Aggregating CL-Signatures Revisited: Extended Functionality and Better Efficiency

FC 2013

Kwangsu Lee, Dong Hoon Lee, and Moti Yung

Korea University and Columbia University, Korea University, Google Inc. and Columbia University

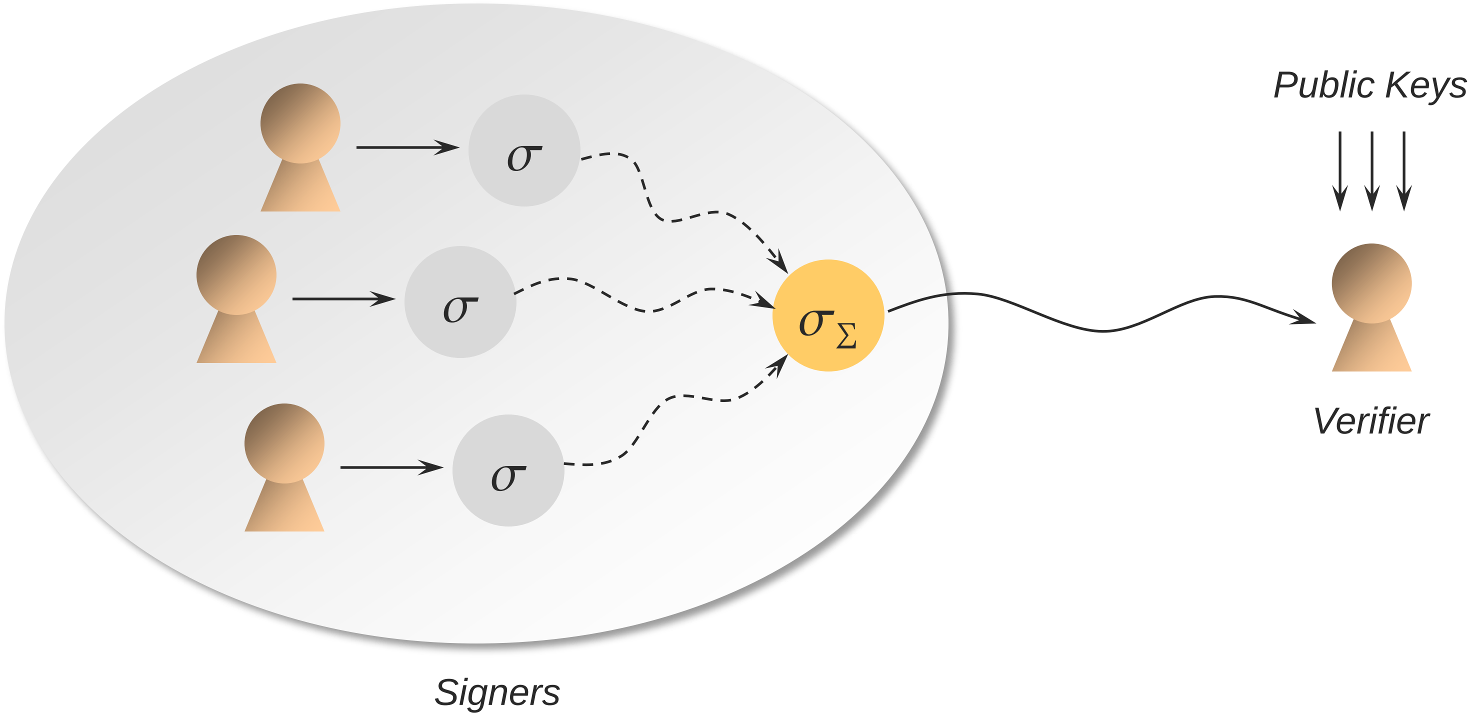
Overview

- Motivation
  - In aggregate signature, it has not been easy to devise a suitable aggregate signature scheme that satisfies the conditions of real applications with reasonable parameters: *short public key size*, *short aggregate signature size*, and *efficient aggregate signing & verification*

- Results
  - We propose two aggregate signature schemes based on the Camenisch-Lysyanskaya (CL) signature scheme
  - The first scheme is an efficient *sequential aggregate signature (SeqAS)* scheme with the shortest size of public keys
  - The second scheme is an efficient *synchronized aggregate signature (SyncAS)* scheme with the shortest size of aggregate signatures

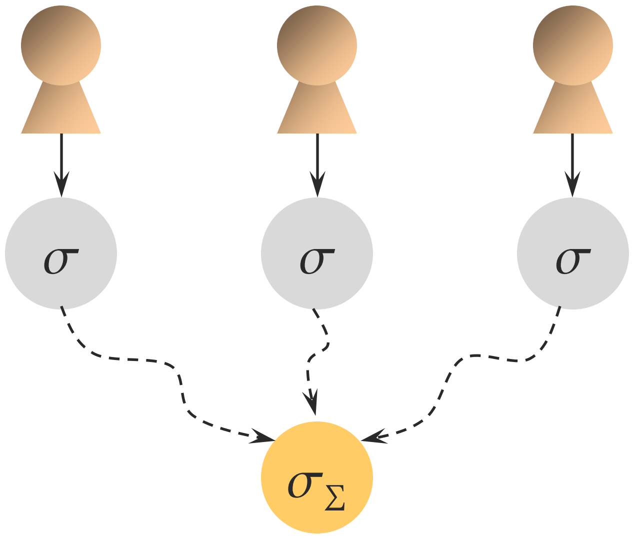
Introduction

- Aggregate Signature
  - Aggregate signature is a new type of PKS which enables any user to combine signatures signed by different signers into a short signature
  - The application includes reducing bandwidth of certificate chains in PKI, secure routing protocols, sensor networks, and secure work flow



