A New (k, n)-Threshold Secret Sharing Scheme and Its Extension

11th Information Security Conference
September 15th–18th, 2008
Taipei, Taiwan

Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima, Toshiaki Tanaka.

KDDI R&D Laboratories, Inc., Japan.

KDDI R&D LABS

Introduction and Background

(k, n)-Threshold Secret Sharing Scheme (Threshold Scheme)

- The concept: Shamir and Blakley introduced in 1979.
- n participants hold shares generated from the secret s.
- **DEFINITION:** A (k, n)-threshold scheme has to satisfy the **PERFECTNESS**:
  - **Perfect Secrecy**:
    $\Rightarrow$ **Any information about s cannot be obtained from $k-1$ or less shares.**
  - **Accessibility**:
    $\Rightarrow$ **s can be completely recovered from k or more shares.**
- **DEFINITION:** A (k, n)-threshold scheme is an **IDEAL** secret sharing scheme if:
  - **The maximum bit-size of share always equals the bit-size of s.**

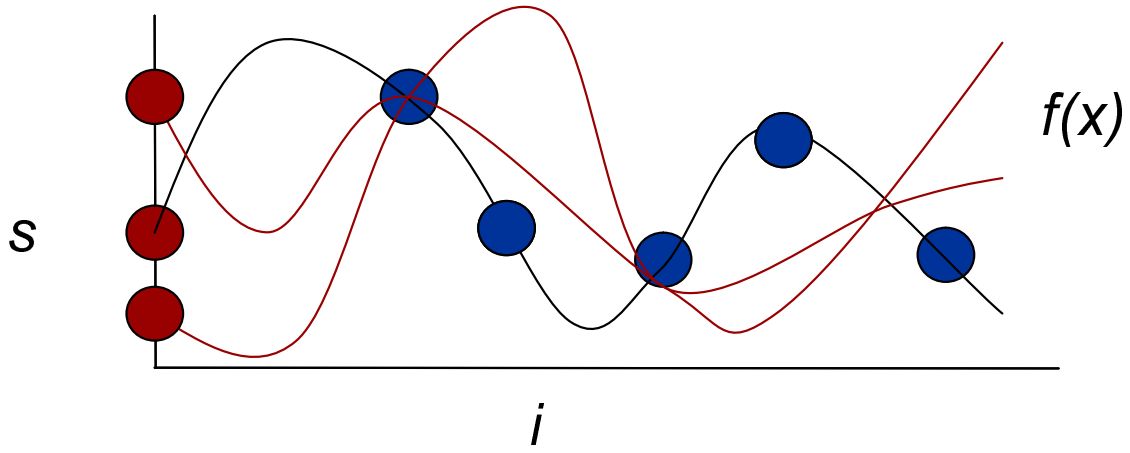
Shamir's (k, n)-Threshold Scheme

- A typical **ideal** secret sharing scheme.
- Input the secret $s \in GF(q)$. (q: prime number)
- Define a polynomial:

$$f(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

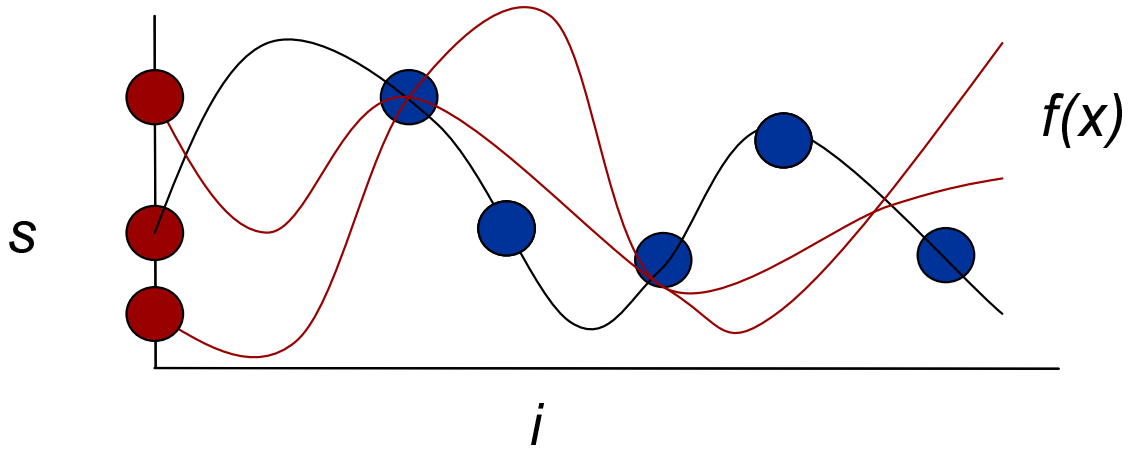
$$(a_1, \ldots, a_{k-1} \in GF(q) : \text{random elements})$$

- Share given to i-th participant P_i: $w_i = f(i)$



Shamir's (k, n)-Threshold Scheme (Cont.)

