Introdução aos protocolos CL-AKA

Anderson dos Santos Paschoalon
Departamento de Engenharia de Computação e Automação Industrial
Universidade Estadual de Campinas (UNICAMP)
Campinas, Brasil
anderson.paschoalon@gmail.com

***Resumo*—A utilização de certificados digitais distribuídos por uma Infraestrutura de chaves Públicas (PKI) é a metodologia mais difundida em protocolos de acordo de chave simétrica. Neste trabalho serão discutidas algumas metodologias alternativas, com um maior enfoque em protocolos CL-AKA, ou seja protocolos de acordo de chave autenticado sem uso de certificados. Serão discutidos os fundamentos teóricos, duas implementações de protocolos, e algumas aplicações e propostas pratias.**

Palavras chaves — CL-AKA; acordo de chaves; criptografia de chave pública, criptografia sem certificados, emparelhamento bilinear

I. INTRODUÇÃO

Para realização de uma comunicação segura entre duas partes, digamos Ana e Beto, o modelo criptográfico que melhor é capaz de garantir confidencialidade e autenticidade mutuamente, para um mesmo tamanho de chave, além de ser computacionalmente mais eficiente, é o de cifragem simétrica, dês de que atenda ao os seguintes requisitos:

- A chave deve ser suficientemente longa, para resistir a ataques de força bruta.
- O algoritmo de cifragem deve ser suficientemente forte, e ao mesmo tempo leve. Até a presente data, tanto o 3DES quanto o AES representam ótimas soluções que satisfazem tal requisito.
- A chave compartilhada entre a Ana e o Beto deve ser secreta; ou seja, conhecida somente por ambos.

Para situações onde a Ana e o Beto podem entregar pessoalmente essa chave um para o outro, o problema está resolvido. Mas na pratica essa situação em muitos casos não é razoável. A situação piora muito se tivermos um numero maior de usuários dentro de um sistema, o que torna o acordo de pares de chaves entre cada usuário impraticável. Temos portanto aqui o problema da distribuição de chaves secretas.

Como alternativa, temos os sistemas criptográficos de chave pública. Eles se baseiam em dois elementos para troca de informações seguras: a chave pública *PU*, e a chave privada *PR*. Neles supõe-se que todos os usuários tem acesso a chave pública, enquanto a chave provada deve permanecer em segredo. (*PR*,*PU*) devem ser relacionadas matematicamente, mas deve ser computacionalmente inviável a partir da pública derivar a privada. E no caso da cifragem de mensagens, uma mensagem cifrada com uma delas, somente pode ser decifrada com a outra.

Sistemas criptográficos de chaves públicas possuem a são capazes de fornecer os serviços de confidencialidade e autenticidade, mesmo sem a existência de um canal seguro.

Há dois tipos principais de sistemas criptográficos de chave pública:

- Sistemas de cifragem de mensagens: realizam a cifragem e a decifragem assimétrica de qualquer mensagem de entrada. Para se garantir confidencialidade, a cifragem deve ser feita utilizando-se a chave pública, sendo necessária a chave privada para a decifragem. Para se fornecer confidencialidade, a cifragem é feita utilizando-se a chave privada, e se utiliza a chave pública para a decifragem. Tal cifragem equivale a uma assinatura, já que somente o dono da chave privada a poderia ter efetuado.

- Sistemas de troca de chaves: realizam o acordo de uma chave secreta entre duas partes, de maneira que somente ambas possam chegar no mesmo valor comum, apenas trocando informações públicas. A Autenticidade e confidencialidade são dadas pela chave secreta simétrica compartilhada. Neste trabalho, o maior foco será em sistemas de troca de chaves.

Porém, apesar da criptografia de chave públca ser capaz de fornecer tais serviços, por si só ela não é capaz de garanti-los. Ataques como o *man-in-the-middle* são possíveis, caso o protocolo de comunicação possua vulnerabilidades.

A solução padrão mais difundida para tal problema atualmente, definida pelo RFC 4949, é a de emissão de certificados digitais por parte de uma Infraestrutura de chaves públicas, ou PKI (*Public Key Infrastructure*). PKI é definido um agregado de hardware, software e pessoas, que tem por responsabilidade gerenciar, armazenar, distribuir e revogar certificados digitais [1].

O funcionamento básico dos certificados emitidos por uma PKI,se baseiam nas seguintes ações:

- Ana entrega sua chave pública a uma PKI, e recebe de volta o certificado. Certificados digitais podem conter diversas informações, mas as principais são as seguintes: um *timestamp T* que indica a data de expiração do certificado, o identificador da usuaria Ana ID_A , e a chave pública da Ana PU_A. Tudo é então cifrado com a chave privada da autoridade PR_{auth}. Aqui representaremos o certificado por C_A:

  $$C_A = E\,(PR_{auth}, "T \,||ID_A\,||PU_A")\qquad (01)$$

- A Chave Pública da PKI deve ser divulgada para todos os demais usuários do sistema.
- Ana deverá publicar seu certificado C_A.
- Caso a chave privada da Ana seja comprometida, e sua validade não tenha sido atingida, ela deverá revogar o certificado, avisando a PKI.
- Ao utilizar certificados de terceiros, a Ana deverá utilizar alguma metodologia de consulta ao PKI para se assegurar que o certificado é valido. Podem ser utilizadas listas de certificados revogados (com definição mais recente no RFC 5280), ou o mais recente protocolo OCSP (RFC 6960), baseado na consulta *online* do *status* do certificado.

Tal solução, porém cria outras dificuldades, como:

- Complexidade na manutenção da criação e manutenção do PKI.
- Dificuldades envolvendo a revogação dos certificados (por exemplo, alguns usuários podem estar utilizando certificados já revogados).
- Custo Adicional relacionado a recuperar e validar o certificado
- Tal estratégia não elimina a necessidade de consulta a uma autoridade (mesmo que ela seja menos freqüente).

Serão abordadas alternativas para o modelo clássico de certificados digitais emitidos por uma PKI. Na próxima sessão serão mostrados alguns modelos já propostos na literatura, e como eles convergiram para o modelo de criptografia de chave pública sem certificado. Em seguida serão discutidos os protocolos CL-AKA (*Certificateless Authenticated Key Agreement Protocol*).

II. Modelos Alternativos para Sistemas Criptográficos de Chave Pública

Como alternativa ao método mais difundido de certificados emitidos e mantidos por uma PKI surgiram diversas propostas de sistemas criptográficos de chave pública alternativos. Os principais são os seguintes:

1. Criptografia de chave pública baseada em Identidade;
2. Criptografia de chave pública autocertificada;
3. Criptografia de chave pública baseada em certificado;
4. Criptografia de chave pública sem certificado.

