分散個体群認証のための秘密分散法について

五太子政史[1] 山澤昌夫[1,2] 山本 博資[1] 藤田 亮[1] 松本義和[2] 白水公康[2] 豊島大朗[2] 瀬瀬考平[2]
近藤 健[3] 辻井 重男[1,2]

概要：IoT などの通信機器において，ボード上に不正なスパイチップが組み込まれる脅威が問題視されている．この対策として，ボードに秘密情報を割当て，それを秘密分散した各データを正規のチップに組み込むという認証方式が提案された．(k, n) 秘密分散法を使うと，チップ n 個中で不正チップが $(n-k-1)$ 個までならば，どれが不正であるかも特定できる．これを実現するための秘密分散方式，秘密情報復元から各チップの認証，及び異常が発見された場合の不正チップの特定までを可能な限り確実に，かつ少ない計算量で実現するための手順を考察した．

Secret Sharing for Authenticating Distributed Items

Masahito Gotaishi[1] Masao Yamasawa[1,2] Hirosuke Yamamoto[1] Ryo Fujita[1] Yoshikazu Matsumoto[2]
Kimiyasu Shirouzu[2] Dairo Toyoshima[2] Kohei Sese[2] Takeshi Kondo[3] Shigeo Tsujii[1,2]

1. 序論

1.1 IoT において要求されるハードウェアのセキュリティ

家庭や製造・事業現場などにおいて IoT が浸透する一方で，それらの攻撃の脅威も大きくなってきた．2010 年代後半から，IoT へのネットワーク攻撃が現実に増加しており，情報セキュリティについては，家庭などへの侵入に繋がりかねない問題，及び IoT 機器が DDOS などのネットワーク犯罪の「踏み台」に使われる可能性が指摘され，わが国では 2020 年オリンピック・パラリンピックに向けて NICT が業務としてインターネットに接続された機器のセキュリティ検査を行うという時限措置が 2018 年 11 月から行われることになった．5G などの通信では，センサや Webcam などの機器において送受信される機微な情報及び商品として価値を持つ情報を守るため，認証，及び暗号化通信などによる侵入・盗聴対策が要求されている [7].

しかしながら，特殊なチップの埋め込まれたマザーボードを使用したサーバーが大手企業に納入された可能性のあることが，2018 年に Bloomberg で報道された [2]．有名な所ではファーウェイのスマートフォンにバックドアが仕込まれていたとして，トランプ大統領が「安全保障上の脅威となる外国企業の通信機器の調達を禁止する」という，多分に同社を意識した大統領令をに署名している [1]．このような「ハードウェア改ざん」を認める企業は現れておらず，情報も十分でないため，このようなハッキング事件が実際に起こったか否かは不確定であるが，技術的にはこのような犯罪はかなり低コストで実行できるとされている．製造後・出荷後にスパイチップとすり替えられる脅威も存在する．

このように，5G のような高度通信の行われる社会では，コンピュータや通信機器の製造業者は部品供給業者をも十分に信頼することはできず，納入されたチップが不正なものでないかをハードウェア立ち上げ時に認証することが求められている．

1.2 不正チップ発見のためのハードウェア認証方式の提案

このための認証に (n, k) しきい値秘密分散を用いることを山澤らは提案した [6].

即ち，ボード上のチップが n 個あるとして，それらに 1 つの秘密情報 S を分散して持たせておく $(x_1, x_2, \ldots, x_n)$ というものである．ボードを製造するときに n 個中 k 個のチップ $C_{\sigma(1)}, \ldots, C_{\sigma(k)}$ $(1 \le \sigma(1) < \sigma(2) < \ldots < \sigma(k) \le n)$ か

[1] 中央大学研究開発機構
Chuo University, Research and Development Initiative
[2] セキュア IoT プラットフォーム協議会
Secure IoT Platform Consortium
[3] 中央コリドー ICT 推進協議会 CCC21ICTCouncil

ら秘密情報を復元し，それらが n 個中 k 個を取る組合せに依存せず一定値であれば全て正当で，異なっていればいずれかのチップが不正であると結論できる．不正なチップの個数が $(n-k-1)$ 以下であれば，例えば $n=10,\ k=8$ であれば，うち 1 個が不正なとき多数決によって S が判明し，どれが不正であるかも判別できる．

