Two-Dimensional Representation of Cover Free Families and its Applications: Short Signatures and More

Shota Yamada
The University of Tokyo

Session ID: CRYP-303

Session Classification: Advanced

RSA CONFERENCE 2012

Our Results

- We proposed a new technique for the use of cover free families.
- We apply the technique to construct
  - q-resilient IBE
  - q-bounded CCA secure PKE
  - m-time signatures
  - Short signatures

  with smaller public key size than previous constructions.

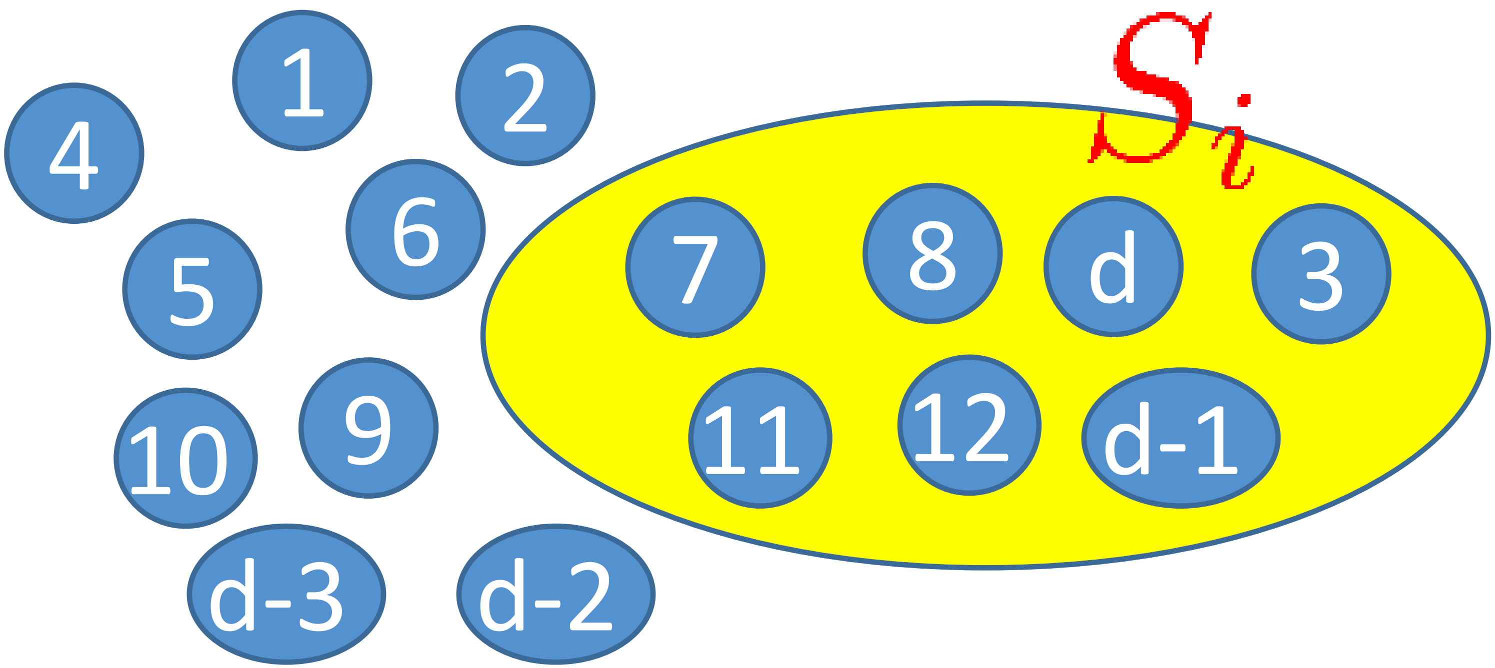
Agenda

- What are Cover Free Families?
- Our Main Idea
- Application(1): q-Resilient IBE
- Application(2): Short Signatures