Introduction

- Types of Aggregate Signature
  - The types of aggregate signatures are categorized as full aggregation, sequential aggregation, and synchronized aggregation
  - (1) In full aggregation, any user can *freely* aggregate different signatures of different signers into a single signature



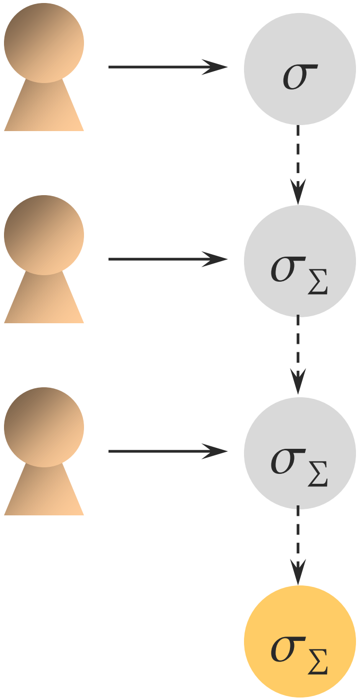
Full aggregation

BGLS03 *ROM*

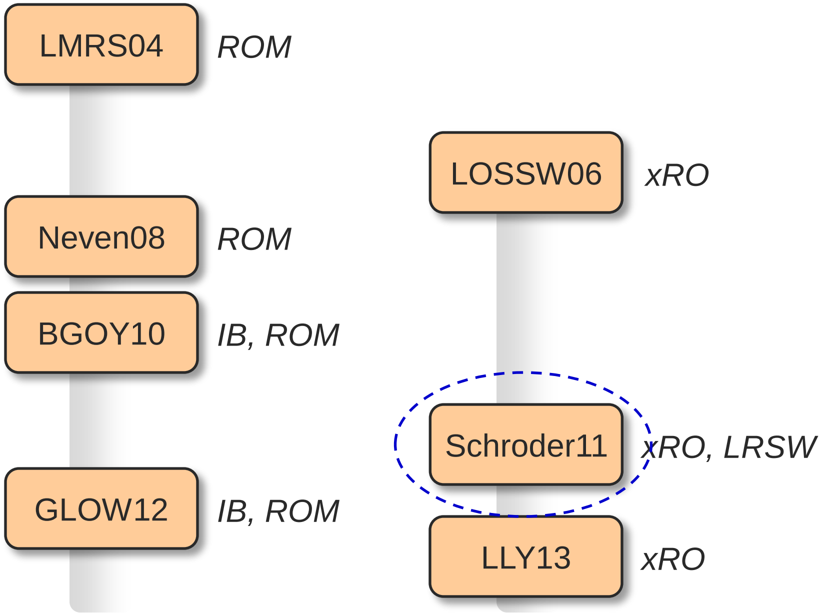
Only one scheme exists!

Introduction

- Types of Aggregate Signature
  - (2) In sequential aggregation, each signer can aggregate his signature into a previously aggregated signature in a *sequential order*