O foco deste trabalho são esquemas de criptografia de chave pública sem certificado, em

especial algoritmos de acordos de chave autenticados sem certificado. Porém, como será explicado mais a frente, estes tipos de esquema surgem de uma combinação dos conceitos do primeiro e terceiro esquemas. Por isso, eles serão brevemente introduzidos.

Antes de prosseguir será definido aqui o conceito de níveis de confiança na autoridade:

- **Nível 1**: a autoridade conhece, ou pode calcular com facilidade as chaves secretas dos usuários, podendo se passar por qualquer entidade sem ser detectada.
- **Nível 2**: a autoridade não conhece as chaves secretas dos usuários, mas pode se passar por qualquer entidade, e emitir chaves públicas sem que seja possível provar que ela o fez.
- **Nível 3**: a autoridade não conhece as chaves públicas dos usuários, pode personificar outras entidades, porém pode ser detectada.

Vemos por exemplo que um esquema que utiliza certificados e uma PKI chega a um nível 3 de confiança. Uma descrição mais detalhada destas propostas, bem como funciona a criptografia de chave pública autocertificada pode ser encontrada em [2].

A. Criptografia de chaves públicas baseada em identidade

Tal esquema se propõe a resolver o problema da autenticidade das chaves públicas, utilizando o próprio identificador do usuário coma a chave pública.Mas como implementar um esquema criptográfico assim? Uma alternativa é assumir que exista uma autoridade, na qual é possível se confiar, e atribuir a ela as seguintes funções:

- Ela deve criar e manter segura uma chave privada PR_{auth}.
- Deve ser responsável por identificar e registrar todos os usuários do sistema.
- Deve ser responsável por calcular as chaves privadas dos usuários $PR_{auth-user}$ (subscrito indica que ela é conhecida pela autoridade e pelo usuário).
- Deve entregar essa chave secreta aos usuários se maneira segura (ou seja, com sigilo e autenticidade).

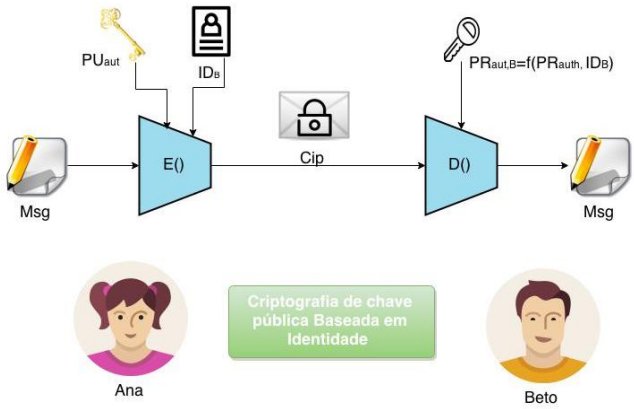

Fig. 1. Esquema de troca de mensagens com a utilização da criptografia de chave pública baseada em identidade. Ana cifra uma mensagem *Msg* com a chave pública a autoridade e a identidade de Beto. Beto então é capaz de decifrar a mensagem utilizando sua chave privada (calculada a partir da chave privada da autoridade, e de sua própria identidade, pela própria autoridade, e entregue a ele de forma segura e confidencial).

Publicar a chave pública PU_{auth} gerada a partir da chave secreta PR_{auth}.

Dessa maneira se tem as seguintes situações possíveis:

- Ana decide enviar uma mensagem para Beto. Para isso, ela cifra seu texto em claro utilizando as informações públicas do sistema: a identidade de Beto e a chave pública da autoridade. Beto decifra o texto cifrado com sua chave privada PR_{auth}.
  $A \rightarrow B: Cip = E(ID_B, PU_{auth}, Msg)$
  $B: Msg = D(PR_{auth-B}, Cip)$
- Para verificar a autenticidade de uma mensagem criada por Beto, cifrada com a chave privada PR_{auth}, basta os usuários a decifrarem com a a identidade de Beto, e a chave publica da autoridade.

Temos algumas vantagens e desvantagens dessa abordagem alternativa.

No modelo tradicional, para um usuário cifrar alguma mensagem, primeiramente ele precisa obter um certificado de seu remetente. Precisa também verificar a assinatura de tal certificado, e em caso se usos futuros, se ele está revogado, o que pode consumir muito tempo. Já no modelo baseado em identidade nenhuma verificação posterior é necessária, o que leva a uma grande economia de

custo de processamento nos usuários e nas autoridades, além de economia de banda.

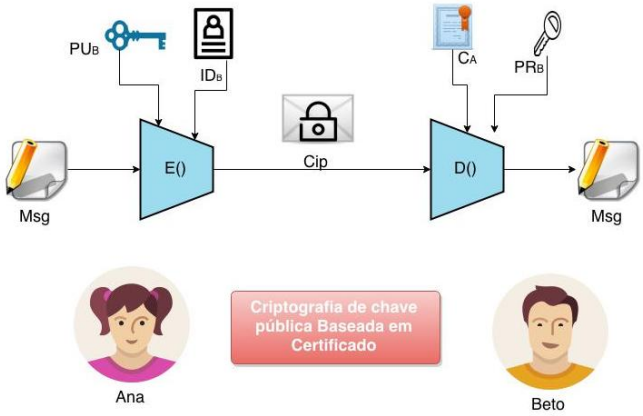

Fig. 2. Esquema de troca de mensagem de forma secreta utilizando criptografia de chave pública baseada em certificado. Nele, o próprio criador do certificado o mantém. Ana deve cifrar a mensagem utilizando a chave pública e a identidade de Beto. Para decifrar, ele utiliza o certificado emitido pela autoridade, e sua própria chave privada.

Porém o modelo apresenta algumas primeiro lugar a autoridade possui a chave privada dos usuários, o que faz com que se demande um nível 1 de confiança. Tal problema é chamado de "custódia de chaves". Há aqui também um risco muito mais elevado que o do convencional caso a chave mestra da autoridade seja comprometida. Isso permitiria que um atacante tivesse acesso as chaves secretas dos participantes do sistema. Por fim há um problema quanto a possibilidade de revogação de identidades.

B. Criptografia de Chave Publica Baseada em certificados

Nesse modelo, proposto pela primeira vez por Gentry [8], tinha por objetivo não só utilizar certificados também como parte da chave de decifrамento.

Nos protocolos baseados em certificados e PKIs, como certificados são revogáveis, os usuários do sistema não poderiam utilizá-los sem antes verificar a validade junto à PKI. Os métodos e protocolos existentes para realizar tal comunicação resultam inevitavelmente em um maior custo computacional, de banda, e eventuais sobrecargas sobre a PKI.

