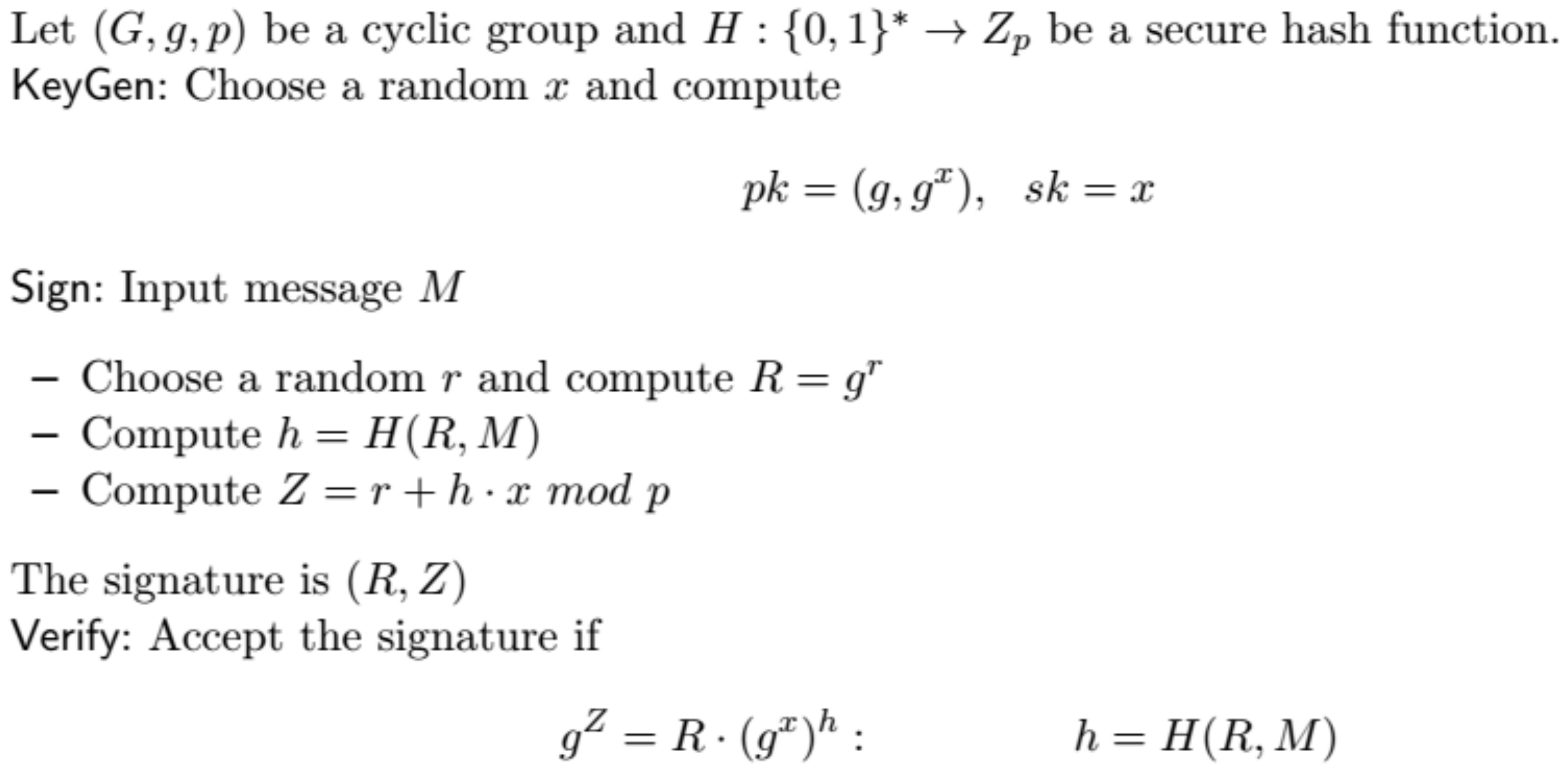
Given a signature, we can compute its private key.

$$S = (sign_1, sign_2) = \left(H(ID)^s \cdot (g^M)^r, g^r\right)$$

$$\boxed{\frac{sign_1}{(sign_2)^M} = \frac{H(ID)^s \cdot (g^M)^r}{g^{r \cdot M}} = H(ID)^s}$$