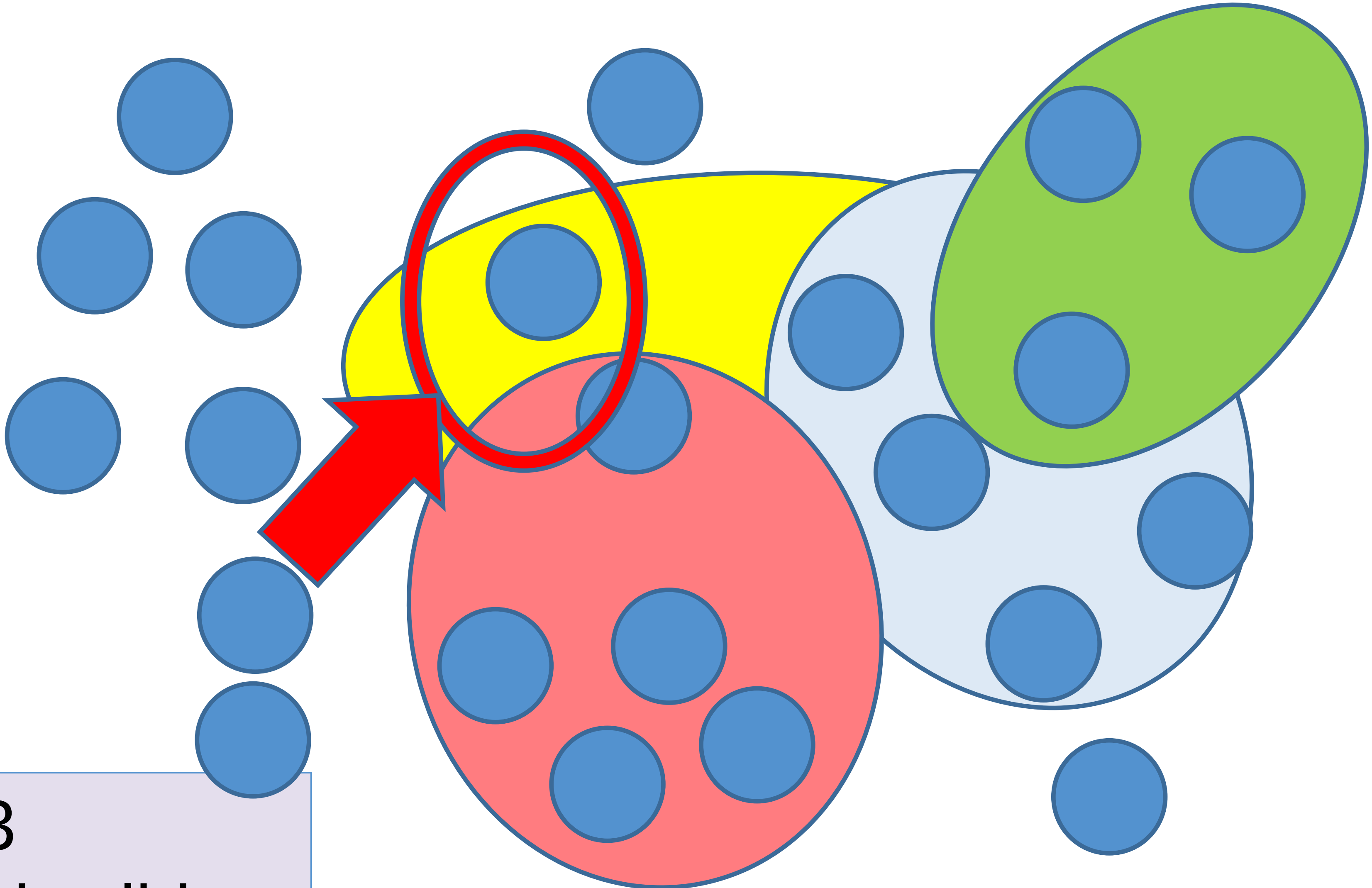
What are Cover Free Families?

m-Cover Free Families

- Index $\{1, 2, \ldots, d\}$
- Family of subsets $\{S_i\}_{i \in [\nu]}$ where $S_i \subset \{1, 2, \ldots, d\}$