A idéia central por traz da criptografia baseada em certificados é a de que os certificados passem a ser parte da senha utilizada no decifrамento das mensagens. Se um certificado for revogado, assim que o usuário que o emitiu tomar conhecimento, as mensagens relacionadas ao protocolo antigo se tornam invalidas, automaticamente. Dessa maneira, não haverá meios de certificados revogados serem utilizados de maneira indevida.

Portanto tal conceito visa combinar características de métodos de criptografia com certificados tradicionais, e do esquema baseado em identidade. O certificado deve ser gerado utilizando um esquema de cifragem baseado em identidade.

Dessa forma o trafego de validação de certificados é eliminado e os certificados são usados apenas pelo próprio usuário criador da mensagem, funcionando como parta de chave privada, mesmo podendo ser transmitido via canal público. Para terceiros, tal certificado não possui função alguma, já que para cifrarem ou verificarem assinatura de A, basta a chave publica e a identidade de A.

Em um cenário que Ana deseja cifrar uma mensagem para Beto, temos as seguintes etapas:

$$A \rightarrow B : Cip = E(ID_B, PU_B, Msg)$$

$$B : Msg = D(C_B, Cip)$$

Comparado com o sistema baseado em identidade, ele apresenta a vantagem de eliminar a custodia de chaves por parte da autoridade, fazendo com que os usuários precisem de um nível 3 de confiança na autoridade. Além disso, não necessita do estabelecimento prévio de um canal sigiloso. Por outro lado, tal sistema ainda exige a existência de uma infraestrutura capaz de emitir certificado, podendo ser então considerada uma variação "mais leve" do modelo baseados em emissão de certificados por uma PKI.

Comparado com o modelo tradicional, o modelo possui algumas desvantagens. Nesse modelo, revogar um certificado significa apenas parar de emiti-los por parte da autoridade. Por isso há a necessidade de se informar todos os demais usuários que a chave pública mudou. Se isso não for feito, aquele que violou a chave privada poderá continuar utilizando o certificado antigo até o fim da validade do mesmo, seja para criar assinaturas falsas, ou ler mensagens cifradas. Devido a isso, para se minimizar os eventuais danos relacionados ao corrompimento de chaves, a validade dos certificados deve ser curta, o que pode sobrecarregar

a autoridade. Além disso o registro da chave pública deverá ser feito sob um canal seguro e autenticado.

C. Criptografia de chaves públicas sem certificados

Se trata de um modelo intermediário entre o modelo de criptografia baseado em identidade e o modelo tradicional baseado em certificados divulgados pela PKI, tendo sido proposto pela primeira vez por Al-Riyami e Patterson, em 2003 [11].

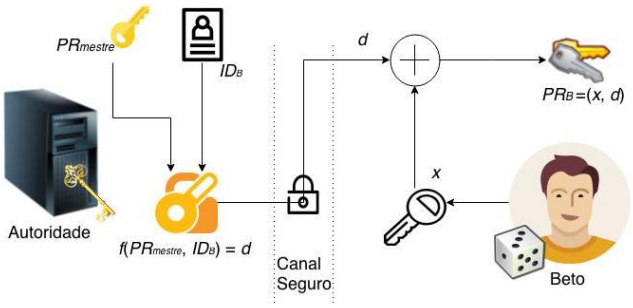

Fig. 3. Criação da chave privada completa a ser utilizada pelo usuário. Ela trata-se da combinação de duas parcelas: a chave privada parcial, que é gerada pela própria autoridade, e entregue por meio de um canal seguro para o usuário; e a chave privada secreta, gerada pelo próprio usuário. Na figura o símbolo de soma (+) não indica a operação aritmética, e sim concatenação.

O modelo deixa de ser baseado em identidade, pois há uma chave pública diferente do identificador do usuário. Porém, ainda é realizado o uso da identidade, o que fornece certificação implícita.

Portanto tal abordagem tenta reunir o melhor dos dois mundos: não há certificados, e não há custódia de chaves.

Neste modelo, há a existência de uma autoridade de confiança. Ela tem por função registrar todos os usuários, e calcular parte das chaves secretas, chamada de segredo parcial *d*.

As chaves parciais calculadas pela autoridade são função da chave privada mestre da autoridade e da identidade do usuário, ou seja $d = f(PR_{auth}, ID_{user})$. Esta chave deve ser entregue pela autoridade por um **canal seguro**, com autenticidade e sigilo. Porem, tal chave parcial não permite a realização de assinaturas nem decifrarem de mensagens, logo não na o problema da custódia de chaves.

O calculo da chave pública do usuário se da da maneira convencional: o usuário escolhe um segredo x_{user} (normalmente um numero aleatório suficientemente grande), e a partir dele calcula a chave pública *PU*, e a publica distribuindo-a por meio de um canal público, ou a depositando em um diretório de chaves publicas. Por fim, a chave privada completa é dada por $PR_{user} = (x_{user}, d)$.

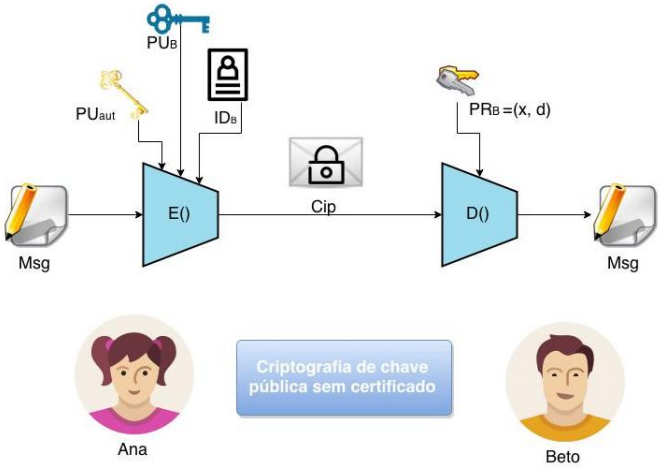

Fig. 4. Esquema de cifragem de uma mensagem, utilizando-se criptografia de chave pública sem certificado. Para Ana enviar uma mensagem criptografada para Beto, ela deve utilizar também a chave pública da autoridade, a chave pública de Beto, e a Identidade de Beto. Beto então, para decifrar a mensagem, utiliza sua chave privada completa, composta pela chave privada parcial (compartilhada entre si e a autoridade), e a chave privada secreta, criada por si próprio.

