非否認性を担保したセキュリティSLA構築手法に関する検討

高橋 健志 †　　Jarmo Harju‡

†(独) 情報通信研究機構
184-8795 東京都小金井市貫井北町 4-2-1
takeshi_takahashi@ieee.org

‡Tampere University of Technology
Korkeakoulunkatu 10, 33720 Tampere, Finland
jarmo.harju@tut.fi

あらまし 現在のオンラインサービスでは、サービスプロバイダが掲示したセキュリティレベルに対し、ユーザが合意する形になっている。そのため、そのセキュリティレベルに満足しないユーザはそのサービスを利用しない、もしくは我慢して利用することになるが、簡単なネゴシエーションを実現することにより彼らの満足度は大幅に改善可能である。そこで本稿では、ユーザとサービスプロバイダの間にてネゴシエーションを実施し、セキュリティSLA(達成すべきセキュリティポリシ) を構築する方式を提案する。同時に提案方式の実現可能性と非否認性、DoS 耐性について議論する。

Building Non-Repudiable Security Service-Level Agreement

Takeshi Takahashi†　　Jarmo Harju‡

†National Institute of Information and Communications Engineering
4-2-1 Nukui-Kitamachi Koganei, Tokyo 184-8795, Japan
takeshi_takahashi@ieee.org

‡Tampere University of Technology
Korkeakoulunkatu 10, 33720 Tampere, Finland
jarmo.harju@tut.fi

Abstract The security features of current online services are mostly defined by the service provider. Users may feel unhappy if the terms do not meet the expected criteria, since they can only agree to user the service with the terms or declin using the service. Even a simple negotiation might result in a more satisfying outcome in many cases. This article proposes a mechanism to build non-repudiable security service level agreements between a user and a service provider. It also demonstrates the feasibility of the mechanism by introducing proof-of-concept implementation, and analyzes its non-repudiability and DoS-resistant feature.

1　はじめに

オンラインサービスは近年大幅に増加・発展を続けているが、それと並行してサイバー社会でのセキュリティとプライバシを脅かすインシデントも増加している。そのようなインシデントからユーザを保護する仕組みが求められおり、様々な対策技術が既に存在しているが、それらのすべてを実装することはコストおよびサービスの利便性の観点から望ましくない。そこでセキュリティと利便性のバランスを考慮する必要があるが、ユーザにより環境やセキュリティに対する要求 (セキュリティ要求, requirement) が異なるため、そのバランスを一様に決定するのは非現実的である。そのため、そのバランスは

ユーザ毎に、またサービス利用の度に決定される必要がある。

そのバランスを考慮する際には、いくつかの技術的課題が存在する。第一に、ユーザの requirement は、機械可読な形で記述されなければならず、構造化されたフォーマットが必要不可欠である。第二に、技術的知識の少ない普通のユーザが必要なセキュリティ技術を特定するのは非常に困難であるため、そのようなユーザが requirement を特定できる技術が必要である。第三に、サービスが満たすべきセキュリティポリシを構築するのに自動ネゴシエーション手法が必要である。ユーザは現在、サービスプロバイダが掲示するセキュリティポリシに対して同意するか拒否するかの二択しかなく、そのプロバイダと必要なセキュリティレベルや技術について交渉する手段を持たない。そして、もしプロバイダがユーザとネゴシエーションしたいと考えても、人手でのネゴシエーションはコストの観点から非現実的である。第四に、ネゴシエーションの結果生成される合意事項は、非否認性を担保していなければならない。ユーザとプロバイダが満たすべきセキュリティポリシに合意しても、セキュリティインシデントが生じないわけではない。そのため、インシデント発生時にその原因が合意事項違反にある場合には、違反をされた側は違反をした側を訴えるべく、その合意事項を証拠として利用できる必要がある。