2. 秘密分散によるハードウェア認証方式の構想

山澤他 [6] の提案した構想は，あるボードに特定の秘密の数値 S を対応させ，その上に装着される n 個のチップのそれぞれに，分散された値 $R_1,\ldots,R_n$ を割当てる．チップメーカーはそれらの値を対応するチップに記録して出荷する．

通信システムを起動するとき，n 個のチップに記録された $R_1,\ldots,R_n$ 個中 k 個の組合せについてデータの復元を試み，全ての復元値が一致するかを確認する．一致しなければ，続いてどのチップが不正であるかを判定する．

但し秘密情報 S はボード上には記録しない．この値を悪用されると復元用のプログラムに細工されて強制的に辻褄が合うようにされる可能性が有るためである．従って，n 個の中に不正チップがあるか否かは，n 個中 k 個を取る全ての組合せについて分散秘密の復元を行い，それらが全て同じか，異なっていたらその復元値の分布がどうなるかによって不正チップの存在の状態を判定することになる．

2.1 分散と認証の具体的手順

秘密情報の復元及びチップの認証は以下のように行う:

(1) n 個から k 個のチップ $C_{\sigma(1)},\ldots,C_{\sigma(k)}$ を取る組合せ全てについて，秘密情報の復元を行う．

(2) $\left(\begin{array}{c} n \\ k \end{array}\right)$ 通りの組合せ全てについて復元値 S_σ が一致すれば異常なしとして，通信を開始する．

(3) 一致しない場合，異常ありとしてシステムの起動を中止し，エラー診断に入る．

(4) 各組合せ σ に対応する S_σ について，最も度数の多いものを真の S とする．

(5) S と一致しない組合せの数が $\left(\begin{array}{c} n \\ k \end{array}\right) - \left(\begin{array}{c} n-h \\ k \end{array}\right)$ $(1 \le h \le k-1)$ であれば，不正なチップは h であると結論し，不正なものを含む k 個のチップの各集合の共通集合が不正チップの集合と判定する．

(6) 全ての組合せで復元値が異なる場合は，不正なチップが $(n-k)$ 個以上あると結論し，ボードを検査に回す．

ボード上に不正なチップが紛れ込むことがあり得るとしても，それほど多くは無い (1 個ないし 2 個) と考えられ，$(n-k)$ は 2〜3 程度となる．認証の際の分散秘密復元の回数は k の指数関数となるため，この程度の数が現実的であろう．

3. 秘密分散法の選定

ハードウェアの認証に秘密分散を用いる場合，計算負荷 (time 及び space complexity) が問題である．秘密分散法として標準的な Shamir の方法では，有限体上の除算が必要であり，これには分散秘密復元を行う時に相当の計算量がかかるため，n 個中 k 個の組合せを全て試みるには適していない．計算済みのパラメータを記憶しておくにしても，全ての組合せに対応したパラメータを持つにはデータ量がかかることが考えられる．また，先に述べた通り，ボード上に秘密の値 S を保持はしないという構想なので，不正チップをできる限り確実に特定できることが望ましい．

秘密分散法には，Shamir の方法 [5] と中国人剰余定理を用いた方法 [8][3]，及び，XOR 演算による秘密分散法 [9][4] などが代表的であるので，これらの中から秘密分散法を選定することになるであろう．今回，著者らは上記のうち，Shamir の方法及び Asmuth と Bloom の方法を検討した．

3.1 Shamir の秘密分散法

最初に提案された秘密分散法であり，「xy 平面上の k 個の点を通る $(k-1)$ 次 (またはそれ以下) 式は 1 つに定まる」という定理を利用したものである．秘密の値 S を，n 個の 2 次元座標 $(x_1,\ y_1),\ (x_2,\ y_2),\ldots,(x_k,\ y_k),\ldots,(x_n,\ y_n)$ の n 個に分割し，うち k 個が揃えばデータを復元できるようにする際の方法は以下である：

3.1.1 分割

(1) 乱数で係数 $c_1,\ c_2,\ldots,c_{k-1}$ を生成し，$(k-1)$ 次式 $f(x) := S + c_1x + c_2x^2 + \ldots + c_{k-1}x^{k-1}$ を作る．