Voltando aos exemplos de comunicação entre a Ana e o Beto, caso Beto decida cifrar uma mensagem para a Ana somente irá precisar de sua chave privada $PR_B = (x_B, d_B)$. Para decifrar, a Ana irá precisar da identidade de Beto, sua chave pública, e a chave pública da autoridade.

Para o processo de assinatura de uma mensagem de Beto para a Ana teremos a seguinte situação:

$$B \rightarrow A : Cip = E(PR_B, Msg)$$

$$A : Msg = D(ID_B, PU_B, PU_{auth})$$

No processo de cifragem de uma mensagem secreta da Ana para o Beto, teremos:

$$A \rightarrow B : Cip = E(ID_B, PU_B, PU_{auth}, Msg)$$

$$B : Msg = D(PR_B, Cip)$$

Agora será verificada a resistência desse sistema contra alguns tipos de ataques. Supomos um primeiro tipo de ataque onde um usuário do sistema, aqui chamado de Caio, tenta utilizar o *man-in-the-middle*, onde ele tenta substituir as chaves públicas sendo trocadas entre a Ana e o Beto com intuito de

através de uma chave publica falsa, se passar por Beto. Por hipótese, Caio não conhece a chave privada mestre da autoridade. Como a chave privada de Beto depende de sua identidade e da chave privada mestre da autoridade, Caio sera incapaz de criar uma chave pública que combinada com a identidade de Beto, e a Chave Pública da autoridade resulte em alguma informação legível. Portanto se diz que a mensagem é implicitamente autenticada.

Essa situação é exemplificada abaixo:

B $\rightarrow$ ~~A~~ C : PU_B

C $\rightarrow$ A: PU_B'

B $\rightarrow$ ~~A~~ C: $Cip = E(PR_B , Msg)$

C $\rightarrow$ A : $Cip' = E(PR_B', Msg')$

A : $Msg = D (ID_B , PU_B, PU_{auth} , Cip')$

A : Msg = {*ilegível*}|"*A chave privada parcial utilizada e incompatível com* PU_{auth}"

Porém, a criação de mensagens indecifráveis, por meio da substituição de chaves públicas é a base para o ataque DoD (*Denial of Decryption*). Nele um atacante altera a chave publica armazenada em um diretório público, ou transmitida em canal público. Apesar de menos danoso, tal ataque vai impossibilitar o destinatário de ser sua mensagem

O modelo de criptografia de chave publica tem uma fraqueza quanto a autoridade mal intencionada. Supomos que por algum motivo a autoridade decida emitir chaves publicas privadas falsas de um usuário do sistema. Ela então poderá interceptar as mensagens enviadas para esse usuário, e decifrá-las, já que possui a chave privada parcial. Para ocultar a fraude, cifra novamente com a chave pública verdadeira, e as repassa para o usuário. O mesmo raciocínio pode ser usado para a realização de falsificação de assinatura. Há meios de se aumentar a segurança, porém não é possível se provar que a autoridade

Além das vantagens ja mencionadas, como a não utilização de certificados digitais e eliminação de custodia de chaves, há outros benefícios da utilização desse método, dentre elas:

- Há uma diminuição do risco relacionado a perda da chave mestra da autoridade. Se houver violação da chave mestra privada da autoridade, as chaves prejudicadas serão as parciais. Portanto o adversário sera incapaz de decifrar textos cifrados criados pelos usuários.
- Outra vantagem é que o processo renovação de chaves públicas independe da autoridade. Ou seja, o usuário não precisa comunicar a autoridade se desejar alterar sua chave pública. Basta distribuí-la para seus pares, ou depositá-la em um diretório público.

Porém o modelo possui suas desvantagens:

- Não há uma solução simples para o ataque chamado *Denial of Decryption* (DoD).
- A autenticação é implícita;
- É necessário o estabelecimento de um canal autentico e sigiloso com a autoridade para a entrega da chave parcial secreta.

Agora que os conceitos fundamentais dos sistemas de criptografia sem certificado foram introduzidos, serão abordados daqui em diante protocolos sem certificado com acordo de chaves autenticado, ou CL-AKA (*Cetificateless Authenticated Key Agreement Protocol*).

III. Protocolos de Acordo de Chave Autenticados Sem Certificados

A diferença essencial entre um algoritmo de acordo de chaves e um de cifragem de chave pública é que enquanto este realiza a criptografia de qualquer tipo de mensagem para esta ser decifrada por seu par, o algoritmo de acordo de chaves tem por objetivo a troca de valores em comum calculados a partir tanto das chaves privadas pessoas, quanto dos dados públicos fornecidos por ser par. O exemplo mais clássico é protocolo de Diffie e Hellman, proposto em 1976, precursor da cifragem assimétrica. Da mesma forma que nos algoritmos de criptografia assimétrica, há a geração de um par de chaves público e privada (*PU, PR*). A chave pública então deve ser entregue de forma autenticada a seus pares, sendo dessa forma capazes de calcular um segredo em comum.

Vamos utilizar o exemplo de comunicação entre a Ana e o Beto. Ambos escolhem uma chave privada, e a partir dela deduzem uma chave pública. Ana deve então entregar sua chave pública de forma autenticada a Beto e este a Ana. Quando isso for

feito, ambos devem utilizar as chaves públicas, juntamente com suas próprias chaves privadas para calcularem uma chave secreta em comum.

Nos protocolos de chave pública sem certificado, cada usuário possui 3 segredos:

1. Valor secreto que gerou a chave pública (conhecimento exclusivo do usuário);
2. Chave parcial secreta (calculada pelo KGC, e compartilhada de maneira segura com o usuário);
3. O segredo temporário de sessão (numero pseudo-aleatório sorteado para o cálculo da chave se sessão);

Mesmo um KGC mal intencionado, ou um adversário com acesso à chave mestra, precisaria corromper outros dois segredos adicionais para ter acesso a sessão de comunicação: o valor secreto gerado pelo usuário, e o segredo temporário.

Lippold, Boyd e González Nieto apresentam em 2009 o primeiro protocolo CL-AKA demonstrado seguro sob um modelo forte de segurança, aqui chamado de LBG. Tal modelo satisfaz a condição de que o comprometimento de dois componentes secretos da chave privada não são suficientes para um atacante descobrir a chave se sessão.