Sequential aggregation



LMRS04 *ROM*

Neven08 *ROM*

BGOY10 *IB, ROM*

GLOW12 *IB, ROM*

LOSSW06 *xRO*

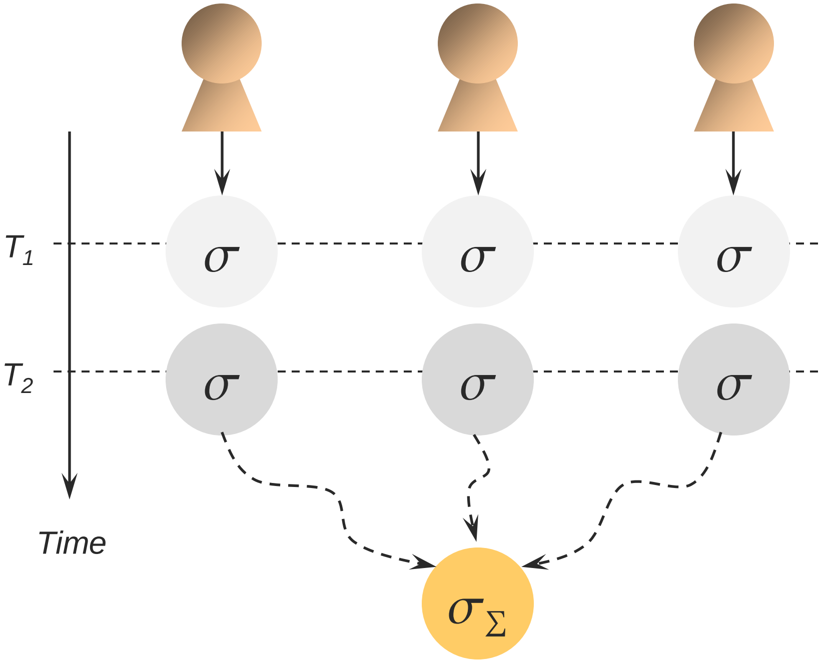
Schroder11 *xRO, LRSW*

LLY13 *xRO*

The scheme of Schroder is based on CL signature

Introduction

- Types of Aggregate Signature
  - (3) In synchronized aggregation, any user can combine different signatures with *the same synchronizing information* into a single signature



Synchronized aggregation

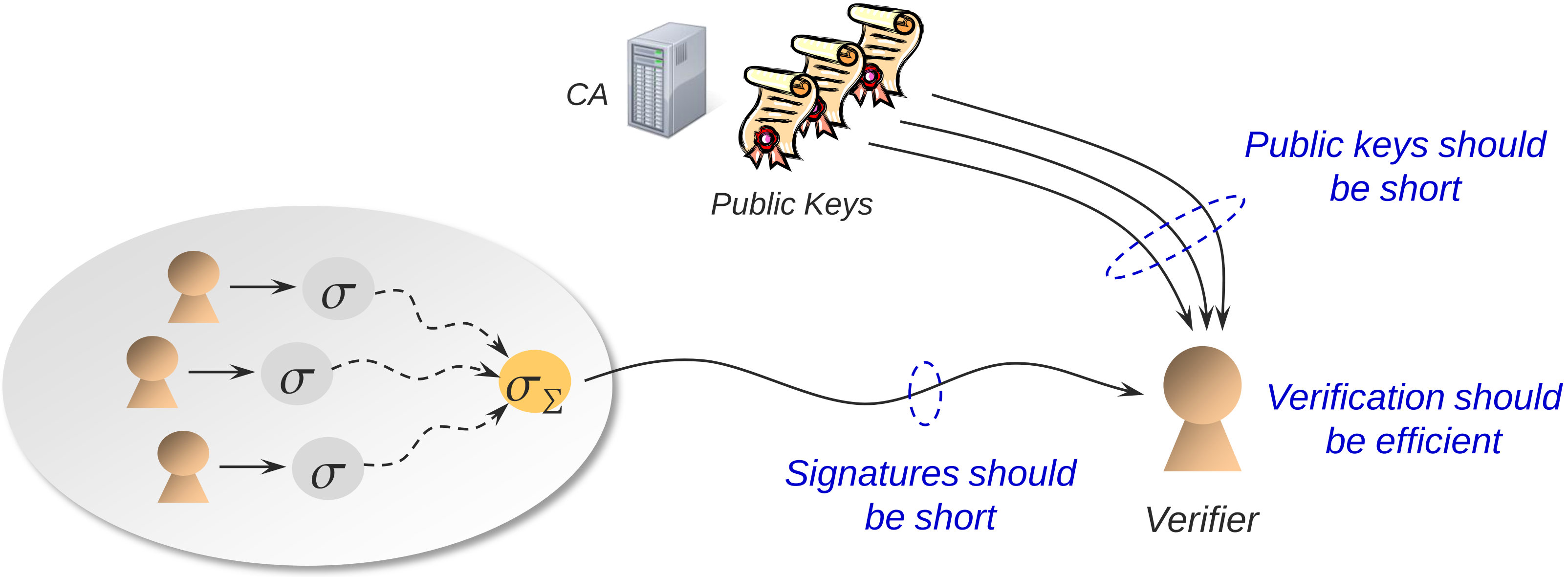
GR06 *IB, ROM*

AGH10 *xRO*

Synchronizing information should be shared!

Introduction

- Motivation
  - For real applications, aggregate signature should satisfy the conditions of short public key size, short aggregate signature size, and efficient aggregate signing & verification
  - However, there is no satisfactory scheme that meets these conditions


CA
Public Keys
Public keys should be short
σ
σ
σ
σ_Σ
Signatures should be short
Verifier
Verification should be efficient

CL Signature

- CL Signature Scheme
  - The CL signature scheme is a PKS scheme in bilinear groups proposed by Camenisch and Lysyanskaya at Crypto 2004
  - The security of the scheme was proven without random oracles under the LRSW assumption