(2) やはり乱数で $x_1,x_2,\ldots,x_n$ を生成する．

(3) $x_1,\ldots,x_n$ を $f(x)$ に代入して $y_0,y_1,\ldots,y_n$ を得る．なお $x_1,\ldots,x_n$ の値は，例えば $1,2,\ldots,n$ など固定でも良い．

復元は Lagrange 補間によって行われる．

3.1.2 復元

Lagrange 補間の式は以下である：

$$y = \sum_{h=1}^{k} \left(\frac{y_h * \prod_{i=1,i\neq h}^{k} (x_{s(i)} - x)}{\prod_{j=1,j\neq h}^{k} (x_{s(j)} - x_{s(h)})} \right) \tag{1}$$

従って，(1) 式で x に 0 を代入した値が S である．

$$S = \sum_{k=1}^{k} \left(\frac{y_h * \prod_{i=1, i\neq h}^{k} x_{s(i)}}{\prod_{j=1, j\neq h}^{k} (x_{s(j)} - x_{s(h)})} \right) \tag{2}$$

上記の (1) 及び (2) 式で，$s(i)$ は n 個から k 個を取り出す組合せで i 番目に大きいものを指す．ここで，秘密分散の際に分散される値は実数などでなく有限体の要素である (そうでないと各分散値の情報量がゼロにならない)．その場合，除算を $(k-1)*k$ 回行っており，この除算は，有限体上ではユークリッドの互除法で逆数を求めて，その逆数を掛けるという手順を取らなければならない．そうでなければ上記の $x_{s(j)} - x_{s(h)}$ の組合せ全てについて逆数を用意しておく必要があり，n^2 の量のデータを記録する必要があると考えられた．しかし Shamir の秘密分散では x_i の値を任意に定めることが可能で，これを $x_i = i$ とすれば，記憶すべき値の数は n で済む．それだけではなく，Shamir の方法では n 個中 k 個から秘密情報を復元する組合せを比較しなくとも n の中に不正チップが含まれているかを知ることが可能であり，実は非常に効率的な方法であることが解った．

3.1.3 復元の計算量

有限体の法を m ビット，分散された各数値も m ビットとする．このときユークリッド互除法の計算量は $\log m$ で乗算の計算量は m^2，加算の計算量が m で有限体上の除算の計算量は $m^2 \log m$，剰余計算 (除算) の計算量が m である．有限体上の演算なので，演算を行うたび剰余を計算するとする．

このとき (2) 式の計算量は，数値を記憶しておかないならば乗算の回数が k^2 回，除算の回数が k^2 で，加算が k 回となるので，計算量は $\mathcal{O}(k^2 * m^2 \log m)$ で，数値を記憶して除算を行わないならば $\mathcal{O}(k^2 * m^2)$ である．

3.2 中国人剰余定理による秘密分散

2012 年に辻ら [8] は，レントゲンなどの医療用画像を，迅速に分散と復元ができる秘密分散方式として中国人剰余定理を使使うことを提案した．

3.3 画像データの中国人剰余定理による分散

保護すべき整数値を S $(0 \leq S < M)$，互いに素な n 個の整数を $Q_1, \ldots, Q_n$ $(GCD(Q_i, Q_j) = 1 \text{ for } i \neq j,\ 1 \leq i, j \leq n)$ とし、ここで n 個中任意の k 個の組合せ $\{Q_{i(1)}, \ldots, Q_{i(k)}\}$ について $\Pi_{h=1}^{k} Q_{i(h)} \geq M$ とする。このとき、S を以下の方法で $R_1, \ldots, R_n$ の n 個に分散する：

$$\begin{aligned} R_1 &:= S \mod Q_1 \\ R_2 &:= S \mod Q_2 \\ &\cdots \\ R_n &:= S \mod Q_n \end{aligned} \tag{3}$$