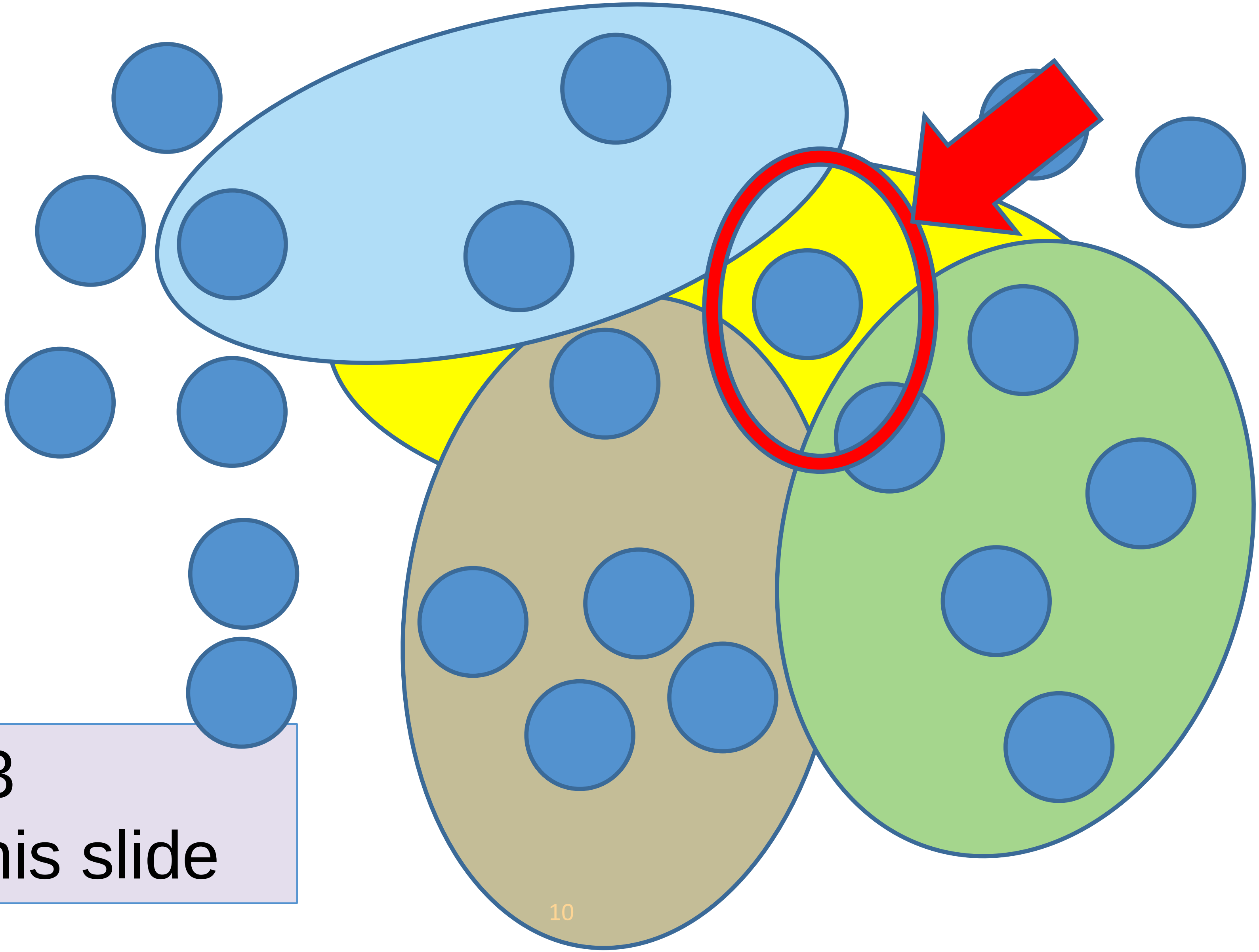


m-Cover Free Families



m=3
In this slide

m-Cover Free Families



m=3
In this slide

Applications of Cover Free Families in Previous Results

Following papers are related to our result:

- [Cramer, Hanaoka, Hofheinz, Imai, Kiltz, Pass, Shelat, Vaikntanathan @ Asiacrypt '07] ([CHH+07])

  Construction of q-bounded CCA secure PKE

- [Hofheinz, Jager, Kiltz @ Asiacrypt'11]

  ([HJK11])

  Construction of short signature schemes

Properties of Schemes Based on Cover Free Families (informal)

The schemes in [CHH+07,HJK11]

- The public key size is very large

  due to the use of cover free family

- Ciphertext/Signature size is very small

We reduce public key size of these schemes while preserving the size of signatures/ciphertext.

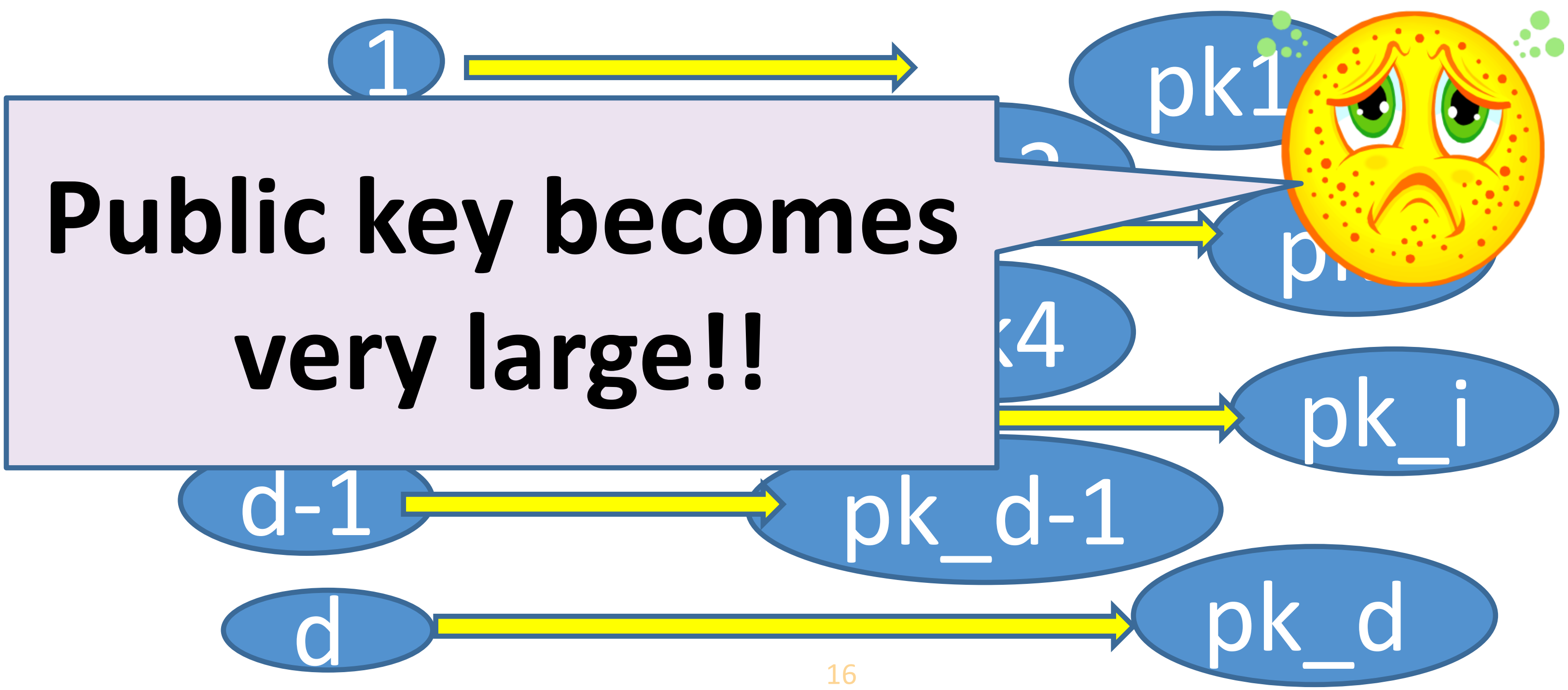
Our Main Idea

A Reason for Large Public Key

- KeyGen process of [CHH+07] and [HJK11]

1.Generate
cover free family
(d is large)

2.Generate
PK components



Idea of Previous Constructions

[CHH+07, HJK11]

1 2 3 4 $\cdots$ d

$$g^{a_1} \quad g^{a_2} \quad g^{a_3} \quad g^{a_4} \quad \cdots \quad g^{a_d}$$

Each index is associated with one group element.