上記の問題に対応し、セキュリティと利便性のバランスを実現すべく、本稿では非否認性を担保したセキュリティSLA(SSLA) を構築する手法を提案する。SSLA とは、ユーザとプロバイダ間で合意した、サービスが実現すべきセキュリティレベルである。提案方式は要素技術としてセキュリティ表現技術と翻訳技術を提供する。セキュリティ表現技術は requirement とセキュリティ対応能力 (capability) を機械可読なフォーマットにて記述する技術である。その記述は複数の視点から行うことができ、その視点のことを次元と呼んでいる。翻訳技術は異なる次元にて記述されたそれらの情報を任意の次元の情報に変換する技術であり、それにより技術的知識の少ないユーザが技術用語を用いずに requirement を記述し、自動的に技術用語へと変換することを可能とする。これにより、ユーザとサービスプロバイダは SSLA 構築に向けたネゴシエーションを実施することが可能になる。提案方式は、そのネゴシエーションの結果として、非否認性を担保した SSLA を構築する。その結果、従来プロバイダが掲示するセキュリティポリシに yes もしくは no のどちらかしか回答できなかったユーザが、お互いに合意できるセキュリティポリシを作り上げることができるようになる。また、本稿では、提案方式の実現可能性、非否認性、DoS 耐性についても考察する。尚、本稿の補足情報は参考文献 [1] を参照されたい。

2　アーキテクチャの概要

提案方式は、ユーザ (User)、サービスプロバイダ (SP)、知識ベース (KB) といった 3 つのロールを定義している。User はオンラインサービスを利用し、SP はユーザにそのサービスを提供する。KB はセキュリティに関する各種情報を保持しており、翻訳を実現するために必須となる辞書を保持している。

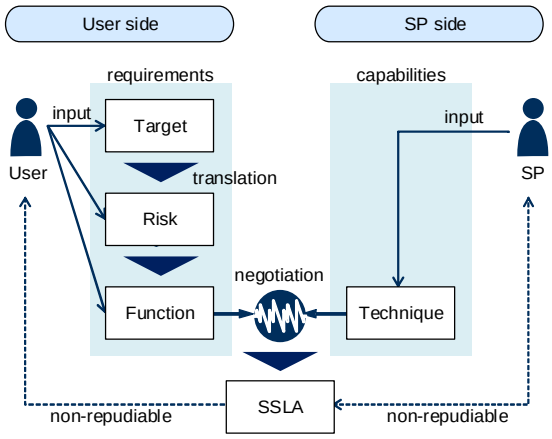

図 2: Process overview

図 2 は、提案方式の概要である。User は複数の requirement を任意の次元で記述し、翻訳技術を利用してそれらを単一の次元へ変換し集約する。その User の requirement と SP の capability を突き合わせ、ネゴシエーションすることにより、サービスが満たすべきセキュリティレベル、すなわち SSLA を構築する。

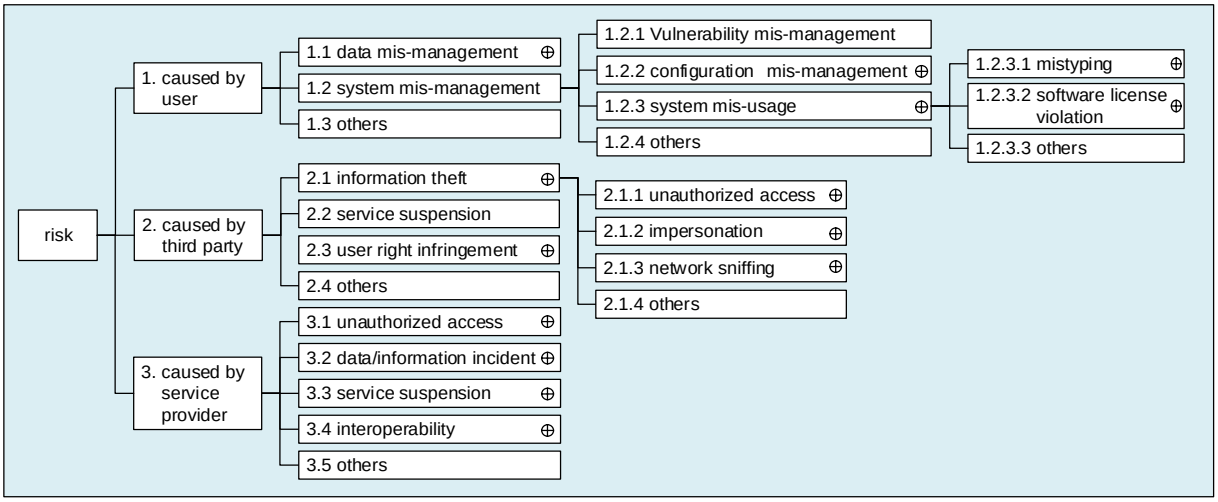

図 1: Risk 辞書の一例 (抜粋)