- From k **shares**, $f(x)$ can be uniquely determined by Lagrange interpolation.
  - $\Rightarrow$ **The secret $s = f(0)$ CAN be recovered.** (Accessibility)
- Given $k - 1$ **shares,** we cannot uniquely determine $f(x)$.
  - $\Rightarrow$ **Any information about $s = f(0)$ CANNOT be obtained.** (Perfect secrecy)



Problem of Shamir's (k, n)-Threshold Scheme

- Processing $k-1$ degree polynomial $f(x)$.
  - $\Rightarrow$ **Heavy computational cost for both distribution and recovery.**
    - $*$ distribution : making shares from s.
    - $*$ recovery : recovering s from shares.
  - $\Rightarrow$ **Shamir's scheme works very SLOWLY**.
- In this talk: I present

A New FAST (k, n)-Threshold Scheme, an alternative to Shamir's scheme.

A New (k, n)-Threshold Scheme

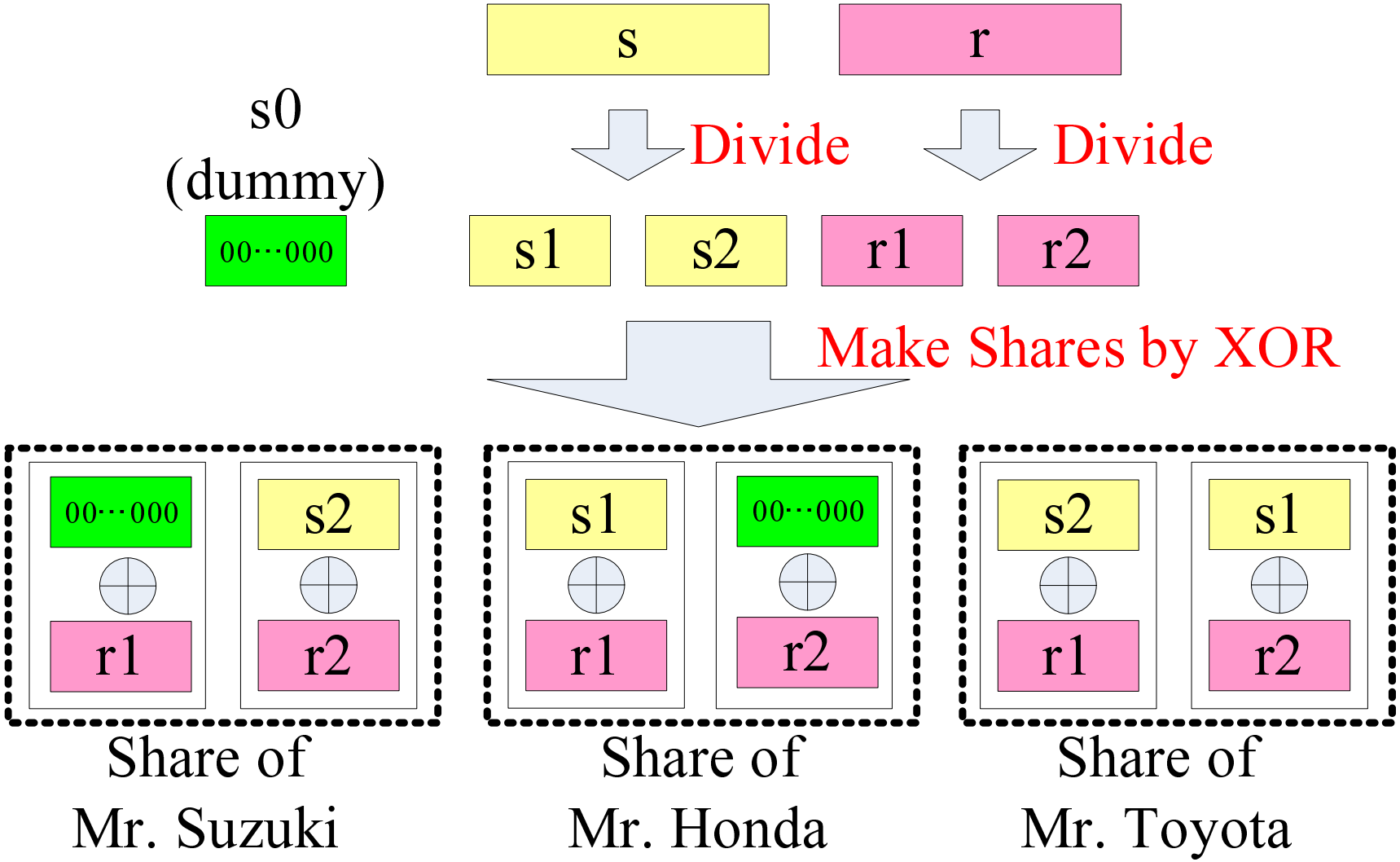
Related Work and Our Contributions

- A fast $(2, n)$-threshold scheme by using just XOR operations.[Fujii, 2005]
  - Satisfying the perfectness and ideality.
  - **Only for $k = 2$.**
- [Open Question] **Fast ideal scheme for arbitrary k and n ?**
  - $\Rightarrow$ **We realized the fast ideal (k, n)-threshold scheme!**

Our Scheme		Shamir's Scheme
FAST	Processing	**SLOW**
EXCLUSIVE-OR (XOR)	Operation	ARITHMETIC $(GF(q))$
O.K. (Proved!)	Perfectness	O.K.
O.K. (Proved!)	Ideality	O.K.
Arbitrary	(k, n)	Arbitrary

