Hierarchy Integrated Signature and Encryption

(Key Separation vs. Key Reuse: Enjoy the Best of Both Worlds)

Yu Chen Qiang Tang Yuyu Wang

ASIACRYPT 2021

Outline

Outline

PKE+SIG

PKE and SIG are workhorse typically used simultaneously to secure communication

- PKE $\Rightarrow$ protect confidentiality
- SIG $\Rightarrow$ protect authenticity: data integrity & authenticated data source

Classical examples

- Secure communication software: PGP, WhatsApp
- Privacy-preserving cryptocurrency: Zcash, Zether, PGC

Joint security (akin to UC)

- EUF-CMA security for SIG: holds even in the presence of $\mathcal{O}_{\text{dec}}$
- IND-CCA security for PKE: holds even in the presence of $\mathcal{O}_{\text{sign}}$

A Subtle Point or a Dilemma

Key Separation vs. Key Reuse

Key Separation: Cartesian-Product Combined Public-Key Scheme

Engineering folklore: using different keypairs for different cryptographic operations

Pros

- joint security is immediate & construction is off-the-shelf
- naturally admits individual key escrow: achieve a balance between user's authenticity requirement and society's auditing requirement

Cons

- double key management complexity and certificate cost[1]
- complicate the design of high-level protocol: tricky address derivation

[1] Certificate costs include but not limit to registration, issuing, storage, transmission, verification, and building/recurring fees.

Key Reuse: Integrated Signature and Encryption

SIG

sk

pk

PKE

Pros

- reduce key management complexity, certificate cost, and cryptographic footprint
- simplify the design of high-level protocol

Cons

- joint security is not immediate (consider textbook RSA) & require careful design
- does not admit individual key escrow
- does not admit classified protection

Deployed in EMV standard, Ping Identity, Zether and PGC

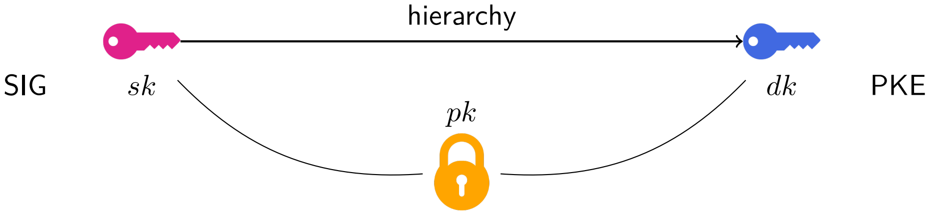
Motivation

We are facing a dilemma between key reuse that brings performance benefit and key separation that supports individual key escrow.

Can we enable individual key escrow mechanism while retaining the merits of key reuse?

Outline

Hierarchy Integrated Signature and Encryption



- Setup$(1^\lambda) \to pp$
- KeyGen$(1^\lambda) \to (pk, sk)$. pk serves as encryption and verification key; sk is the signing key, serving as master secret key.
- Derive$(sk) \to dk$ used only for decryption
- Enc$(pk, m) \to c$
- Dec$(dk, c) \to m$
- Sign$(sk, \tilde{m}) \to \sigma$
- Vefy$(pk, \tilde{m}, \sigma) \to 0/1$

Strong Joint Security

IND-CCA security in the presence of a signing oracle (unrestricted access)

$$\Pr\left[b = b' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}, \mathcal{O}_{\mathsf{sign}}}(pp, pk); \\ b \xleftarrow{\mathsf{R}} \{0,1\}, c^* \leftarrow \mathsf{Enc}(pk, m_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{dec}}, \mathcal{O}_{\mathsf{sign}}}(c^*); \end{array} \right] - \frac{1}{2} \leq \mathsf{negl}(\lambda).$$

EUF-CMA security in the presence of a decryption key

$$\Pr\left[\begin{array}{c} \mathsf{Vrfy}(pk, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin \mathcal{Q} \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ dk \leftarrow \mathsf{Derive}(sk); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}}(pp, pk, \boxed{dk}); \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Application of HISE