3 提案方式を構成する技術

提案方式は、複数の要素技術から構成され、本節では、そのそれぞれについて説明する。

3.1 セキュリティ表現技術

SSLA は User と SP の間で合意されたセキュリティレベルに関する情報である。SSLA の構築には、requirement と capability が明示される必要がある。ここで、requirement はどのようなセキュリティもしくはセキュリティ技術が必要かを記述した情報であり、capability はどのような技術を持っているか、もしくは何を実現できるかを記述した情報である。これらの情報は KB 内に保存された辞書内の語彙を用いて記述される。提案方式は、機械による処理を実現すべく、自由記述を最小化することを目指しており、そのため、それらの各語彙に一意の識別子を付与している。本識別子は、Object Identifier (OID)[2] の形式で表現される。そして SSLA は、この requirement と capability を User と SP の間で突き合わせることにより構築される。

様々なユーザが自由自在に requirement と capability を記述できるようにすべく、提案方式は Target、Risk、Function、Technique の 4 つの次元を用意している。そしてそのそれぞれの次元ごとに語彙とそれに対応する識別子を記載した辞書を用意している。Target 次元は守るべき対象を指定する。Target 辞書内に記載されている語彙から選んで記述するが、例えばユーザの「個人情報」などが存在する。Risk 次元は避けるべきリスク種別を指定する。Risk 辞書内に記載されている語彙から選んで記述するが、例えば「通信傍受」のリスクなどが存在する。Function 次元は実装すべき機能を指定する。Function 辞書内に記載されている語彙から選んで記述するが、例えば「ユーザデータの暗号化」や「ユーザの認証」などが存在する。Technique 次元は実装すべきセキュリティ技術・ツールを指定する。Technique 辞書内に記載されている語彙から選んで記述するが、例えば「AES」や「SHA」などが存在する。

これらの語彙の識別子は、それぞれの次元ごとに TARGET、RISK、FUNCTION、TECHNIQUE のいずれかの OID arc から始まる。図 1 に、Risk 辞書の一例を抜粋して示す。それぞれの辞書ごとに、TARGET、Risk、Function、Technique の OID arc に続く形で、各項目の識別子が記述される。KB 毎に、独自の辞書を保持してもよい。User も SP も通常は複数の requirement と capability を保持しているため、requirement も capability も実際にはこれらの OID のリストとして表現される。

場合によっては、また、User によっては、複数の次元の語彙を使ってセキュリティを表現したいケースも存在する。その際には、コロンを使っ

表 1: 翻訳対応表例 (抜粋のみ)

(a) [target, risk] 対応表

target	risk
TARGET.1.1.1	RISK.2.3.4
	RISK.3.2.3
TARGET.1.1.2	RISK.2.2.5
	RISK.2.3.2
	RISK.3.1.3

(b) [risk, function] 対応表

risk	Function
RISK.1.1.1	FUNCTION.12.1.3
	FUNCTION.17
	FUNCTION.23.3
RISK.1.1.2	FUNCTION.15
	FUNCTION.19.12.2

て複数の次元の語彙を掛け合わせて表記することができる。例えば、特定の Risk に対応する Function を指定した際には、Risk と Function の語彙をコロンを用いて接続することが可能であり、例えば”Risk.1.1.2:Function.19.12.2.”のように記述することができる。

3.2 翻訳技術

複数の次元を用意することにより、User は requirement を様々な視点から指定可能であり、結果として重要な requirement を指定できないリスクを低減することができる。しかしながら次元の異なる情報をコンピュータで自動処理するためには、それらの情報を任意の次元に翻訳する技術が必要となる。