$$PK = [(p, \mathbb{G}, \mathbb{G}_T, e), g, X = g^x, Y = g^y]$$

$$SK = [x, y]$$

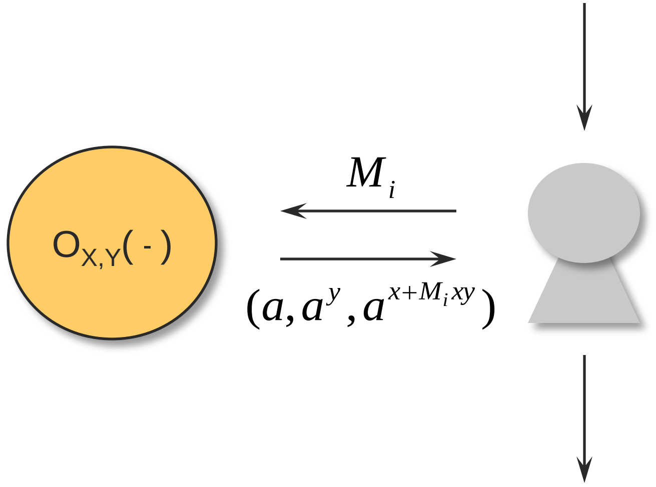
$$\sigma = [A = g^r, B = Y^r, C = A^x B^{xM}] \; where\ M \in \mathbb{Z}_p^*$$

$$e(A, Y) = e(B, g) \ \wedge \ e(C, g) = e(A, X) \cdot e(B, X)^M$$

CL Signature

- LRSW Assumption
  - The LRSW assumption was introduced by Lysyanskaya, Rivest, Sahai, and Wolf and adapted to bilinear groups
  - It is secure under the generic group model defined by Shoup

$$(p, \mathbb{G}, \mathbb{G}_T, e, g, X = g^x, Y = g^y)$$

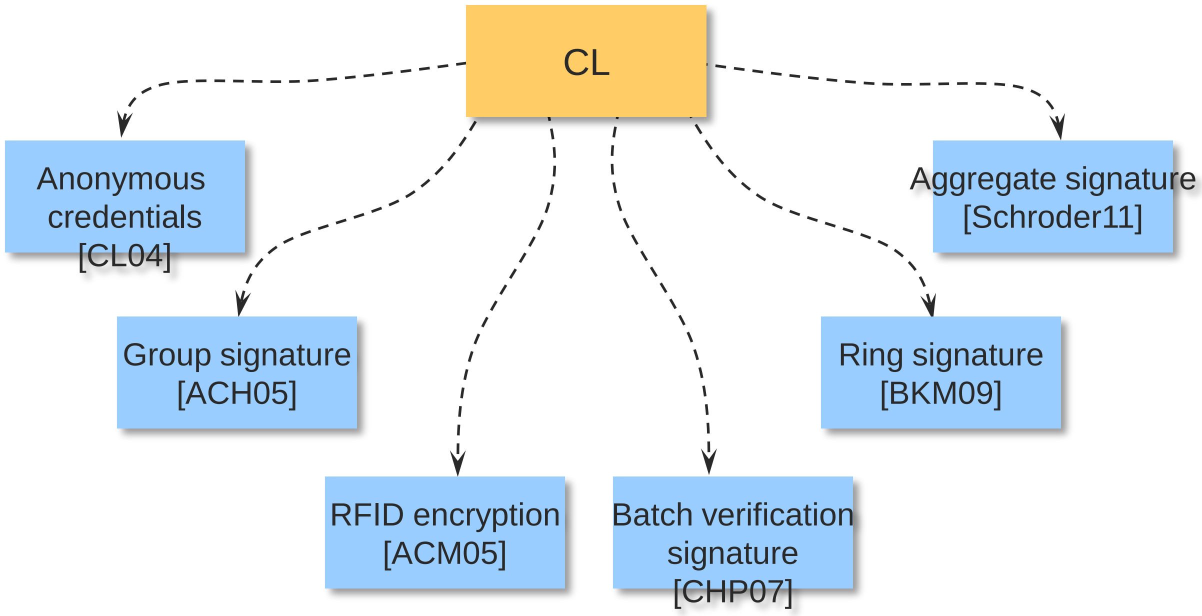


$$(M, a, b, c)$$

$$M \notin \{M_i\} \wedge M \in \mathbb{Z}_p^* \wedge a \in \mathbb{G} \wedge b = a^y \wedge c = a^{x+Mxy}$$

CL Signature

- Applications
  - The CL signature scheme is flexible enough for a range of possible applications such as anonymous credential systems, group signature, RFID encryption, batch verification signature, ring signature, and aggregate signature


CL
Anonymous credentials [CL04]
Group signature [ACH05]
RFID encryption [ACM05]
Batch verification signature [CHP07]
Ring signature [BKM09]
Aggregate signature [Schroder11]

Sequential Aggregate Signature

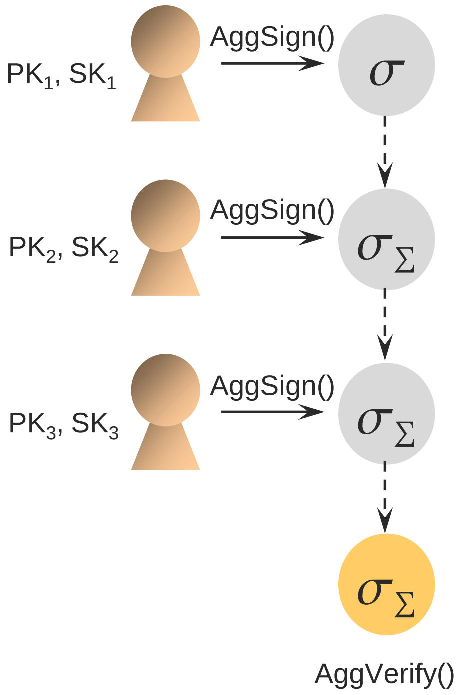
- Definition
  - SeqAS is a special type of PKAS that allows each signer to sequentially add his signature to the previous aggregate signature
  - A SeqAS scheme consists of four algorithms Setup, KeyGen, AggSign, and AggVerify