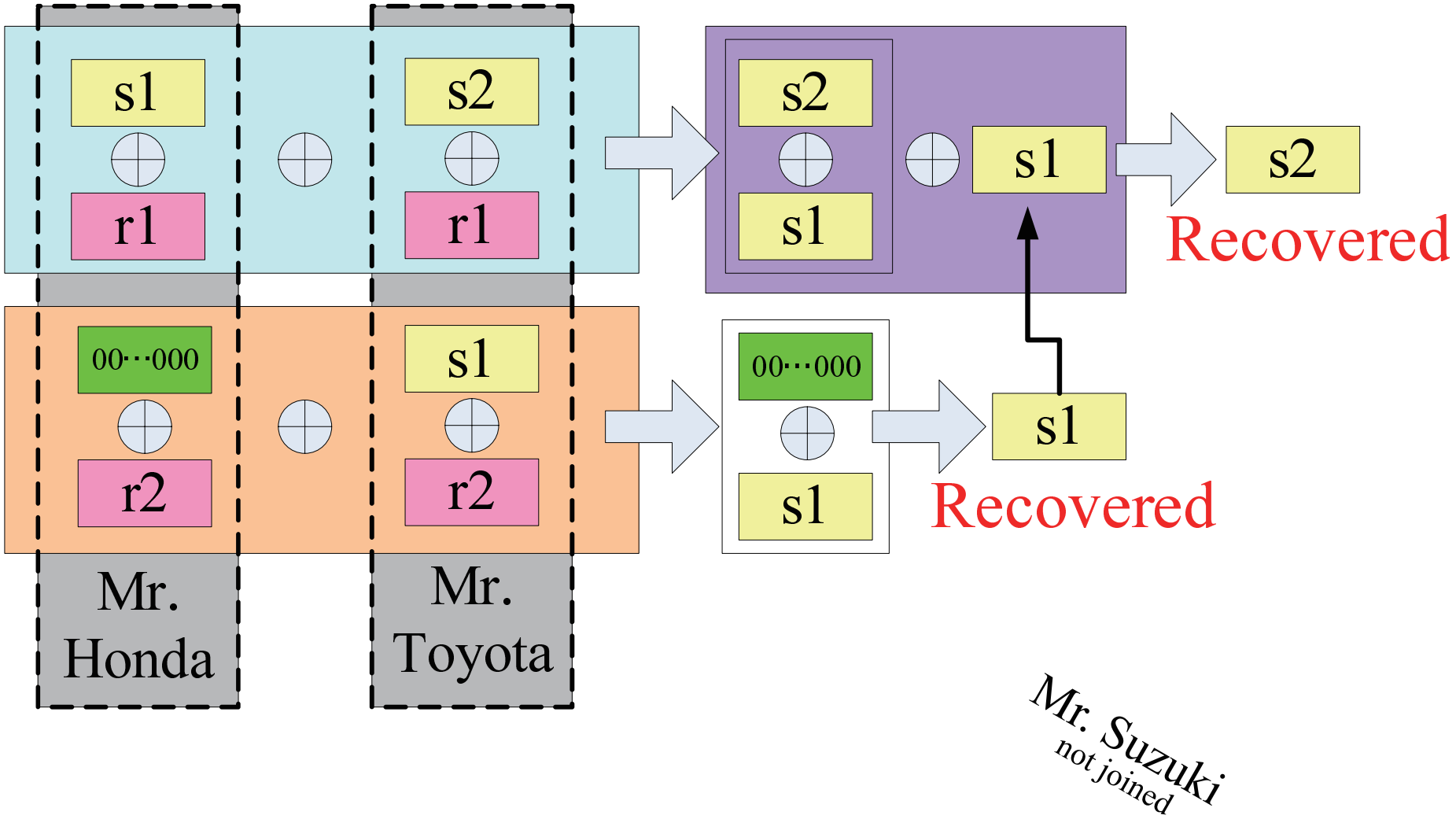
Rough Sketch of Our Scheme

- Distribution of a $(2, 3)$-threshold scheme:



Rough Sketch of Our Scheme (Cont.)

- Recovery from two shares of Mr. Honda and Mr. Toyota.



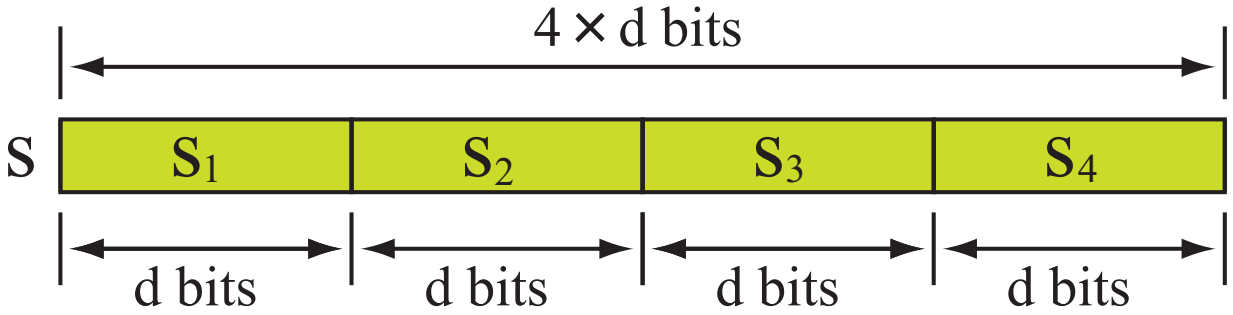
Preliminaries: Notations and Defininitions

- $\oplus$: A bit-wise XOR operation
- $\|$: Concatenation of bit sequences
- n_p : **A prime number that is $n_p \geq n$.**
- w_i: A share given to i-th participant P_i. $(i = 0, \ldots, n-1)$
- s : The secret ($s \in \{0,1\}^{d(n_p-1)}$, $d > 0$)
- **Values of indexes are elements of $GF(n_p)$ ($X_{c(a \pm b)} = X_{c(a \pm b) \bmod n_p}$)**
- **If n is COMPOSITE, a (k, n)-threshold scheme is realized by using $w_0, \ldots, w_{n-1}$ of a (k, n_p)-threshold scheme.**

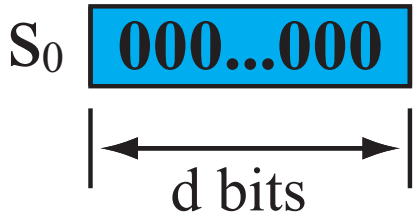
Distribution Algorithm

- In this talk: **Introduction using an example for $(k, n) = (3, 5)$ for $n = n_p$.**
- **Step 1)**: Divide s into **$(n_p - 1)$ pieces of d-bit segment equally. (4 pieces)**