Os protocolos CL-AKA podem ser sumarizados nas seguintes fases, segundo Goya *et al.*[3]: inicialização do sistema, geração de chaves do usuário e acordo de chaves. No entanto aqui neste trabalho, quando os protocolos forem apresentados, a ultima etapa será dividida em duas: acordo de chaves e calculo das chaves secretas. Tais fases realizam as seguintes tarefas:

1. **Inicialização do sistema**: A autoridade inicialmente gera a chave mestra pública e a secreta, a partir dos parâmetros públicos do sistema (corpo, funções hash, emparelhamento bilinear, etc). Em seguida, utilizando, os parâmetros públicos, a chave pública da autoridade, as identidades das partes, e a chave mestra secreta, a autoridade calcula as chaves parciais secretas, e as distribuem de forma segura.
2. **Geração de chaves do usuário**: Utilizando suas próprias identidades e os parâmetros públicos e a chave pública da autoridade, os usuários calculam os valores secretos, e os valores públicos, a serem compartilhados.
3. **Acordo de chaves**: São executados passos nos quais as informações públicas dos usuários são trocadas.
4. **Calculo das chaves secretas**: a partir das informações obtidas na etapa anterior, dos parâmetros públicos, e das chaves privadas a chave se sessão é calculada pelas partes.

Neste trabalho serão estudadas 3 propostas de protocolos propostos o LBG já citado, a proposta de Goya, Okida e Terada chamado de GOT[3], e o CAKA [6].

IV. Fundamentos Teóricos

Antes de introduzir os protocolos CL-AKA, será discutido os fundamentos teóricos por traz dos mesmos. Esta sessão sera dividida em três partes.

Na primeira parte, será discutido o emparelhamento bilinear. Em protocolos algoritmos criptográficos de chave pública se faz necessário o uso de *funções de mão única com segredo*. Tais funções caracterizam-se pela propriedade de que calculá-las é sempre computacionalmente viável ou seja, existe uma solução para com complexidade polinomial, porém calcular a função inversa, é computacionalmente muito difícil. Isso se dá pelo fato de que não se conhece algum algoritmo com complexidade menor que exponencial que resolva o problema. Porém, conhecendo-se um segredo adicional, se torna factível o calculo da função inversa. Emparelhamentos bilineares se mostraram uma classe de funções convenientes para a criptografia de chave pública sem certificados. Como Goya comenta em [2], não é uma condição necessário algoritmos de criptografia sem chave pública utilizarem emparelhamentos bilineares, mas na pratica a grande maioria deles utilizam esse recurso.

Em seguida serão definidos os problemas ditos computacionalmente difíceis de serem resolvidos, ou seja, o ponto de definem as funções de mão única como tal, problemas estes sobre os quais os algoritmos de acordo de chave apóiam sua segurança.

Por fim, será discutido o que são modelos de segurança formais e os requisitos de segurança que protocolos CL-AKA possuem.

A. Emparelhamento Bilinear

Definição 1: É dito **grupo cíclico**, um grupo que pode ser gerado a partir de um único elemento a e da aplicação de uma operação binária "•" sucessivas vezes. Ou seja, o grupo pode ser expresso por $G = \{e, a, a \bullet a, a \bullet a \bullet a, \dots\}$, onde e é o elemento neutro dessa operação. a é chamado gerador de G. Caso essa operação seja uma soma, chamamos ele de grupo **cíclico aditivo**, e caso essa operação seja uma multiplicação, de **grupo cíclico multiplicativo**. Além disso, caso o grupo seja finito, ou seja possua um numero q de elementos, ele é dito **grupo cíclico finito de ordem q**.

Definição 2: Sendo $\mathbb{G}_1$ um grupo cíclico aditivo e $\mathbb{G}_2$ um grupo cíclico multiplicativo, ambos finitos de ordem q, com q primo. Chamamos de $e()$ **emparelhamento bilinear admissível**, um mapeamento bilinear $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, com as seguintes propriedades:

1. **Bilinearidade**: para $\forall P, Q \in \mathbb{G}_1$ e $\forall a, b \in \mathbb{Z}$, tal que $e(aP, bQ) = e(P, Q)^{ab}$
2. **Não-degeneração**: Para todo $P \neq 1_{\mathbb{G}_1}$, nós temos $e(P, P) \neq 1_{\mathbb{G}_1}$
3. **Computabilidade**: Para $\forall P, Q \in \mathbb{G}_1$ é possível se computar $e(P, Q) \in \mathbb{G}_2$ com complexidade polinomial.

B. Problemas considerados computacionalmente difíceis

A segurança dos métodos relacionados a criptografia de chave publica sem certificado, de acordo com [12], se baseiam na infactibilidade da solução dos problemas listados abaixo. Consideremos para as seguintes definições, um grupo cíclico aditivo $\mathbb{G}_1$ de ordem q (primo) e P sendo seu elemento gerador, sendo P e q publicamente conhecidos.

Problema Computacional de Diffie-Hellman, ou CDH (*Computational Diffie-Hellman Problem)*: dados $P^a, P^b \in \mathbb{G}^2$ como entradas, onde a, b são aleatoriamente selecionados de $\mathbb{Z}_q^*$, calcular P^{ab}.

Problema Bilinear de Diffie-Hellman, ou ou BDH (*Bilinear Diffie-Hellman Problem*): dados (*P, aP, bP, cP, κ*), com $P, \kappa \in \mathbb{G}_1$, e $a, b, c \in \mathbb{Z}_q$, calcular *a, b, c*, tal que $\kappa = e(P, P)^{abc}$.

Problema Bilinear Diffie-Hellman Lacunar, ou *GapBilinear Diffie-Hellman* (Gap-BDH) *Problem*: Dados (*P, aP, bP, cP*), calcular $e(P,P)^{abc}$, com a ajuda do oráculo de ajuda para o problema Bilinear de Diffie-Hellman *BDDH*( ), que dados (*P, aP, bP, cP, κ*), responde com 1 caso $\kappa = e(P, P)^{abc}$, e com 0 caso contrário.

C. Segurança demonstrável, e requisitos de segurança para protocolos CL-AKA

É denominada segurança demonstrável uma metodologia que estabelece modelos de segurança e uma prova de segurança formal para verificação se um determinado protocolo é de fato seguro.

Uma prova formal de segurança parte inicialmente da existência de um problema computacional difícil, ou seja, um *que não se conhece a existência solução em tempo polinomial*. Supõe-se então a existência de um adversário com certos poderes previamente definidos. É mostrado então, que o adversário ser capaz de atingir seu objetivo, implicaria na existência de um algoritmo de tempo polinomial para a solução do problema difícil, o que entra em contradição com a primeira hipótese.

Goldwasser e Micali em 1984 estabeleceram as bases para a metodologia de segurança demonstrável, enquanto em 1993 Bellare e Rogaway, propõe o modelo do oráculo aleatório [2].