Setup(1^λ) → PP

KeyGen(PP) → PK, SK

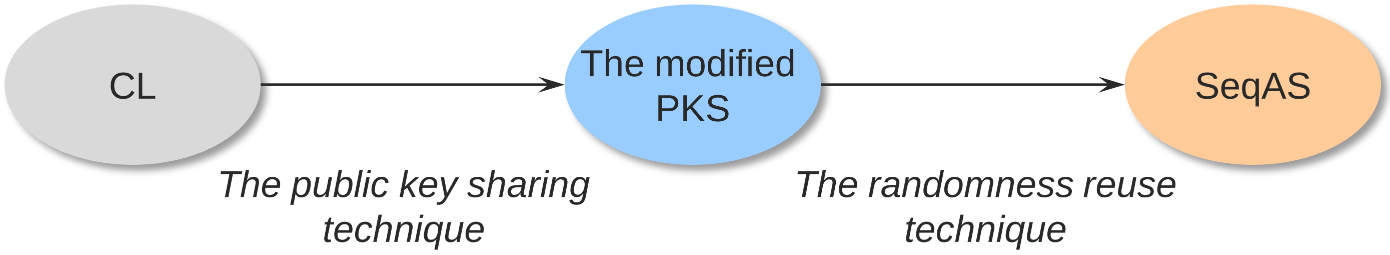
AggSign(σ'_Σ, $\{M_i\}$, $\{PK_i\}$, M, SK, PP) → σ_Σ

AggVerify(σ_Σ, $\{M_i\}$, $\{PK_i\}$, PP) → 1 or 0



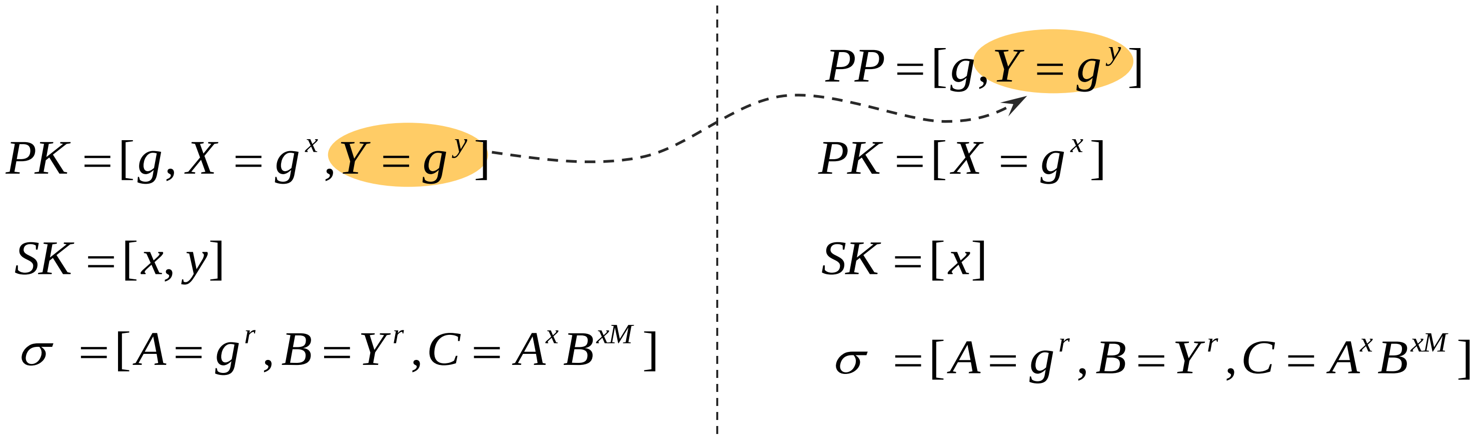
Sequential Aggregate Signature

- Design Principle
  - First, we use the *public key sharing* technique such that the element Y is shared among all signers
  - Next, we apply the *randomness re-use* technique of Lu et al. to sequentially aggregate signatures


CL
The modified PKS
SeqAS
The public key sharing technique
The randomness reuse technique

