JavaによるMISTYソケット通信ライブラリの実現

4H－03

松田 規[1]　米田 健[1]　野村 立[2]　中川路 哲男[1]
[1]三菱電機（株） 情報技術総合研究所
[2]三菱電機（株） 電力・産業システム事業所

1. はじめに

近年、電力産業システムの分野において、JavaVMを搭載することによる小型制御機器の高機能化が検討されている。PLCモデムやPHS等と組み合わせて使用することにより、電力メータの遠隔検針や、制御機器の遠隔保守が実現でき、保守にかかるコストが削減される。また、Javaアプリケーションによる制御機能のカスタマイズが可能となるため、この技術を電力メータに適用すると、ユーザによる家電機器の遠隔制御などにも利用できると期待される。しかし、電力使用量や制御コマンドなどの情報はプライベートな情報であり、盗聴や改竄から保護されなければならない。また、制御機器の不正制御などに対する対策も必要である。

そこで、我々は小型制御機器との通信データのセキュリティを確保するため、Java環境向けの通信データ暗号化機能を開発した。検討の結果、リソースが限られる小型制御機器には共通鍵暗号のみを用いた方式が最適と考え、高速かつ小型実装が可能なMISTY[1]で通信データを暗号化した。また、他ベンダが開発した制御機器にも適用できるように、Java言語[2]のみで記述を行った。さらに、Java標準のソケット・ファクトリ機構を利用する事により、事前に制御機器に組み込まれたアプリケーションに関しても、一切の改修なしに暗号化通信が可能となった。

2. 遠隔制御システムを実現する上での課題

遠隔制御システムの一形態として、図1に示すモデルを想定する。これは、電力会社内の検針センタが、各家庭に設置された電力メータの遠隔検針を行うモデルである。ここで、センターメータ間は、TCP/IP通信がサポートされたネットワークで接続されていると仮定する。

検針センタには、メータに搭載されたHTTPサーバと通信を行い電力使用量を取得する検針プログラムと、メータ上で動作するサービスプログラムをオンラインで提供するダウンロードサーバが搭載される。

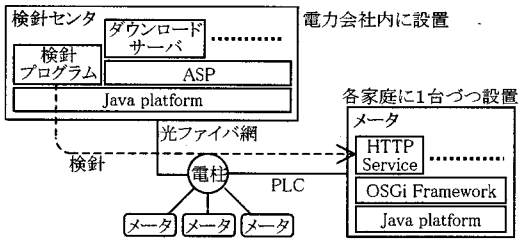

図1：遠隔検針システムの概要

各家庭に設置されるメータは、複数ベンダによって製造されるが、どのベンダの機器でもJava実行環境に加えOSGiフレームワーク[3]が搭載される。これは、組込み機器向けに開発された環境であり、サービスと呼ばれる機能単位での動的ローディング機能を提供する。この機能により、アプリケーション実行時の省メモリ化を図ることができる。また、サービスとして提供されるHTTPサーバを用いて電力使用量等の情報提供を行う。

これらのシステムを用いて遠隔検針を実現する場合、次に示す課題を解決する必要がある。

- 通信データの保護
  電力使用量やサービスプログラムなどのデータを、盗聴・改竄から保護する
- 不正ユーザによる電力使用量取得の防止
  検針センタ以外のユーザによる電力使用量取得を検出・防止する
- 他社製品への上記機能の組込み方式の確立
  コードの改修が困難な他社製品に対して、セキュリティ通信を行う機能の組込みを行う

そこで、これらの課題を解決するため、MISTYソケット通信ライブラリの開発を行った。

Implementation of secure socket library using MISTY on Java platform.
N.Matsuda[1], T.Yoneda[1], R.Nomura[2], T..Nakakawaji[1]
[1]Information Technology R&D Center,
[2]Energy and Industrial Systems Center
Mitsubishi Electric Corporation

3. 実現方式

通信データのセキュリティを確保するため、暗号技術の適用を行った。検討の結果、小型制御機器には高速な共通鍵暗号が最適であると考え、MISTY による暗号化通信を実現した。以下、その実現方式について述べる。