WorkShop

Schnorr Signature



Let (G, g, p) be a cyclic group and $H : \{0,1\}^* \rightarrow Z_p$ be a secure hash function.
KeyGen: Choose a random x and compute

$$pk = (g, g^x), \quad sk = x$$

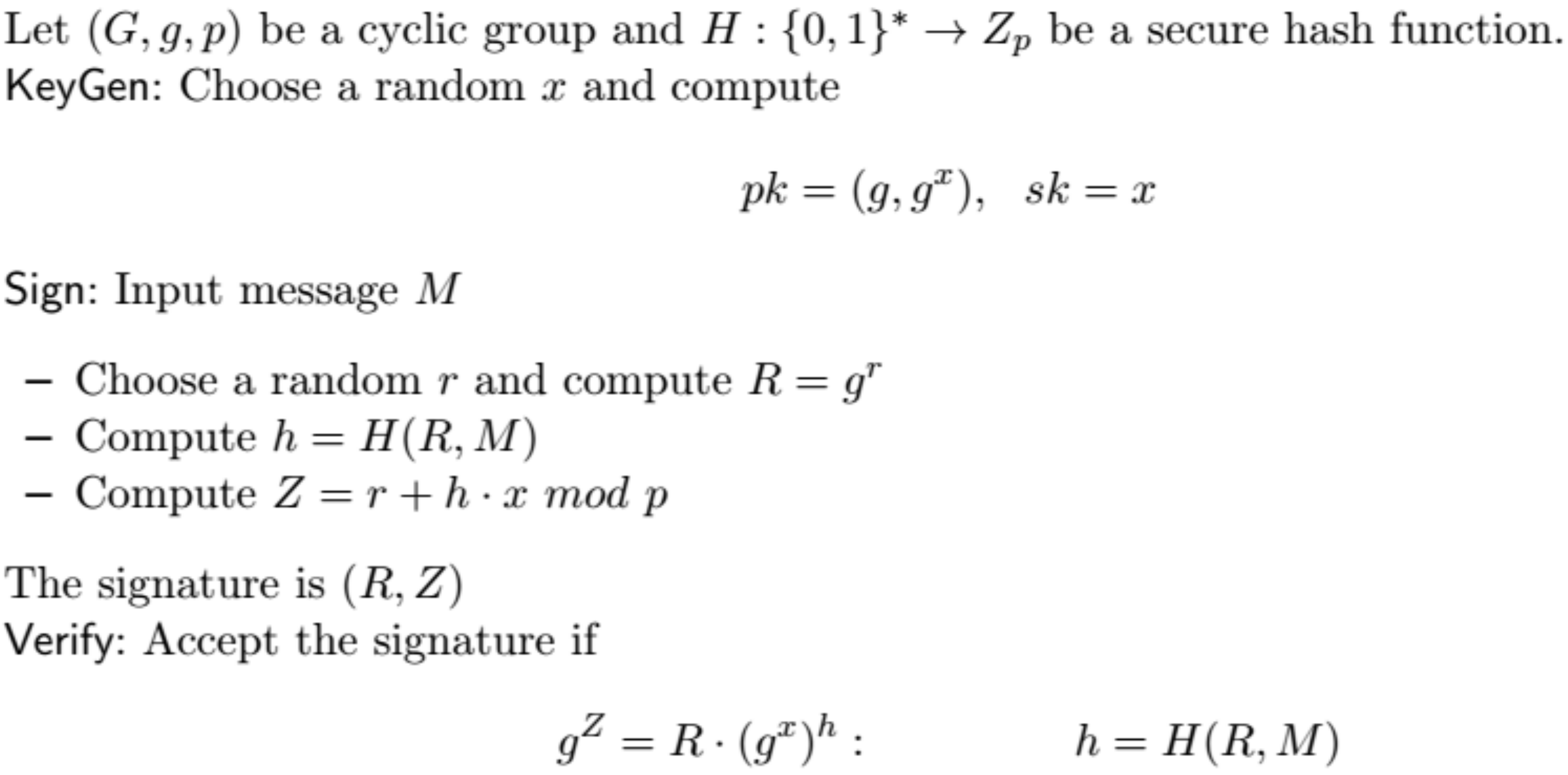
Sign: Input message M

- Choose a random r and compute $R = g^r$
- Compute $h = H(R, M)$
- Compute $Z = r + h \cdot x \ mod \ p$

The signature is (R, Z)
Verify: Accept the signature if

$$g^Z = R \cdot (g^x)^h : \qquad\qquad h = H(R, M)$$

Schnorr Signature (Insecure 1)