Sequential Aggregate Signature

- Modified CL Signature Scheme
  - The original CL signature scheme can be modified to share the element Y with all other signers
  - The signature of the modified one is the same as that of the original one, and the modified one is still secure under the LRSW assumption



$$PK = [g, X = g^x, Y = g^y]$$

$$SK = [x, y]$$

$$\sigma = [A = g^r, B = Y^r, C = A^x B^{xM}]$$

$$PP = [g, Y = g^y]$$

$$PK = [X = g^x]$$

$$SK = [x]$$

$$\sigma = [A = g^r, B = Y^r, C = A^x B^{xM}]$$

$$\sigma = [g^r, (g^r)^y, (g^r)^{x+xyM}]$$

Sequential Aggregate Signature

- SeqAS Scheme
  - The modified CL signature scheme can be converted to a SeqAS scheme by using the randomness re-use technique
  - The resulting signature should be re-randomized to prevent an attack

$$PP = [(p, \mathbb{G}, \mathbb{G}_T, e), g, Y = g^y]$$

$$PK_i = [X_i = g^{x_i}]$$

$$SK_i = [x_i]$$

Re-randomization

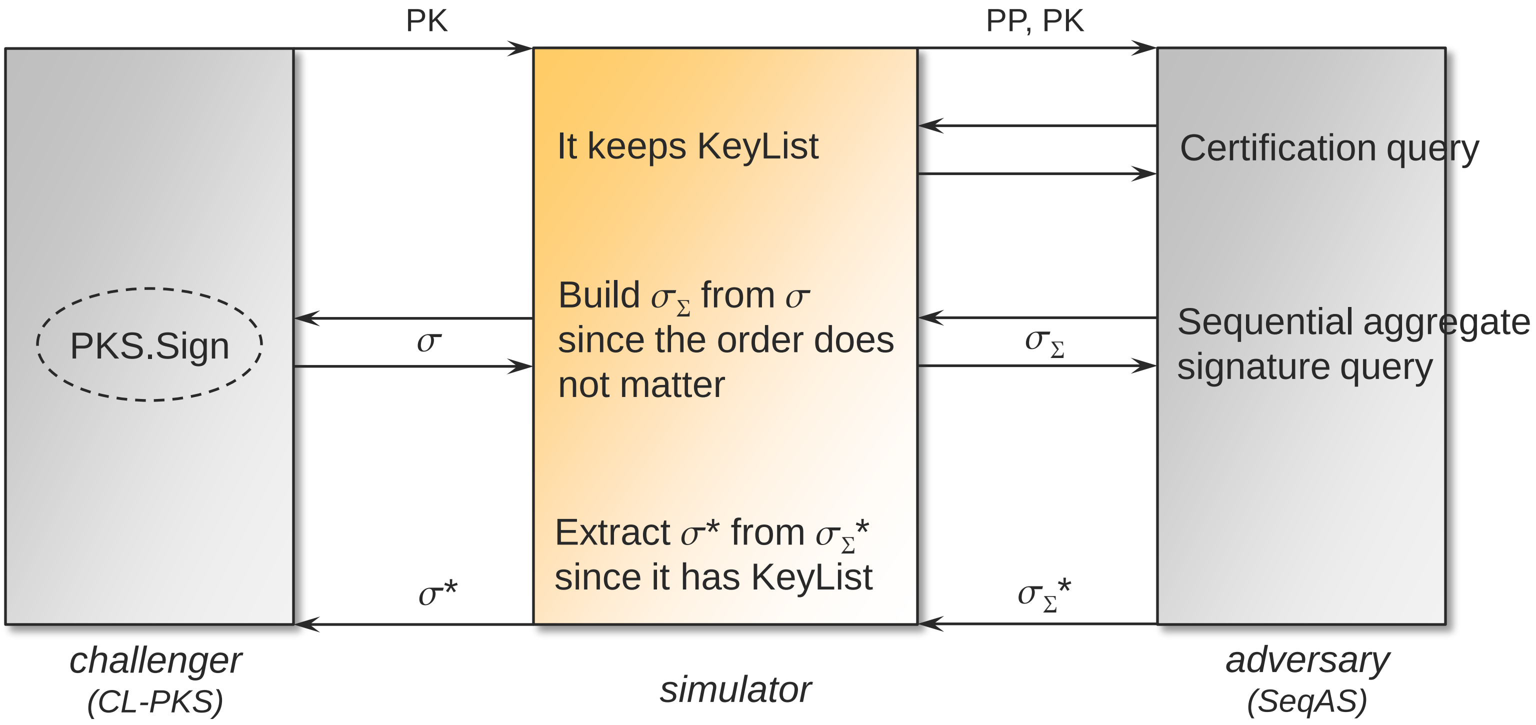
$$\sigma_\Sigma = [A = (A')^r, B = (B')^r, C = (C'(A')^{x_i}(B')^{x_i M_i})^r]$$

$$\textit{where } (A', B', C') \textit{ is an aggregate-so-far}$$

$$e(A, Y) = e(B, g) \ \wedge \ e(C, g) = e(A, \prod_{i=1}^{l} X_i) \cdot e(B, \prod_{i=1}^{l} X_i^{M_i})$$

Sequential Aggregate Signature

- Security Analysis
  - The proof uses two facts that the aggregated signature is independent of the order of aggregation and the simulator possesses the private keys of other signers



PK

PP, PK

It keeps KeyList

Certification query

PKS.Sign

σ

Build σ_Σ from σ since the order does not matter

σ_Σ

Sequential aggregate signature query

σ^*

Extract σ^* from $\sigma_\Sigma{}^*$ since it has KeyList

$\sigma_\Sigma{}^*$

challenger
(CL-PKS)

simulator

adversary
(SeqAS)

