Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

ワイヤレスセンサネットワークにおけるMicaZを用いたデータアグリゲーション実装の検討

伊澤 和也[†1] 宮地 充子[†1] 面 和成[†1]

センサネットワーク (WSNs) のセキュリティ（完全性，機密性）に関する研究が数多くなされている．WSNs のセキュリティを考慮することが重要であることは言うまでもないが，一方で WSNs にセキュリティを付加するには追加の演算がどうしても必要になる．このとき，演算時間やメモリ量，通信遅延がどの程度必要になるのかは実装してみないと分からない部分も多い．また，近年センサなどの小型機器で動作する OS(TinyOS) で利用できる暗号ライブラリ (RELIC-toolkit) が開発された．

本稿では，上記 RELIC-toolkit を用いて宮地・面のアグリゲーション方式を MICAz 上で実装・評価することを目的とし，まずはセンサ上の演算の実装評価を行う．具体的には，2 種類のデータアグリゲーション方式を取り上げ，完全性を満たす宮地・面方式と機密性を満たす PEQ 方式を比較・考察することで，完全性・機密性をそれぞれ実現する際のオーバーヘッドを評価する．

Consideration of Implementation of Data Aggregation on MicaZ for Wireless Sensor Networks

Kazuya Izawa,[†1] Atsuko Miyaji[†1]
and Kazumasa Omote [†1]

Data security(data integrity and confidentiality) of WSNs has been a very popular research topin in recent years. For preserving data security, WSNs require extra operations that lead to larger resource consumption. TinyOS is an operating system that is suitable for using WSNs. Recently, RELIC-toolkit is developed, it is a very useful cryptographic library used inside the TinyOS.

The goal of our work is to implement secure data aggregation of Miyaji and Omote on MICAz sensors by using RELIC-toolkit. In this paper, we implement and evaluate the operation of two data aggregation schemes, data aggregation of Miyaji and Omote and Vu's scheme, on MICAz. And we assess the two schemes by estimating memory consumption and operation time.

1. はじめに

本論文ではワイヤレスセンサネットワーク (WSNs) について取り上げている．WSNs とは山や平野などの空間に小さな (性能の限られた) 多数のセンサノードを配布し各センサノードが観測したデータをネットワークを用いて基地局に送付する方式の内通信が無線 (ワイヤレス) なものを特に WSNs と呼称する．特徴として基地局の通信能力及び計算能力は極めて高いが，各センサノードの通信能力及び計算能力はバッテリ駆動であることから総じて低いことが挙げられる．

WSNs を使用するに当たり，大抵の場合データの観測や送付などにおいて人間が介在することはない．昨今は省力化及び自動化の観点から WSNs の需要が増大しており，今後も増大する見込みである．使用用途として挙げられるのが主にビルやダムといった設備の稼働情報収集，野生動物の行動モニタリングといった人間が管理するには手に余る，足を踏み入れるには困難が伴う場所などで使用されている．WSNs を使用する際，WSNs が使用される環境によってはデータの盗聴や改ざんへの対処として個々のセンサノードが観測したデータの機密性や完全性が要求される．WSNs が使用される環境・用途として先に挙げたビルやダムなどの設備の稼働情報収集の場合，テロリストなどの攻撃者によってセンサノードが観測したデータや通信の盗聴や改ざんが行われた場合，セキュリティ及びプライバシーの観点から言って明らかに好ましいことではない．個々のセンサノードが観測したデータの機密性や完全性が要求される．

WSNs においてはセンサノードのバッテリが限られていることは既に述べたが，ゆえに WSNs においては消費電力の抑制によるバッテリの延命が課題であり，要求を満たすための処理方式の 1 つがデータアグリゲーションである．データアグリゲーションは各センサノードの値を処理を加え 1 つにすることで，全体の通信回数を削減することにより，消費電力の抑制を可能とする．

また前記の機密性や完全性を実装する場合，実装に用いられるハッシュ関数や疑似乱数生成関数がセンサノード上で動作するに当たり，どの程度実行時間を費やすのか，どの程度メモリを費やすのかについて検討することは非常に重要である．理論的に動作することはわかるが実際の機器上で動作させる場合，演算時間やメモリ量が必要なのかは実装してみないとわからない部分も多い．本研究では，2 種類のデータアグリゲーション方式を取り上げ，デー

†1 Japan Advanced Institute of Science and Technology (JAIST)

Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

タの完全性を満たすことに注力した宮地・面の方式とデータの機密性を満たすことに注力した PEQ 方式のそれぞれのセンサノード上の演算を MicaZ 上で実装した．そして, 双方の方式が要求するメモリ量及び演算時間を計測することでフィージビリティスタディを行った．

以降の段落構成は以下のとおりとなっている．2 章ではマイクロプロセッサの 1 つである MICAz 及び使用した OS である TinyOS と暗号関係の処理を行うために導入した relic-toolkit ライブラリの説明を行う．3 章では今回 WSNs が行っているデータ処理の 1 つであるデータアグリゲーションの説明について行う．4 章では実験を行うに当たっての目的・実験環境・実験結果についての記述を行う．5 章では 4 章でなされた実験結果の評価・考察について行う．6 章では最後に本研究のまとめについて行う．

2. MICAz における実装について

2.1 MICAz

MICAz は米 Crossbow Technology 社が開発したセンサネットワーク用の無線端末の名称であり, 光, 温度, 音を観測する能力を持っている.[9] 使用プロセッサが ATMega128L(8bit マイクロプロセッサ), クロック数が 7.37MHz, プログラムメモリが 128kB, RAM が 4kB, フラッシュメモリが 512kB, 無線用チップに CC2420 を装備しており, 無線使用周波数が 2400MHz, 最高通信速度が 250kbps である．但し作成したプログラムはフラッシュメモリに書きこまれる

2.2 TinyOS

TinyOS はインターネット上の有志の手によって開発されたオープンソースの OS である[11]．無料でダウンロード及びインストールが可能である．名前に Tiny(小さな) とつく通り MICAz をはじめとするマイクロプロセッサ等を動作させるのに適した OS である．また実装環境として C ライクなプログラミング言語である nesC をサポートしている．

2.3 RELIC-toolkit

RELIC-toolkit は C/nesC で利用できる暗号ライブラリの 1 つであり, 暗号などの実装を行う際に暗号化処理に用いられるハッシュ関数や疑似乱数生成関数, 楕円曲線暗号, ペアリング暗号, RSA 等の暗号プロトコルをサポートしている[10]．

2.4 Avrora

Avrora は Tizer らが開発したシミュレーションソフトウェアであり, Java 言語で実装されている[5]．クロック数やシミュレーションの対象となる演算が必要とするメモリ (ROM や RAM) の容量も計算する．今回は MICAz 自体を用いた実験も行うが, 通信の遅延などの発生による結果の揺らぎを考慮し, 演算をシミュレーションする Avrora を比較対象として使用した．

3. データアグリゲーション方式

3.1 データアグリゲーション

データアグリゲーション行う理由として挙げられるのが, センサノードのバッテリが極めて限られていることに起因するセンサノードの消費エネルギー抑制である．センサノードのエネルギー消費として最も大きいのがデータ送付であり, 全体の 8 割程度を占めていることからもデータアグリゲーションを導入するメリットは大きい[2]．データアグリゲーションは WSNs におけるデータ処理の 1 形態である．データアグリゲーションが用いられている WSNs においては各センサノードから別のセンサノード (無論最終送付先は基地局である) にデータを送付する際に観測したデータに処理を加え 1 つのデータにしたのちに別のセンサノードに送付する．データアグリゲーションを用いると各センサノードの計算時間を費やすことになるが, 代わりに通信のデータ量は減る結果となる．データアグリゲーションが使用できる環境として基地局がネットワークに属する個々のセンサノードデータではなくセンサノードデータを統計処理された結果を必要としている場合に有効である．データアグリゲーションはセンサノードが読み取ったデータを基地局に送付する際に, 途中のセンサノードでデータを統合 (アグリゲーション) することで, 中間センサノードの通信コストを下げることを可能とする．

データアグリゲーションを使用する有効性はセンサノードが観測したデータを送付する際に発揮されるわけであるが, 具体例として以下の図を用いて説明する．図 1 は標準的なツリー構造を取る WSNs を想定したものであり, 一番上に書かれている灰色の丸が基地局, 白い丸がセンサノード, 丸と丸を結んでいる線がネットワークのリンクを表している．最初に基地局が各センサノードにデータを送付するようにクエリを直属のセンサノードに送付し, 以下受信したセンサノードは自分より下位となるセンサノードにクエリを転送する．葉ノードまでクエリが到達すると, 葉ノードは自分が観測したデータを上位となるセンサノード (クエリを送付してきたセンサノード) に送付する．図 1 においてデータアグリゲーションを用いなかった場合と用いた場合を比較すると, 前者は受信データ数は自分より下位となるセンサノード数全てと同じであり, 送信データ数は受信データ数に自分自身を加えた数値となる．対して後者は受信データは自分の直接所属しているセンサノード数のみであり, 送信データ数に至ってはデータアグリゲーションを行っているため, 1 となる．

Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

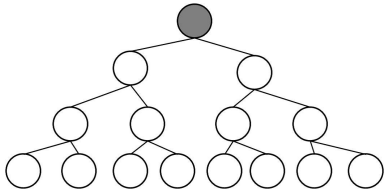

図 1 ワイヤレスセンサネットワーク (ツリー構造の場合)

データアグリゲーションの処理方式としては加算, 平均値, 最大値, 最小値, 中間値等が存在しているが, 前記のうち平均値や分散値も計算できる加算方式が使用用途の多さと計算が容易であることから代表的であり, 加算方式のみを実装するデータアグリゲーション方式もある. データアグリゲーション方式の中では加算方式であることを利用し, 斬新な方法で機密性を満たす方式も存在する[3]. その他ハッシュ関数と MAC 関数を用いてデータの完全性を確保する方式[4], XOR 関数を用いて処理を高速化しつつデータの機密性を満たす方式[6], ヒストグラムを用いてデータアグリゲーションを行い, 結果は近似値となるものの加算以外の方式もサポートする方式[7] が存在する.

本稿では宮地・面方式を実装評価することを目的として, まずは宮地・面方式の演算部分を実装評価する. また, 比較対象として演算処理が非常に軽量な PEQ 方式も実装評価する (主演算は XOR). 宮地・面方式は完全性のみを満たしているデータアグリゲーション方式であり, MAC 関数やハッシュ関数を用いた効率的な方式であり, 計算量やメモリ量 (コード量は含まれない) 通信の遅延, 動作に必要なメモリといった実用的な観点から評価する. 一方 PEQ 方式は機密性のみを満たしている方式であり, 演算の中心に XOR 関数を使用しているためきわめて動作が軽量である. この方式はデータアグリゲーション方式であるが, 個々のセンサデータを保持したままパケットが基地局に送付される.

3.2 宮地・面方式

データアグリゲーションを実現する上で完全性を満たすことは重要である. 例えばある町の一帯における電力消費や水道消費の度合いを統計的に知りたい場合にデータアグリゲーションを利用する例が紹介されている[3]. 宮地・面方式では Optimal security を一番の目玉としている. Optimal security とは任意のセンサノードが攻撃者によって乗っ取られたとしても, 攻撃による影響がリンク先のセンサノードに及ばず存在を局限できるという安全性である. 既存のデータアグリゲーション方式では, 攻撃者によって複数のセンサノードが乗っ取られた場合の処理として, 1 回の通信において各センサノードと基地局間で 2 往復の通信が必要であるのに対して, 宮地・面方式では多数の corrupted センサノードへの攻撃に対して 1 往復の通信で済む方式となっている.

宮地・面方式の特徴としてネットワーク構造がツリー型を採用していることやデータの完全性確保のために TESLA 方式を基にハッシュ関数と MAC 関数を使用していること. 攻撃者に強めの仮定を置いている. またデータアグリゲーション方式として加算をメインにしているが, 理由としては加算方式のみであっても平均, 分散や標準偏差を派生的に計算可能だからである[3].

宮地・面方式を用いる際に各センサノード s_i はメッセージ $M_{i,t}$ を持っており, <count, value, identifier, commitment, confirmation> である. count はセンサノード s_i とリンクを持つセンサノードの内下位センサノードの数であり, value は count に属するセンサノードが持つデータを全て足した値である. identifier はセンサノード s_i に対して $I_i = k_{i,0} = H^{\ell}(k_{i,\ell})$ と計算される値であり, l は最大セッション数, $k_{i,\ell}$ はセンサノード s_i と基地局間で共有される秘密鍵である. $k_{i,t} = H^{\ell-t}(k_{i,\ell})$ はセッション $t(1 \leq t \leq \ell-1)$ における MAC 関数の秘密鍵であり, 次のセッション $t+1$ において $I_i = H^{t}(k_{i,t})$ であるかどうかを確認する identifier としても扱われる. commitment は暗号化に用いるコミットメントであり, confirmation は結果の確認である. 説明を踏まえたうえで, セッション t におけるセンサノード s_i がメッセージ $M_{i,t}$ を計算する式を表すと以下のようになる.