提案方式は、ある次元の OID が別の次元のどの OID に相当するかを紐づけた翻訳対応表を用意し、それを参照することにより翻訳を実現する。この対応表も KB 内に保存されており、[target, risk]、[risk, function]、[function, technique] の 3 種類の対応表に分かれて存在している。それらの対応表は 2 つのコラムで構成されており、一方のコラムの OID がもう一方のコラムの複数の OID に対応する形になっている。例えば、[risk, function] の対応表では、Risk を表す OID と、それに対応する一つ以上の Function で構成されている。一つの Risk に対して一つ以上の Function が紐づいているのは、実際、あるリスクに対応するには複数の機能を要するケースが存在するためである。参考のため、表 1 に対応表例の抜粋を示す。

3.3 ネゴシエーションプロトコル

提案方式は、KB 参照と SSLA ネゴシエーションの 2 種類の通信を実施する。KB 参照は、Translation-request と-reply メッセージを用い、様々な次元で表現されている requirement と capability を翻訳する。Translation-request には requirement と capability が入っており、それを受け取った KB は次元変換を実施し、その結果を Translation-reply に入れて返信する。

SSLA ネゴシエーションは SSLA-proposal と-confirmation メッセージを用い、SSLA を構築する。SSLA-proposal は requirement と capability を保持しており、もしそのメッセージの受信者が提案された requirement に合意した際には、SSLA-confimration を送信する。もし内容に合意しない際には、SSLA-confirmation を送る代わりに別の requirement と capability 情報を入れた新たな SSLA-proposal を返信する。どちらか一方が SSLA-confirmation を送信するか、ネゴシエーションを中止するまで、本手続きは続く。SSLA-confirmation メッセージが届いた際に、本ネゴシエーションは終了し、その際の requirement のリストが SSLA となる。

上述の通り、提案方式は複数回のメッセージ交換を許可しているが、議論を簡略化するため、図 3 に一ラウンドで終了するネゴシエーション例を示す。ここで、KB_U と KB_{SP} は User と SP がそれぞれ信頼している KB である。ネゴシエーション開始に先立ち、User は KB_U にコンタクトをし、様々な次元で記述した requirement を Function 次元へと変換する。User は、その変換結果および KB_U の URI を入れた SSLA-