Sequential Aggregate Signature

- Discussions
  - The public key and the aggregate signature of our SeqAS scheme consist of *one* group element and *three* group elements respectively, and the aggregate verification algorithm requires *five* pairing operations and *l* exponentiations
  - If we instantiate our SeqAS scheme by using asymmetric bilinear groups (175-bit MNT curve), then the size of public key is 525 bits and the size of aggregate signature is 525 bits
  - A new PKS scheme (the modified CL signature scheme) can be derived from our SeqAS scheme, and it is secure under the LRSW assumption

Synchronized Aggregate Signature

- Definition
  - SyncAS is a special type of PKAS that allows anyone to aggregate signer's signatures with the same time period into an aggregate signature
  - A SyncAS scheme consists of six algorithms Setup, KeyGen, Sign, Verify, Aggregate, and AggVerify

Setup(1^λ) → PP

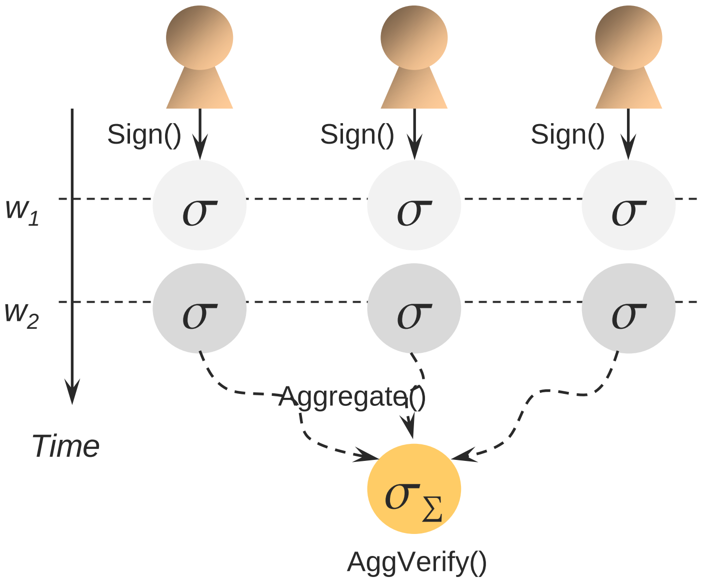
KeyGen(PP) → PK, SK

Sign(M, w, SK, PP) → σ

Verify(σ, M, PK, PP) → 1 or 0

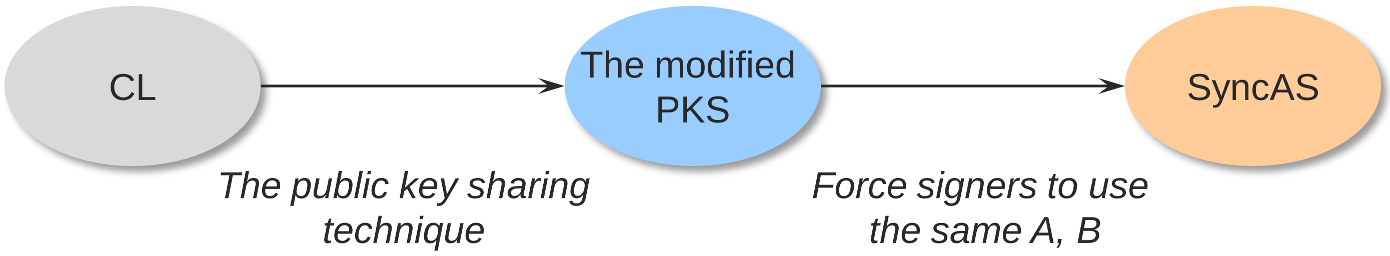
Aggregate($\{\sigma_i\}$, $\{M_i\}$, $\{PK_i\}$, PP) → σ_Σ

AggVerify(σ_Σ, $\{M_i\}$, $\{PK_i\}$, PP) → 1 or 0



Synchronized Aggregate Signature

- Design Principle
  - In the modified CL signature scheme, aggregation is easy if all signers use the same *A*, *B* in the signature
  - In synchronized aggregate signature, we can force signers to use the same *A*, *B* by hashing the same time period *w*


CL
The modified PKS
SyncAS
The public key sharing technique
Force signers to use the same A, B