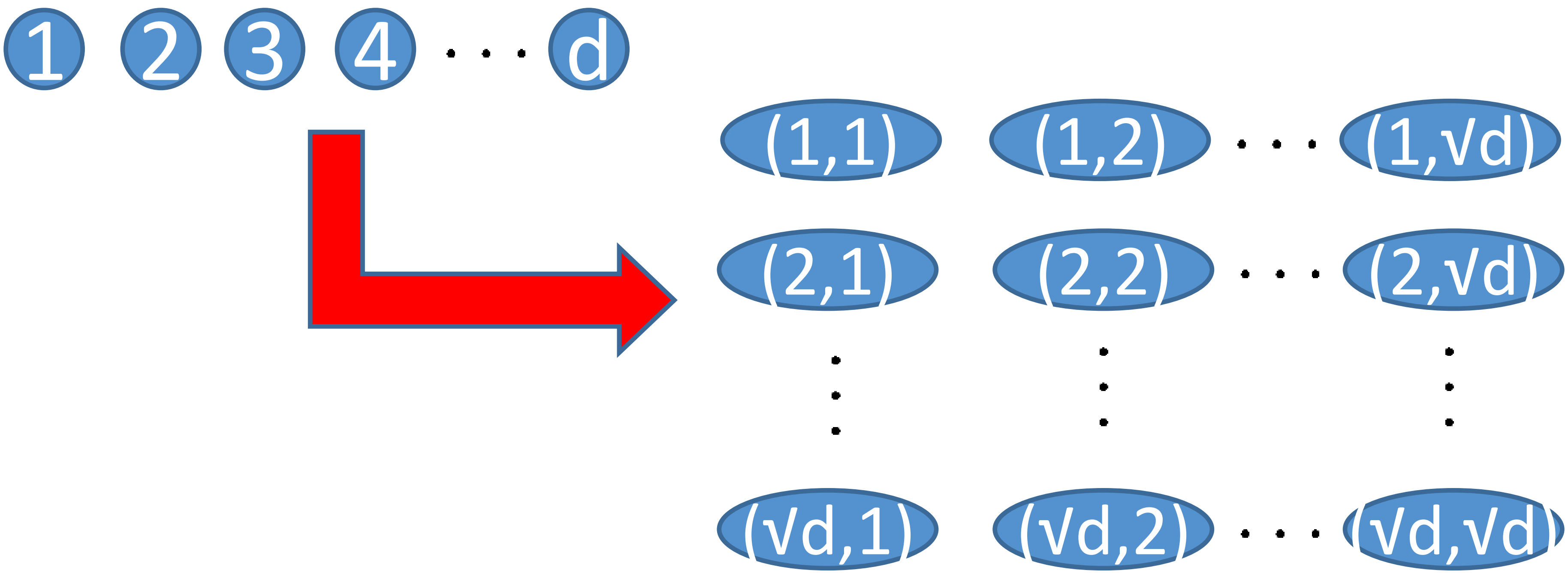
The public key size becomes O(d)

d is large!!