Merit of HISE

- compact public key size
- reduce key management complexity
- simplify the design and analysis of high-level protocols

suitable for scenarios that simutaneously require privacy, authenticity and key escrow

Zether/PGP

(pk, sk)

outsource costly operations
e.g., expensive decryption

auditing

Application of HISE

Merit of HISE

- compact public key size
- reduce key management complexity
- simplify the design and analysis of high-level protocols

suitable for scenarios that simutaneously require privacy, authenticity and key escrow

outsource costly operations
e.g., expensive decryption

Zether/PGP

delegate decryption capability

sk

(pk, sk)

auditing

compromise the security of SIG

Application of HISE

Merit of HISE

- compact public key size
- reduce key management complexity
- simplify the design and analysis of high-level protocols

> suitable for scenarios that simutaneously require privacy, authenticity and key escrow

Zether/PGP

(pk, sk)

delegate decryption capability

$\longrightarrow$

dk

retain the security of SIG

outsource costly operations
e.g., expensive decryption

auditing

Outline

Starting Point: ISE from IBE

Paterson et al. [PSST11] give an elegant ISE construction from IBE.

(mpk, msk)

Naor transform | CHK transform

$I_0 = 0|\mathbf{v},\ \mathbf{v} \in \{0,1\}^n$

bit prefix partition trick
$\Rightarrow$ joint security

$I_1 = 1|\mathbf{v},\ \mathbf{v} \in \{0,1\}^n$

Naor transform:

$$\sigma \leftarrow \mathsf{Extract}(msk, 0|\tilde{m})$$
$$m \xleftarrow{R} M$$
$$c \leftarrow \mathsf{Enc}(mpk, 0|\tilde{m}, m)$$
$$m \stackrel{?}{=} \mathsf{Dec}(\sigma, c)$$

(pk, sk)

CHK transform:

$$(vk, sk) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$$
$$c \leftarrow \mathsf{Enc}(mpk, 1|vk, m)$$
$$\sigma \leftarrow \mathsf{OTS.Sign}(sk, c)$$
$$1 \stackrel{?}{=} \mathsf{OTS.Vefy}(vk, c, \sigma);$$
$$m \leftarrow \mathsf{Dec}(sk_{1|vk}, c)$$

ISE from IBE does not lend itself to HISE

msk plays the role of both sk and dk $\rightsquigarrow$ compromise strong joint security

HISE from Constrained IBE for Prefix Predicate

Main idea: msk acts as sk, secret keys for identities in I_1 as decryption key

Technical hurdle: decryption key should be short $\rightsquigarrow$ we need a succinct representation for all secret keys for identites in I_1 $\Leftarrow$ constrained IBE for prefix predicates $\Leftarrow$ BTE

(mpk, msk)

Naor transform

$I_0 = 0|\mathbf{v},\ \mathbf{v} \in \{0,1\}^n$

$\sigma \leftarrow \text{Extract}(msk, 0|\tilde{m})$

$m \xleftarrow{R} M$

$c \leftarrow \text{Enc}(mpk, 0|\tilde{m}, m)$

$m \stackrel{?}{=} \text{Dec}(\sigma, c)$

bit prefix partition trick

$\Rightarrow$ joint security

(pk, sk)

$dk \leftarrow \text{Constrain}(sk, 1|\mathbf{v})$

CHK transform

$I_1 = 1|\mathbf{v},\ \mathbf{v} \in \{0,1\}^n$

$(vk, sk) \leftarrow \text{OTS.KeyGen}(1^\lambda)$

$c \leftarrow \text{Enc}(mpk, 1|vk, m)$

$\sigma \leftarrow \text{OTS.Sign}(sk, c)$

$1 \stackrel{?}{=} \text{OTS.Vefy}(vk, c, \sigma)$

$m \leftarrow \text{Dec}(sk_{1|vk}, c)$

Outline

HISE from PKE and NIZKPoK (HI coversion)