$$M_{i,t} = < c, v, k_{i,t-1}, \mu_{k_{i,t}}, (\oplus\lambda_{k_{j,t}}) \oplus \lambda_{k_{i,t}} >, \quad (1)$$

where $\mu_{k_{i,t}} = MAC_{k_{i,t}}(c||v)$ and $\lambda_{k_{j,t}} = MAC_{k_{j,t}}(p_{j,t-1})$

c はセンサノード s_i とリンクを持つセンサノードの内下位センサノードの数であり, $v = v_{i,t} + \sum v_{j,t}$ は c に属するセンサノードが持つデータを全て足した値である. $v_{j,t}$ はセンサノード s_i が読み取ったデータである. $k_{i,t-1}$ は各センサノードの識別に使われるハッシュ関数である. $\mu_{k_{i,t}}$ はデータの完全性確保のために使用されるコミットメントであり, c と v を連結して MAC 値を取っている. $\lambda_{k_{j,t}}$ は確認のための値であり, $p_{j,t}$ はセッション t におけるセンサノード s_i の MAC 関数の結果であり, $p_{j,t} \in OK, NG$ である. ちなみにもしセンサノード s_i が葉ノードであった場合, $c = 1, v = v_{i,t}$ かつ confirmation $= \phi$ である. $M_{i,t}$ は $M_t =< 1, v, k_{t-1}, MAC_{k_t}(1||v), \phi >$ である.

3.3 PEQ 方式

WSNs の使用用途として人間の脈拍や血圧といった身体情報を収集し, 医療診断に役立てる場合も考えられるが, 人間の身体情報は個人情報であるので許可された者以外が情報にアクセスすることは好ましくない. 以上のことからデータの機密性を満たすことは重要であ

Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

るといえる．データのプライバシーを満たしつつ WSNs におけるデータアグリゲーションのクエリを正しく評価することを重視し, GP^{2}S 方式[7] や PDA 方式[3] の両者の長所を併せ持ったデータアグリゲーション方式である．

個々のデータのプライバシーを満たしつつほとんどすべての種類のクエリ (加算, 最大値/最小値, 平均, 中間値, ヒストグラム) に対応した方式である．本方式はセンサノード間の通信でハッシュ関数を使用せず, XOR 関数のみの使用を実現している動作が極めて高速な方式である．PEQ 方式の特徴として宮地・面方式と同じくセンサノードはツリー型を採用しており, 加算のみの PDA 方式[3] と比較して加算以外の方式をサポートしていること, 処理結果が近似となる GP^{2}S 方式[7] に対して処理結果は非近似値となることが挙げられる．また XOR 関数はデータの機密性確保のためにも使用される．

PEQ 方式を用いる際に各センサノードが計算する式はセンサノードが葉ノードが中間ノードであるかによって変わるが, 最初に共通処理として以下の計算を行う．

$$x = c_1 * e_1 \oplus c_2 * e_2 \oplus \ldots \oplus c_m * e_m \quad (2)$$

$(c_1, c_2, \ldots, c_m)$ はセンサノードが観測したデータを 2 進数表現したビット列であり, $e_1, e_2, \ldots, e_m$ は基地局から割り当てられた機密性確保のための秘密ベクトル (秘密鍵) である．x は対象となるセンサノードが生成した処理結果である．対象となるセンサノードが葉ノードの場合は先ほどの処理で終了である．センサノードが中間ノードの場合は最初に全ての下位センサノードからデータ $x_1, \ldots, x_k$ が送付されるまで待ったのち $z = y \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_k$ を計算し z を送付する．y は中間ノード自身が観測したデータを処理したものであり, z は中間ノードがさらに上位ノードに送付するデータである．

4. 実　　装

4.1 実験目的

実験の目的は, 実際の使用を想定した場合に各方式がどの程度の実現性なのか検討を行うことである．具体的には, 宮地・面方式及び PEQ 方式で使用される関数が計算に費やす時間及びメモリ量 (コード量, データ量) を測定し, これらの方式におけるセンサノードの演算時間及びメモリ量を見積もることである．WSNs で使用されるセンサノードを想定した MICAz(TinyOS 搭載) を使用する．

WSNs では公開鍵ベースの方式と共通鍵ベースの方式が提案されているが，今回対象とする 2 つの方式は共通鍵ベースの方式である．既存の WSNs の実装に関する研究では，各センサでの演算処理時間は共通鍵ベースの方式で数ミリ秒から数百ミリ秒，公開鍵ベースの方式で数秒から数十秒かかることが分かっている．これらのことを踏まえて，今回の実験では各センサで演算処理時間が $500ms$ 以内であることを目標値として定め，これら 2 つの方式がこの目標値をクリアしているかどうかを検証した．