Com relação a acordo de chaves com autenticação sem certificados (CL-AKA), em 2009, Swandon e Jao propõe um modelo mais robusto de segurança, e mostra que todos os protocolos CL-AKA propostos até então eram inseguros. Posteriormente, ainda em 2009, Lippold, Boyd e González Nieto [5] aprimora o modelo de segurança, e apresenta o primeiro protocolo de acordo de chaves demonstrado seguro sob um modelo forte de segurança. Tal moderno de segurança costuma ser referenciado como LBG. Os protocolos CL-AKA demonstrados fortes, satisfazem a condição de que dois componentes secretos comprometidos não são suficientes para um atacante descobrir uma chave de sessão. Goya *et al.*[3] cita o seguinte conjunto de requisitos de segurança para protocolos CL-AKA:

- Resistência a ataques de personificação básicos: adversários que não conheçam chaves secretas, não podem se passar por outros usuários;
- Resistência a ataques de compartilhamento desconhecido de chaves: deve ser inviável convencer um participante A que ele está compartilhando uma chave com B, quando na realidade ele está compartilhando com um usuário registrado C, quando C sabe que está compartilhando com A.
- Segurança de chave conhecida: a execução do protocolo deve produzir chaves de sessão únicas, e o protocolo deve permanecer seguro caso algumas chaves se sessão sejam descobertas pelo adversário.
- Resistência de ataques de personificação pelo comprometimento de chaves secretas: Caso a chave secreta de longa duração de A seja comprometida, um atacante B não pode ser capaz de personificar um usuário B para A.
- Segurança no futuro completa fraca:deve ser inviável para um atacante recuperar uma chave secreta de sessão, mesmo que no futuro venha a corromper todas as chaves de longa duração de todos os participantes envolvidos naquela sessão.
- Resistência a vazamento de segredos temporários: o vazamento de segredos temporários não deve comprometer outras sessões.

V. Propostas de Protocolos de acordo de Chave

Caros nesta sessão será discutida a implementação de protocolos CL-AKA. Serão analisadas duas implementações diferentes. Será vista primeiro o protocolo LBG, e protocolo GOT, proposto em Goya *et al.* [3]. A descrição dos protocolos será feita utilizando como referência as quatro fases citadas anteriormente.

A. Protocolo CL-AKA LBG

Inicialização do sistema: o KGC escolhe dois grupos primos de ordem q $\mathbb{G}_1$e $\mathbb{G}_2$, sendo P o gerador de $\mathbb{G}_1$. Escolhe Ttambém um emparelhamento bilinear $e()$ um emparelhamento bilinear admissível, com $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$São definidas também 3 funções hash $H_1()$, $H_2()$ e $H_3()$. . Escolhe um numero aleatório $s \in \mathbb{Z}_q^*$, e o define como chave privada mestre. Daí, calcula a chave pública como sendo $PU_{KGC} = sP$. Todas os parâmetros, exceto s são tornados públicos.

Geração de chaves do usuário: o usuário A escolhe um número aleatório $x_A \in \mathbb{Z}_q^*$, e define sua chave pública como $PU_A = x_AP$. O usuário recebe então do KGC sua chave secreta parcial, dada por: $d_A = \{sH_1(ID_A), sH_3(H_1(ID_U))\}$. Além disso, A também gera um segredo temporário $r_A \in \mathbb{Z}_q^*$.

Acordo de chaves: são então trocadas as seguintes mensagens:

A → B : $E_A = (r_AP, x_AP)$

B → A : $E_B = (r_BP, x_BP)$

Calculo das chaves secretas: A partir das informações trocadas, cada usuário calcula:

$K_A = e(H_1(ID_B), sP)^{rA}e(sH_1(ID_A), r_BP) = K_B$ [02]

$K'_A = e(H_3(H_1(ID_B)), sP)^{rA}e(sH_3(H_1(ID_A)),r_BP) = K'_B$ [03]

$L_A = e(H_1(ID_B), sP)^{xA}e(sH_1(ID_A), x_BP) = L_B$ [03]

$L'_A = e(H_3(H_1(ID_B)),sP)^{xA}e(sH_3(H_1(ID_A)),x_BP) = L'_B$ [04]

$N_A = e(H_1(ID_B), sH_1(ID_A)) = N_B$ [05]

$N'_A = e(H_3(H_1(ID_B)), sH_3(H_1(ID_A))) = N'_B$ [06]

Cada parte então é capaz de calcular a chave secreta de sessão, calculando:

$K_s = H_2(ID_A, ID_B, E_A, E_B, r_Ar_BP, x_Ax_BP, r_Ax_BP, x_Ar_BP, K, K', L, L', N, N')$ [07]

B. Protocolo CL-AKA GOT

Inicialização do sistema:Da mesma forma que feito anteriormente, são escolhidos os corpos de ordem q, o gerador P, as funções hash, a chave mestra, e a chave pública da autoridade.

Geração de chaves do usuário: Da mesma forma, cada usuário A escolhe um valor aleatório x_A, e o torna parte da chave privada, e calcula a chave pública como x_AP. Além disso, recebe da autoridade de forma segura a chave privada parcial d_A =

$sH_1(ID_A) = sQ_A$. Cada usuário também escolhe valores aleatórios $y_A, t_A \in \mathbb{Z}_q^*$.

Acordo de chaves: são então trocadas as seguintes mensagens:

A→B: $E_A=(ID_A,x_AP,t_AP,y_AP)= (ID_A,PU_A,C_A,Y_A)$

B→A: $E_B=(ID_B,x_BP,t_BP,y_BP) =(ID_B,PU_B,C_B,Y_B)$

Calculo das chaves secretas: Cada usuário é capaz então de computar um mesmo valor K_s, dado por:

$$K_s = H_0(K_A,K_B,K_{AB},M_1,M_2,sid,pid) \quad [08]$$

onde:

$$K_{AB} = y_B Y_A \quad [09]$$

$$M_1 = (x_B PU_A, y_B PU_A, x_B Y_A) \quad [10]$$

$$M_2 = ((t_B + y_B)(C_A + Y_A),Z_{A1} ,Z_{B1}) \quad [11]$$

$$sid = (E_A, E_B) \quad [12]$$

$$pid = (ID_A, ID_B, app_{id}) \quad [13]$$

$$Z_{A1} = x_B C_A \quad [14]$$

$$Z_{A2} = e(C_A, d_B) \quad [15]$$

$$K_A = H_2(Z_{A1}, Z_{A2}, ID_B, PU_B, C_A) \quad [16]$$

$$Z_{B1} = t_B PU_A \quad [17]$$

$$Z_{B2} = e(\underline{t_B}.PU_{KGC}, Q_A) \quad [18]$$

$$K_B = H_2(Z_{B1}, Z_{B2}, ID_A, PK_A, C_B) \quad [20]$$

VI. Aplicações

Paralelamente as pesquisas relacionadas a criação de protocolos CL-AKA eficientes e seguros, também existem iniciativas de aplicações praticas que utilizem essa tecnologia, numa tentativa de mostrar a sua viabilidade no mercado. Na sessão seguinte serão descritas brevemente algumas propostas encontrados na literatura. Em seguida, serão sugeridas algumas aplicações e soluções a problemas práticos envolvendo protocolos CL-AKA.