$A = g^r$
$B = Y^r$
$C = A^x B^{xM}$

$A = H(0||w)$
$B = H(1||w)$
$C = A^x B^{xM}$

Synchronized Aggregate Signature

- SyncAS Scheme
  - The modified CL signature scheme can be converted to a synchronized aggregate signature since all signers share the same time period *w*
  - However, the time period *w* in the signature should not be used before

$$PP = [(p, \mathbb{G}, \mathbb{G}_T, e), g, H_1, H_2]$$

$$PK_i = [X_i = g^{x_i}]$$

$$SK_i = [x_i]$$

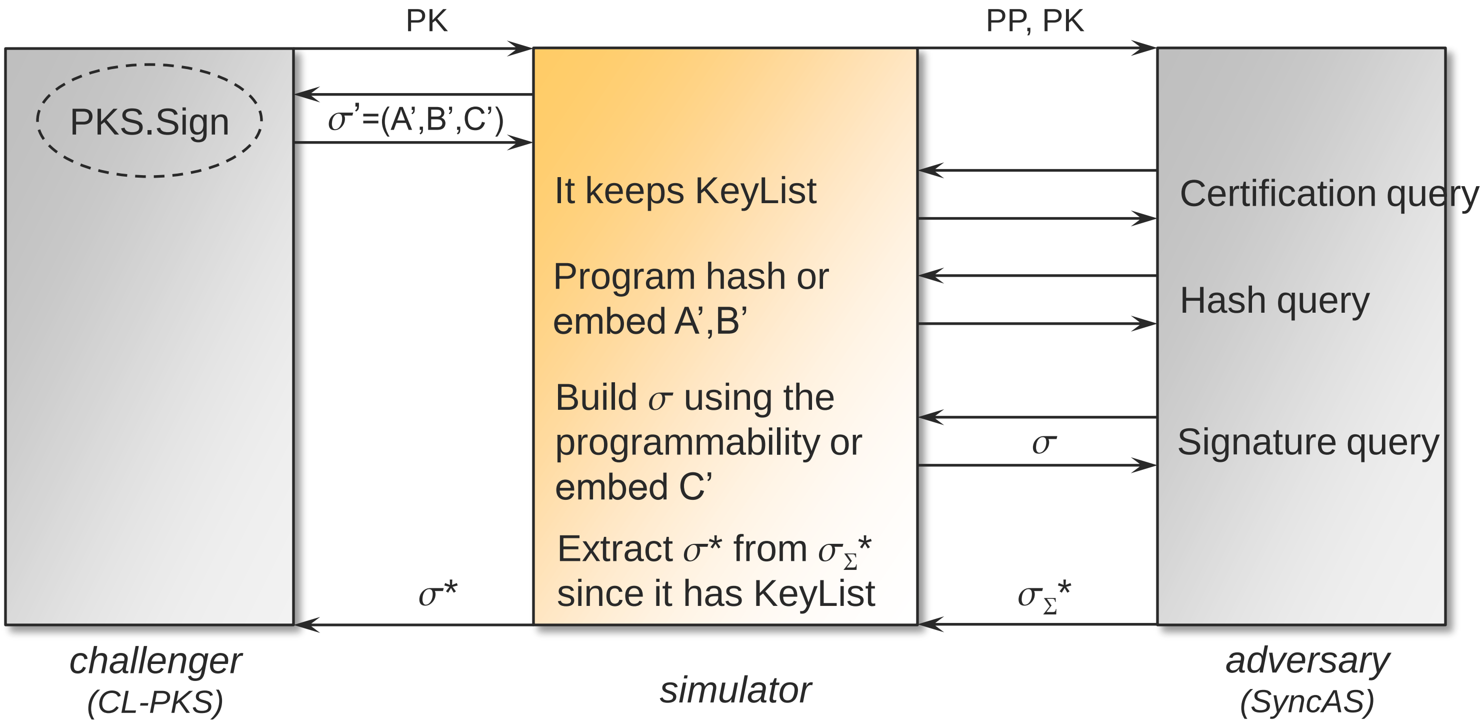
$$\sigma = [C = H_1(0 \| w)^{x_i} H_1(1 \| w)^{x_i H_2(M_i \| w)}, w]$$

$$\sigma_{\Sigma} = [C = \prod_{i=1}^{l} C_i, w]$$

$$e(C, g) = e(H_1(0 \| w), \prod_{i=1}^{l} X_i) \cdot e(H_2(1 \| w), \prod_{i=1}^{l} X_i^{H_2(M_i \| w)_i})$$

Synchronized Aggregate Signature

- Security Analysis
  - The proof uses the facts that the random oracle model supports the programmability, the adversary request just one signature per one time, and the simulator possesses the private keys of other signers



Synchronized Aggregate Signature

- Discussions
  - The aggregate signature of our SyncAS scheme consist of *one* group element and *one* integer, and the aggregate verification algorithm requires *three* pairing operations and *l* exponentiations
  - If we instantiate our SyncAS scheme by using asymmetric bilinear groups (175-bit MNT curve), then the size of aggregate signature is 207 bits
  - A combined aggregate signature scheme that supports sequential aggregation and synchronized aggregation at the same time can be derived
  - The security of our SyncAS scheme can be proven under *one-time* LRSW (OT-LRSW) assumption which is a static assumption
  - If the number of messages is restricted to be polynomial, then we can remove the random oracles

Conclusion

- Final Remarks
  - We proposed one sequential aggregate signature scheme and one synchronized aggregate signature scheme and proved their security under the security of the CL signature scheme
  - Our two aggregate signature schemes sufficiently satisfy the efficiency conditions of real applications

  - An interesting problem is to prove the security of our SeqAS scheme under *static assumptions* instead of the interactive LRSW assumption

Thank You