$$s \longrightarrow s_1 \parallel s_2 \parallel s_3 \parallel s_4.$$

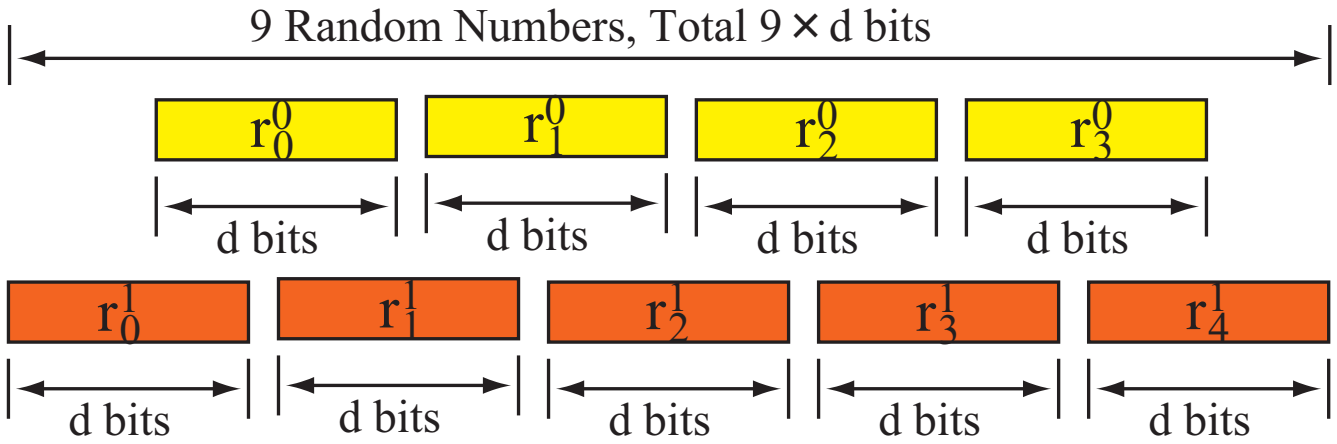


- **Step 2)**: Prepare s_0 as a **d-bit ZERO sequence.**



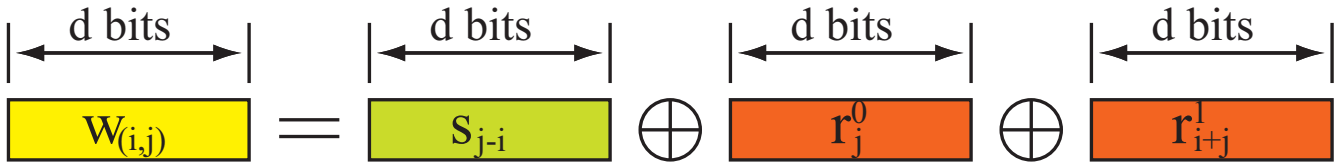
Distribution Algorithm (Cont.)

- **Step 3)**: Generate $(k-1)n_p - 1$ **of d-bit random numbers (9 pieces)** independently from each other with uniform probability.



- **Step 4)**: Execute XOR operations by the following equation:

$$w_{(i,j)} = \left(\bigoplus_{h=0}^{k-2} r_{h \cdot i+j}^{h} \right) \oplus s_{j-i} = \underline{s_{j-i} \oplus r_j^0 \oplus r_{i+j}^1.}$$



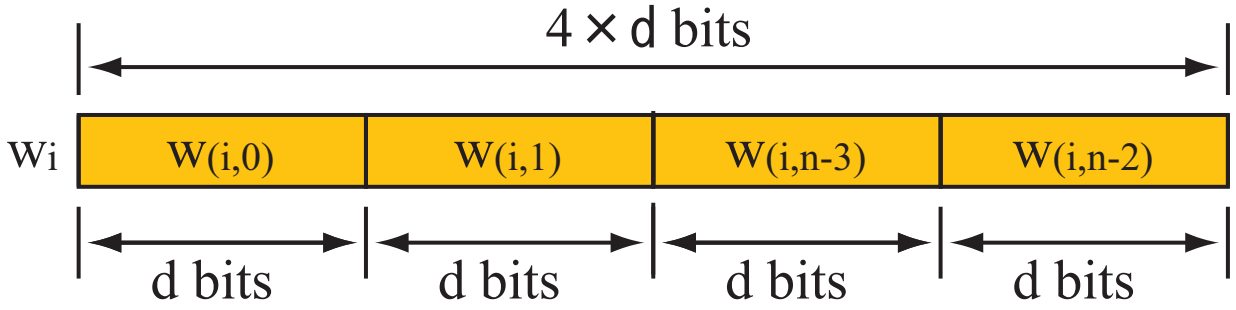
Distribution Algorithm (Cont.)