Goal: add signing functionality to PKE in a generic manner

- bootstrap PKE in-use to HISE $\rightsquigarrow$ enables a seamless upgrade

Idea: create hierarchical key structure via OWF

1. picks $sk \xleftarrow{R} \{0,1\}^n$ as signing key
2. maps sk to randomness r via uniform OWF: $\mathsf{F}(sk) \rightarrow r$
3. runs $\mathsf{PKE.KeyGen}(r) \rightarrow (pk, dk)$

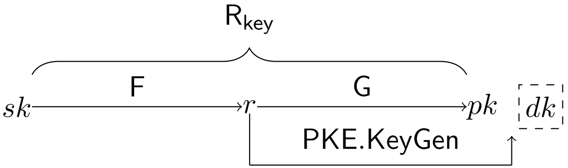


Figure: The hierarchical key structure

Design of HISE from PKE

The encryption component of HISE is simple: same as that of the underlying PKE.

But, we are facing the following technical hurdle when designing signature:

- sk is unstructured bit string, how to design signature?
- the signature should remain secure even in the presence of dk (partial leakage of sk) $\Rightarrow$ strong joint security

Solution

- using general-purpose public-coin ZKPoK to prove knowledge of sk
- prove $\mathsf{R}_{\mathsf{key}}$ is leakage-resilient one-way w.r.t. leakage r and thus certainly dk
  - minimum requirement on G: target-collision resistant

Strong joint security:

- SIG component: Sigma protocol for leakage-resilient one-way relation $\rightsquigarrow$ leakage-resilient SIG
- PKE component: zero-knowledge property $\rightsquigarrow$ $\mathcal{O}_{\mathsf{sign}}$ is useless + uniformity of F admits security reduction

Outline

Motivation of Global Escrow

Motivating example: large-scale collaborative working Apps such as Slack is getting popular $\rightsquigarrow$ encrypted communication may contain proprietary information

- employer may have the right to get access to all private communications for various reasons
  - naive solution: collect individual decryption key one by one $\Rightarrow$ impractical and inefficient
- employees need to be assured that even a malicious employer cannot slander them by forging signatures for fabricated communications

We further expect global escrow property

- there is a "super" key that can decrypt any ciphertext under any public key
- signature remains secure even in the presence of the "super" key

To attain global escrow property for HISE in a generic manner, we take a detour to revist global escrow PKE

Global Escrow PKE

Global escrow PKE: an escrow agent holds a global escrow decryption key that can decrypt ciphertexts encrypted under any public key

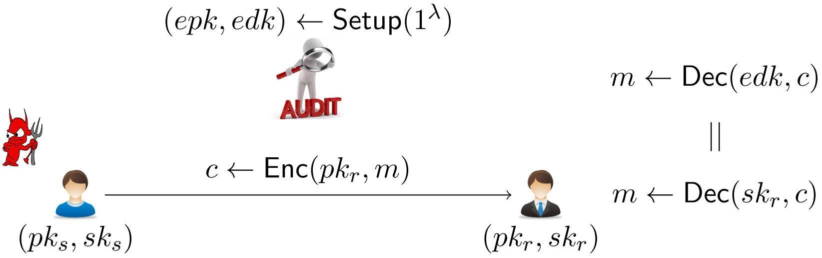
 $\cdots$  $\cdots$ 

(pk_1, sk_1) (pk_i, sk_i) (pk_n, sk_n)

The state of the art of global escrow PKE is less satisfactory

- long overdue for formal definition and generic construction
- the only known practical scheme based on standard assumption is the escrow ElGamal PKE proposed by Boneh and Franklin from bilinear maps

Formal Definition



Correctness: honestly generated CTs decrypting to the same result under edk and sk_r

Consistency: no PPT adversary can generate an ill-formed CT decrypting different results under edk and sk_r

Failure attempts

1. Identity-based encryption: does not know how to extend to the public-key setting (users must be able to generate keypairs themselves)
2. Broadcast encryption: sender could be malicious especially when he has incentive to evade oversight