このとき、n 個から k 個を取る任意の組合せを $s(n,k)$, $s(n,k)$ で i 番目の要素を $s(n,k,i)$ とすると $(s(n,k,i) < s(n,k,j)$ for $i < j)$，S は下の式で求められる．

$$S = \sum_{h=1}^{k} \left(R_h * P_{s(n,k),h} * \prod_{i=1, i\neq h}^{k} Q_{s(n,k,i)} \right) \mod \prod_{i=1}^{k} Q_{s(n,k,i)} \tag{4}$$

式 (4) で，$P_{s(n,k),h}$ は，

$$P_{s(n,k),h} * \left(\prod_{i=1, i\neq h}^{k} Q_{s(n,k,i)} \right) \mod Q_h = 1 \tag{5}$$

となるような整数である．

この方法の問題点は，Shamir の秘密分散とは異なり，それぞれに分散された値が S に関する情報を持っていることにある．秘密の値が 30 で，それを $7, 11, 13$ による剰余の形で分割すると 7 で割った剰余は 2 となり，この情報から，値が 0〜$1000 (= 7 \times 11 \times 13 - 1)$ のうち $2, 9, 16, \ldots, 996$ のどれかである，と 143 個に候補が絞れることになる．即ち，分割されたそれぞれの値の持つ情報量が 0 ではないため，秘密分散法としては問題である．

3.4 Asmuth & Bloom の提案した方法

Asmuth と Bloom[3] は，剰余として分割されたそれぞれの値に元データに関する情報が含まれないようにする方法を提案していた．

筆者らが検討の対象にした手法はこの秘密分散法である．

3.4.1 分割

上記の手法の問題点は，秘密の m ビット値 S を k 個に分散するときに分散値のデータ長が m/k ビット程度に減ることにある．分散するときに剰余を取る法の各値を S のレンジよりも大きくすればよい．しかしそれでは剰余を取るときに値が変わらないので，工夫を加える．

(1) S の取り得る範囲を $0 \leq S \leq M-1$ として，分割するための剰余を取る n 個の互いに素な各整数を $Q_1 < Q_2 \ldots < Q_n$ とする．このとき，$M < Q_1$ とする．閾値を k とする．

(2) このとき S に対して乱数 $1 \leq r < M^{k-1}$ を発生させて $S' := S + r * M$ を得る．$0 < S' \leq \prod_{i=1}^{k} Q_i$ である

(3) S' を各 Q_i で割った剰余を R_i として分散保存する．

3.4.2 復元

復元は以下の手順で行う：

(1) k 個の値 $R_{s(1)}, R_{s(2)}, \ldots, R_{s(k)}$ を得れば S' が一意に求まることは中国人の剰余定理で保証されている．

(2) 具体的には，以下の計算を行う：

$$S' := \left(\sum_{i=1}^{k} R_{s(i)} * P_{s,i} * \prod_{j=1,j\neq i}^{k} Q_{s(j)} \right) \mod \prod_{i=1}^{k} Q_{s(i)} \tag{6}$$

$P_{s,i}$ は，

$$P_{s,i} * \left(\prod_{j=1,j\neq i}^{k} Q_{s(j)} \right) \mod Q_{s(i)} = 1 \tag{7}$$

となるような整数である．

(3) S' を M で割った剰余が S である．

3.4.3 復元の計算量

中国人剰余定理に基づいて秘密の値を復元することは演算操作的には，Lagrange 補間で平面上の点からそれらを通る代数曲線を求める手法に似たところがある．違いは，中国人剰余定理では一部のパラメータは復元に当っての分散データの選び方に依存しないようにできる，つまり復元時でなく事前に計算しておける部分が大きいことである．これを使って計算を高速化する．

k 個選んだ分散データ $R_{s(1)}, \ldots, R_{s(k)}$ について，以下のような方法で復元することができる：

(1) 以下を計算する：

$$\alpha := \sum_{i=1}^{k} R_{s(i)} * P_{s(i)} * \prod_{j=1,j\neq s(i)}^{n} Q_j \tag{8}$$

例によって，P_h は $P_h * \left(\prod_{i=1,i\neq h}^{n} Q_i \right) \mod Q_h = 1$ となるような整数である．