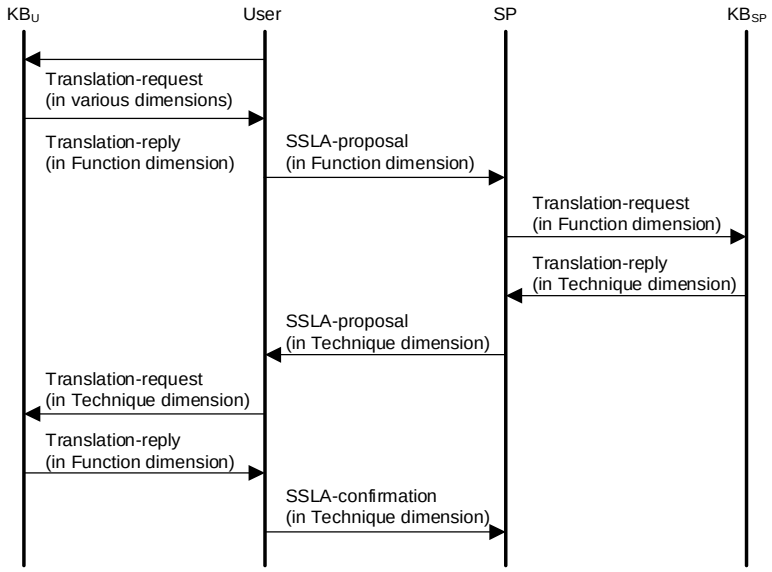

図 3: ネゴシエーション手続き例

proposal メッセージを SP に送る。それを受領すると、SP は自身が提案された requirement を満たせるかどうかチェックし、また、requirement を曖昧度の残る Function 次元ではなく、より具体化された Technique 次元にて合意し、責任範囲を限定したいと考える。そのため、KB_{SP} と通信し、User から受領した requirement を満たす Technique 次元の requirement のリストを問い合わせる。そして、その結果および KB_{SP} の URI を入れた新たな SSLA-proposal メッセージを構築し、User へと送る。それを受領した User は、提案された Technique のリストで、自らが元々実現したかった Function 次元での requirement を満たせるかどうかを確認すべく、再度 KB_U に問い合わせる。User は SP から受領した Technique 次元の requirement のリストを KB_U に送り、Function 次元へと変換された requirement のリストを受け取るので、それに基づき、元々の自身の requirement が満たされるかどうかを確認することができるのである。問題ないことが確認できた後、User は SP に対し SSLA-confirmation メッセージを送り、最終的な SSLA が合意されるに至る。尚、本手続きの冒頭では User は Function 次元の requirement を送っているが、最終的に合意されている SSLA は Technique 次元であることに留意されたい。

提案方式では、ネゴシエーション結果として生成される SSLA の非否認性を担保すべく、ネゴシエーションプロトコルのメッセージには暗号識別子とデジタル署名を利用している。SSLA-proposal メッセージは requirement と capability、通信をしている二者の識別子、ネゴシエーション ID、ランダムな文字列 (nonce)、タイムスタンプ、署名、そして proof-of-work トークンを内包している。二者の識別子についてはそれぞれの公開鍵を利用し、公開鍵もしくは証明書を一方向ハッシュ関数に通した値を利用している。ネゴシエーション全体を通じて変わることない値を持つネゴシエーション ID、そして nonce は送信者をリプレイアタックから守る。メッセージは、送信者の秘密鍵 (署名鍵) を用いて署名してある。Proof-of-work トークンは、hashcash スタンプ [3] を利用して生成する。本トークンを生成するには、送信者は計算をしなければならず、その結果、DoS 攻撃リスクを抑制するのに貢献する。SSLA-confirmation メッセージは SSLA-proposal メッセージに入っていたコンテンツを持っており、また、ssla-confirmation メッセージ同様、SSLA-confirmation メッセージ送信者の署名と nonce を持っている。

SSLA-confirmation メッセージが受領された時、送信者と受信者は requirement のリスト、すなわち SSLA を署名付きで共有している。これにより、両者とも、同一の情報を合意した証

拠として保持ることが可能となっている。

3.4 SSLA 決定アルゴリズム

上述のプロトコルの通り、SSLA-proposal メッセージが送られた際には、その受信者はそれを受け入れるか、対案を掲示するか、ネゴシエーションを終了するかを決断しなければならない。この決定は手動でも可能であるものの、多数の顧客とやり取りするためには SP は自動的に実現する方が望ましい。

一つの単純なアプローチは、相手から提案された requirement に対し十分な capability を自身が持っていればその requirement を SSLA として了承するというものであり、アルゴリズム 1 に図示する。ここで、req と caps は requirement リスト内の一エントリーと capability がそれぞれ入った変数であり、dimension サブルーチンは引数の記述の次元を返し、translate サブルーチンは第 1 引数を、第 2 引数で指定した dimension に翻訳する。尚、この翻訳は上述の KB 参照によりなされる。本アルゴリズムは、結果として 1 もしくは 0 を返答するが、1 であれば caps が req を満たすに十分であり、0 であれば不十分であることを示す。すべての requirement について本アルゴリズムを実施し、回答がすべて 1 であれば、受信者は caps が requirement をすべて満たすことができ、セキュリティSSLA として十分であることがわかる。逆に、もし requirement の中で caps では満たせないものがあった際には、その受信者は受け取った requirement 一覧を SSLA として認めることができない。そのため、その受信者はカウンター提案をするか、もしくはネゴシエーションを終了する。

4 考察

本節では、提案方式の実現可能性と非否認性、DoS 耐性について考察する。

4.1 概念実証実装

提案方式の実現可能性を検証すべく、我々は概念実証実装を実施した。概念実証は Java ベースにて実装し、requirement、capability は OID 形式の項目リストとして表現し、JSON オブジェクト内に格納して利用している。本概念実証実装を用いることにより、非否認性を担保した SSLA をネゴシエーションを通じて生成できることを確認した。尚、本実装に用いたソースコードは、オンラインにて公開 [4] している。