Outline

Global Escrow PKE from NIZK and PKE GE conversion)

escrow center

$$pp_{\mathsf{nizk}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$$
$$pp_{\mathsf{pke}} \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$$
$$(epk, edk) \leftarrow \mathsf{PKE.KeyGen}(pp_{\mathsf{pke}})$$
$$pp = (pp_{\mathsf{nike}}, pk_{\mathsf{pke}}, epk), edk$$

$$\mathsf{NIZK.Verify}(c_1, c_2, \pi) \stackrel{?}{=} 1$$
$$m \leftarrow \mathsf{PKE.Dec}(edk, c_2)$$

sender

$$c_1 \leftarrow \mathsf{PKE.Enc}(pk, m; r_1)$$
$$c_2 \leftarrow \mathsf{PKE.Enc}(epk, m; r_2)$$
$$\pi \leftarrow \mathsf{NIZK.Prove}(c_1, c_2, (m, r_1, r_2))$$

receiver

(pk, sk)

$$\mathsf{NIZK.Verify}(c_1, c_2, \pi) \stackrel{?}{=} 1$$
$$m \leftarrow \mathsf{PKE.Dec}(sk, c_1)$$

Give a generic approach to compile any PKE into global escrow PKE

- enrich the application scope of the Naor-Yung transform beyond CCA security
- achieve CCA security with no overhead

Instantiation of the First Approach

Choices of primitives

- PKE: ElGamal PKE in EC groups
- NIZK: Groth-Sahai proof in standard model or Sigma proof in random oracle model

Improvement

- When PKE satisfies the "randomness fusion" property [BMV16], we can safely reuse the randomness and then apply twisted Naor-Yung transform ⇒ better efficiency

plenty of PKE schemes from the DDH, quadratic residuosity, and subset sum assumptions satisfy randomness fusion property.

Outline

Multiparty NIKE

$\mathsf{Setup}(1^\lambda, n) \to pp_{\mathsf{nike}}$

mild property
PK is efficiently recognizable

(pk_1, sk_1) … (pk_i, sk_i) … (pk_n, sk_n)

$\mathsf{ShareKey}(sk_i, \{pk_1, \ldots, pk_n\}) \to k$

pseudorandom

- $n = 2$: Diffie-Hellman key exchange [DH76]
- $n = 3$: Joux's key exchange [Jou04] from bilinear maps
- n is any positive integer
  - Boneh and Silverberg [BS02] using multilinear maps
  - Alamati et al. [AMPR19] using composable input homomorphic weak PRF

Global Escrow PKE from 3-party NIKE

escrow center

$$pp_{\mathsf{nike}} \leftarrow \mathsf{NIKE.Setup}(1^\lambda, 3)$$
$$(pk_\gamma, sk_\gamma) \leftarrow \mathsf{NIKE.KeyGen}(pp_{\mathsf{nike}})$$
$$pp = (pp_{\mathsf{nike}}, pk_\gamma), edk = sk_\gamma$$

$$k \leftarrow \mathsf{NIKE.ShareKey}(sk_\gamma, S)$$
$$m \leftarrow \mathsf{SKE.Dec}(k, c)$$

running 3-party NIKE in-the-head

$$S = \{pk_\alpha, pk_\beta, pk_\gamma\}$$

sender

$$(pk_\alpha, sk_\alpha) \leftarrow \mathsf{NIKE.KeyGen}(pp_{\mathsf{nike}})$$
$$k \leftarrow \mathsf{NIKE.ShareKey}(sk_\alpha, S)$$
$$c \leftarrow \mathsf{SKE.Enc}(k, m)$$

receiver

$$(pk_\beta, sk_\beta)$$

$$k \leftarrow \mathsf{NIKE.ShareKey}(sk_\beta, S)$$
$$m \leftarrow \mathsf{SKE.Dec}(k, c)$$

- pseudorandomness of shared key $k \Rightarrow$ IND-CPA/CCA security
- PK is efficiently recognizable $\Rightarrow$ consistency

Instantiation of the Second Approach: First Attempt

Joux's 3-party NIZK from symmetric pairing

(a, g^a)