A. Propostas de aplicações de protocolos CL-AKA encontradas na literatura

A primeira proposta de solução aqui descrita, descrita por Goya et al. [3], trata do uso do protocolo CL-AKA no projeto Borboleta [13]. O projeto Borboleta trata-se de uma iniciativa que visa criar um suporta a tecnologia de informação em programas de atendimento médico domiciliar. Tal aplicação, proposta no mesmo artigo em que o protocolo é apresentado, trata-se do uso deste protocolo como ferramenta de troca de chaves secretas entre dispositivos móveis de acesso remoto (*smatphones* e *PDAs*), e um ponto de acesso remoto, conectado a uma máquina de acesso restrito. Dessa forma, prontuários preenchidos remotamente, seriam enviados de maneira segura para a base de dados hospitalar do SUS (Sistema Único de Saúde). Inicialmente o KGC foi instalado em uma maquina de acesso restrito, junto aos *PDAs* e *smartphones*, cria seu par de chaves publica e privada, bem como entrega a cada um dos aparelhos sua chave privada parcial. Em seguida eles calculam suas chaves privada e pública. Após a etapa de inicialização, os aparelhos têm a função de iniciar a comunicação para troca de chaves de sessão com o ponto de acesso remoto. No projeto, as implementações foram feitas em Java. Levando em conta o baixo poder computacional dos dispositivos, mostra-se, portanto que o resultado foi de fato efetivo em sua proposta.

Em uma proposta mais recente, feita por Farouk et al. [7] é proposto um protocolo do tipo CL-AKA para troca se chaves em um ambiente de *grid computing*, ou seja vários computadores trabalhando em conjunto sob um mesmo ambiente virtualizados. O que difere tal sugestão das demais apresentadas aqui até agora, é que esta não utiliza emparelhamentos bilineares. Os cálculos são feitos utilizando-se multiplicação sobre curvas elípticas, calculo de multiplicativos inversos e funções hash. Por esse motivo, os autores chamam o protocolo de GPC-AKA, um acrônimo para *Grid Computing Pairing-free Certificateless Authenticated Key Agreement protocol*. A segurança de tal protocolo foi testada por meio da versão 0.8 de uma ferramenta chamada Scyther (http://www.cs.ox.ac.uk/people/cas.cremers/scyther/), que fornece um framework que é capaz de modelar diferentes adversários para análises de seguranças de protocolos. Este software é capaz de analisar um protocolo com relação a noções de segurança futura completa fraca (*Weak Perfect Forward Secrecy*), ataques de personificação pelo comprometimento de chave secreta (*Key Compromise Impersonation*), e resistência a vazamento de segredos temporários. No entanto, os

autores não indicam se a ferramenta é capaz de suportar todos os demais requisitos de segurança para protocolos CL-AKA, como descritos por Goya et al.[3].

B. Sugestão de adaptação do protocolo GOT para suporte de autenticação explicita

Como dito anteriormente, os métodos de criptografia com chave publica sem certificados são implicitamente autenticados. Nos protocolos CL-AKA propostos, por exemplo, alguma das mensagens seja corrompida por um atacante, os usuários somente irão notar isso assim que decifrarem a primeira mensagem cifrada com a chave simétrica calculada. Isso pode ser um problema, por vários motivos, dentre eles:

- Há claramente um desperdício de recursos computacionais, visto que os usuários já iniciaram um novo processo, acreditando que a chave simétrica calculada era igual para ambas as partes.
- Além disso, se faz necessário um método de detecção na não validade da mensagem decifrada, o que pode não ser trivial.
- Com o deciframento de apenas um pacote. não é possível se ter certeza se o pacote modificado foi o utilizado no acordo de chaves, ou o cifrado com as chaves simétricas. Nesse ultimo caso, a chave simétrica calculada pode inclusive estar correta.

Aqui será então proposta uma melhoria para o protocolo GOT [3], de forma que ele possa oferecer certificação explicita, utilizando como base a proposta descrita na por Krawczyk [10]. Consideremos aqui K_s como sendo a chave calculada, e para as demais variáveis utilizaremos a mesma notação das sessões anteriores, com:

$$E_A = (ID_A, x_A P, t_A P, y_A P) \quad [21]$$

$$E_B = E(ID_B, x_B P, t_A P, y_A P) \quad [22]$$

Teremos então, a seguinte toca de mensagens:

$$A \rightarrow B : E_A$$

$$B \rightarrow A : E_B || MAC(K_s, "0")$$

$$A \rightarrow B : MAC(Ks, "1")$$

Nas mensagens representadas acima, “1” e “0” são simplesmente bytes com os valores um e zero, e $MAC()$ representa o resultado de um código autenticador de mensagem, ou função MAC.

C. Sugestão de alteração no modelo de comunicação do protocolo SSL

Por fim, será apresentado aqui mais uma possível aplicação: a proposição de uma adaptação no *handshaking* do protocolo SSL, para este poder suportar protocolos CL-AKA. Por simplificação, assume-se que não é feito o uso de diretórios de chaves publicas.

Inicialmente é tomado como referência o seguinte diagrama apresentado na figura 5, retirado da referencia [14], que apresenta de forma simplificada as principais etapas de comunicação do protocolo SSL realizadas entre o cliente e o servidor SSL. Será assumido que tanto o servidor SSL quanto o cliente já possuem suas chaves privadas completas (pessoal e parcial).