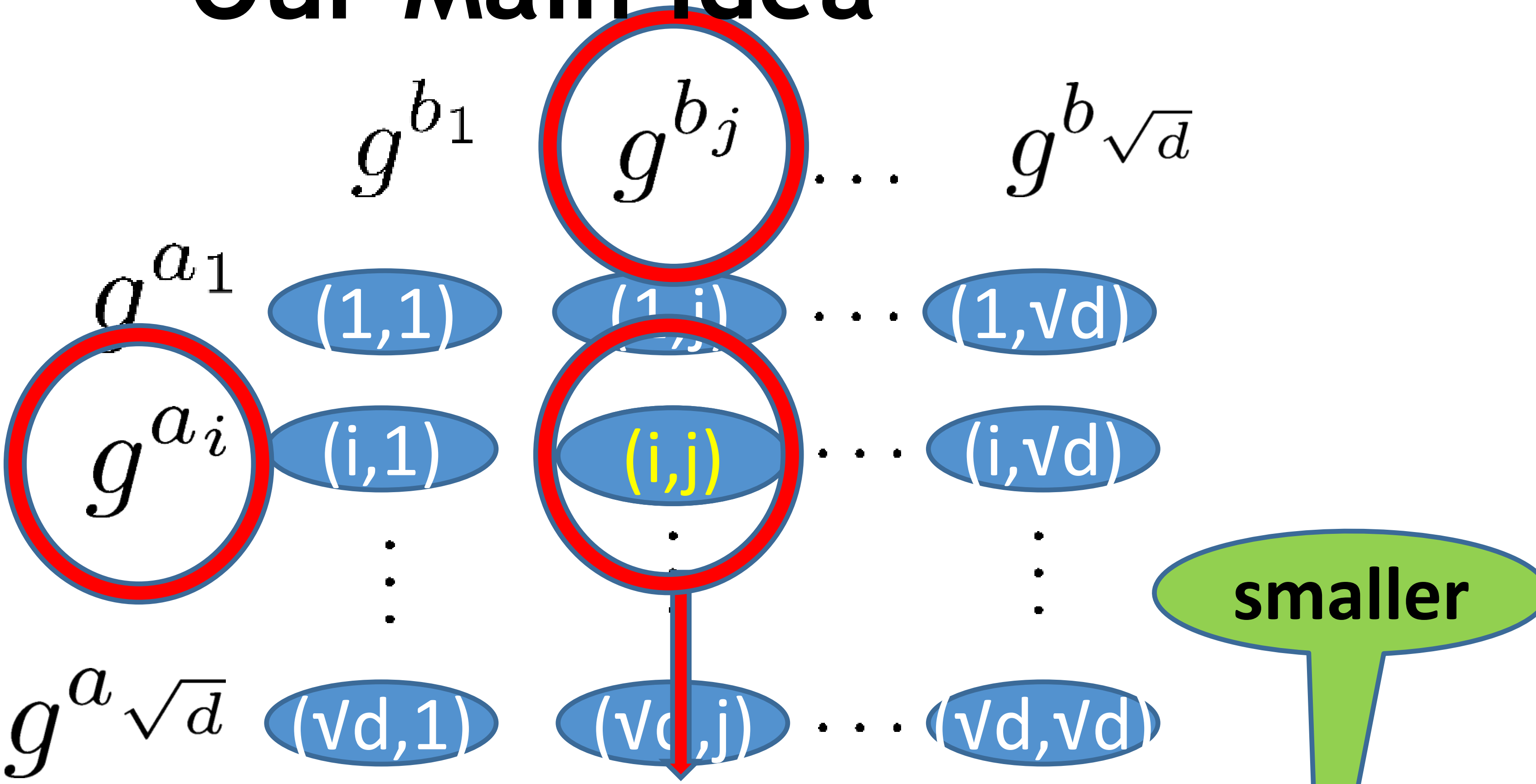
Our Main Idea

- We change the set of indices from $\{1, 2, \ldots, d\}$

to $\{(1,1), (1,2), \ldots, (\sqrt{d}, \sqrt{d})\}$



Our Main Idea



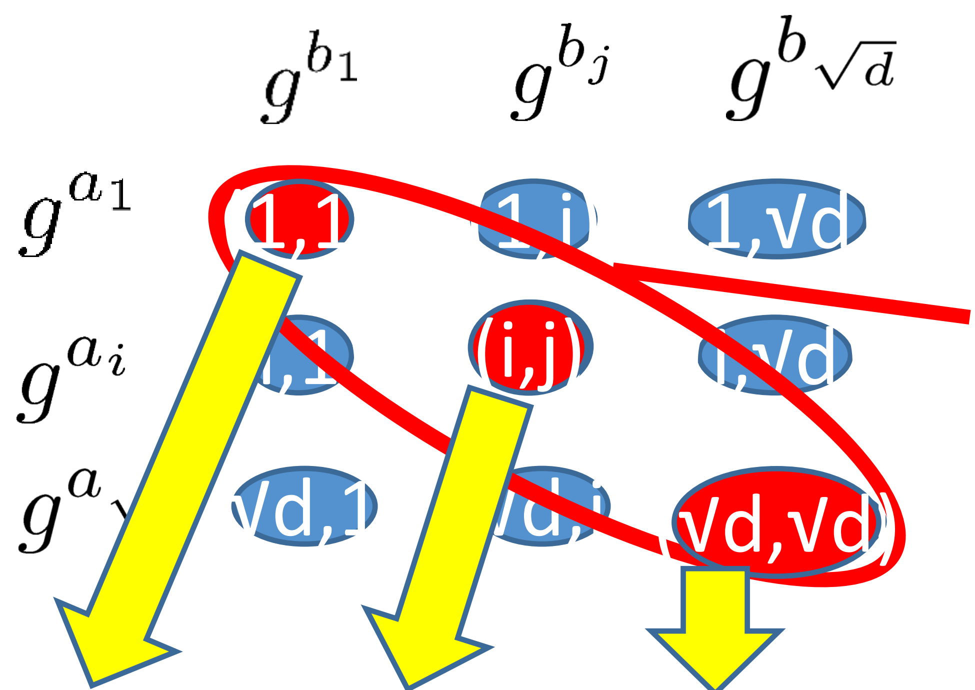
We associat… …up element
with each "… …lumn".
Size of publi… …omes $O(\sqrt{d})$

Our Main Idea

Associate $D \in \mathcal{D}$ with $H(D) \in \mathbb{G}$

ID / Message

Private key for ID /Signature



$$S : \mathcal{D} \to 2^{\sqrt{d}\times\sqrt{d}}$$

$$S(D) \subset [\sqrt{d}] \times [\sqrt{d}]$$

$$g^{a_1 b_1} \times g^{a_i b_j} \times g^{a_{\sqrt{d}} b_{\sqrt{d}}} = \prod_{(i,j)\in S(D)} g^{a_i b_j} = H(D)$$

Why "Two" Dimensions?