(b, g^b)

$$pp = (\mathbb{G}, \mathbb{G}_T, e, g)$$
$$k \leftarrow e(g, g)^{abc}$$

(c, g^c)

supersingular curve ss-1536

$$|\mathbb{G}| = 1536$$
$$|\mathbb{G}_T| = 1536$$
$$|\mathbb{Z}_p| = 256$$

Instantiation of the Second Approach: First Attempt

Joux's 3-party NIZK from symmetric pairing

(a, g^a)

(b, g^b)

$pp = (\mathbb{G}, \mathbb{G}_T, e, g)$
$k \leftarrow e(g, g)^{abc}$

(c, g^c)

supersingular curve `ss-1536`
$|\mathbb{G}| = 1536$
$|\mathbb{G}_T| = 1536$
$|\mathbb{Z}_p| = 256$

Boneh-Franklin escrow ElGamal PKE

- Setup(1^λ): $edk \xleftarrow{R} \mathbb{Z}_p$, $epk \leftarrow g^{edk}$.
- KeyGen(pp): $sk \xleftarrow{R} \mathbb{Z}_p$, $pk \leftarrow g^{sk}$.
- Enc(pk, m): $sk_t \xleftarrow{R} \mathbb{Z}_p$, $pk_t \leftarrow g^{sk_t}$; $k \leftarrow$ ShareKey$(sk_t, S = \{pk_t, pk, epk\})$, $c = (pk_t, m \oplus k)$
- Dec(sk, c): $k \leftarrow$ ShareKey$(sk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$.
- Dec$'(edk, c)$: $k \leftarrow$ ShareKey$(edk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$.

Instantiation of the Second Approach: First Attempt

Joux's 3-party NIZK from symmetric pairing

(a, g^a)

(b, g^b)

$$pp = (\mathbb{G}, \mathbb{G}_T, e, g)$$
$$k \leftarrow e(g, g)^{abc}$$

(c, g^c)

supersingular curve ss-1536

$|\mathbb{G}| = 1536$

$|\mathbb{G}_T| = 1536$

$|\mathbb{Z}_p| = 256$

symmetric pairing is too slow

Boneh-Franklin escrow ElGamal PKE

- Setup(1^λ): $edk \xleftarrow{R} \mathbb{Z}_p$, $epk \leftarrow g^{edk}$.
- KeyGen(pp): $sk \xleftarrow{R} \mathbb{Z}_p$, $pk \leftarrow g^{sk}$.
- Enc(pk, m): $sk_t \xleftarrow{R} \mathbb{Z}_p$, $pk_t \leftarrow g^{sk_t}$; $k \leftarrow$ ShareKey$(sk_t, S = \{pk_t, pk, epk\})$, $c = (pk_t, m \oplus k)$
- Dec(sk, c): $k \leftarrow$ ShareKey$(sk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$.
- Dec$'(edk, c)$: $k \leftarrow$ ShareKey$(edk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$.

Instantiation of the Second Approach: Second Attempt

The above instantiation explain the only prior known global escrow PKE in theory

But

- symmetric pairing is too slow
- the original Joux's protocol inherently relies on symmetric pairing

Idea: adapt Joux's protocol with asymmetric pairing

Instantiation of the Second Approach: Second Attempt

The above instantiation explain the only prior known global escrow PKE in theory

But

- symmetric pairing is too slow
- the original Joux's protocol inherently relies on symmetric pairing

Idea: adapt Joux's protocol with asymmetric pairing

Joux's 3-party NIZK from asymmetric pairing

(a, g_1^a, g_2^a)

(b, g_1^b, g_2^b)

$$pp = (\mathbb{G}, \mathbb{G}_T, e, g)$$
$$k \leftarrow e(g_1, g_2)^{abc}$$

(c, g_1^c, g_2^c)

Instantiation of the Second Approach: Second Attempt

The above instantiation explain the only prior known global escrow PKE in theory

But

- symmetric pairing is too slow
- the original Joux's protocol inherently relies on symmetric pairing

Idea: adapt Joux's protocol with asymmetric pairing

Joux's 3-party NIZK from asymmetric pairing

(a, g_1^a, g_2^a)

(b, g_1^b, g_2^b)