4.2 非否認性

提案方式により創造される SSLA は、署名が施されているため、第三者の存在に頼ることなく、非否認性を担保できている。これは、提案方式が User と SP の間で、第 3 者を介さずに実施していることからも明白であり、署名により SSLA の内容を検証することができる。現実社会で契約手続きをする際には、契約を結ぶ 2 者で同一の紙を 2 枚用意し、お互いにサインをするが、それと同様のことを電子通信の世界でも実施し、非否認性を担保している。

4.3 DoS 耐性

提案方式では、SP は User から SSLA-proposal を受け取ると、軽くない様々な処理を行う必要があるため、DoS 耐性について考慮する必要がある。DoS 攻撃を実施する攻撃者は、計算機能力や通信帯域、ステートを維持する能力などの有限な資源を枯渇させる。攻撃者は能力を増強することにより、どのようなサービスに対しても攻撃をすることができるが、プロトコルの弱点があると、攻撃者がそこをついて攻撃しやすくなってしまう。そのため、DoS 耐性とは通常、攻撃者が被害者に比べて明確に多くの作業をしなければ攻撃できない性質を指す。もし攻撃者が被害者よりも多くのリソースを要するのであれば、経済的には被害者のほうが有利である。

プロトコルは、DoS 耐性を持つためには、サーバがリソース割当・消費をする前に通信相手が十分な量の作業にコミットすることの証明が必要である。そして、この証明を検証するのにはほとんど労力を要さないことが必要である。

提案方式では上記に対応するため、hashcash stamp[3] を利用している。これは、送信者に対

Algorithm 1 SSLA 決定アルゴリズム例

```
1: if dimension(req) = Technique then
2:   if req ∈ caps then
3:     return 1
4:   else
5:     return 0
6:   end if
7: else
8:   translated-reqs ← Translate(req, Function)
9:   for all entry in translated-reqs do
10:    if entry ∉ Translate(caps, Function) then
11:      return 0
12:    end if
13:  end for
14:  return 1
15: end if
```

して相応の計算負荷を課すことができる proof-of-work を作成する。受信者は、受け取った hash-cash スタンプを効率的かつ容易に検証可能である。これにより、提案方式は通信開始者にパフォーマンス面でのペナルティを課し、それによってサーバの視点から DoS の危険性を軽減することが可能となる

5 結論

提案方式はセキュリティ表現技術、翻訳技術、ネゴシエーションプロトコル、SSLA 決定アルゴリズムを用い、非否認性を担保した SSLA を構築する。本稿では提案方式の実現可能性、非否認性、DoS 耐性についても考察し、本提案方式の有効性を示した。しかしながら、本稿で示した技術は今後さらなる研究を経て、発展されていく必要がある。例えば、requirement や capability の記述技術については、ユーザが自分で指定するのは、たとえ多数の次元が用意されていても、またたとえ知識があっても、時間を要し、面倒になりがちである。そこで状況やユーザを鑑みて、自動的に requirement や capability を記述してくれる技術が望まれる。特に、モバイル端末など、画面サイズが小さいものや、利便性が限定される場合には、その重要性はより高いものとなる。これらの研究を続けることにより、我々は、セキュリティと利便性のバランスをユーザ毎に最適化する研究に貢献していけるものと信じている。

参考文献

[1] T. Takahashi, J. Kannisto, J. Harju, S. Heikkinen, B. Silverajan, M. Helenius, and S. Matsuo, “Tailored Security: Building Nonrepudiable Security Service-Level Agreements,” *IEEE Vehicular Technology Magazine*, Sep. 2013.

[2] International Telecommunication Union, “Information technology - Open Systems Interconnection - Procedures for the operation of Object Identifier Registration Authorities: General procedures and top arcs of the International Object Identifier tree,” *X.660*, 2011.

[3] (2013, Apr.) Hashcash. [Online]. Available: http://hashcash.org/

[4] (2013, Jun.) Nict-tut collaboration project. [Online]. Available: https://github.com/securitySLAnegotiation