- Three or more dimensions technique does not seem to work.
  - Verification does not work in the case of a signature scheme.

$$e(g^{ab}, g) = e(g^a, g^b) \qquad \times \quad e(g^{abc}, g) = \cdots ??$$

  - Encryption does not work in the case of q-resilient IBE scheme.
- Because we resort to bilinear map.

Our technique could be extended to higher dimensions if there exists multi-linear form and appropriate computationally hard problem.

$$e(g^{a_1}, g^{a_2}, \cdots, g^{a_k}) = e(g, \ldots, g)^{a_1 a_2 \cdots a_k} \ ??$$

Novelty of Our Technique

- In fact, “matrix like” or “two dimensional” technique has been used in many previous papers.
  - [PW08@STOC],[HJKS10@PKC],[BW10@ACNS] etc.

- Our work adapted the technique to the case where cover free families are used for the first time.
- It is also the first time the technique is used for a construction of signature schemes.

(to the best of our knowledge)

Application(1): q-resilient IB-KEM

Application(1):q-Resilient IBE

- q-Resilient secure IBE scheme (actually, IB-KEM)

The scheme is q-resilient/bounded secure if the scheme is semantically secure against adversaries who cannot make more than q KeyGen/Decryption queries.

q-Resilient secure IBE

CHK transform

Naor transform

q-Bounded CCA secure PKE

q-Time signature

Our q-Resilient IBE Scheme

Public key $g^{a_1}, \ldots, g^{a_{\sqrt{d}}}, g^{b_1}, \ldots, g^{b_{\sqrt{d}}}$

Master secret key $a_1, \ldots, a_{\sqrt{d}}, b_1, \ldots, b_{\sqrt{d}}$

Private key for ID $SK_{ID} = \prod_{(i,j) \in S(ID)} g^{a_i b_j} = H(ID)$

where $S(ID) \subset [\sqrt{d}] \times [\sqrt{d}]$

Ciphertext $C = g^r$

KEM key $K = \prod_{(i,j) \in S(ID)} e(g^{a_i}, g^{b_j})^r = e(g^r, H(ID))$

Comparison (q-Resilient IB-KEM)

	Ciphertext size	Public key size	Private key size	Assumption
[CHH+07] (implicit)	$1 \times \lvert g \rvert$	$16q^2\lambda \times \lvert g \rvert$	$1 \times Z_p$	DDH
Ours	$1 \times \lvert g \rvert$	$8q\sqrt{\lambda} \times \lvert g \rvert$	$1 \times Z_p$	DBDH
(Heng, Kurosawa'04	$2 \times \lvert g \rvert$	$(q+3) \times \lvert g \rvert$	$2 \times Z_p$	DDH)

q: Upper bound of number of KeyGen query

λ: Security parameter

Our q-Bounded CCA Secure PKE

Apply CHK transform (+ idea of BMW) to our proposed IB-KEM

Public key $g^{a_1}, \ldots, g^{a_{\sqrt{d}}}, g^{b_1}, \ldots, g^{b_{\sqrt{d}}}$

Secret key $a_1, \ldots, a_{\sqrt{d}}, b_1, \ldots, b_{\sqrt{d}}$

Ciphertext $C = g^r$

KEM key $K = \prod_{(i,j) \in S(C)} e(g^{a_i}, g^{b_j})^r = e(g^r, H(C))$

$$S(C) \subset [\sqrt{d}] \times [\sqrt{d}]$$

Comparison (q-Bounded CCA PKE)

	Ciphertext size	Public key size	Assumption
[CHH+07]	$1 \times \lvert g \rvert$	$16q^2\lambda \times \lvert g \rvert$	DDH
Ours	$1 \times \lvert g \rvert$	$8q\sqrt{\lambda} \times \lvert g \rvert$	DBDH

q: Upper bound of number of KeyGen query

λ: Security parameter

Our m-Time Signature

Apply Naor transform to
our proposed IB-KEM

Public (Verification) key $g^{a_1}, \ldots, g^{a_{\sqrt{d}}}, g^{b_1}, \ldots, g^{b_{\sqrt{d}}}$

Secret (Signing) key $a_1, \ldots, a_{\sqrt{d}}, b_1, \ldots, b_{\sqrt{d}}$

Signature on M $$\sigma = \prod_{(i,j)\in S(M)} g^{a_i b_j}$$

$$S(M) \subset [\sqrt{d}] \times [\sqrt{d}]$$

Verification $$e(g,\sigma) \stackrel{?}{=} \prod_{(i,j)\in S(M)} e(g^{a_i}, g^{b_j})$$

Comparison（m-Time Signature）

	Signature size	Public key size	Assumption
Ours	$1 \times \lvert g \rvert$ 	$8m\sqrt{\lambda} \times \lvert g \rvert$ 	CDH 
[Zaverucha-Stinson'10]	$1 \times \lvert g \rvert + 10\text{bits}$ 	$16m^2\lambda \times \lvert g \rvert$ 	DL 

Application(2): Short Signature

Application(2):Short Signature

[HJK'11]

For 80-bit security,

- The signature length is only 200-bits.

- Public key size is 26,000,000-bit long.

The public key size is very large, due to the use of cover free family.

We can reduce the size by our technique.