$pp = (\mathbb{G}, \mathbb{G}_T, e, g)$
$k \leftarrow e(g_1, g_2)^{abc}$

(c, g_1^c, g_2^c)

shortcomings

key and ciphertext size get doubled

decryption is expensive

pairing is needed to check if ciphertext is valid

$e(g_1, g_2^b) \stackrel{?}{=} e(g_1^b, g_2)$

Improved Scheme based on a Relaxed Version of NIKE: Final Attempt

relaxed 3-party NIKE from asymmetric pairing

keypairs could be of different types

type A: (a, g_1^a, g_2^a)

$$pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$$
$$k \leftarrow e(g_1, g_2)^{abc}$$

type B: (b, g_1^b)

type C: (c, g_2^c)

curve bls12-381
$|\mathbb{G}_1| = 381$
$|\mathbb{G}_2| = 762$
$|\mathbb{G}_T| = 1524$
$|\mathbb{Z}_p| = 256$

much faster and compact

New Global Escrow PKE

- Setup(1^λ): $edk \xleftarrow{R} \mathbb{Z}_p$, $epk = (g_1^{edk}, g_2^{edk})$ (type A)
- KeyGen(pp): $sk \xleftarrow{R} \mathbb{Z}_p$, $pk \leftarrow g_2^{sk}$ (type B)
- Enc(pk, m): $sk_t \xleftarrow{R} \mathbb{Z}_p$, $pk_t \leftarrow g_1^{sk_t}$ (type C);
  $k \leftarrow$ ShareKey$(sk_t, S = \{pk_t, pk, epk\})$, $c = (pk_t, m \oplus k)$
- Dec(sk, c): $k \leftarrow$ ShareKey$(sk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$
- Dec$'(edk, c)$: $k \leftarrow$ ShareKey$(edk, S = \{pk_t, pk, epk\})$, $m \leftarrow c_2 \oplus k$

Global Escrow HISE

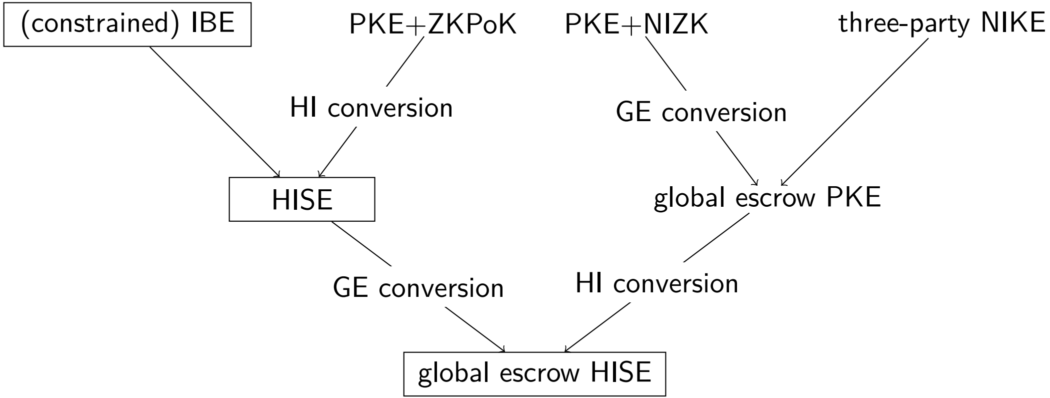


Figure: Technology roadmap of global escrow HISE. The rectangles denote our newly introduced cryptographic schemes.

Applications of Global Escrow HISE

super key
edk

slack 钉钉

(pk_1, sk_1) $\ldots$ (pk_i, sk_i) $\ldots$ (pk_n, sk_n)

- The employer can perform efficient large-scale supervision over private communications with "super" key.
- The employees are assured that even a malicious boss of the "super" key cannot slander them by forging signatures for fabricated communications.