4.2 実験環境

ここでは実験に用いた環境について述べる．本実験で用いた実験環境は 3 形態ありそれぞれ PC, Avrora, MICAz の 3 つである．また実験に用いた関数は XOR 関数, ハッシュ関数, MAC 関数の 3 つである．

本実験においては使用している WindowsPC に VMWare Player をインストールすることで Ubuntu10.10 を動作させ, 併せて TinyOS や RELIC-toolkit をインストールしている．使用した PC の性能は使用プロセッサが Intel Core2 Duo, クロック数が 2.93MHz, RAM が 2GB であるが, VMWare Player のメモリは 372MB とした．

既に RELIC-toolkit 内のライブラリにはハッシュ関数は用意されていたが, MAC 関数は用意されていなかったので, 各関数のシミュレーションを行う上で公平性を期すため, RFC2202 を参考に実装しライブラリに組み込んだ[1]．

シミュレーション環境として Avrora を導入するが, 理由として MICAz のシミュレータである Avrora を用いて事前にどの程度の実行時間がかかるかを測定することで, 実験の評価・考察に深みを持たせることが挙げられる.[8]

4.3 実験手順

- PC 上での実験
  - (1) RELIC-toolkit を利用して演算部分を C 言語で実装
  - (2) CRELIC-toolkit 付属のベンチマーク機能を利用して時間測定
- シミュレーション (Avrora) での実験
  - (1) C 言語プログラムを NesC に成形
  - (2) MICAz が解釈できる実行ファイル（main.exe）の作成
  - (3) Avrora が解釈できるファイル（main.od）の作成
    avr − objdump − zhDbuild/micaz/main.exe > main.od
  - (4) Avrora で実行
    avrora − monitors = call − profilemain.od
- MICAz 上での実験
  - (1) main.exe を MICAz へコピー
    makemicazreinstallmib520, /dev/ttyUSB0

Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

(2) MICAz でライトを点灯させ, 時間を測定
- XOR(160bit):1.0×10^5 回実行させ測定結果の平均値を導出
- XOR(2^{13}bit):1.0×10^4 回実行させ測定結果の平均値を導出
- SHA1:1.0×10^4 回実行させ測定結果の平均値を導出
- HMAC-SHA1:1.0×10^3 回実行させ測定結果の平均値を導出

4.4 実験結果

実験結果を以下に記す.

表 1 各演算の演算時間

	PC(ns)	シミュレーション (ns)	MICAz(ns)	コード量 (kB)
XOR(160-bit)	22.0	9.99×10^4	10.4×10^4	181
XOR(2^{13}-bit)	831	3.13×10^6	3.13×10^6	181
SHA1	2.83×10^3	7.97×10^6	7.99×10^6	183
HMAC-SHA1	1.08×10^4	3.21×10^7	3.23×10^7	181

表 1 の 1 行目に書かれている項目の内最初の 3 つである PC, シミュレーション, MICAz の単位は () 内に書かれているとおりである. シミュレーションとは Avrora を用いた MICAz シミュレーションのことである. コード量とは 4 種類の演算を MICAz で実行する際に作られる main.exe のコード量である. XOR に関する項目が 2 種類存在するのは 160bit は宮地・面方式で使用されているビット数に合わせたものであり, 2^{13}bit は次の評価・考察で説明するが, PEQ 方式での使用を想定したものである.

5. 評価・考察

センサノード数 s が $511(2^9 - 1)$, センサデータのバイト数を 2Byte, ネットワーク構造が完全 2 分木 (木の深さは 8 段) であるとする. この章では 4 章で評価した各演算の結果を用いて宮地・面方式と PEQ 方式の演算時間がどの程度かかるかを見積もる. 2 つの方式の計算量は宮地・面方式の演算回数は XOR 関数を XOR, ハッシュ関数を $Hash$, MAC 関数を MAC とすると, 葉ノードの場合は $Hash + MAC$ であり, 中間ノードの場合は $2XOR + 7Hash + 8MAC$ である. PEQ 方式の演算回数はセンサノードが表現できるビット数を m とすると, 演算回数を評価する式は葉ノードの場合は $(m-1)XOR$ となり, 中間ノードの場合は $(m+1)XOR$ となる. 宮地・面方式と PEQ 方式の演算時間を以下に記す.