	$j=0$	$j=1$	$j=2$	$j=3$
$w_{(0,j)}$	$s_0 \oplus r_0^0 \oplus r_0^1$	$s_1 \oplus r_1^0 \oplus r_1^1$	$s_2 \oplus r_2^0 \oplus r_2^1$	$s_3 \oplus r_3^0 \oplus r_3^1$
$w_{(1,j)}$	$s_4 \oplus r_0^0 \oplus r_1^1$	$s_0 \oplus r_1^0 \oplus r_2^1$	$s_1 \oplus r_2^0 \oplus r_3^1$	$s_2 \oplus r_3^0 \oplus r_4^1$
$w_{(2,j)}$	$s_3 \oplus r_0^0 \oplus r_2^1$	$s_4 \oplus r_1^0 \oplus r_3^1$	$s_0 \oplus r_2^0 \oplus r_4^1$	$s_1 \oplus r_3^0 \oplus r_0^1$
$w_{(3,j)}$	$s_2 \oplus r_0^0 \oplus r_3^1$	$s_3 \oplus r_1^0 \oplus r_4^1$	$s_4 \oplus r_2^0 \oplus r_0^1$	$s_0 \oplus r_3^0 \oplus r_1^1$
$w_{(4,j)}$	$s_1 \oplus r_0^0 \oplus r_4^1$	$s_2 \oplus r_1^0 \oplus r_0^1$	$s_3 \oplus r_2^0 \oplus r_1^1$	$s_4 \oplus r_3^0 \oplus r_2^1$

Construction of Shares for $(k, n) = (3, 5)$

- **Step 5)**: Concatenate $n_p - 1$ pieces (4 pieces) and generate a share w_i given to P_i.

$$w_{(i,0)} \parallel w_{(i,1)} \parallel w_{(i,2)} \parallel w_{(i,3)} \rightarrow w_i.$$



Recovery Algorithm

- **Introduction using an example for $(k, n) = (3, 5)$ for $n = n_p$.**
- **Recovery from three shares: $w_{t_0}, w_{t_1}, w_{t_2}$. ($t_i \in GF(n_p), t_i \neq t_j$ if $i \neq j$.)**
- **Step 1)**: Obtain the **binary matrices $G_{t_0}, G_{t_1}, G_{t_2}$ such that $w_i = G_i \cdot r$.**

$$\left(\boldsymbol{w}_i = \left(w_{(i,0)}, w_{(i,1)}, w_{(i,2)}, w_{(i,3)}\right)^{\mathrm{T}}, \quad \boldsymbol{r} = \left(r_0^0, \ldots, r_3^0, r_0^1, \ldots, r_4^1, s_1, \ldots, s_4\right)^{\mathrm{T}}. \right)$$

$$\underline{\boldsymbol{G}_i = \left(\boldsymbol{I}_4, \boldsymbol{E}_i, \boldsymbol{L}_{-i}\right).}$$

$\boldsymbol{I}_\alpha$: $\alpha \times \alpha$ identity matrix.

$$\boldsymbol{E}_\beta = \left(\begin{array}{ccc|c|c} 0 & \cdots & 0 & 0 & \multirow{2}{*}{$\boldsymbol{I}_{n_p-\beta}$} \\ \vdots & \ddots & \vdots & \vdots & \\ 0 & \cdots & 0 & 0 & \\ \hline \multicolumn{3}{c|}{\multirow{3}{*}{$\boldsymbol{I}_{\beta-1}$}} & 0 & 0 \quad \cdots \quad 0 \\ & & & \vdots & \vdots \quad \ddots \quad \vdots \\ & & & 0 & 0 \quad \cdots \quad 0 \end{array}\right) = \left(\begin{array}{c|c} \vdots & \\ 1 & \boldsymbol{L}_\beta \\ \vdots & \end{array}\right).$$

$\Uparrow$ Omit!
Because it's s_0,
d-bit zero sequence.

Recovery Algorithm (Cont.)