Our Short Signature Scheme (simplified form)

Public (Verification) key

$$g^{a_1}, \ldots, g^{a_{\sqrt{d}}}, g^{b_1}, \ldots, g^{b_{\sqrt{d}}}, X = g^x$$

Secret (Signing) key

$$a_1, \ldots, a_{\sqrt{d}}, b_1, \ldots, b_{\sqrt{d}}, x$$

Signature on message M

$$(s, \sigma) =$$

$$s \in_R \{0,1\}^l \qquad \sigma = \Big(\prod_{(i,j) \in S(M)} g^{a_i b_j} \Big)^{1/(x+s)}$$

Verification

$$e(g^s X, \sigma) \stackrel{?}{=} \prod_{(i,j) \in S(M)} e(g^{a_i}, g^{b_j})$$

Comparison (Short Signature)

	Signature size	Public key size	Efficiency (Sign)	Efficiency (Verify)
[HJK'11]	200	2.6×10^7	$1 \times$ Exp	$2 \times$ Pairing
Ours (1)	200	2.7×10^6	$1 \times$ Exp	$11 \times$ Pairing
Ours (2)	200	2.0×10^5	$1 \times$ Exp	$200 \times$ Pairing

Secure under q-DH assumption. 80 bit security.

Conclusion

- We proposed a new technique for the use of cover free family.
- Based on our idea, we can compress the size of public keys in
  - q-resilient IB-KEM
  - q-bounded CCA secure KEM
  - m-time signature
  - short signature
- Signature/Ciphertext size of the resulting schemes are very short whereas the size of the public key are shorter than previous constructions.

Secure Computation, I/O efficient algorithms and Distributed Signatures

Jonas Kölker, w. Damgaard & Toft

Aarhus University

Session ID: CRYP-303

Session Classification: Advanced

RSACONFERENCE2012

The Motivating Scenario

- You put some data in the cloud
- Your friends put their data in the cloud
- You want to compute on that data, securely

- Some of them are not really friends (or hacked)
- We don’t really trust the cloud completely either

- Storage is dear; we want to compress our data
- We want the cloud-side programs to be simple

Formalising The Scenario

- Players $p_1, \ldots, pn$ (you and your friends)
- Servers $D_1, \ldots, Dm$(in the cloud)

- Store data in blocks: $blk = (x_1, \ldots, xk)$
- Choose f of degree $\leq d$, uniformly randomly, subject to $f(-i) = xi$; give $f(j)$ to server j

- Secure vs. $d\ - k$ bad servers; pick k in $\Theta(m)$
- We must care about I/O-efficiency of algorithms

epona@cs.au.dk

Universally Composable Functionality

- Input(i, v) – memory[v] := player[i].recv()
- Output(v) – player[all].send(memory[v])
- Operation(•, v1, v2, v3)
  - memory[v3] := memory[v1] • memory[v2]
  - "•" is one of +, -, * or ≤ (which returns 0 or 1)
- Const(v, x) – memory[v] := x
- Random(v) – memory[v] := sample
- Write(adrs, blkid) – disk[blkid] := memory[adrs]
- Read(adrs, blkid) – memory[adrs] := disk[blkid]

Three Related Read/Write Protocol Pairs

- Passively Secure

- Information theoretically and actively secure
- Computationally and actively (statically) secure

- The latter two are extensions of the former
- Focus is on the computationally secure

The Passively Secure Write Protocol

- Generate $d-k-1$ shared random values:
  - $[r_1], \ldots, [r_{d-k-1}]$
- For j = $1, \ldots, m$, let:
  - $[f(j)] = \sum_{i=1}^{k} \lambda_i^j \left[x_{adrs_i}\right] + \sum_{i=k+1}^{d+1} \lambda_i^j [r_{i-k}]$
- For $j = 1, \ldots, m$, each player sends "write blkid" and their share of $[f(j)]$ to server j.
- Each server j reconstructs $f(j)$ and stores it at address $blkid$, i.e. $disk_j[blkid] \coloneqq f(j)$

The Passively Secure Read Protocol

- Each player sends “read $blkid$” to each server
- Each server j shares its $f(j)$ among the players
  - (It recalls $f(j)$ as $disk_j[blkid]$)
- Each player computes $[x_{adrs_i}] \coloneqq \sum_{j=1}^{m} \delta_j^i [f(j)]$

- Lemma 1 and 2: the λs and δs exist
  - That’s basically Lagrange interpolation

- Security: degrees vs. size of corruption sets

epona@cs.au.dk

Handling Active Corruption

The Template For The Active Protocols

- To be secure against actively corrupted servers, sign all the data sent to the servers
- To detect replays, use sequence numbers
- To detect wrong sequence numbers, use majority vote

- Two kinds of signature schemes: information theoretically secure and computationally secure
- We're going to use Schnorr's signatures

Using Schnorr's Signature Scheme

- Public keys: $\alpha, \beta \in G$
- Secret key: a such that $\beta = \alpha^a$
- $\mathrm{Sig}(c) = (\gamma, \delta)$ such that $\gamma = \alpha^\delta \beta^{H(\gamma,c)}$

- Players hold a sharing $[a]$ of the secret key
- For efficiency, sign a Pedersen commitment to the message, as $c = gmhr$ can safely be public.