(2) S' を下式で求める：

$$S' = \alpha \mod \prod_{i=1}^{k} Q_{s(i)} \tag{9}$$

(3) $S' \mod M$ で S を求める．

上記の各 $P_h, \prod_{j=1,j\neq h}^{n} Q_j$ はデータの選び方に依存しないことがわかるであろう．分散データ R_h が固定で良いならば，その積 $U_h := R_h * P_h * \prod_{j=1,j\neq h}^{n} Q_j$ を分散値として保存していれば良い．すなわち，この場合は分散保存された k 個の数値を加算して，剰余計算すればよいことになる．1 回の復元に要する計算量は各分散値に対応する法 Q_i と最終的に秘密の値を求める際に使う法」M のビット長を m とすると，分散された秘密情報の復元を行う時の k 個の Q_i の最小公倍数のビット長は km である．秘密情報の復元では，k 回の加算と剰余計算を行うことになるので計算量は $\mathcal{O}(km)$，チップの認証のために n 個から k 個を取る組合せ全てについて復元を行うならば計算量は $\mathcal{O}(km * n^k)$ となる．

4. 不正チップの発見

4.1 Shamir の秘密分散を使う場合

Shamir の方法のメリットは，S を n 個に分散したうち，例えば $(k+1)$ 個中 1 個以上不正なデータがある場合にそれを判別できることである．n 個の分散データがあったとしても，それら全てから Lagrange 補間を行って得られる多項式は，$(k-1)$ 次式となる．即ち，k 次以上の項の係数は全て 0 になる．

n 個中 $C_1, \ldots, C_{k+1}$ の $(k+1)$ 個からデータの復元を行ったとき，その k 次項の係数が 0 であれば，$C_1, \ldots, C_{k+1}$ までの全てが認証されることになる．k 次，すなわち最高次の項の係数は以下の式で求められる：

$$\sum_{h=1}^{k+1} \left(\frac{y_h}{\prod_{j=1,j\neq h}^{k+1} (x_h - x_j)} \right) \tag{10}$$

ここで，$x_1 = 1, x_2 = 2, \ldots, x_k = k, \ldots, x_n = n$ と決めると，

$$\prod_{j=1,j\neq h}^{k+1} (x_h - x_j) = (-1)^{h-1}(h-1)!(k-h+1)! \tag{11}$$

となるので，k 次項の係数は，

$$\sum_{h=1}^{k+1} \frac{(-1)^{h-1} y_h}{(h-1)!(k-h+1)!} \tag{12}$$

となる．

この計算量は，逆数を記憶させる場合は $\mathcal{O}(km)$ である．

4.1.1 認証の手順

n 個あるチップにどれも異常が無く，そのことを確認する手順は以下のようになる：

(1) 1〜$(k+1)$ 番目のチップについて Lagrange 補間を行い，k 次 (最高次) の係数が 0 であることを確認する．

(2) 1 番目の分散情報を除外し，2 番目〜$(k+2)$ の $(k+1)$ 個について同様に Lagrange 補間で多項式を求め，k 次の係数が 0 であることを確認する．

(3) これを $(n-k-1)$〜n 番目まで繰り返す．

この場合の計算量は $\mathcal{O}(m*k*(n-k))$ である．異常が見つかる場合は以下の手順になる．1〜k 番目までが正常なチップである場合，以降はノーマルシーケンスに従って $(k+2)$〜n 番目までのどれが異常であるかは特定できる．問題は k 番目までに不正が見つかる場合である．

(1) 1〜$k+1+i$ 番目まで復元に使うチップを増やすことを，最高次の項の係数が 0 になるまで繰り返す．

(2) n 番目まで試しても最高次の係数が 0 にならない場合 (不正チップが $(n-k)$ 個以上ある)，秘密分散による

認証は停止し，台帳の参照などオフラインで不正チップを探す．

(3) $k+s+1$ ($s \leq n-k-2$) 番目で最高次係数が 0 になった場合，1〜$(k+s+1)$ 番目までに不正チップは s 個あることが解るので．$(k+s+1)$ 個中 $(k+1)$ 個を選ぶ組合せを全て検証することにより，不正チップが特定される．

(4) $(k+s+2)$ 番目以降のチップについては，ノーマルシーケンスと同じ手順で正当性が検証できる．

この場合の計算量は $\mathcal{O}(m*k*s*n^s)$ である．復元された数値について一致をチェックしたり多数決で正当な数値を判定したりすることなく，不正の存在が結論できるという点が好ましいと著者らは考える．