- **Step 2)**: Execute the **Gaussian Elimination on** $GF(2)$ and obtain the matrix $\boldsymbol{M}$.

$$\boldsymbol{G} = \left(\begin{array}{c|c} \boldsymbol{G}_{t_0} & \\ \boldsymbol{G}_{t_1} & \boldsymbol{I}_{12} \\ \boldsymbol{G}_{t_2} & \end{array}\right) = \left(\begin{array}{ccc|ccc} \boldsymbol{I}_4 & \boldsymbol{E}_{t_0} & \boldsymbol{L}_{-t_0} & 1 & & \\ \boldsymbol{I}_4 & \boldsymbol{E}_{t_1} & \boldsymbol{L}_{-t_1} & & \ddots & \\ \boldsymbol{I}_4 & \boldsymbol{E}_{t_2} & \boldsymbol{L}_{-t_2} & & & 1 \end{array}\right)$$

$$\Downarrow \textbf{ Gaussian Elimination on } GF(2)$$

$$\boldsymbol{G}' = \left(\begin{array}{ccc|c} & * & & * \\ \hline \boldsymbol{\emptyset} & \boldsymbol{\emptyset} & \boldsymbol{I}_4 & \boldsymbol{M} \end{array}\right).$$

Solving the system of equation for s_1, s_2, s_3 **and** s_4**.**

- **Step 3)**: Execute the following operation and obtain the secret:

$$(s_1, s_2, s_3, s_4) = \boldsymbol{M} \cdot \boldsymbol{w}_i$$

Perfectness and Ideality

- The generator matrix $\boldsymbol{G}$ for L participants, $P_{t_0}, \ldots, P_{t_{L-1}}$:

$$\boldsymbol{G} = \left(\begin{array}{cccc|c} & & \boldsymbol{U} & & \boldsymbol{V} \\ \boldsymbol{I}_{n_p-1} & \boldsymbol{E}_{t_0} & \boldsymbol{E}_{2t_0} & \cdots & \boldsymbol{E}_{(k-2)t_0} & \boldsymbol{L}_{-t_0} \\ \boldsymbol{I}_{n_p-1} & \boldsymbol{E}_{t_1} & \boldsymbol{E}_{2t_1} & \cdots & \boldsymbol{E}_{(k-2)t_1} & \boldsymbol{L}_{-t_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{I}_{n_p-1} & \boldsymbol{E}_{t_{L-1}} & \boldsymbol{E}_{2t_{L-1}} & \cdots & \boldsymbol{E}_{(k-2)t_{L-1}} & \boldsymbol{L}_{-t_{L-1}} \end{array} \right)$$

$\boldsymbol{G}$: $L(n_p - 1) \times kn_p - 2$ matrix.

$\boldsymbol{U}$: $L(n_p - 1) \times (k-1)n_p - 1$ matrix represents the combination of random numbers.

$\boldsymbol{V}$: $L(n_p - 1) \times (n_p - 1)$ matrix represents the combination of pieces of the secret.

Perfectness and Ideality (Cont.)

- If $L \leq k-1$:

$$rank\,(\boldsymbol{U}) = L(n_p - 1)$$

  - $\Rightarrow$ **All rows of U, and hence those of G, are linearly independent.**
  - $\Rightarrow$ **Perfect secrecy**.

- If $L \geq k$:

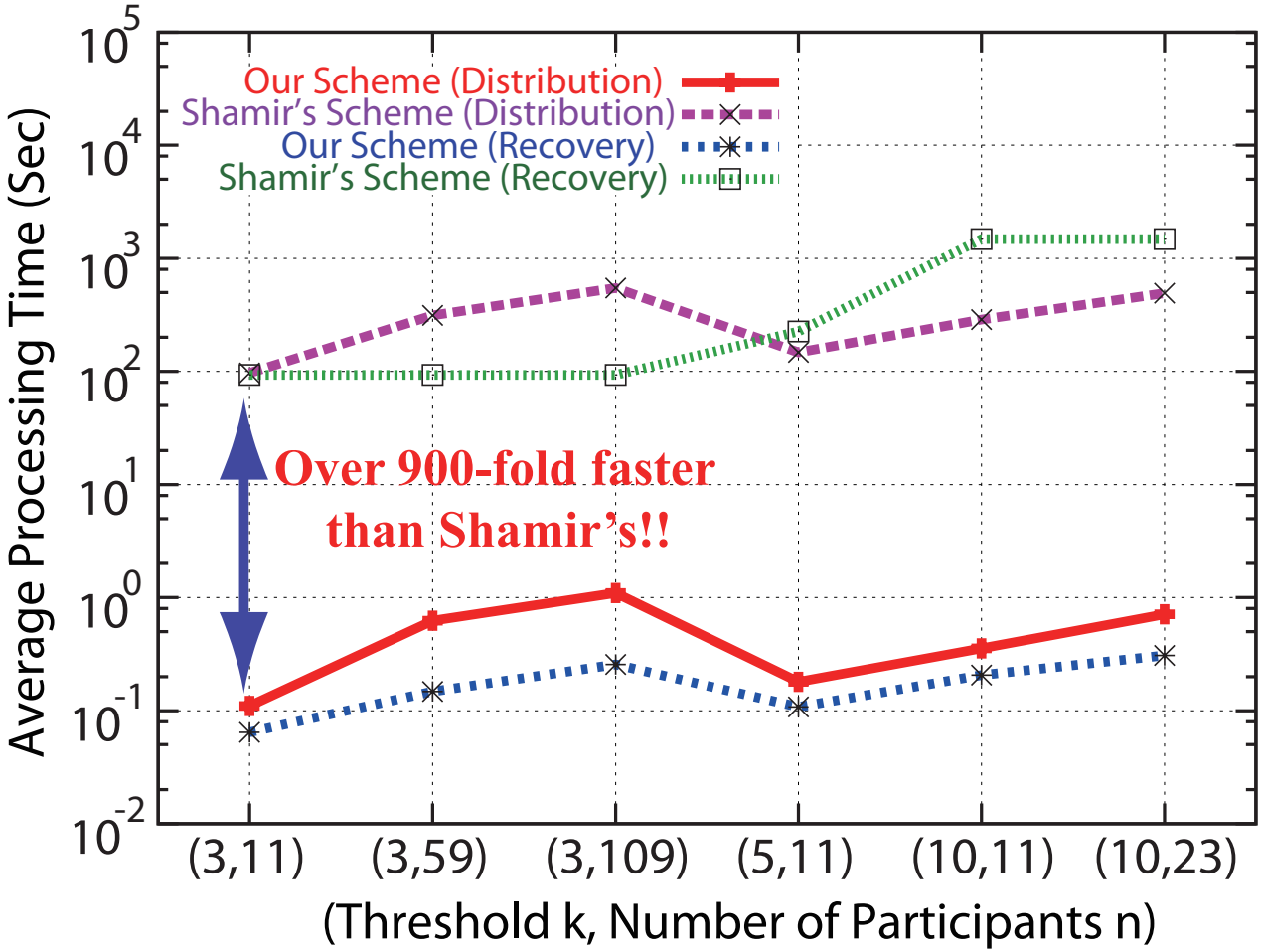
$$rank\,(\boldsymbol{U}) = (k-1)(n_p - 1) \quad \text{and} \quad rank\,(\boldsymbol{G}) = k(n_p - 1).$$

  - $\Rightarrow$ **Rows of U are linearly dependent and $rank(\boldsymbol{G}) - rank(\boldsymbol{U}) = n_p - 1$.**
  - $\Rightarrow$ **All pieces of the secret can be recovered.**
  - $\Rightarrow$ **Accessibility**.

- **The length of each share equals the length of the secret.**
  - $\Rightarrow$ **An IDEAL secret sharing scheme.**

Efficiency

- Computer Simulation: Comparing the processing time with Shamir's scheme.



- 4.5MB Secret
- Debian GNU/Linux 4.0
- GCC 4.1
- Pentium4 3GHz
- 2GB RAM

- Under the simulation, **our scheme realizes much faster oparations than Shamir's.**

Efficiency (Cont.)

- The number of operations:

Our Scheme	Distribution	: $O(kn) \cdot \log_2 \lvert s \rvert$ **times XOR (average)**
	Recovery	: $O(kn_p) \cdot \log_2 \lvert s \rvert$ **times XOR (upper bound)**
	Gaussian Elim.	: $O(k^3 n_p^3)$ **times XOR (upper bound)**
Shamir's Scheme	Distribution	: $O(kn)$ **times ARITHMETIC** ($GF(q)$)
	Recovery	: $O(k \log^2 k)$ **times ARITHMETHIC** ($GF(q)$)

- **Distribution**: Cost is lineary increasing with k and n for both schemes.
- **Recovery**:
  - **Shamir's scheme: Cost is constant and independent of n for fixed k.**
  - **Our scheme depends on n_p for fixed k.**
- Our scheme may NOT perform faster **RECOVERY** processing than Shamir's **if n_p is extremely large**.

Conclusions

Conclusions

- **A new (k, n)-threshold scheme using just XOR operations.**
- **An extension to a fast (k, L, n)-threshold ramp scheme.**
  - Sorry, please refer to my paper.
- The detailed version will appear in the **Cryptology ePrint Archive**, soon!

Thank you!

QUESTIONS or COMMENTS?

⟨kurihara@kddilabs.jp⟩