Outline

Comparision with Cartesian-Product CPK and ISE

Table: Comparison between CP-CPK, ISE, and our (global escrow) HISE

Scheme	strong joint security	individual escrow	global escrow	key reuse	certificate cost
CP-CPK [PSST11]	✓	✓	✗	✗	$\times 2$
ISE [PSST11]	✗	✗	✗	✓	$\times 1$
HISE	✓	✓	✗	✓	$\times 1$
global escrow HISE	✓	✓	✓	✓	$\times 1$

For certificate cost, $\times 1$ (resp. $\times 2$) means the cost associated with one (resp. two) certificate(s). As aforementioned, certificate costs include but not limit to registration, issuing, storage, transmission, verification, and building/recurring fees. Take SSL certificate as an example, one certificate is roughly 1KB, takes roughly 200∼300ms to transmit in WAN setting with 50Mbps network bandwidth and 8ms to verify. The monetary cost for an SSL certificate varies depending on features and business needs. While the cost of an SSL certificate for common usage is $10∼$2000/year, the banks and large financial institutions could spend up to $500,000/year on an SSL certificate with high-level security guranttee.

Instantiation of (Global Escrow) HISE

Instantiation of HISE

- HISE scheme 1: Naor+CHK transform (Boneh-Franklin IBE)
- HISE scheme 2: HI conversion (ElGamal PKE+Poseidon hash+Spartan)

Instantiation of global escrow HISE

- global escrow HISE scheme 1: twsited Naor-Yung transform (HISE scheme 1)
- global escrow HISE scheme 2: HI conversion (global escrow PKE from 3-party NIKE+Poseidon hash+Spartan)

Experimental Results

Table: Efficiency comparison of CPK and our proposed (global escrow) HISE schemes

Scheme	efficiency (ms) [# exp, #pairing]							sizes (bits) [# $\mathbb{G}$, # $\mathbb{Z}_p$]			
	KGen	Sign	Vrfy	Enc	Dec	Der	Dec′	$\|pk\|$	$\|sk\|$	$\|c\|$	$\|\sigma\|$
CP-CPK	0.015 [2, 0]	0.064 [1, 0]	0.120 [2, 0]	0.118 [2, 0]	0.056 [1, 0]	⊘ ⊘	⊘ ⊘	512 $2\mathbb{G}$	512 $2\mathbb{Z}_p$	512 $2\mathbb{G}$	512 $[\mathbb{G}, \mathbb{Z}_p]$
HISE scheme 1	0.057 [1, 0]	0.148 [1, 0]	0.733 [0, 2]	0.569 [2, 1]	0.364 [0, 1]	0.148 [1, 0]	⊘ ⊘	381 $\mathbb{G}_1$	256 $\mathbb{Z}_p$	1905 $[\mathbb{G}_1, \mathbb{G}_T]$	762 $\mathbb{G}_2$
HISE scheme 2	0.058 [1, 0]	3.5s N/A	250 N/A	0.115 [2, 0]	0.056 [1, 0]	0.0004 N/A	⊘ ⊘	256 $\mathbb{G}$	256 $\mathbb{Z}_p$	512 $2\mathbb{G}$	40K N/A
global escrow HISE scheme 1	0.057 [1, 0]	0.148 [1, 0]	0.733 [0, 2]	1.462 [5, 2]	1.505 [4, 1]	0.148 [1, 0]	1.505 [4, 1]	381 $\mathbb{G}_1$	256 $\mathbb{Z}_p$	5590 $[2\mathbb{G}_1, 3\mathbb{G}_T, \mathbb{Z}_p]$	762 $\mathbb{G}_2$
global escrow HISE scheme 2	0.057 [1, 0]	3.5s N/A	250 N/A	0.629 [2, 1]	0.531 [1, 1]	0.0004 N/A	0.532 [1, 1]	381 $\mathbb{G}_1$	256 $\mathbb{Z}_p$	2286 $[\mathbb{G}_2, \mathbb{G}_T]$	40K N/A

Performance of Cartesian product CPK and (global escrow) HISE schemes with 128-bit security level. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ refers to asymmetric pairing groups. $\mathbb{G}$ refers to ordinary elliptic group. The symbol ⊘ indicates that there is no corresponding algorithm. The symbol N/A indicates that the efficiency (or bandwidth) is hard to measure by algebra operations (or elements).