4.2 Asmuth & Bloom の秘密分散を使う場合

加算と剰余計算だけで分散された数値から秘密を復元できるのがこの手法のメリットであるが，正しい値と比較しない限り，復元を行った k 個のチップの中に不正なものが無いかは判定できない．この点，著者らは不安に思う．異常が無い場合は以下のようにして確認できる：

(1) 1〜$(n-k)$ 番目を除く，$(n-k+1)$〜n 番目のチップについて秘密情報を復元する．

(2) 同様にして，$(n-k+1)$〜$(2n-2k)$ 番目を除いたチップについて秘密情報を復元する．以降，$\lceil \frac{n}{n-k} \rceil$ 回これを繰り返す．

(3) 復元された値が全て一致したら正当性が検証できる．

上記の手順で，一致しないものがあった場合，それらの組合せの中には全て 1 個以上の不正チップが含まれると考えられるので，集合の共通部分を比較することにより不正チップは特定される．どの組み合わせも一致しない場合，$(n-k)$ 個以上不正チップが存在すると結論し，秘密分散による認証は停止する．

異常が無い場合，それを認証するまでの計算量は $\mathcal{O}(km*\lceil \frac{n}{n-k} \rceil)$ である．異常がある場合も秘密復元の段階では同じ手順である．

5. 考察

上記目的のため，復元の計算量が少なくて済む秘密分散を検討していたが，意外に Shamir 式の秘密分散は確実かつ高速で実行できる．Asmuth & Bloom の手法も高速化には役立つが，意外に秘密分散をするための法 $Q_1, \ldots, Q_n$ の選定に苦労するので，実装には Shamir 式の秘密分散の方が良いのではないかと考えられる．

6. 今後の課題

秘密分散法には，上に述べたような藤井達 [9] の手法もあり，こちらも検討すべきかと思われる．但しこの手法では XOR 演算自体は高速だが復元の際に分散データ集合に対応した行列を生成する過程があり，実際には負荷が大きくなるのではとも思われた．今後こちらについても検討を行い，ボード上のチップの正当性を検証できるようにするための手法，及び分散秘密を守ったままボード上に正しいチップ集合を組み込むための生産管理手段を検討することが必要である．

参考文献

[1] オランダでファーウェイ製品の「バックドア」発見、政府が調査, https://forbesjapan.com/articles/detail/27317.

[2] 中国、マイクロチップ使ってアマゾンやアップルにハッキング，https://www.bloomberg.co.jp/news/articles/2018-10-04/PG2CZY6TTDS801.

[3] Asmuth, C. and Bloom, J.: A modular approach to key safeguarding, *IEEE Transactions on Information Theory*, Vol. 29, No. 2, pp. 208–210 (1983).

[4] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: A New (k, n)-Threshold Secret Sharing Scheme and Its Extension, *International Conference on Information Security*, Springer, pp. 455–470 (2008).

[5] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).

[6] 山澤昌夫，五太子政史，山本博資，松本義和，白水公康，豊島大朗，瀬瀬考平，　近藤健，辻井重男：分散個体群を認証するための秘密分散法要件の一検討 (発表予定)，*Multimedia, Distributed, Cooperative and Mobile (DICOMO), Symposium Collected Papers*, Vol. 2023 (2020).

[7] 松本義和，辻井重男，白水公康，瀬瀬考平：重要 IoT デバイスへの PKI 電子認証の実装 −セキュア IoT 基盤が形成するトラストチェーン−，コンピュータセキュリティシンポジウム 2018 論文集， Vol. 2018, No. 2, pp. 838–841.

[8] 辻敏雄, 笠原正雄：中国人剰余定理による秘密分散法とその応用，電子情報通信学会技術研究報告.，Vol. 112, No. 306, pp. 61–68 (2012).

[9] 藤井吉弘，栃窪孝也，保坂範和，多田美奈子，加藤岳久：排他的論理和を用いた (k,n) しきい値法の構成法，電子情報通信学会技術研究報告. ISEC, 情報セキュリティ， Vol. 107, No. 44, pp. 31–38（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/110006292266/⟩ (2007).