3.1. 暗号方式設計

通信の秘匿を行うための暗号方式として、公開鍵暗号方式と、共通鍵暗号方式の２方式が主流として用いられる。表１に、各暗号方式の特徴を示す。

表１：公開鍵暗号と共通鍵暗号の特徴

非各項目	公開鍵暗号	共通鍵暗号
秘匿通信	可能（低速）	可能（高速）
改竄検出	可能（低速）	可能（高速）
鍵の配布	容易	困難

PC 等の高速な CPU を搭載したシステムにおいては、この双方を組み合わせて利用するのが一般的である。しかし、小型制御機器は CPU パワーが小さいため、共通鍵暗号と比較して 1000 倍ほどの CPU パワーを必要とする公開鍵暗号は不向きである。そこで、図２に示すように、本ライブラリでは、共通鍵暗号である MISTY のみを用いる事とした。MISTY は他の共通鍵暗号アルゴリズムと比較しても高速な暗号アルゴリズムであり、CPU パワーが限られた小型制御機器に適すると考えられる。

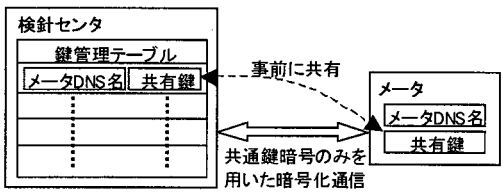

図２：暗号化アルゴリズムと鍵管理

また、共通鍵暗号を用いる場合は、通信を行う２者間で、事前に安全な通信チャネルにて共有鍵を持ち合う必要がある。しかし、電力メータは取り外しが極めて困難なため、設置後にサービスセンタ等で共有鍵をインストールすることは難しい。そこで、設置の際に、通信に用いる DNS 名等を設定するため、このタイミングでメータ内の ROM 等に焼き込む事とした。

3.2. クラス設計

通信を行うクラスの作成には、(1)Socket クラスの派生クラスを作成する方式、(2)SocketFactory を用いて既存クラスを置き換える方式が考えられる。

前者の方式(1)では、アプリケーションの要件に応じて平文通信と暗号化通信を使い分けられるため望ましいと考えられるが、既存アプリケーションに変更が必要となる。後者の方式(2)では、既存アプリケーションに対する改修は不要であるが、平文通信・暗号化通信のいずれか一方しか対応できないと言う問題点がある。

本ライブラリでは、他ベンダが開発した既に運用中で変更が困難なメータ内アプリケーションに対しても暗号化通信機能を付加することが必要であるため、後者の方式(2)を採用した。図３に、開発したクラスの構成を示す。

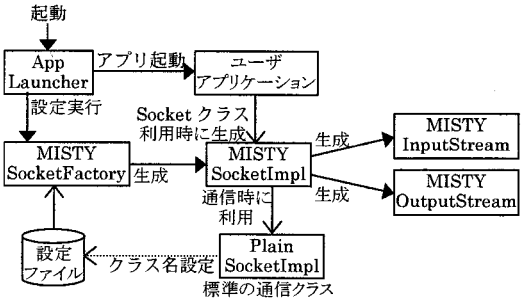

図４：クラス構成概要

図のように、ユーザが AppLauncher を起動すると、本ライブラリの Factory 設定がなされた後、ユーザアプリケーションが起動される。Factory 設定が行われると、Socket クラス内部で利用されるクラスは、全て本ライブラリのクラスに置き換わるため、アプリケーションに改修を加えることなく暗号化通信が可能となる。

4. まとめ

本論文では、小型制御機器の遠隔検針に向けた Java を用いた暗号通信方式について検討した。検討の結果、共通鍵暗号で暗号化を行う方式が最適と判断し、MISTY による暗号化通信を提案した。

今後は、小型制御機器の一つである電力メータに本方式を適用し、実速度の測定・評価を行う。また、リプレイ攻撃などの各種攻撃に対応可能なセキュリティ機能の追加を行っていきたいと考えている。

参考文献

[1] M.Matsui:"ブロック暗号アルゴリズム MISTY", 信学技報, ISEC96-11(1996-07)

[2] JDK1.1.7 API documentation. http://java.sun.com/

[3] OSGi Service Platform Release 2 Specification http://www.osgi.com/resource/spec_download.asp