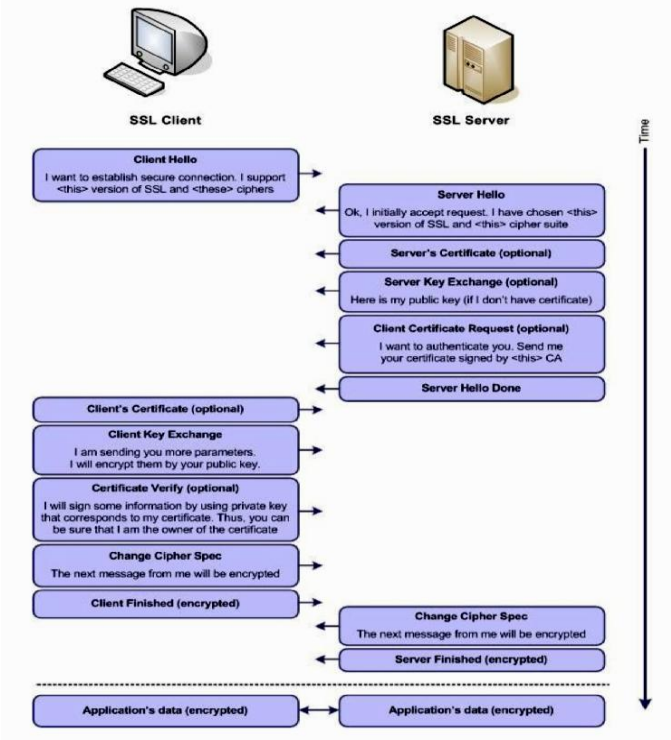

Fig. 5. Trocas de mensagens realizadas entre um cliente e um servidor SSL. Fonte: Web Service Security Tutorial, Disponível em < https://sites.google.com/site/ddmwsst/create-your-own-certificate-and-ca/ssl-socket-communication > , Acessado em 8 de Junho de 2015.

Nessa adaptação as operações de trocas de certificados serão substituídas pelo *handshaking* proposto como adaptação para o protocolo GOT. O resultado é o seguinte:

$Server \rightarrow Client : "Server\ Hello"$
$Server \rightarrow Client : E_{Server}$
$Client \rightarrow Server : E_{Client} || MAC(K_s, "0")$
$Server \rightarrow Client : MAC(Ks, "1")$
$Server \rightarrow Client : "Server\ Hello\ Done"$
$Client \rightarrow Server : "Change\ Chiper\ Spec"$
$Client \rightarrow Server : E(K_s, "\text{Client Finished}")$
$Server \rightarrow Client : "Change\ Cipher\ Spec"$
$Server \rightarrow Client : E(Ks, "Server\ Finished")$
$< ApplicationData >$

VII. CONCLUSÃO

Neste trabalho foi apresentada uma introdução a soluções alternativas a utilização de certificados emitidos por PKIs, dentre elas ceriptografia de chave pública por identidade, com certificados, e sem certificados. Esta ultima, trata-se de uma solução de compromisso entre as duas outras abordagens. Depois de apresentado este paradigma, voltou o foco para protocolos CL-AKA, ou seja, protocolos de acordo de chaves autenticado sem certificados, que utiliza os fundamentos da criptogradia sem certificados, porém não para o transporte de mensagens, e sim para o acordo de chaves simétricas. Existe hoje uma certa variedade desses tipos de protocolos demonstrados seguros, e aqui foram apresentados dois: o LBG[5] e o GOT[3]. Por fim, foram mostrados algumas aplicações praticas já realizadas, envolvendo algumas implementações, e também foram sugeridas algumas propostas de implementações, dentre elas uma a adaptação do GOT para suportar certificação explicita e adaptação no modelo de comunicação do SSL.

REFERÊNCIAS

[1] Stallings, Willian. "Chapter 14: Key Management and Distribution." Em Cryptography and Network Security Principles and Practice, 731. Sixth ed. Upper Saddle River, New Jersey: Pearson, 2014.

[2] Goya, Denise Hideko. "Criptografia de chave pública sem certificado". 2011-12-16. 152 folhas. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística Universidade de São Paulo. São Paulo, 2011. http://www.teses.usp.br/teses/disponiveis/45/45134/tde-02082012-191831/pt-br.php

[3] Goya, Denise, Cleber Okida, and Routo Terada. "Acordo De Chave Com Autenticação Sem Certificado Digital." 9th International Information and Telecommunication Technologies Symposium, 2010, pp.1-8. http://www3.ime.usp.br/~isc2002/lsd/attachments/062_i2ts_Acordo_final_DeniseCleberRouto.pdf.

[4] C. Swanson and D. Jao, "A study of two-party certificateless authenticated key-agreement protocols " in INDOCRYPT '09: Proceedings of the 10th International Conference on Cryptology in India, ser. Lecture Notes in Computer Science, vol. 5922. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 57–71.

[5] G. Lippold, C. Boyd, and J. Gonzalez Nieto, "Strongly secure certificateless key agreement" in Pairing '09: Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography, ser. Lecture Notes in Computer Science, vol. 5671. Berlin, Heidelberg: Springer-Verlag, 2009, pag. 206–230, cryptology ePrint Archive, Report 2009/219,

[6] Srivastava, Renu, and A.K. Misra. "An Improved Certificateless Authentication Key Agreement Protocol." International Journal of Computer Applications 70, no. 3 (2013): pp. 15-19. doi:10.5120/11942-7736.

[7] Farouk, Amr, Mohamed Fouad, and Ahmed Abdelhafez. "Analysis and Improvement of Pairing-free Certificate-less Two-party Authenticated Key Agreement Protocol for Grid Computing." International Journal of Security, Privacy and Trust Management (IJSPTM) 3, no. 1 (2014): 14. doi:DOI:10.5121/ijsptm.2014.3103.

[8] Gentry, Craig. "Certificate-Based Encryption and the Certificate Revocation Problem." Advances in Cryptology — EUROCRYPT 2003 Lecture Notes in Computer Science International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 2656:21. doi:10.1007/3-540-39200-9_17.

[9] P1363 Standard Specifications for Public-Key Cryptography, IEEE, 2000.

[10] H. Krawczyk, "Hmqv: a high-performance secure diffie-hellman protocol" in Advances in Cryptology a CRYPTO 2005, LNCS 3621 , 2005, p. 1-62.

[11] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in ASIACRYPT 2003, ser. Lecture Notes in Computer Science, vol. 2894. Springer, 2003, cryptology ePrint Archive, Report 2003/126, http://eprint.iacr.org/.

[12] Yang, Guomin, and Chik-How Tan. "Strongly Secure Certificateless Key Exchange without Pairing." ASIACCS 2011 6th ACM Symposium on Information, Computer and Communications Security, 2011, pp. 71-79.

[13] "Projeto Borboleta: Sistema Integrado De Computação Móvel Para Atendimento Domicilar De Saúde." Fapesp. October 31, 2010. Acessado em 8 de Junho, 2015 <http://www.bv.fapesp.br/pt/auxilios/23929/projeto-borboleta-sistema-integrado-de-computacao-movel-para-atendimento-domicilar-de-saude/>.

[14] "SSL Socket Communication." Web Service Security Tutorial. Acessado em 8 de Junho, 2015. <https://sites.google.com/site/ddmwsst/create-your-own-certificate-and-ca/ssl-socket-communication>.