上記の結果に対する各項目の説明を行う. 1 行目に書かれている項目の内最初に見える

表 2 宮地・面方式と PEQ 方式の演算時間の比較

	ノードの位置	計算量	演算時間 (ms)
宮地・面方式	葉ノード	$Hash + MAC$	40.1
	中間ノード	$6XOR + 7Hash + 8MAC$	310
PEQ 方式	葉ノード	$15XOR$	47.0
	中間ノード	$17XOR$	53.2

ノードの位置とは対象となるセンサノードがどの位置にあるかを表している. 葉ノードとは図 1 において一番下のセンサノードである. 中間ノードとは先に述べた葉ノードと基地局以外のセンサノードのことである. 結果より宮地・面方式の演算時間は $40.1ms$ と $310ms$ であり, PEQ 方式は葉ノードが $47.0ms$ と $53.2ms$ となっている. これらの結果より, 2 つの方式では各センサでの演算処理時間が我々の定めた目標値である $500ms$ 以内を達成できたといえる.

PEQ 方式は確かに XOR 関数のみを使用し演算のみで考えると高速ではあるが, 実際に WSNs 全体で考えると演算時間が短いとは限らない, 特に PEQ 方式は演算量がセンサノードが表現できるビット数に比例している. 上記の点に依存することは WSNs としてフィージビリティを考えた場合好ましくないといえる. また PEQ 方式の特徴として葉ノードと中間ノードにおいて演算時間に差がほとんど見られないが, これは中間ノードは葉ノードに比べて子の数だけ多く XOR 関数を使用するだけだからである. また使えるデータストレージ (フラッシュメモリ) の容量は 512kB であるのに対してコード量が 183kB なので, データ量としては約 330kB しか使用できないことが分かった.

6. おわりに

本論文では, データアグリゲーション実装の前段階として機密性や完全性を実現する場合, 実装に用いられるハッシュ関数や MAC 関数がセンサノード上で動作するに当たり, どの程度実行時間を費やすのか, どの程度メモリを費やすのかについて検討した. 理論的に動作することはわかるが実際の機器上で動作させる場合, 演算時間やメモリ量がどの程度必要なのかは実装してみないとわからない部分が多い. そこで, 今回 2 種類のデータアグリゲーション方式を取り上げ, データの完全性を満たすことに注力した宮地・面の方式とデータの機密性を満たすことに注力した PEQ 方式のそれぞれのセンサノード上の演算を MicaZ 上で実装した. また双方の方式が要求するメモリ量及び演算時間を見積もることでフィージビリティスタディを行った.

Vol.2011-DPS-146 No.32
Vol.2011-CSEC-52 No.32
2011/3/11

7. 謝　　辞

本研究における MICAz の実験環境構築に関して, Ta Dat Tien 氏, 佐々木廉氏及び Pham Anh Cam 氏に深く感謝の意を表します.

参 考 文 献

1) R. Cheng, R. Glenn, Test Cases for HMAC-MD5 and HMAC-SHA-1, RFC2202, 1997.
2) Peiman Ghaffariyan, An Effective Data Aggregation Mechanism for Wireless Sensor Networks, WiCOM2010, pp.1-4.
3) Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher, PDA:Privacy-preserving data aggregation in wireless sensor netoworks, INFOCOM2007, pp.2045-2053.
4) A.Miyaji and K.Omote, Efficient and optimally secure in-network aggregation in wireless sensor networks, WISA2010, Springer-Verlag, 2010, pp.135-149.
5) Ben L. Titzer, Daniel K. Lee, Jens Palsberg, Avrora: Scalable Sensor Network Simulation with Precise Timing, IPSN2005, pp.477-482.
6) Hai Vu, Thuc Nguyen, Neeraj Mittal and S. Venkatesan, PEQ: A Privacy-preserving Scheme for Exact Query Evaluation in Distributed Sensor Data Networks, SRDS2009, pp.189-198.
7) Wensheng Zhang, Chuang Wang and Taiming Feng, GP^{2}S Generic Privacy-Preservation Solutions for Approximate Aggregation of Sensor Data, PerCom2008, pp179-184.
8) Avrora, http://compilers.cs.ucla.edu/avrora/
9) Crossbow Technology, http://www.xbow.com/
10) RELIC-toolkit, http://code.google.com/p/relic-toolkit/
11) TinyOS, http://www.tinyos.net/