- Need random $[r]$s w. α^r and $([u],[v])$s w. $g^u h^v$.

epona@cs.au.dk

The Actively Secure Write Protocol

- Each player sends "Begin write at $blkid$" to each server, receives c_{blk} by majority, increments it
- Create random sharings, $[r_1], \ldots, [r_{(d-(k-1))}]$
- Each player computes their share of D_j's share
  - $[s_j] = \sum_{i=1}^{k} \lambda_i^j \, [x_{adrs_i}] + \sum_{i=k+1}^{d+1} \lambda_i^j [rk-k]$
- Players generate $c_j' = g^{u_j} h^{v_j}, [u_j], [v_j]$ and $[x]$.
- Players compute $[sj - uj]$ and open to p_u. He reconstructs $\tau_j = sj - uj$ and broadcasts those.

The Actively Secure Write Protocol (cont)

- Players open x, check $\sum_j x^j([s_j] - [u_j] - \tau_j) \stackrel{?}{=} 0$
- Players compute $c_j = g^{\tau_j} c'_j$, get $[r_i]$ and $\gamma_i = \alpha^{r_i}$
- Players compute $[\delta_j] = [r_j] - [a]H(\gamma_j, cj, cblk)$
- Players send "Write $blkid$ with $([s_j], [v_j], [\delta_j], \gamma_j)$"
- Servers compute $s_j, vj, \delta_j, \gamma_j$, with error correction and majority decision, increment c_{blk}, store it
  - i.e. $disk_j[blkid] = (sj, vj, \delta_j, \gamma_j)$

- This is secure…

The Actively Secure Read Protocol

- Players send "Read at blk to p_u" to each server
- Servers send γ_j, δ_j, cj to p_u and $c_{blk}, [s_j'], [v_j']$ to all
- Players produce $[t_j], [w_j], g^{t_j}h^{w_j}$ for $j = 1, \ldots, m$
- Players open $[s_j' - tj], [v_j' - wj]$ to p_u
- p_u reconstructs $x_j = s_j' - tj$ and $y_j = v_j' - wj$.
- p_u validates (γ_j, δ_j) against $(c_j, cblk)$
  - and checks that $c_j = g^{x_j}h^{y_j} \cdot g^{t_j}h^{w_j}$
- p_u broadcasts $\gamma_j, \delta_j, cj, xj, yj$

epona@cs.au.dk

The Actively Secure Read Protocol (cont)

- Players verify $(\gamma j, \delta j)$ against $(cj, cblk)$ and c_j against x_j, yj, tj, wj, i.e. that $c_j = g^{x_j} h^{y_j} \cdot g^{t_j} h^{w_j}$
- The players compute $\left[x_{adrs_i}\right] = \sum_{j=1}^{m} \delta_j' \, [t + xj]$

- This is secure…

epona@cs.au.dk

Generating Randomness

Producing Randomness With Related Data

- A protocol for batch producing $([r], \alpha^r)$

- Generate $[r_b^a]$ and $[xa]$ for $a = 1 \dots n$, $b = 0 \dots m$
- In parallel, for $a = 1, \dots, n$:
  - Each player opens $[r_b^a]$ to p_a for $b = 0, \dots, m$
  - p_a broadcasts $\chi_b^a = \alpha^{r_a^b}$ for $b = 0, \dots, m$
  - Everybody broadcasts their shares of $[x_a]$
  - Players compute $[ya] = [\sum_{b=0}^{m} x_a{}^b r_b^a]$
  - All players check that $\alpha^{y_a} = \prod_{b=0}^{m} (\chi_b^a)^{x_a{}^b}$

Producing Randomness (cont)

- Form column vectors V_b for $b = 1, \ldots, m$ with n entries; entry a is $([r_b^a], \alpha^{r_b^a})$
- Players compute a new column vector, $M \cdot Vb$
  - Let $\gamma_1, \ldots, \gamma_n$ be the i'th row of M. Then the i'th entry of $M \cdot Vb$ is $\left(\left[\sum_a \gamma_a r_b^a\right], \prod_a \alpha^{\gamma_a r_b^a}\right)$
  - For efficiency, we do this in a delegate-and-verify way
- Output $n - tp$ first entries of $M \cdot Vb$ for $b = 1, \ldots, m$

Delegate And Verify (AmortizedExp)

- Each player p_i computes a part of the result, $\beta_b^i = \prod_{a=1}^{n} \alpha^{\gamma_a r_b^a}$ for $b = 1, \dots, m$, where $(\gamma_1, \dots, \gamma_n)$ is the i'th row of M, then broadcasts β_b^i.
- The players generate a random value, x.
- Players compute $(\delta_0, \dots, \delta_n) = (x^0, \dots, x^{n-1}) \cdot M$
  - i.e. a linear combination of rows of M
- Players check that $\prod_{i=1}^{n} (\beta_b^i)^{x^{i-1}} \stackrel{?}{=} \prod_{a=1}^{n} \alpha^{r_b^a \gamma_a}$
- Disqualify any cheaters and output the β_b^is

epona@cs.au.dk

Application

Applying Ideas, In Particular These

- Read and understand the ideas
- Implement the ideas
- Run the implementation of the ideas

Specifically:

- Read “Secure Computation, I/O-Efficient Algorithms and Distributed Signatures”
- Extend VIFF, http://www.viff.dk
- Run your extended version of VIFF