Let (G, g, p) be a cyclic group and $H : \{0,1\}^* \to Z_p$ be a secure hash function.
KeyGen: Choose a random x and compute

$$pk = (g, g^x), \quad sk = x$$

Sign: Input message M

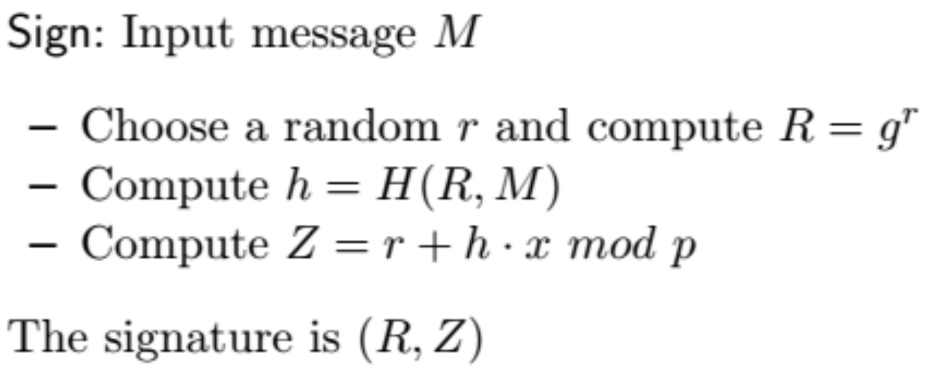
- Choose a random r and compute $R = g^r$
- Compute $h = H(R, M)$
- Compute $Z = r + h \cdot x \ mod \ p$

The signature is (R, Z)
Verify: Accept the signature if

$$g^Z = R \cdot (g^x)^h : \qquad\qquad h = H(R, M)$$

Prove that the signature scheme is not secure if the signature is (r, R,Z)

Schnorr Signature (Insecure 1)



Sign: Input message M

- Choose a random r and compute $R = g^r$
- Compute $h = H(R, M)$
- Compute $Z = r + h \cdot x \ mod \ p$

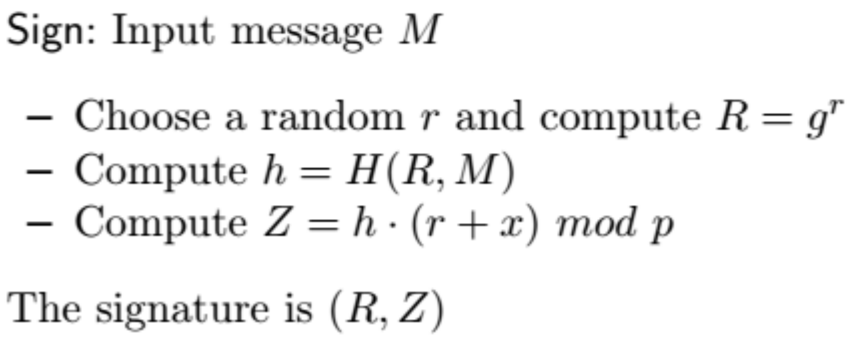
The signature is (R, Z)

Prove that the signature scheme is not secure if the signature is (r, R,Z)

1. Compute w suc that r+w=0 mod p (That is, w=-r)
2. Compute Z+w, which is equal to a=h*x mod p
3. Run EEA algorithm to compute the inverse of h denoted by b such that b*h=1 mod p
4. Compute b*a mod p= b*h*x =1*x=x mod p

That is, we can compute the secre key x from the signature

Schnorr Signature (Insecure 2)



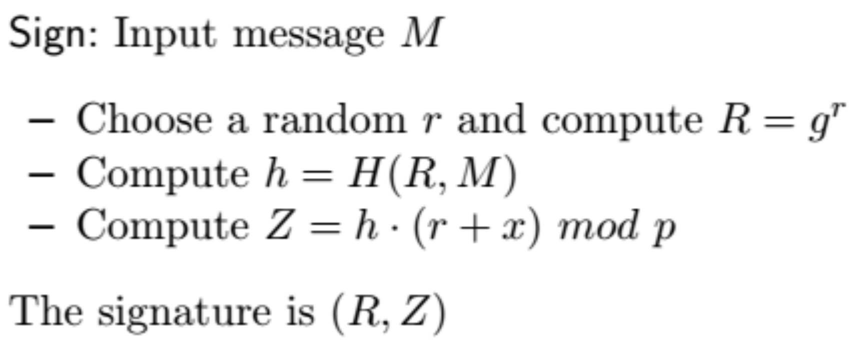
Sign: Input message M

- Choose a random r and compute $R = g^r$
- Compute $h = H(R, M)$
- Compute $Z = h \cdot (r + x) \ mod \ p$

The signature is (R, Z)

Prove that the signature scheme is not EUF-CMA secure.

Schnorr Signature (Insecure 2)



Sign: Input message M

- Choose a random r and compute $R = g^r$
- Compute $h = H(R, M)$
- Compute $Z = h \cdot (r + x)\ mod\ p$

The signature is (R, Z)

Prove that the signature scheme is not EUF-CMA secure.

1. Compute the inverse of group element g^x, denoted by a=g^{-x}
2. Choose a random w and compute b=g^w
3. Compute R=b*a=g^{w-x} (That is, r=w-x)
4. Compute h=H(R, M)
5. Compute Z=h*w= h(w-x + x)

That is, we can forge a signature on M without knowing x.