A Byproduct: Global Escrow PKE

Table: Comparison of escrow ElGamal PKE [BF03] and our global escrow PKE

Scheme	efficiency (ms) [# exp, #pairing]					sizes (bits) [# $\mathbb{G}$, # $\mathbb{Z}_p$]				
	Setup	KGen	Enc	Dec	Dec$'$	$\lvert pp\rvert$	$\lvert edk\rvert$	$\lvert pk\rvert$	$\lvert sk\rvert$	$\lvert c\rvert$
Boneh-Franklin escrow ElGamal PKE	2.879 [2, 0]	2.014 [1, 0]	8.723 [2, 1]	6.654 [1, 1]	6.745 [1, 1]	3072 $2\mathbb{G}$	256 $\mathbb{Z}_p$	1536 $\mathbb{G}$	256 $\mathbb{Z}_p$	3072 $[\mathbb{G}, \mathbb{G}_T]$
our proposed global escrow PKE	0.243 [4, 0]	0.058 [1, 0]	0.680 [2, 1]	0.579 [1, 1]	0.586 [1, 1]	2286 $[2\mathbb{G}_1, 2\mathbb{G}_2]$	256 $\mathbb{Z}_p$	381 $\mathbb{G}_1$	256 $\mathbb{Z}_p$	2286 $[\mathbb{G}_2, \mathbb{G}_T]$

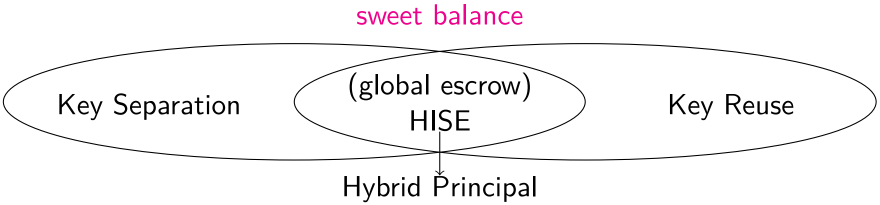
Performance of global escrow PKE schemes with 128-bit security level. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ refers to asymmetric pairing groups. $(\mathbb{G}, \mathbb{G}_T)$ refers to symmetric pairing groups. We report times for setup, key generation, encryption, and (escrow) decryption, as well as the sizes of public parameters pp, global escrow decryption key edk, public key pk, secret key sk, and ciphertext c.

$12 \sim 30\times$ speed up

Our implementation is released on Github: https://github.com/yuchen1024/HISE

Outline

Summary



HISE (formal definition + generic constructions)

- reconcile the apparent conflict between key separation and key resue
- resolve the problem left open in Verheul [Ver01]
- can be used as a drop-in replacement of PKE+SIG in scenarios that requires authenticity, confidentiality and auditibility simutaneously
- both users and authority have incentives to deploy

Global escrow PKE revisit (formal definition + generic constructions)

- indicate a new application of Naor-Yung paradigm
- estabilish a connection from 3-party NIKE

Thanks for Your Attention!

Any Questions?

Reference I

[AMPR19] Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Arnab Roy. Minicrypt primitives with algebraic structure and applications. In *Advances in Cryptology - EUROCRYPT 2019*, volume 11477 of *Lecture Notes in Computer Science*, pages 55–82. Springer, 2019.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computation*, 32:586–615, 2003.

[BMV16] Silvio Biagioni, Daniel Masny, and Daniele Venturi. Naor-yung paradigm with shared randomness and applications. In *Security and Cryptography for Networks - 10th International Conference, SCN 2016*, volume 9841 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2016.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. 2002. `http://eprint.iacr.org/2002/080`.

[DH76] Whitefield Diffie and Martin E. Hellman. New directions in cryptograpgy. *IEEE Transactions on Infomation Theory*, 22(6):644–654, 1976.

[Jou04] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 17(4):263–276, 2004.

[PSST11] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. On the joint security of encryption and signature, revisited. In *Advances in Cryptology - ASIACRYPT 2011*, pages 161–178, 2011.

[Ver01] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2001.